Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

ΠΡΟΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

Εκπόνηση Πτυχιακής Εργασίας:

Εύρεση Αλγορίθμου Τερματισμού στα Opportunistic Networks του Μελλοντικού Διαδικτύου

Παπαδόπουλος Σωκράτης

Πειραιάς 2011

Περίληψη

Η ανάγκη του Μελλοντικού Διαδικτύου για νέες εφαρμογές και υπηρεσίες καθώς και η εκτεταμένη χρήση της ασύρματης επικοινωνίας, απαιτούν την απαλλαγή από παρωχημένες νοοτροπίες δικτύωσης. Τα εξελισσόμενα Opportunistic Networks θέτουν τις βάσεις για την εξασφάλιση αυτών των απαιτήσεων. Αποτελούν προσωρινή προέκταση της βασικής υποδομής του δικτύου με κύριο χαρακτηριστικό την αποδοτικότερη χρήση των πόρων του. Ακόμη, χαρακτηρίζονται από την καινοτομική δομή τους και την επιλεκτική προσθήκη νέων συσκευών, ενώ οι εφαρμογές τους χρησιμοποιούνται ολοένα και περισσότερο σε ποικίλους τομείς. Μεγάλο ενδιαφέρον επιστημονικής έρευνας παρουσιάζουν οι φάσεις λειτουργικότητας του κύκλου ζωής τους, με αποκορύφωμα την εύρεση κατάλληλου αλγορίθμου για τον τερματισμό του. Η δημιουργία ενός τέτοιου μηχανισμού μελετάται και μοντελοποιείται στα πλαίσια αυτής της εργασίας, με απώτερο σκοπό την υλοποίηση και εκτέλεσή του. Έπειτα, χρησιμοποιώντας αυστηρά καθορισμένα σενάρια και τοπολογίες εφαρμόζεται σε πρόγραμμα προσομοιωτή, με τα αποτελέσματά του να ισχυροποιούν τη σπουδαιότητα του αλγορίθμου.

Λέξεις Κλειδιά

Opportunistic network, oppnet, μελλοντικό διαδίκτυο, cognitive management systems, χαρακτηριστικά oppnet, φάσεις λειτουργικότητας, αλγόριθμος τερματισμού, υλοποίηση αλγορίθμου, σενάρια κόμβων, προσομοίωση, σύγκριση αποτελεσμάτων.

Περιεχόμενα

Κεφάλαιο 1
Opportunistic Networks στο Μελλοντικό Διαδίκτυο

Εισαγωγή

Μία από τις βασικές απαιτήσεις του Μελλοντικού Διαδικτύου (ΜΔ), που επηρεάζει ποικιλοτρόπως τα οικονομικά, κοινωνικά και τεχνοκρατικά κριτήρια των μελλοντικών δικτύων είναι η ανάγκη για νέες εφαρμογές/υπηρεσίες καθώς και η επέκταση της χρήσης των ασύρματων επικοινωνιών. Σήμερα το διαδίκτυο είναι τόσο πετυχημένο γιατί προσφέρει μεγάλη ευλυγισία στους χρήστες του, αφού είναι προσβάσιμο μέσω πολλαπλών φυσικών μέσων και υποστηρίζει πολλές και διαφορετικές εφαρμογές και τύπους δεδομένων. Στις πρώτες δεκαετίες της λειτουργίας του, η ενσύρματη πρόσβαση ήταν η επικρατέστερη ενώ οι τύποι των εφαρμογών που υποστηρίζονταν ήταν περιορισμένοι συμπεριλαμβάνοντας κυρίως τη μεταφορά αρχείων, το ηλεκτρονικό ταχυδρομείο, τις client-server web εφαρμογές/υπηρεσίες. Στο ΜΔ ο τρόπος σύνδεσης που τείνει να εδραιωθεί σε πολλά σημεία του είναι ο ασύρματος, ενώ ταυτόχρονα εξαπλώνεται το ενδιαφέρον για νέες εφαρμογές, στηριζόμενο στην τάση της εποχής για πρόσβαση στο διαδίκτυο σε κάθε πτυχή της καθημερινότητάς μας. Λόγου χάρη, οι υπηρεσίες τηλεπικοινωνίας και οι πληροφορίες ανά τον κόσμο αναρτώνται στα εξελισσόμενα δίκτυα κοινωνικοποίησης (social networks) με αποτέλεσμα την εκτεταμένη χρήση έξυπνων συσκευών (smart devices). Αυτό υποδηλώνει την ανάγκη κατάλληλης σύνδεσης και παροχής υπηρεσιών σε οποιοδήποτε σημείο θεωρηθεί απαραίτητο, σε οποιαδήποτε ώρα της ημέρας. Φυσικά, υπάρχουν και άλλοι τομείς που θα έχουν προνόμια από ένα τέτοιο ΜΔ, όπως η διοίκηση της υποδομής δικτύων που απαρτίζονται από ασύρματες συσκευές/κόμβους/τερματικά, η παρακολούθηση και προστασία του περιβάλλοντος, οι επιχειρήσεις από τη σκοπιά δημιουργίας καινοτόμων προϊόντων ή την παροχή νέων ψηφιακών συσκευών που αφορούν την υγεία, την εκπαίδευση κλπ, καθένας από τους οποίους απαιτεί ένα ικανοποιητικό σύνολο πόρων μέσα στο δίκτυο.

Επίσης, η ανάγκη αύξησης της αποδοτικότητας των παρεχόμενων πόρων (efficient resource provision) και της χρήσης τους παραμένει ένα σημαντικό θέμα, το οποίο μέχρι σήμερα διευθετείται βασισμένο στο χειρότερο σενάριο (σενάριο ώρας αιχμής). Το γεγονός αυτό, έχει οδηγήσει σε έναν τέτοιο σχεδιασμό ο οποίος δεν είναι αποδοτικός σε όλες τις υπόλοιπες ώρες λόγω της υπέρ-παροχής αχρησιμοποίητων πόρων. Αυτή η αλόγιστη χρήση πρέπει να εκλείψει διότι όπως όλοι γνωρίζουμε οι ασύρματοι πόροι είναι πεπερασμένοι και ταυτόχρονα δαπανηροί, σε αντίθεση με τους ενσύρματους. Έτσι, πολλοί είναι οι πάροχοι που έχουν λάβει ορισμένα προσωρινά μέτρα για την αντιμετώπιση αυτού του προβλήματος, όπως για παράδειγμα η πρόσθεση Wi-Fi σημείων πρόσβασης για τη μείωση της κινητικότητας, σε τοπολογίες υψηλής συμφόρησης στο δίκτυο υποδομής τους.

Οι παραπάνω απαιτήσεις του ΜΔ γεννούν ένα νέο κίνητρο αναζήτησης για επιπλέον αποδοτικότητα και συνεπώς την αναδιανομή των παρεχόμενων πόρων. Για να επιτευχθεί αυτό, πρέπει με την πάροδο του χρόνου να υλοποιηθούν ορισμένοι στόχοι όπως, υψηλότερη αποδοτικότητα των πηγών, μείωση της κατανάλωσης ενέργειας και ισχύος, μείωση του συνολικού κόστους κατοχής (total cost of ownership). Η προτεινόμενη λύση του OneFit Project [1], παρατίθεται παρακάτω και αφορά πρώτον τους προαναφερθέντες στόχους και δεύτερον προσφέρει νέες ευκαιρίες στην ασύρματη βιομηχανία καθώς και στους χρήστες της.

Opportunistic Networks (OppNets): είναι δίκτυα που ελέγχονται αυστηρά και μόνο από τους παρόχους σε θέματα πολιτικής, εύρους ζώνης, παρακολούθησης, διατήρησης καθώς και τερματισμού, με δύο βασικά χαρακτηριστικά. Το πρώτο αφορά τη διάρκεια ζωής τους, η οποία είναι σχετικά σύντομη αφού αποτελούν προσωρινή προέκταση της βασικής υποδομής του δικτύου, περιλαμβάνοντας κατά δεύτερον στοιχεία και συσκευές οι οποίες είναι σκόπιμα τοποθετημένες με τέτοιον τρόπο, ώστε μετά τη διεκπεραίωσή του να έχουν τη δυνατότητα μεταφοράς (handover) σε άλλο σημείο του δικτύου.

Cognitive Management Systems (CMS): είναι ένα εμπειρικό σύστημα διαχείρισης το οποίο θα είναι υπεύθυνο για την παρακολούθηση όλων των συστατικών ενός OppNet και θα παρέχει τη δυνατότητα στη βασική υποδομή του δικτύου (δηλαδή στο infrastructure) να ελέγχει τη λειτουργία ενός ή περισσότερων δικτύων OppNet καθώς και να συμβάλει στο συντονισμό τους.

1.1 Opportunistic Networks

Τα OppNets είναι προσωρινά δίκτυα που ελέγχονται αυστηρά και μόνο από τους παρόχους και αποτελούν προέκταση της βασικής υποδομής τους. Η δημιουργία τους είναι άκρως δυναμική βασιζόμενη στο φάσμα, στην πολιτική, στις πληροφορίες, στη γνώση του παρόχου και σε οποιαδήποτε στιγμή κριθεί απαραίτητη η μεταφορά δεδομένων προς ασύρματους κόμβους, με την καλύτερη δυνατή απόδοση. Τα τερματικά και οι συσκευές που περιλαμβάνουν τοποθετούνται με έναν τέτοιο τρόπο ώστε να εξυπηρετείται η όσο το δυνατόν μακροβιότερη λειτουργία τους, ενώ πολλά από αυτά είναι πιθανό να ανήκουν στη βασική υποδομή δικτύου (Εικ. 1.1). Τα OppNets αποτελούν μια σοβαρή πρόταση για τη λύση των προβλημάτων του ΜΔ αφού τα χαρακτηριστικά τους [2] είναι εκείνα που τα εδραιώνουν σε πλεονάζουσα θέση.

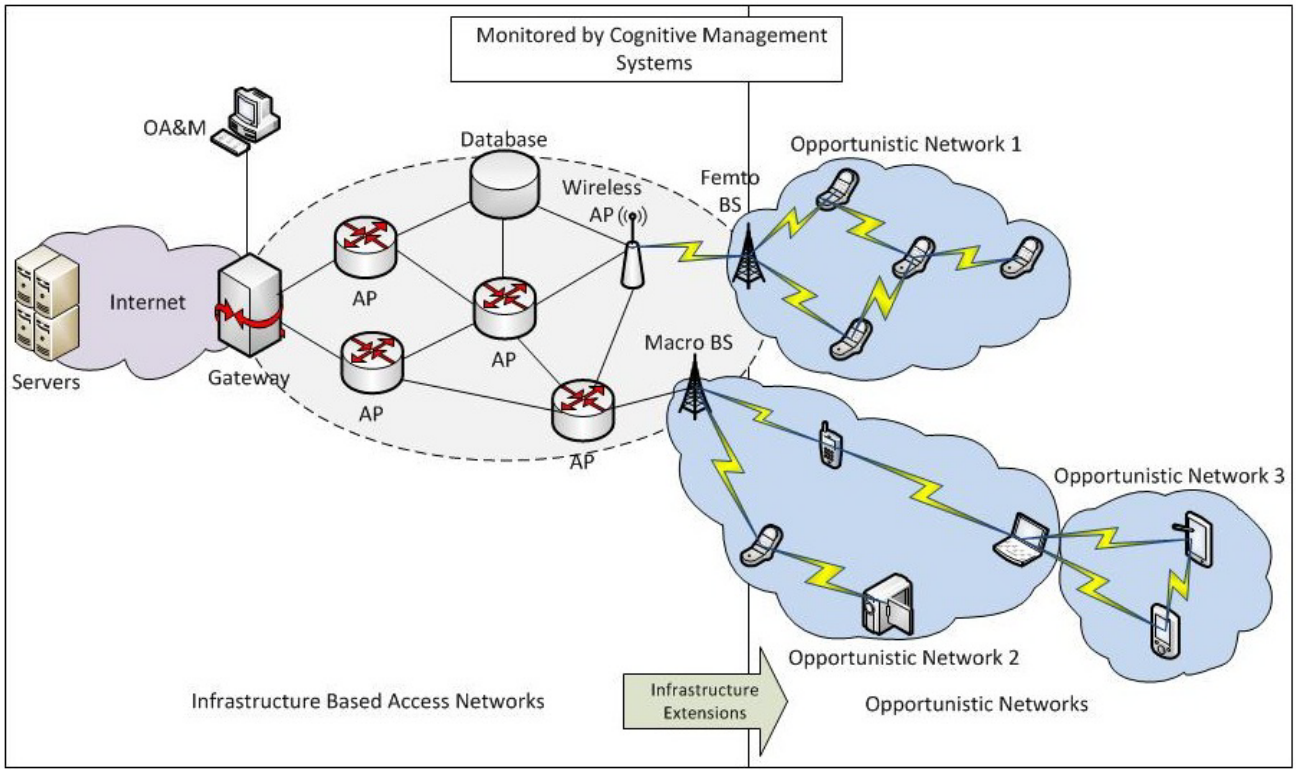
Ένα από τα σπουδαιότερα χαρακτηριστικά τους είναι ο συνεχής και ακατάπαυστος έλεγχος (operator governed), μέσω της διαθεσιμότητας των πόρων και των πολιτικών που ακολουθούν σε θέματα τεχνολογίας, χρέωσης, ασφάλειας κλπ. καθώς και του ευρύτερου πλαισίου πληροφοριών που χρησιμοποιούν, στοιχεία τα οποία είναι απαραίτητα για τη δημιουργία τους αλλά και τη μετέπειτα διατήρησή τους.

Επιπρόσθετα, αποτελούν επέκταση της βασικής υποδομής του δικτύου (infrastructure) η οποία απαρτίζεται από πολλαπλές συσκευές και τερματικά, σκοπίμως τοποθετημένα με ένα χωρίς υποδομή σχεδιασμό (infrastructure-less) κυρίως για λόγους συνδεσιμότητας και δρομολόγησης.

Σημαντικό χαρακτηριστικό των OppNets είναι η προσωρινή διάρκεια ζωής τους. Αυτή εξαρτάται από διάφορους παράγοντες, όπως για παράδειγμα η ολοκλήρωση ή μη της παροχής υπηρεσιών/εφαρμογών στους τελικούς χρήστες. Όσο εκείνοι συνεχίζουν να τις χρησιμοποιούν το δίκτυο θα πρέπει να παραμένει συνδεδεμένο. Ένα άλλο παράδειγμα θα μπορούσε να είναι η υποστήριξη μίας εταιρίας σε συγκεκριμένη περιοχή και χρόνο για την ανάπτυξη ενός προϊόντος.

Στα χαμηλότερα επίπεδα δικτύου ο πάροχος, είναι αυτός που καθορίζει το φάσμα που θα χρησιμοποιηθεί για την επικοινωνία των κόμβων του OppNet. Το αδειοδοτημένο φάσμα δηλαδή θα διαμοιραστεί σε συνεργασία με τη βασική υποδομή του.

Τέλος, το επίπεδο δικτύου εκμεταλλεύεται την επίγνωση σε θέματα πολιτικής, υλικού, προφίλ και πληροφοριών για να βελτιστοποιήσει τη δρομολόγηση και παράδοση των πακέτων.



Εικ. 1.1 Opportunistic Networks υπό τον έλεγχο των Cognitive Management Systems.

1.2 Cognitive Management Systems

Η θεμελιώδης ιδέα για τη χρήση των Cognitive Management Systems (CMS) βασίζεται στο ότι θα παρέχουν τα αναγκαία εργαλεία για να πραγματοποιηθεί μία στενή και αξιόλογη συνεργασία μεταξύ της υποδομής και του OppNet (Εικ 1.1). Αυτή η συνεργασία είναι κρίσιμη για να

διασφαλιστεί η βιωσιμότητα, η ανάπτυξη και η αξία του συστήματος για όλους τους ενδιαφερόμενους στον κλάδο (επενδυτές, μετόχους, παρόχους).

Τα CMS θα είναι υπεύθυνα για την απόφαση καταλληλότητας (suitability determination), δημιουργίας (creating), τροποποίησης (modifying) και αντιμετώπισης του βεβιασμένου τερματισμού (forced termination) των OppNets. Ταυτόχρονα τους παρέχεται η δυνατότητα προσαρμογής στις νέες περιβαλλοντικές συνθήκες και απαιτήσεις, με δυναμική αναδιαμόρφωση και αυτοδιαχείριση [3].

Όσον αφορά την απόφαση καταλληλότητας ενός OppNet, το CMS αναμένεται να υποδεικνύει τις δυνατότητες του φάσματος και να λαμβάνει την τελική απόφαση για το αν είναι πραγματοποιήσιμη αλλά και επικερδής η προσπάθεια εγκατάστασής του. Επίσης θα παρέχει τις πολιτικές, τους πόρους αλλά και το σύνολο όλων των πληροφοριών του συστήματος που θα ελέγχουν το δίκτυο. Ακόμα, θα είναι υπεύθυνο για τη δημιουργία και την παρακολούθηση του δικτύου, η δομή του οποίου θα είναι σύμφωνη με το πλαίσιο των κανόνων που προβλέπεται από αυτό. Τέλος, θα λαμβάνει τις αποφάσεις και θα διαχειρίζεται τον τερματισμό του είτε είναι βεβιασμένος είτε φυσιολογικός.

Κεφάλαιο 2
Χαρακτηριστικά και Εφαρμογές των Opportunistic Networks

2.1 Απαλοιφή των προβλημάτων επικοινωνίας

Ένας από τους στόχους των OppNets, είναι να κινητοποιήσει τον πλούτο των διάχυτων πόρων και δυνατοτήτων στα σύγχρονα δίκτυα. Αυτός είναι συχνά ένας θησαυρός που παραμένει αχρησιμοποίητος λόγω της έλλειψης επικοινωνίας μεταξύ των κόμβων. Διαφορετικές συσκευές και συστήματα είτε είναι ανίκανα να «μιλήσουν» το ένα στο άλλο, είτε δεν προσπαθούν καν να επικοινωνήσουν.

Αυτό συμβαίνει παρόλο που όλες αυτές οι συσκευές χαρακτηρίζονται από αυτονομία, ικανότητες αυτό-οργάνωσης, προσαρμοστικότητα στις μεταβαλλόμενες συνθήκες, ή ακόμη και αντιμετώπιση των προβλημάτων που προκαλούνται από βλάβες στο υλιστικό τους (hardware) ή και κακόβουλες επιθέσεις τρίτων. Είναι λοιπόν άξιο λόγου, ότι τέτοιες τόσο έξυπνες συσκευές δεν έχουν την ικανότητα να επικοινωνούν μεταξύ τους.

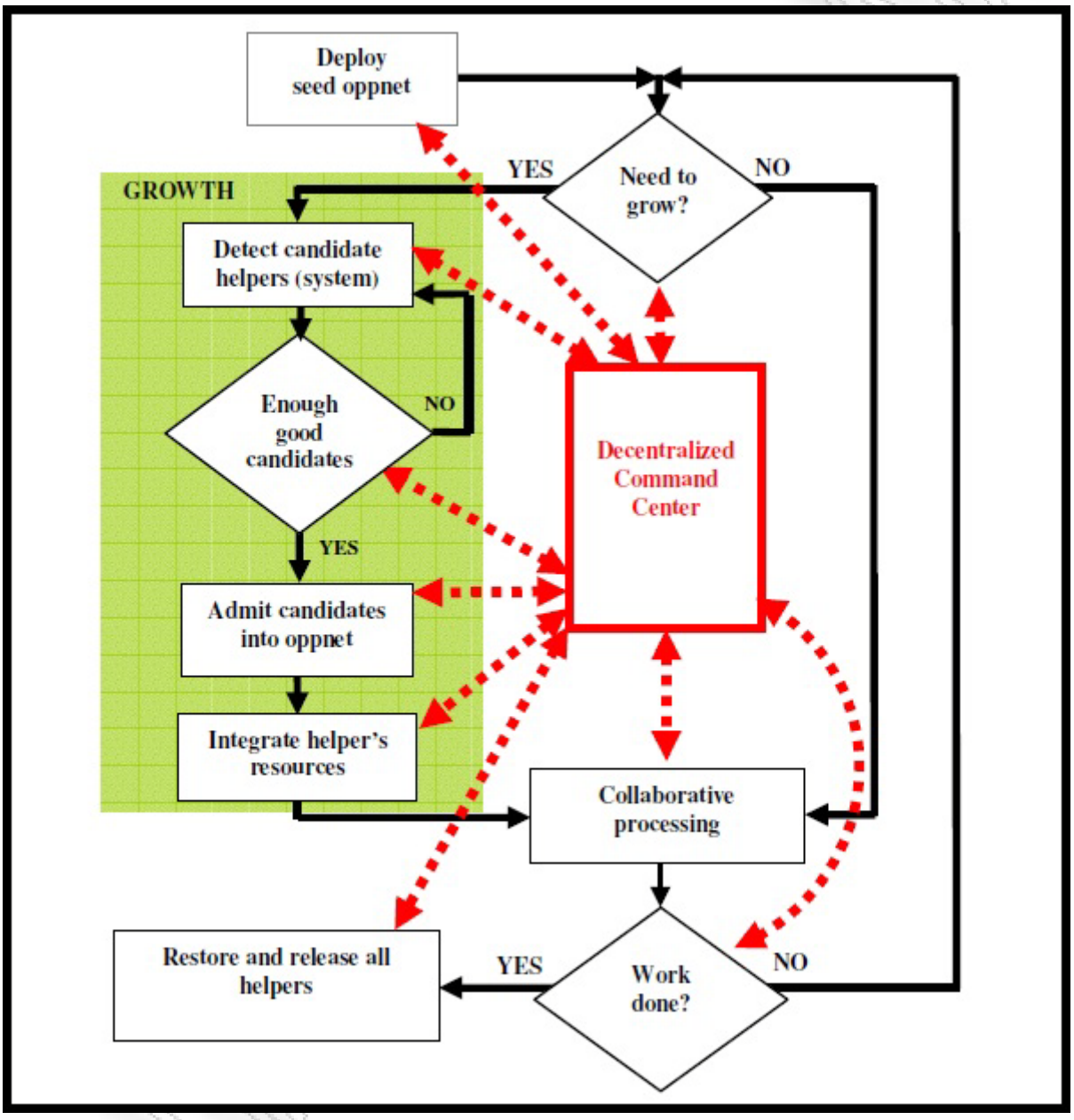
Οι λόγοι έλλειψης επικοινωνίας οφείλονται σε πολλαπλές και διαφορετικές αιτίες κάθε φορά, όπως χρήση διαφορετικών διαθέσιμων τεχνολογιών ασύρματης πρόσβασης (RATs), συμφόρηση λόγω υψηλής κυκλοφοριακής κίνησης, κόμβοι που ανήκουν σε διαφορετικά δίκτυα ή παρόχους.

Στα OppNets, οι ασύρματοι κόμβοι είναι ικανοί να επικοινωνούν ο ένας με τον άλλον ακόμα και αν δεν υπήρξε πότε ένα δρομολόγιο το οποίο να τους συνέδεε απευθείας. Όσον αφορά τους κόμβους, δεν καθίσταται αναγκαίο να κατέχουν ή να αποκτούν κάποια πληροφορία για την τοπολογία του δικτύου. Τα μονοπάτια χτίζονται δυναμικά, καθώς τα πακέτα δρομολογούνται μεταξύ του αποστολέα και του παραλήπτη, ενώ κάθε πιθανός κόμβος είναι δυνατόν να χρησιμοποιηθεί καιροσκοπικά ως επόμενος-κόμβος (next-hop), δεδομένου ότι το πακέτο πλησιάζει όλο και πιο κοντά στο τελικό.

2.2 Η δομή του OppNet

Κάθε δίκτυο OppNet στην αρχική του μορφή, ξεκινά να αναπτύσσεται από ένα seed το οποίο αποτελείται από κόμβους που εδραιώνονται κατά την αρχική εγκατάστασή του. Το seed αυτό, είναι ένα προσχεδιασμένο δίκτυο και στη χειρότερη περίπτωση μπορεί να αποτελείται από ένα μοναδικό κόμβο [4].

Στη συνέχεια το seed επεκτείνεται σε μεγαλύτερο δίκτυο (expanded network) (Εικ. 2.1) μέσω προσκλήσεων για την ένταξη ξένων συσκευών, ομαδοποιημένων κόμβων (clusters) ή οποιουδήποτε άλλου συστήματος ικανού να επικοινωνήσει μαζί του [5]. Αξίζει να αναφερθεί ότι ένας απλός κόμβος πριν εισέλθει στην οικογένεια του OppNet, υποχρεούται πρώτα να γνωστοποιήσει τη διαθεσιμότητά του ως υποψήφιος κόμβος προς είσοδο. Επίσης οι κόμβοι είναι εκ των προτέρων άγνωστοι μεταξύ τους.



Εικ. 2.1 Επέκταση OppNet με την ένταξη νέων βοηθών.

Κάθε νέος κόμβος που εισέρχεται στο δίκτυο, μόλις γίνει ολοκληρωμένο μέλος του, ονομάζεται βοηθός (Helper) και έχει το δικαίωμα να καλεί νέους προς ένταξη. Προσκαλώντας ελεύθερα λοιπόν τέτοιες συνεργάσιμες συσκευές/τερματικά, το OppNet έχει και τη δυνατότητα να παρουσιαστεί ως ένα αρκετά ανταγωνίσιμο δίκτυο στον οικονομικό τομέα. Τα θέματα που κρίνεται αναγκαίο να διευθετηθούν, αφορούν αφενός τα κατάλληλα κίνητρα έτσι ώστε οι

προσκεκλημένοι κόμβοι να είναι πρόθυμοι να ενταχθούν και αφετέρου τους, ενδεχομένως χαμηλής αξιοπιστίας, κόμβους. Οι βοηθοί συνεργάζονται για την επίτευξη των στόχων του OppNet, με αποτέλεσμα να μπορούν να χρησιμοποιηθούν για κάθε είδους εργασία, παρόλο που δεν έχουν σχεδιαστεί για να λειτουργήσουν σε ένα τέτοιο δίκτυο το οποίο τους προσκάλεσε προσωρινά για υποστήριξη.

Το σύνολο των βοηθών περιλαμβάνει και οντότητες που δε θεωρούνται συνήθως δικτυακοί κόμβοι, τόσο ενσύρματους όσο και ασύρματους. Ακόμα και κόμβοι που δεν έχουν τη δυνατότητα ανίχνευσης, όπως ασύρματες συσκευές σε αυτοκίνητα, μπορούν να συνεισφέρουν σημαντικά στο επικοινωνιακό έργο ενός OppNet. Έτσι και αλλιώς, κάθε συσκευή ή επεξεργαστής περιορισμένης λειτουργικότητας που είναι συνδεδεμένος σε κάποιο δίκτυο, διαθέτει ορισμένα χαρακτηριστικά επεξεργασίας και επικοινωνίας. Για παράδειγμα, πληροφορίες που αφορούν τις διαδικτυακές συνήθειες ή την τοποθεσία ενός χρήστη, μπορούν να καταγράφονται στον Η/Υ ή στο κινητό του αντίστοιχα. Τέτοιοι κόμβοι μπορούν να δεχθούν προσκλήσεις υποψηφιότητας βοηθού στο OppNet. Αυτό μπορεί να πραγματοποιηθεί επιτυχώς, μόλις το seed αναγνωρίσει ένα σύνολο IP διευθύνσεων που επισημαίνονται από ενδιαφέροντα για αυτό στοιχεία, σε μία συγκεκριμένη περιοχή που έχει ζήτηση.

Το δίκτυο σταματά να προσκαλεί νέους κόμβους όταν αποκτά τόσους βοηθούς ώστε να παρέχονται επαρκείς δυνατότητες επεξεργασίας και επικοινωνίας. Βασικό στοιχείο λοιπόν, αποτελεί η αποφυγή στρατολόγησης κόμβων οι οποίοι όχι μόνο δε θα συνεισφέρουν, αλλά πιθανόν θα μειώνουν την απόδοσή του δικτύου, χρησιμοποιώντας άσκοπα πόρους.

2.3 Αποδοτική επιλογή και τύποι κόμβων

Στην ενότητα 2.2, παρατέθηκε η δομή του δικτύου OppNet αποτελούμενη αρχικά από ένα seed δίκτυο λιγοστών κόμβων, μέχρι την τελική επέκτασή του στο ολοκληρωμένο πια δίκτυο. Η επέκταση αυτή πραγματοποιήθηκε βήμα-βήμα μέσω των βοηθών του, αφού πρώτα εγκρίθηκαν ως μέλη του. Όμως ποιά είναι τα κριτήρια αυτά που θα κρίνουν τον κάθε κόμβο ως επαρκή και αναγκαίο για τη συμμετοχή του στο δίκτυο και μετά την επιλογή τους, ποιος ο ρόλος που τους αποδίδεται σε αυτό; Αυτά τα ερωτήματα, καθώς επίσης και η διαδικασία που τελικά καθορίζει τον υποψήφιο βοηθό ως μελλοντικό στοιχείο του OppNet, αναλύονται στην παρούσα ενότητα.

Προκειμένου να αποκτήσουν επίγνωση της κατάστασης ενός υποψήφιου κόμβου για την είσοδό του στο δίκτυο, παρακολουθούνται συγκεκριμένα χαρακτηριστικά. Αυτά περιλαμβάνουν το επίπεδο ενέργειας (energy level), το επίπεδο διαθεσιμότητας (availability level) και την πιθανότητα παράδοσης (delivery probability).

Τα παραπάνω στοιχεία, παρέχουν την είσοδο σε μία συνάρτηση καταλληλότητας (fitness function) η οποία σύμφωνα με ένα προκαθορισμένο κατώφλι (threshold), αποφασίζει για την αποδοχή ή την απόρριψη ενός υποψηφίου στο OppNet. Ο ορισμός του κατάλληλου τύπου της συνάρτησης είναι υψίστης σημασίας αφού αυτός θα αποτελεί και το μοναδικό κριτήριο ένταξης των κόμβων.

Η παρακάτω σχέση παρουσιάζει τη συνάρτηση καταλληλότητας όπως χρησιμοποιείται σε αυτή την εργασία:

$$\text{Fitness Function} = f(e_i, d_i, a_i) \quad [6]$$

Όπου το e_i συμβολίζει το επίπεδο ενέργειας του i-στού κόμβου σε ένα ακριβές χρονικό σημείο,

το d_i συμβολίζει την πιθανότητα παράδοσης των πακέτων του i-στού κόμβου σε ένα ακριβές χρονικό σημείο,

και το a_i συμβολίζει το επίπεδο διαθεσιμότητας του i-στού κόμβου σε ένα ακριβές χρονικό σημείο.

Η ενέργεια των υποψήφιων κόμβων είναι ένα χαρακτηριστικό που πρέπει να λαμβάνεται σοβαρά υπόψη, διότι οι κόμβοι, όντας ασύρματες συσκευές, είναι πιθανό να έχουν περιορισμένη διάρκεια ζωής σε επίπεδο μπαταρίας ή πολύ απλά μην είναι ιδιαίτερα φορτισμένοι τη συγκεκριμένη χρονική στιγμή. Αυτό έχει ως αποτέλεσμα, να μην επιτρέπεται η είσοδός τους στο δίκτυο και να απορρίπτονται.

Το δεύτερο χαρακτηριστικό που μελετάται, είναι η πιθανότητα παράδοσης ενός κόμβου. Η τιμή της υπολογίζεται από τον λόγο: πλήθος πακέτων που επιτυχώς παραδόθηκαν στον προορισμό τους, προς πλήθος πακέτων που συνολικά στάλθηκαν στη διάρκεια ενός χρονικού διαστήματος.

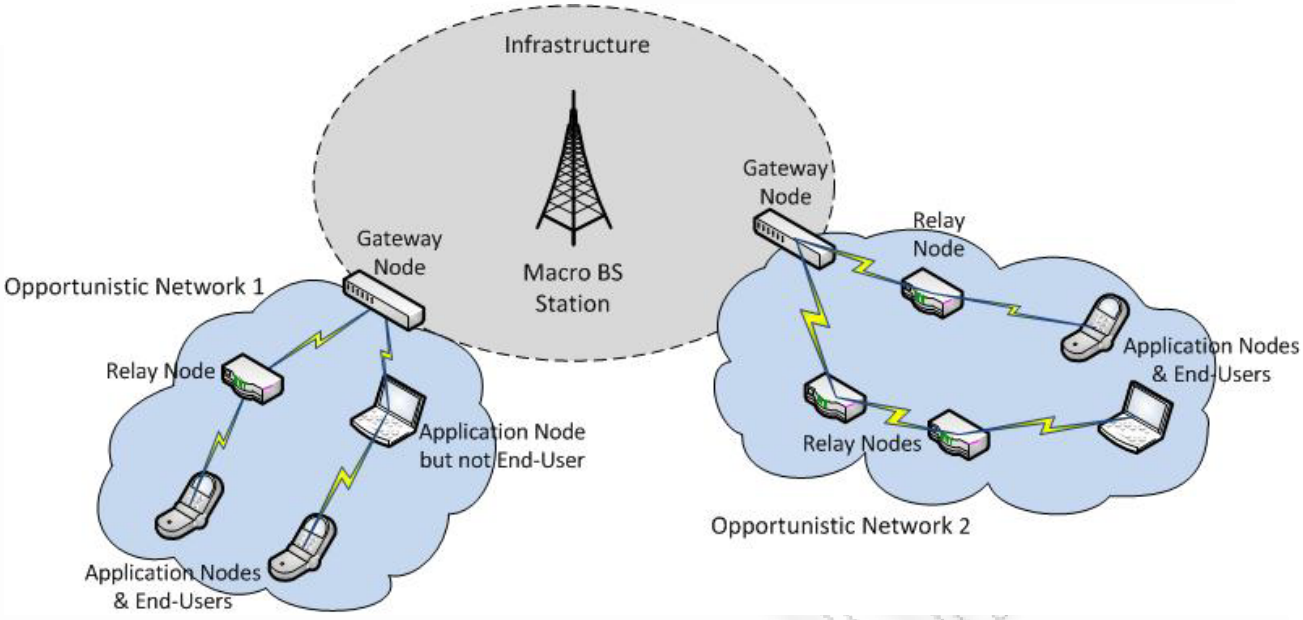
Επιπρόσθετα, γεγονός αποτελεί ότι πολλές φορές τα στοιχεία του δικτύου δεν είναι διαθέσιμα για επικοινωνία, λόγω της συνεχούς κινητικότητάς τους. Αυτό σημαίνει ότι είναι απαραίτητο να ληφθεί υπόψη η μεταβλητή της διαθεσιμότητας η οποία εξαρτάται από ένα δεύτερο σύνολο μεταβλητών όπως είναι το διαθέσιμο φάσμα σε μία γεωγραφική περιοχή, οι διεπαφές (interfaces) και η ποιότητα των συνδέσεων (quality of links).

Συμπερασματικά, είναι εμφανές ότι η ύπαρξη της συνάρτησης καταλληλότητας είναι πάρα πολύ σημαντική, αφού χάρη σε αυτή καθορίζεται η είσοδος κάθε πιθανού βοηθού στο OppNet. Όμως όλη αυτή η διαδικασία είναι μόνο η αρχή στο ρόλο που διαδραματίζουν οι επιλεγμένοι βοηθοί μέσα στο δίκτυο, ρόλοι οι οποίοι κατηγοριοποιούνται μεταξύ τριών βασικών τύπων: κόμβοι πύλης (gateway nodes), κόμβοι αναμετάδοσης (relay nodes) και κόμβοι εφαρμογών (application nodes) (Εικ. 2.2).

Κόμβοι πύλης: είναι κόμβοι οι οποίοι εκτός από το OppNet ανήκουν και στη βασική υποδομή του δικτύου (infrastructure) αποτελώντας το μοναδικό δίαυλο επικοινωνίας μεταξύ τους. Μετά το τέλος της λειτουργίας του, παραμένουν ενεργά μέλη της υποδομής.

Κόμβοι αναμετάδοσης: αποτελούν εκείνους τους κόμβους του OppNet οι οποίοι χρησιμοποιούνται ως δρομολογητές για την προώθηση των πακέτων.

Κόμβοι εφαρμογών: είναι οι κόμβοι που χρησιμοποιούν μία ή και περισσότερες εφαρμογές υπηρεσιών. Σε αυτούς περιλαμβάνονται και οι τελικοί χρήστες (end-users).



Εικ. 2.2 Βασικοί τύποι κόμβων δικτύου OppNet και ο ρόλος τους.

2.4 Εφαρμογή των OppNets

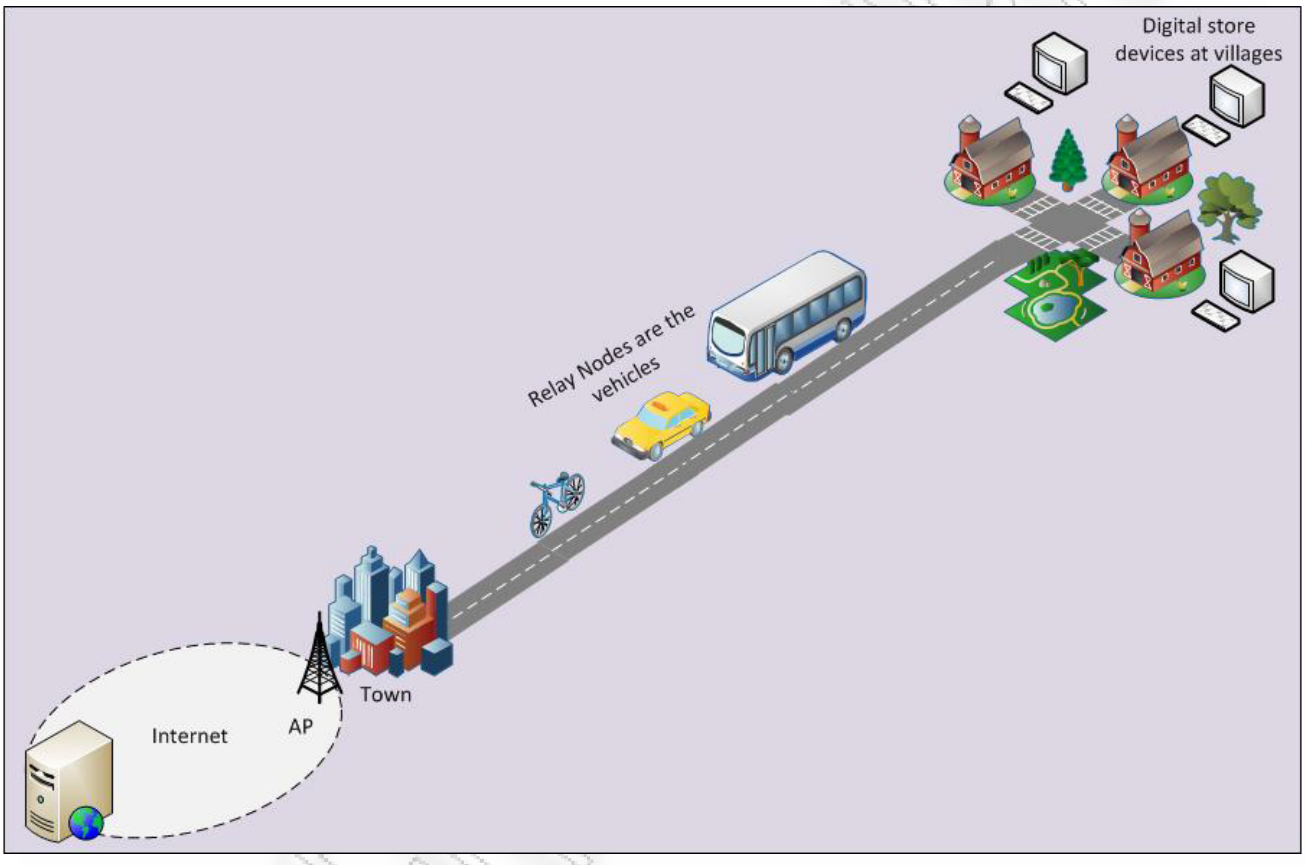
Αν κάνουμε μία ανασκόπηση περί των πτυχών που καλύπτει η εφαρμογή των OppNets παρατηρούμε ότι κυρίως αφορά τους στόχους του Μελλοντικού Διαδικτύου, όπως καλύτερη απόδοση των πόρων, μείωση του φορτίου μεταγωγής κατά την ώρα αιχμής (high peak), εξυπηρέτηση τερματικών εκτός εμβέλειας της βασικής υποδομής δικτύου, αποσυμφόρηση και υψηλή Ποιότητα Υπηρεσιών (QoS). Όμως το πλαίσιο της εφαρμογής τους δεν περιορίζεται μονάχα σε αυτά, αλλά περιλαμβάνει ήδη και άλλα σπουδαία επιτεύγματα όπως την εξυπηρέτηση αναπτυσσόμενων περιοχών και την παρακολούθηση της άγριας φύσης [7].

2.4.1 OppNets για τις αναπτυσσόμενες περιοχές

Τα OppNets έχουν τη δυνατότητα να παρέχουν διακοπτόμενη σύνδεση στο διαδίκτυο σε αγροτικές και αναπτυσσόμενες περιοχές, αποτελώντας συνήθως τον οικονομικότερο τρόπο για τη γεφύρωση του ψηφιακού χάσματος. Ένα τέτοιο παράδειγμα είναι το DakNet Project, το οποίο στοχεύει στην υλοποίηση ενός ασύγχρονου δικτύου υποδομής με πολύ χαμηλό κόστος για την παροχή συνδεσιμότητας σε αγροτικές περιοχές της Ινδίας, εκεί όπου δεν είναι κοστολογικά εφικτό να αναπτυχθεί κάποιου άλλου είδους πρόσβαση στο διαδίκτυο (π.χ. ένα Base Station).

Σύμφωνα με το σχέδιο DakNet [8] (Εικ. 2.3), τοποθετούνται κιόσκια σε διάφορα χωριά τις Ινδίας και εξοπλίζονται με συσκευές ψηφιακής αποθήκευσης μικρής εμβέλειας, για ασύρματη επικοινωνία. Ασύρματα Σημεία Πρόσβασης (Mobile Access Points, δηλαδή MAPS) εγκαθίστανται

σε λεωφορεία, μοτοσυκλέτες, αυτοκίνητα ή ακόμη και ποδήλατα που διασχίζουν τα χωριά και ανταλλάσσουν δεδομένα ασύρματα. Τα MAPs έχουν τη δυνατότητα να ανεβάζουν και ύστερα να κατεβάζουν τα αιτούμενα δεδομένα από τα κιόσκια. Ακολούθως, περνώντας από τα μεγάλα σημεία πρόσβασης (Access Points) της πόλης, τα ανεβάζουν στο διαδίκτυο και έπειτα μπορούν να κατεβάσουν τις ζητούμενες πληροφορίες επιστρέφοντάς τες τελικώς στα χωριά, ολοκληρώνοντας το κύκλο. Ο ρόλος λοιπόν των MAPs μέσα στο OppNet είναι κομβικός, αφού αποτελούν τους μοναδικούς κόμβους αναμετάδοσης του δικτύου, χάρη στους οποίους τα πακέτα δρομολογούνται από τα χωριά στο εσωτερικό της εμβέλειας πυλών, τοποθετημένων στην πόλη.



Εικ. 2.3 Τοπολογία δικτύου Project DakNet.

Μία ακόμη εφαρμογή των OppNets που εξυπηρετεί τις αναπτυσσόμενες περιοχές αποτελεί το Saami Network Connectivity (SNC) project [9], το οποίο αποσκοπεί στη διαδικτυακή επικοινωνία του πληθυσμού των Saamis, βοσκών των ταράνδων, οι οποίοι κατοικούν σε απομακρυσμένες περιοχές της Σουηδικής Λαπωνίας και επανατοποθετούν τη βάση τους σύμφωνα με τον ετήσιο κύκλο που υπαγορεύεται από τη φυσική συμπεριφορά των ταράνδων. Οι βοσκοί μέχρι πρότινος δεν είχαν αξιόπιστες ενσύρματες, ασύρματες ή δορυφορικές δυνατότητες επικοινωνίας σε βασικές περιοχές μέσα στις οποίες εργάζονταν και ζούσαν. Η παροχή συνδεσιμότητας δικτύου προστατεύει και υπερασπίζει τις συνήθειές τους, τον πολιτισμό, τις παραδόσεις ενώ παράλληλα στηρίζει την ένταξή τους στη σύγχρονη κοινωνία της χώρας τους.

2.4.2 OppNets για την παρακολούθηση της άγριας φύσης

Η παρακολούθηση της άγριας φύσης είναι ένα ενδιαφέρον πεδίο εφαρμογής των OppNets. Κυρίως, επικεντρώνεται στην παρακολούθηση άγριων ειδών για να ερευνηθεί σε βάθος η συμπεριφορά τους και να κατανοηθούν οι αλληλεπιδράσεις και οι επιρροές που ασκούνται μεταξύ τους, καθώς και η αντίδρασή τους στις κλιματολογικές αλλαγές που προκαλούν οι ανθρώπινες δραστηριότητες. Οι ερευνητές λοιπόν, χρησιμοποιούν τα OppNets ως μία αξιόπιστη, οικονομικά αποδοτική, και αδιάκριτη λύση κατά την παρακολούθηση της μετακίνησης μεγάλων πληθυσμών σε αχανείς πεδιάδες.

Τα συστήματα παρακολούθησης περιλαμβάνουν ειδικές ετικέτες αναγνώρισης (tags) με αισθητήρες. Αυτές τοποθετούνται στα ζώα υπό μελέτη και έπειτα τα δεδομένα που συλλέγονται αποστέλλονται σε έναν ή περισσότερους σταθμούς βάσης (base station).Οι σταθμοί με τη σειρά τους τα δρομολογούν προς το κέντρο επεξεργασίας. Επίσης, έχει οριστεί ένα δικτυακό πρωτόκολλο για να φιλτράρεται η ροή των πληροφοριών προς τους σταθμούς. Οι σταθμοί βάσης μπορεί να είναι είτε σταθεροί είτε κινητοί, και στις δύο περιπτώσεις όμως η συλλογή των δεδομένων από όλες τις ετικέτες των ζώων είναι πάντα μία μεγάλη πρόκληση. Ως εκ τούτου, είναι γενικά χρήσιμο κατά τη συνεύρεση των ζώων στην ίδια περιοχή, να επιτρέπεται η ανταλλαγή πληροφοριών που έχουν ήδη συλλεχθεί μεταξύ τους. Κατά συνέπεια, κάθε ζώο κατέχει τις δικές του πληροφορίες, καθώς και δεδομένα όλων των υπόλοιπων άγριων ζώων που συνάντησε στη διάρκεια της περιπλάνησής του, στις τεράστιες περιοχές του οικοσυστήματος.

Το ZebraNet [10] είναι ένα διεπιστημονικό project που ξεκίνησε από το πανεπιστήμιο του Princeton και ο στόχος του ήταν η ανάπτυξη ενός μηχανισμού για την παρακολούθηση της ζέβρας στις μεγάλες εκτάσεις σαβάνας της κεντρικής Κένυας. Οι ζέβρες εντοπίζονται χάρη σε ένα ειδικό κολάρο που φορούν, ενώ ο σταθμός βάσης προς επικοινωνία είναι τοποθετημένος μέσα σε ένα κινητό όχημα. Το όχημα αυτό αποτελείται από διάφορους ερευνητές και μετακινείται ανά περιόδους στην περιοχή της σαβάνας, μαζεύοντας δεδομένα από τα κολάρα. Υπάρχουν δύο εναλλακτικά πρωτόκολλα για τη συλλογή δεδομένων στο ZebraNet. Το πρώτο θυμίζει πρωτόκολλο πλημμύρας (flooding), αφού κάθε κολάρο στέλνει όλα τα δεδομένα του σε κάθε γειτονικό που συναντά μέχρις ότου αυτά να φτάσουν στο σταθμό βάσης. Το δεύτερο ονομάζεται πρωτόκολλο ιστορικού (history-based protocol, HBP) και επιλέγει μόνο έναν κόμβο για να μεταφέρει τα δεδομένα, λαμβάνοντας υπόψη την υψηλότερη πιθανότητα για να καταλήξει το πακέτο στο σταθμό βάσης. Αποτελέσματα προσομοίωσης δείχνουν ότι και τα δύο πρωτόκολλα υπερτερούν σε σύγκριση με το άμεσο (direct protocol), στο οποίο κάθε κολάρο πρέπει να επικοινωνήσει απευθείας με το σταθμό για τη μεταφορά. Επιπρόσθετα, το HBP υπερτερεί του flooding όσον αφορά την κατανάλωση ενέργειας και το εύρος ζώνης. Το ZebraNet έχει τεθεί σε εφαρμογή στο Ερευνητικό Κέντρο της Κένυας και είναι υπό δοκιμή. Τα πρώτα αποτελέσματα από την πραγματική χρήση του είναι ήδη διαθέσιμα και έχουν πρόσφατα χρησιμοποιηθεί για να καθορίσουν το μοντέλο κινητικότητας που χρησιμοποιείται σε ορισμένες τεχνικές προώθησης πακέτων.

2.5 Σύγκριση Opportunistic Network – Mobile Ad Hoc Network – Delay Tolerant Network

Στο σημείο αυτό, θα ήταν διαφωτιστικό να πραγματοποιηθεί μία σύγκριση των δικτύων OppNet με τα MANets (Mobile Ad Hoc Networks) και τα DTNs (Delay Tolerant Networks), ώστε να γίνει ευκολότερα κατανοητό σε ποιά χαρακτηριστικά διαφέρουν και σε ποιά παρουσιάζουν ομοιότητες. Στον πίνακα που ακολουθεί, αναρτώνται ορισμένα από αυτά.

Χαρακτηριστικό	**OppNet**	**MANet**	**DTN**
Ελεγχόμενο από τον πάροχο.	Ναι	Όχι	Όχι
Οι κόμβοι έχουν επίγνωση της τοπολογίας.	Όχι	Ναι	Ναι
Χρήση Store & Forward Protocol.	Ναι	Όχι	Ναι
Επικοινωνία μεταξύ ετερογενών δικτύων.	Ναι	Όχι	Ναι

Πίνακας σύγκρισης ασύρματων δικτύων

Διαπιστώνεται λοιπόν ότι τα OppNets διαφέρουν από τα MANets σε όλους τους τομείς που έχουν τεθεί υπό σύγκριση στον πίνακα, ενώ παρουσιάζουν ομοιότητες στους μισούς σε σχέση με τα DTNs.

Ο έλεγχος του δικτύου από τα συστήματα των παρόχων, είναι μοναδικό στοιχείο του OppNet ενώ οι κόμβοι του δεν έχουν εκ των προτέρων επίγνωση της τοπολογίας του δικτύου, παρά μόνο για τους γειτονικούς κόμβους κατά την προώθηση πακέτων. Βέβαια, επειδή τα OppNets είναι δίκτυα υπό μελέτη και εξελίσσονται με την πάροδο του χρόνου, η επίγνωση της τοπολογίας για τους κόμβους είναι ένα χαρακτηριστικό το οποίο μπορεί να αλλάξει μελλοντικά.

Ακόμη, τα MANets κάνουν χρήση των πρωτοκόλλων AODV (Ad-Hoc on-Demand Distance Vector) και DSR (Dynamic Source Rooting) τα οποία πρώτα εγκαθιστούν την ολοκληρωμένη δρομολόγηση του πακέτου και ύστερα το προωθούν. Αντιθέτως, στα υπόλοιπα δύο δίκτυα, συναντάμε τη χρήση ενός εντελώς διαφορετικού τύπου πρωτοκόλλου με την ιδιότητα του S&F (Store & Forward). Αυτός ο τύπος πρωτοκόλλων λειτουργεί μεταφέροντας σταδιακά τα δεδομένα και αποθηκεύοντάς τα σε όλο το δίκτυο με την ελπίδα ότι θα φτάσουν τελικά στον προορισμό τους.

Τέλος, η ασύρματη επικοινωνία μεταξύ δύο ή και περισσότερων ετερογενών δικτύων που χρησιμοποιούν διαφορετικές τεχνολογίες επικοινωνίας και βασίζονται στη δική τους στοίβα

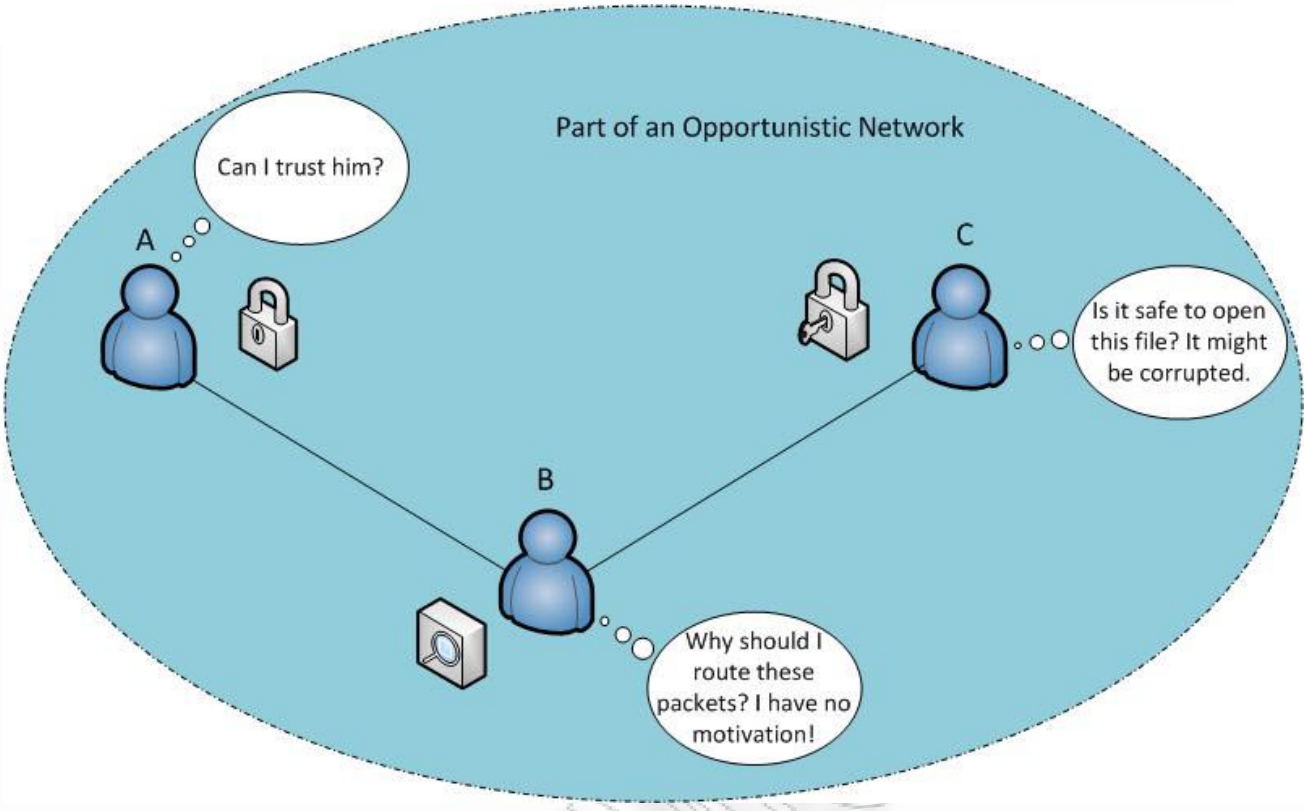
πρωτοκόλλων που ταιριάζει καλύτερα στην υποδομή τους [11], εξυπηρετείται μόνο από τα OppNets και τα DTNs.

2.6 Θέματα διασφάλισης της ιδιωτικότητας και παροχής κινήτρων

Το γεγονός ότι ένας χρήστης εντάσσει μία προσωπική του συσκευή ως κόμβο ενός OppNet και αλληλεπιδρά με άλλους χρήστες άγνωστους σε αυτόν, αποτελεί την εκκίνηση μίας νέας συνεργασίας. Η συνεργασία αυτή εγείρει ερωτηματικά σχετικά με δύο σημαντικές ανθρώπινες πτυχές: τη διασφάλιση της ιδιωτικότητάς και τα κίνητρα που τον παρακινούν να συμμετέχει σε αυτή [12]. Είναι σημαντικό να σημειωθεί ότι η ιδιωτικότητά του χρήστη πιθανόν βρίσκεται σε κίνδυνο, καθώς η είσοδός του στο δίκτυο μπορεί να εκθέσει προσωπικές πληροφορίες. Επιπλέον, καταλυτικό παράγοντα για την ενθάρρυνση της συμμετοχής των υποψήφιων συσκευών στο δίκτυο, αποτελεί η παροχή κινήτρων. Και τα δύο αυτά θέματα πρέπει να ληφθούν σοβαρά υπόψη, προκειμένου να αυξηθεί η πρόσβαση των χρηστών σε αυτά.

Για παράδειγμα, θεωρούμε ένα σημείο δικτύου (Εικ. 2.4) στο οποίο έχουν ενταχθεί τρεις κόμβοι (A,B και C), όλοι άγνωστοι μεταξύ τους. Ο κόμβος A θέλει να ανταλλάξει δεδομένα με τον κόμβο C, όμως ο A βρίσκεται στην εμβέλεια επικοινωνίας του B και όχι του C, ενώ ο B βρίσκεται στην εμβέλεια επικοινωνίας του C. Αν λοιπόν ο κόμβος A επιθυμεί να επικοινωνήσει με τον C, είναι επιτακτική ανάγκη να το κάνει αποκλειστικά μέσω του B. Όμως η ανωνυμία που τους διακατέχει επιφέρει και ένα σημαντικό αντίκτυπο στην επικοινωνία τους αφού τίθενται δύο καίρια ερωτήματα:

1. Ποιό είναι το κίνητρο του κόμβου B να δρομολογήσει τα πακέτα μεταξύ των A και C, εφόσον για να το διεκπεραιώσει θα χρειαστεί να καταναλώσει μέρος της ενέργειάς του;
2. Γιατί οι κόμβοι A και C να εμπιστευτούν την επικοινωνία τους στον κόμβο B, αφού είναι πιθανό εκείνος να «κρυφακούσει», χειραγωγήσει ή ακόμα και να απορρίψει τα μηνύματά τους;



Εικ. 2.4 Επιπλοκές και προβλήματα στην επικοινωνία των κόμβων.

Για αυτό, το μοντέλο του OppNet θα μπορούσε να υποστηρίξει έναν ασύρματο τύπο one-hop επικοινωνίας στον οποίο να επιτρέπεται μόνο στους απευθείας συνδεδεμένους κόμβους να ανταλλάσουν μηνύματα. Οι απευθείας συνδεδεμένοι κόμβοι διαθέτουν μεγαλύτερο κίνητρο να συμμετέχουν στο δίκτυο, καθώς έχουν τη δυνατότητα να ικανοποιούν τις δικές τους επιθυμίες στη διαχείριση των πληροφοριών. Επίσης, οι τεχνικές προστασίας της ιδιωτικότητας είναι εφαρμόσιμες, λόγω του one-hop τύπου επικοινωνίας.

Επιπρόσθετα το δίκτυο θα είναι σημαντικό να παρέχει τη δυνατότητα έκφρασης προσωπικών ενδιαφερόντων σε συγκεκριμένους τύπους πληροφοριών. Έτσι οι πληροφορίες που θα διαμοιράζονται μέσα σε μία εφαρμογή είναι αναγκαίο να κατηγοριοποιηθούν καταλλήλως, για την εύκολη αντιστοίχιση των προσωπικών ενδιαφερόντων. Η εγκυρότητα των πληροφοριών θα πρέπει να περιορίζεται με το χρόνο και την τοποθεσία.

Συμπερασματικά, η διασφάλιση της ιδιωτικότητας και η παροχή κινήτρων για την ένταξη σε ένα OppNet αποτελούν δύο σοβαρά ζητήματα που είναι ανάγκη να διευθετηθούν στο άμεσο μέλλον για τη βιωσιμότητα και την ακεραιότητα του δικτύου. Είναι πιθανό να παρουσιάζονται πάντα νέες ανάγκες και προβλήματα στο μονοπάτι της αντιμετώπισής τους, παρόλα αυτά όμως υπάρχουν ορισμένες ιδέες για την επίλυσή τους όπως η ανωνυμία, η ψευδωνυμία, η αυθεντικοποίηση, η απευθείας σύνδεση, η επιλεκτική διασπορά δεδομένων βασισμένη σε μοντέλα, ακόμη και η επιβράβευση των συμμετεχόντων που βοήθησαν στο διαμοιρασμό της πληροφορίας.

Κεφάλαιο 3
Φάσεις Λειτουργικότητας και Μοντελοποίηση Αλγορίθμου

3.1 Οι φάσεις λειτουργικότητας των OppNets

Βάση όσων έχουν λεχθεί ως τώρα σχετικά με τα OppNets, θα μπορούσαμε εν συντομία να πούμε, ότι σκοπός τους είναι η υποστήριξη της βασικής υποδομής του δικτύου για ποικίλους λόγους. Είναι σημαντικό λοιπόν ο πάροχος ενός τέτοιου δικτύου, να έχει την ευχέρεια να αποφασίσει αν και πότε ένα OppNet είναι απαραίτητο να εγκατασταθεί. Έπειτα η ανάγκη εγκατάστασης οδηγεί στην υλοποίηση και δημιουργία του. Αυτό θα έχει ως αποτέλεσμα να χρειαστεί η παρακολούθησή του καθ' όλη τη διάρκεια λειτουργίας του. Στο τέλος η κατάλληλη απόφαση για τον τερματισμό του είναι κομβική, αφού θα πρέπει να συντονιστεί η διοχέτευση της κυκλοφορίας των εναπομεινάντων πακέτων προς νέα μονοπάτια.

Η διαδικασία που μόλις περιγράφηκε, αποτελεί περιληπτικά τον κύκλο γέννησης και θανάτου ενός δικτύου OppNet [13]. Αυτές είναι και οι τέσσερις φάσεις που θα μας απασχολήσουν σε αυτό το κεφάλαιο δίνοντας ιδιαίτερη έμφαση στην τελευταία.

3.1.1 Πρώτη Φάση - Προσδιορισμός καταλληλότητας

Στη φάση αυτή, αρχικά εντοπίζονται και συλλέγονται ορισμένα χαρακτηριστικά του περιβάλλοντος δικτύου και των υποψήφιων κόμβων με σκοπό να τεθούν υπό μελέτη. Κάποια από αυτά τα στοιχεία αφορούν κυρίως τις πολιτικές του, τις δυνατότητες των κόμβων και το επίπεδο κινητικότητάς τους, τα προφίλ των χρηστών ή και το διαθέσιμο φάσμα.

Όμως το πιο σημαντικό στοιχείο που μελετάται, είναι τα κέρδη που θα αποκομίσει η εγκατάσταση ενός τέτοιου δικτύου. Τα πιθανά κέρδη μπορεί να προέρχονται μέσω της παροχής υπηρεσιών με αξιόλογο QoS, την αποδοτική χρήση του φάσματος και τη χαμηλή ισχύ μεταφοράς. Αυτά οδηγούν με τη σειρά τους σε χαμηλή κατανάλωση ενέργειας για το σταθμό βάσης (BS) του παρόχου.

Το αποτέλεσμα αυτής της έρευνας δείχνει αν είναι κατάλληλο να στηθεί ή όχι ένα OppNet σε συγκεκριμένο χρόνο και τοπολογία. Η αξιολόγηση της καταλληλότητάς του αποτελεί μία πρώτη απόφαση για τη δημιουργία του και προκύπτει ως αποτέλεσμα μίας πρώιμης ανάλυσης.

3.1.2 Δεύτερη Φάση – Δημιουργία OppNet

Κατ' ακολουθίαν της απόφασης καταλληλότητας και στηριζόμενη στα δεδομένα που συλλέχθηκαν, έρχεται η φάση δημιουργίας. Αυτή επικεντρώνεται στην επιλογή βέλτιστων ασύρματων μονοπατιών σε επίπεδο φάσματος και ισχύος σύμφωνα πάντα με το υπάρχον σύστημα δρομολόγησης. Έτσι επιτυγχάνεται και διασφαλίζεται το βέλτιστο QoS για όλες τις γραμμές.

Ακόμη, εκτελεί τις απαιτούμενες διαδικασίες για να συνδέσει αποτελεσματικά τα μέλη του δικτύου και για να εξασφαλίσει όσο είναι δυνατό συνεχόμενη παροχή υπηρεσιών χωρίς απρόσμενες διακοπές. Επίσης είναι υπεύθυνη για το handover των κόμβων από την υποδομή στην οποία ανήκαν μέχρι πρότινος προς το OppNet όταν αυτό κριθεί αναγκαίο.

3.1.3 Τρίτη Φάση – Παρακολούθηση και συντήρηση OppNet

Το δίκτυο θα πρέπει να είναι δυναμικό καθ' όλη τη διάρκεια ζωής του. Για να επιτευχθεί αυτό, μόλις η δημιουργία του ολοκληρωθεί, είναι επιτακτική ανάγκη η παρακολούθηση και συντήρησή του.

Προκειμένου να διατηρηθεί η αποδοτική λειτουργικότητα του OppNet και να παρέχεται προσαρμοστικότητα στις αλλαγές των περιβαλλοντικών συνθηκών, η φάση αυτή είναι υπεύθυνη για την εφαρμογή των κατάλληλων ρυθμίσεων στη διαμόρφωσή (configuration) του.

Συνεπάγεται λοιπόν ότι κατά τη φάση της συντήρησης θα πρέπει να παρακολουθούνται οι κόμβοι, το φάσμα, οι πολιτικές αλλά και το QoS των συνδέσεων. Επίσης στο σημείο αυτό λαμβάνονται αποφάσεις για το αν είναι εφικτό, επιθυμητό και επικερδές να πραγματοποιηθεί μία συγχώνευση ή διάσπαση δικτύων. Απόρροια αυτών των γεγονότων είναι συνήθως η αναδιαμόρφωση του δικτύου (reconfiguration) με τα νέα δεδομένα.

3.1.4 Τέταρτη Φάση – Τερματισμός OppNet

Ο τερματισμός του OppNet ολοκληρώνει τον κύκλο ζωής του υποδηλώνοντας ότι δεν είναι πλέον απαραίτητη η συνεισφορά του στο έργο της βασικής υποδομής του δικτύου. Αυτό όμως δεν είναι πάντα αλήθεια. Θα ήταν πιθανό, ο τερματισμός του να οφείλεται σε άλλους λόγους και οι υπηρεσίες του να ήταν αναγκαίες ανεξάρτητα από τη λήξη λειτουργίας του.

Ένα δίκτυο OppNet λοιπόν, είναι επιθυμητό να τερματίσει ομαλά, αλλά υπάρχει πάντα το ενδεχόμενο του βεβιασμένου τερματισμού. Συνεπώς ο τερματισμός του θα μπορούσε να κατηγοριοποιηθεί αυθαίρετα σε ομαλό και βεβιασμένο. Ο πρώτος θα σήμαινε την ομαλή διεξαγωγή του δικτύου μέχρις ότου όλοι οι κόμβοι του να διακόψουν τις υπηρεσίες που χρησιμοποιούσαν. Σε αυτή την περίπτωση, οι πόροι που είχαν δεσμευτεί από το δίκτυο απελευθερώνονται. Αντίθετα, όταν τερματίζεται αναγκαστικά, παρουσιάζεται η ανάγκη μεταφοράς των ενεργών συνδέσεών του στην υποδομή του δικτύου. Με άλλα λόγια κατά το

βεβιασμένο τερματισμό του, που κυρίως οφείλεται στην έλλειψη πόρων, δεν αποδεσμεύονται μόνο οι πόροι του αλλά ξεκινά και η διαδικασία του Handover.

Στη δεύτερη περίπτωση λοιπόν, η διατήρηση της συνεχόμενης ροής δεδομένων προς τους τελικούς χρήστες είναι μία μεγάλη πρόκληση. Όμως ακόμα μεγαλύτερη είναι η εύρεση κατάλληλου μηχανισμού ο οποίος θα αποφασίζει τον τύπο τερματισμού βασιζόμενος σε συγκεκριμένες μετρήσεις.

3.2 Μοντελοποίηση αλγορίθμου κατά τον τερματισμό OppNet

Όπως έχουμε ήδη αναφέρει οι τέσσερις φάσεις που απαρτίζουν τον κύκλο ζωής του OppNet είναι εξίσου σημαντικές, καθώς κάθε μία, διαδραματίζει έναν ξεχωριστό ιδιαίτερο ρόλο. Όμως η φάση που θα μας απασχολήσει και θα αναλυθεί εκτενέστερα από εδώ και στο εξής αφορά μονάχα τον τερματισμό του δικτύου. Όπως προαναφέρθηκε, δεν υπάρχει μόνο ένας τύπος τερματισμού αλλά τουλάχιστον δύο.

Συνεπώς κατά τη λειτουργία του δικτύου, παρουσιάζεται η ανάγκη εύρεσης ενός αλγορίθμου [14] ο οποίος θα είναι υπεύθυνος για τον εντοπισμό ενός συμβάντος το οποίο θα επιφέρει τον αντίστοιχο τύπο τερματισμού. Ανάλογα λοιπόν με το συμβάν που θα έχει προηγηθεί, ο αλγόριθμος αυτός, θα οδηγεί μέσω συγκεκριμένων διαδικασιών και αποφάσεων το OppNet, σε απελευθέρωση των πόρων ή και σε Handover αν αυτό κρίνεται αναγκαίο.

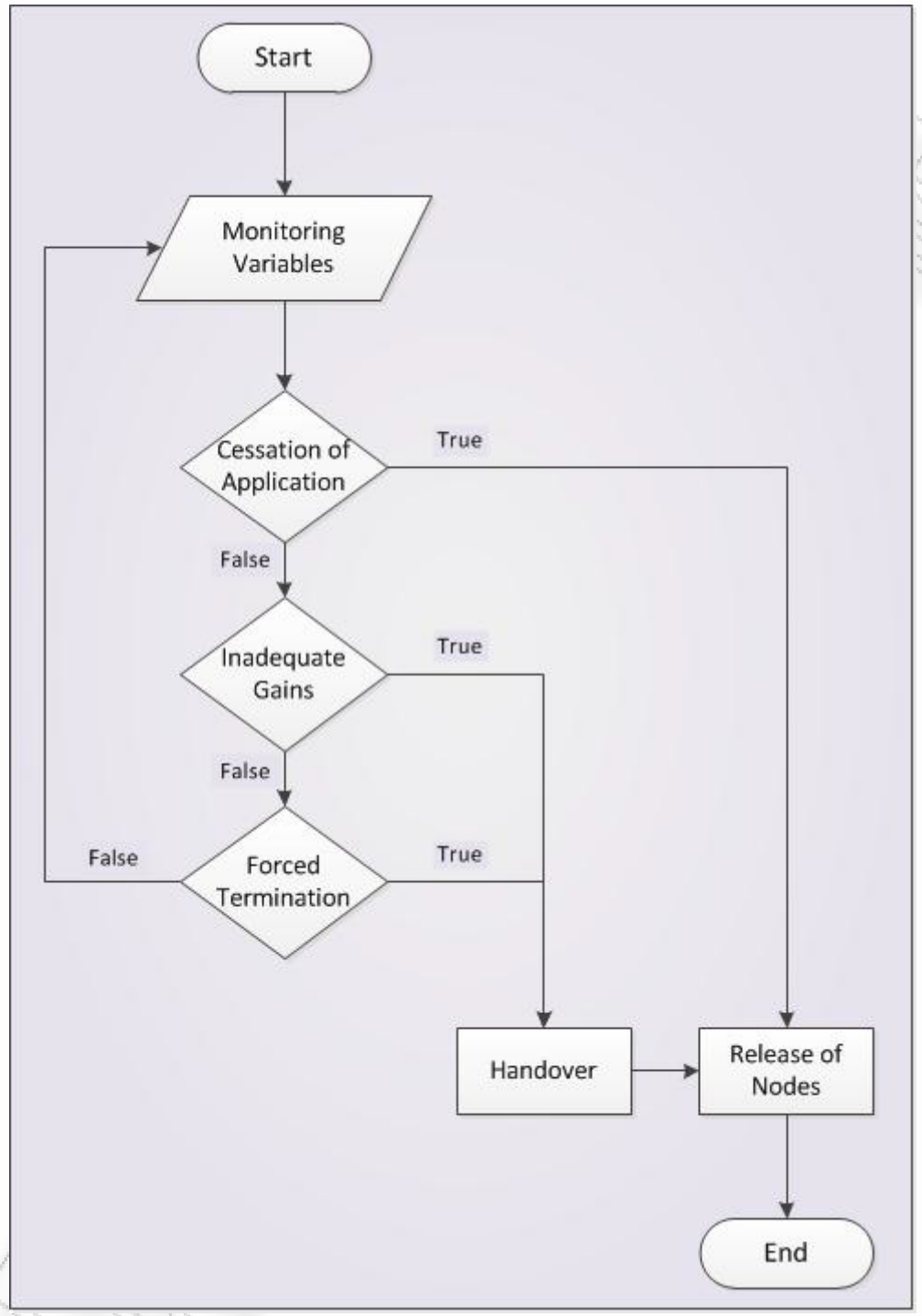
Επιπρόσθετα αυτός ο μηχανισμός για να είναι αποτελεσματικός, θα πρέπει να βρίσκεται σε επαγρύπνηση καθ' όλη τη διάρκεια λειτουργίας του δικτύου. Έτσι θα πραγματοποιεί συνεχή και ακατάπαυστο έλεγχο σε εκείνα τα χαρακτηριστικά που θεωρούνται σχετικά με τη λήξη του.

Στην ενότητα αυτή, παρουσιάζεται ένας τέτοιου είδους αλγόριθμος. Για την καλύτερη κατανόησή του, η μοντελοποίηση της εικόνας 3.1 αποδίδει μία σαφή και ολοκληρωμένη εικόνα του τρόπου λειτουργίας του.

Ο αλγόριθμος εφαρμόζεται (start) κατά το σημείο εκκίνησης του OppNet, αμέσως μετά δηλαδή την ολοκλήρωση της δημιουργίας του και κατά την έναρξη επίβλεψής του. Έπειτα το πρώτο μέλημα είναι η παρακολούθηση ορισμένων μεταβλητών (Monitoring Variables). Οι μεταβλητές αυτές αφορούν το επίπεδο παροχής των εφαρμογών/υπηρεσιών που χρησιμοποιούνται από τους τελικούς χρήστες, το ποσοστό κέρδους του δικτύου, τους διαθέσιμους πόρους καθώς και την ποιότητα των υπηρεσιών (QoS).

Βασισμένοι στις μεταβλητές αυτές έχουν οριστεί τρεις τύποι τερματισμού:

- Διακοπής παροχής των εφαρμογών (Cessation of application provision)
- Ανεπαρκούς κέρδους (Inadequate gains)
- Βεβιασμένου τερματισμού (Forced termination)



Εικ. 3.1 Μοντελοποίηση αλγορίθμου τερματισμού

Όμως, για να δοθεί το έναυσμα ενός εκ των τύπων τερματισμού, απαραίτητη προϋπόθεση είναι οι τιμές των μεταβλητών να είναι χαμηλότερες ή ίσες σε σύγκριση με τις τιμές κατωφλίου (threshold) που έχουν οριστεί από το σύστημα. Το κατώφλι αποτελείται από αυστηρά προκαθορισμένες τιμές που σκοπό έχουν να διακόψουν τη διεξαγωγή του δικτύου, όταν οι μετρήσεις είναι κατώτερες των επιτρεπόμενων.

Αυτό το κατώτερο όριο τιμών αποδίδεται στον αλγόριθμο με τη χρήση συνθηκών. Κάθε τύπος τερματισμού χρησιμοποιεί μία συνθήκη. Όταν η συνθήκη είναι αληθής σημαίνει ότι ο τερματισμός έχει πυροδοτηθεί. Στην αντίθετη περίπτωση ελέγχεται η συνθήκη για τον επόμενο τύπο. Αν καμία από τις συνθήκες δεν είναι αληθής, σημαίνει ότι το δίκτυο δε χρειάστηκε να τερματιστεί αφού οι τιμές των μεταβλητών του δεν ήταν κατώτερες του κατωφλίου.

Συνεπώς, το κατώφλι για τον πρώτο τύπο τερματισμού (Cessation of application provision) δεν είναι άλλο παρά να ολοκληρωθούν όλες οι εφαρμογές και οι υπηρεσίες που λάμβαναν χώρα στο OppNet. Με λίγα λόγια το δίκτυο δεν είναι πλέον αποδοτικό αφού κανείς δεν το χρησιμοποιεί, με αποτέλεσμα τον ομαλό τερματισμό του απελευθερώνοντας τους πόρους.

Στον δεύτερο τύπο τερματισμού (Inadequate gains) το κατώφλι περιλαμβάνει μόνο το ποσοστό του κέρδους. Αν αυτό δεν είναι επαρκές, τότε το δίκτυο τερματίζεται αναγκαστικά, αγνοώντας αν έχει ολοκληρωθεί η παροχή υπηρεσιών. Το αποτέλεσμα είναι αρχικά να πραγματοποιηθεί handover στο υπόλοιπο δίκτυο και έπειτα ακολουθεί η απελευθέρωση των πόρων.

Ο τρίτος και τελευταίος τερματισμός προκαλείται χάρη στην έλλειψη πόρων ή στην κακή ποιότητα υπηρεσιών. Παρομοίως με τον προηγούμενο τύπο το OppNet τερματίζεται υποχρεωτικά, εκτελώντας handover και ύστερα απελευθερώνει τους πόρους του.

Κεφάλαιο 4
Υλοποίηση Αλγορίθμου

4.1 Υλοποίηση αλγορίθμου και εργαλεία χρήσης

Μετά την εύρεση και μοντελοποίηση ενός αλγορίθμου για τον εντοπισμό συμβάντος τερματισμού του OppNet, παρουσιάζεται μία νέα πρόκληση που αφορά την υλοποίηση και εκτέλεσή του. Το γεγονός αυτό θα επιφέρει αποτελέσματα τα οποία θα αποδείξουν την αξία εφαρμογής του σε ένα τέτοιο σύστημα. Επιπρόσθετα παρέχεται η δυνατότητα επισήμανσης στοιχείων που χρειάζονται περαιτέρω μελέτη και έχουν την τάση να βελτιωθούν.

Για τους λόγους αυτούς η υλοποίηση του συγκεκριμένου αλγορίθμου είναι πλέον γεγονός. Για τους σκοπούς αυτής της υλοποίησης χρησιμοποιήθηκαν τα εξής δύο εργαλεία:

- Γλώσσα προγραμματισμού τρίτης γενιάς Java [15], [16]
- Σύστημα βάσης δεδομένων MySQL 5.5 [17]

Η επιλογή των παραπάνω δεν είναι τυχαία και οφείλεται στην άψογη συνεργασία τους. Η χρήση του MySQL-Java Connector 5.1.15 [18] επιτρέπει στη Java να κάνει χρήση εντολών MySQL και να έχει πρόσβαση στη βάση δεδομένων. Συνεπώς προσφέρεται η δυνατότητα ανάγνωσης, τροποποίησης, ενημέρωσης ακόμα και διαγραφής δεδομένων μέσω του πηγαίου κώδικα της java.

4.2 Σενάριο και πρόγραμμα επτά κόμβων

Ο αλγόριθμος λοιπόν θα υλοποιηθεί με τη δημιουργία ενός προγράμματος το οποίο θα βασίζεται σε ένα σενάριο με σκοπό να είναι αληθοφανές και να έχει μία λογική βάση. Θεωρείται λοιπόν ένα ON το οποίο αποτελείται από 7 κόμβους, καθένας με το δικό του ρόλο μέσα σε αυτό. Δύο χαρακτηρίζονται ως κόμβοι πύλης με σκοπό να συνδέουν το δίκτυο με την υποδομή του. Οι δύο επόμενοι διαδραματίζουν το ρόλο των κόμβων αναμετάδοσης προωθώντας τα πακέτα από και προς τους τελικούς χρήστες ενώ οι τρεις τελευταίοι αποτελούν τους κόμβους εφαρμογών. (σχήμα)

Αρχικά, πριν την εκτέλεση του προγράμματος δημιουργείται μία βάση δεδομένων στην οποία θα καταγράφονται όλες οι μετρήσεις που θα συλλέγονται μελλοντικά. Έπειτα, εφόσον η βάση είναι συνδεδεμένη, ακολουθεί η ενεργοποίηση του εξυπηρετητή («σηκώνεται» ο server) αναμένοντας αιτήματα.

Οι ενέργειες που εκτελεί σειριακά το πρόγραμμα είναι οι εξής:

1. Καταγραφή και μεταφορά μεταβλητών δικτύου προς τη βάση δεδομένων.

2. Έλεγχο μετρήσεων με τις προκαθορισμένες τιμές κατωφλίου.
3. Ενεργοποίηση κατάλληλου μηχανισμού τερματισμού όταν κριθεί απαραίτητο.
4. Αναφορά αποτελεσμάτων σε γραφικό περιβάλλον.

Κατά την εκκίνηση του προγράμματος θεωρείται ότι το ON έχει μόλις δημιουργηθεί ύστερα από μία έγκυρη απόφαση καταλληλότητας. Ουσιαστικά λοιπόν το πρόγραμμα εφαρμόζεται κατά τη φάση της παρακολούθησης των συστατικών του. Η διεργασία αυτή αποτελεί και το πρώτο του μέλημα καθώς η συγκομιδή των μετρήσεων περί των διάφορων μεταβλητών του δικτύου όπως το φάσμα και το επίπεδο ενέργειας θα μεταφέρονται απευθείας στη βάση δεδομένων. Μετέπειτα τα δεδομένα αυτά υπόκεινται σε υποχρεωτικό έλεγχο έτσι ώστε να διασφαλιστεί η ομαλή λειτουργία του δικτύου.

Η διαδικασία που περιγράφηκε παραπάνω επαναλαμβάνεται κάθε χρονική στιγμή για όλους τους συμμετέχοντες κόμβους μέχρις ότου να συμβεί κάτι «ασυνήθιστο» για το δίκτυο. Όταν κάποια μέτρηση είναι κατώτερη του ορίου τότε ενεργοποιείται ένας εκ των μηχανισμών τερματισμού. Σε εκείνο το χρονικό σημείο στέλνονται και τα τελευταία δείγματα δεδομένων προς τη βάση οδηγώντας το δίκτυο στην απελευθέρωση πόρων ή και εφαρμόζοντας handover όταν κριθεί απαραίτητο.

Μετά το τέλος διεξαγωγής του προγράμματος, παρέχεται η δυνατότητα παρατήρησης και ανάλυσης όλων των μετρήσεων δικτύου μέσω ειδικά σχεδιασμένου γραφικού περιβάλλοντος (Graphical User Interface, GUI). Το περιβάλλον αυτό παραθέτει όλα τα στοιχεία που συλλέχθηκαν στη βάση δεδομένων. Ο διαχειριστής/χρήστης παρατηρεί το χρονικό σημείο που συνέβη ο τερματισμός του δικτύου αλλά και το λόγο αυτού του γεγονότος. Επίσης έχει την ευχέρεια σύγκρισης τιμών σε διαφορετικές χρονικές στιγμές καθώς και επίβλεψης της μεταβλητής συμπεριφοράς ενός συγκεκριμένου κόμβου.

Τέλος, η έξοδος από το γραφικό περιβάλλον σηματοδοτεί την αποσύνδεση από τη βάση δεδομένων, ενώ όλα τα στοιχεία που έχουν συλλεχθεί διαγράφονται. Λόγος της διαγραφής είναι μία επόμενη εκτέλεση του προγράμματος όπου δε θα ήταν επιθυμητό να συνευρεθούν στον ίδιο χώρο παλιές και νέες μετρήσεις.

4.3 Ανάλυση κώδικα

Όσον αφορά το κομμάτι του προγραμματισμού, ήταν προτιμότερο και επιδιώχθηκε να χρησιμοποιηθούν περισσότερες από δύο κλάσεις για την καλύτερη ομαδοποίηση των μεθόδων. Με αυτόν τον τρόπο κάθε κλάση διαδραματίζει έναν ξεχωριστό και ξεκάθαρο ρόλο κατά τη φάση υλοποίησης. Επίσης αυτός ο τρόπος προγραμματισμού καθιστά ευκολότερη την κατανόηση του κώδικα από τρίτους.

Συνεπώς ο βασικός κορμός του προγράμματος απαρτίζεται από τέσσερις κλάσεις οι οποίες αλληλεπιδρούν μεταξύ τους. Αυτές είναι οι εξής:

- Main
- TerminationModel
- VariableChoice
- TableModelTermination

Στις ενότητες που ακολουθούν πραγματοποιείται μία εκτενέστερη ανάλυση σε κάθε μία από αυτές ξεχωριστά, εμπλέκοντας όμως τη μία με την άλλη ώστε να γίνει απόλυτα κατανοητή η λογική σειρά εκτέλεσης του κώδικα.

4.3.1 Η κλάση Main

Η κλάση Main (Εικ. 4.1) ξεκινά την εκτέλεση του προγράμματος καλώντας τη μέθοδο public static void main έχοντας κάνει import το πακέτο javax.swing.JFrame. Ο κώδικας που απαρτίζει την main είναι περιεκτικός, δηλώνοντας αρχικά το αντικείμενο model της κλάσης TerminationModel (γραμμή 14) που θα αναλυθεί παρακάτω. Το model είναι αναγκαίο για το κάλεσμα των μεθόδων setVariables και DisplayResults της κλάσης TerminationModel (γραμμές 15-16) οι οποίες ορίζουν τις μεταβλητές δικτύου και τυπώνουν στην οθόνη τα αποτελέσματα όλης της διαδικασίας τερματισμού. Έπειτα καθορίζονται ορισμένες επιλογές για τη δημιουργία του πλαισίου στο οποίο θα αποτυπωθούν τα αποτελέσματα μετά τη λήξη του δικτύου (γραμμές 18-21).

```
1  package TerminationExecute;
2
3  /**
4   *
5   * @author Papadopoulos Sokratis
6   */
7
8  import javax.swing.JFrame;
9
10 public class Main {
11
12     public static void main(String[] args) {
13
14         TerminationModel model = new TerminationModel();
15         model.setVariables();
16         model.DisplayResults();
17
18         model.setSize( 721, 705 );
19         model.setVisible( true );
20         model.setResizable( false );
21         model.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
22
23     }
24 }
```

Εικ. 4.1 Η κλάση Main.

4.3.2 Η κλάση TerminationModel

Η κλάση TerminationModel εισάγει πολλαπλά πακέτα κλάσεων τύπου java.awt, java.swing και java.sql (Εικ. 4.2). Οι δύο πρώτοι τύποι χρησιμοποιούνται για τη δημιουργία του γραφικού περιβάλλοντος (border layout, grid layout κτλ), αλλά και των συστατικών στοιχείων που θα το απαρτίζουν (buttons, panels, text fields κτλ). Οι κλάσεις του πακέτου java.sql είναι απαραίτητες αρχικά για τη σύνδεση με τη βάση δεδομένων και έπειτα για την προσθήκη, ανάκτηση και διαγραφή δεδομένων.

```
1  package TerminationExecute;
2
3  /**
4   *
5   * @author Papadopoulos Sokratis
6   */
7
8  import java.awt.BorderLayout;
9  import java.awt.GridLayout;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12 import java.sql.Connection;
13 import java.sql.Statement;
14 import java.sql.DriverManager;
15 import java.sql.SQLException;
16 import java.sql.ResultSet;
17 import javax.swing.JTextArea;
18 import javax.swing.JScrollPane;
19 import javax.swing.ScrollPaneConstants;
20 import javax.swing.JFrame;
21 import javax.swing.JTable;
22 import javax.swing.JButton;
23 import javax.swing.JPanel;
24 import javax.swing.JLabel;
25 import javax.swing.JTextField;
26 import javax.swing.JOptionPane;
27
```

Εικ. 4.2 Εισαγόμενα πακέτα κλάσεων στην TerminationModel.

Κατά τη δήλωσή της, η TerminationModel επεκτείνει την κλάση JFrame με σκοπό να κάνει χρήση των μεθόδων της οι οποίες είναι απαραίτητες για την υλοποίηση του γραφικού περιβάλλοντος (Εικ. 4.3 γραμμή 28). Έπειτα δηλώνονται πέντε σταθερές που σχετίζονται με τη σύνδεση στη βάση δεδομένων: ο οδηγός της βάσης (Java Database Connector Driver), η διεύθυνση URL στην οποία είναι τοποθετημένη η βάση στο σύστημα, το όνομα χρήστη, ο κωδικός πρόσβασης και το προεπιλεγμένο ερώτημα (γραμμές 30-34).

Πριν την ανάλυση των μεθόδων της κλάσης, ορίζονται και ορισμένες private μεταβλητές των οποίων οι τιμές είναι επιθυμητό να χρησιμοποιούνται από όλες τις μεθόδους (γραμμές 36-40). Η χρήση των μεταβλητών αυτών θα γίνει αισθητή στην περιγραφή των μεθόδων που ακολουθούν.

Ο βασικός κορμός της TerminationModel λοιπόν αποτελείται από τρεις μεθόδους. Η πρώτη (γραμμή 42-45) είναι η συνάρτηση δημιουργίας της κλάσης και για αυτό το λόγο χρησιμοποιεί το ίδιο όνομα με αυτή. Ο ρόλος της συνάρτησης δημιουργίας είναι περιορισμένος, αφού ο μοναδικός σκοπός δημιουργίας της είναι να προσδίδει έναν τίτλο στο παράθυρο διεπαφής που εμφανίζεται στην οθόνη μετά τον τερματισμό του δικτύου με τίτλο «Monitoring Opportunistic Networks' Results».

```
28  public class TerminationModel extends JFrame {
29
30      static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
31      static final String DATABASE_URL = "jdbc:mysql://localhost/oNetResults";
32      static final String USERNAME = "*****";
33      static final String PASSWORD = "*******";
34      static final String DEFAULT_QUERY = "SELECT * FROM netNodes";
35
36      private int attempt=1;
37      private TableModelTermination tableModel;
38      private BorderLayout borderLayout = new BorderLayout();
39      private JTextField textField1;
40      private JTextField textField2;
41
42      public TerminationModel()
43      {
44          super( "Monitoring Opportunistic Networks' Results" );
45      }
46
47
48      public void setVariables() {...}
518
519
520
521     public void DisplayResults() {...}
704 }
705
```

Εικ. 4.3 Η κλάση TerminationModel.

Οι διεργασίες που εκτελούν οι ακόλουθες μέθοδοι είναι υψίστης σημασίας αφού ουσιαστικά όλη η ιδέα του αλγορίθμου εξελίσσεται στο εσωτερικό τους (γραμμές 48-517, γραμμές 521-703). Ως εκ τούτου η περιγραφή της λειτουργικότητάς τους παρατίθεται στις αμέσως επόμενες υποενότητες του κεφαλαίου αυτού.

4.3.2.1 Η μέθοδος setVariables

Ο βασικός ρόλος της μεθόδου setVariables είναι όπως υπονοεί και το όνομά της να θέτει σε πρώτη φάση τις μεταβλητές δικτύου και έπειτα να τις αποθηκεύει στη βάση δεδομένων. Εκτός όμως από αυτές τις διεργασίες εκτελεί και μερικές ακόμη, όπως τη δημιουργία μίας περιοχής κειμένου και φυσικά τη σύνδεση με τη βάση δεδομένων.

Αρχικά η setVariables ορίζει τον τύπο της διαχείρισης διατάξεων του γραφικού περιβάλλοντος μέσω της μεθόδου setLayout. Έτσι εισάγοντας σε αυτή το στιγμιότυπο borderLayout που δημιουργήθηκε προηγουμένως (Εικ. 4.3 γραμμή 38), ορίζεται η διάταξη BorderLayout (Εικ. 4.4 γραμμή 50). Αυτό σημαίνει ότι τα συστατικά θα τακτοποιούνται σε πέντε περιοχές (NORTH, SOUTH, EAST, WEST, CENTER).

Το πρώτο συστατικό που προστίθεται στη βόρεια περιοχή της διεπαφής (γραμμή 61) είναι ένα JTextArea (γραμμή 52-55). Ο ρόλος αυτής της περιοχής κειμένου θα είναι παθητικός, αφού δε θα είναι δυνατό να τροποποιηθούν τα στοιχεία του, παρά μόνο να τεθούν προς ανάγνωση (γραμμή 53). Συνεπώς ο ρόλος του στο γραφικό περιβάλλον θα περιορίζεται στην παρακολούθηση της λειτουργίας του δικτύου.

```
48  public void setVariables() {
49
50      setLayout( borderLayout );
51
52      JTextArea textArea = new JTextArea( 13, 100 );
53      textArea.setEditable( false );
54      textArea.setWrapStyleWord( true );
55      textArea.setLineWrap( true );
56
57      JScrollPane scrollPane = new JScrollPane( textArea,
58              ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
59              ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER  );
60
61      add( scrollPane, BorderLayout.NORTH );
```

Εικ. 4.4 Δημιουργία TextArea.

Στη συνέχεια δημιουργούνται 8 αντικείμενα της κλάσης VariableChoice (Εικ. 4.5 γραμμές 68-75). Ο αριθμός αυτός δεν είναι τυχαίος καθώς συμβαδίζει με το πλήθος των κόμβων του σεναρίου που υλοποιείται. Ως απόρροια τούτου δύο αντιπροσωπεύουν τα χαρακτηριστικά των κόμβων πύλης, δύο των κόμβων αναμετάδοσης και τρεις των κόμβων εφαρμογής. Το όγδοο αντικείμενο χρησιμοποιείται για την καταμέτρηση του φάσματος δικτύου καθώς και για τη μέση τιμή ορισμένων μεταβλητών.

Στις γραμμές 78-81 δηλώνονται οι τιμές κατωφλίου για κάποιες μεταβλητές δικτύου. Οι τιμές αυτές είναι με τέτοιο τρόπο προσδιορισμένες ώστε να ανταποκρίνονται όσο το δυνατόν περισσότερο στην πραγματικότητα. Για παράδειγμα το κατώφλι για το bit rate ανέρχεται στα 120 Kbps ενώ για το κέρδος δικτύου στο 20%.

```
62
63          /** node1, node2 are gateway nodes
64           *  node3, node4 are relay nodes
65           *  node5, node6, node7 are application nodes
66           *  node8 doesn't exist but we assign spectrum & total network values
67           */
68          VariableChoice node1 = new VariableChoice();
69          VariableChoice node2 = new VariableChoice();
70          VariableChoice node3 = new VariableChoice();
71          VariableChoice node4 = new VariableChoice();
72          VariableChoice node5 = new VariableChoice();
73          VariableChoice node6 = new VariableChoice();
74          VariableChoice node7 = new VariableChoice();
75          VariableChoice networkValues = new VariableChoice();
76
77          //Thresholds in order to trigger the termination
78            int interfaceThreshold = 1, bitRateThreshold = 120, key = 0,
79                    delayThreshold = 23, spectrumThreshold = 10;
80
81            double gainThreshold = 0.2, berThreshold = 0.025;
```

Εικ. 4.5 Δημιουργία 8 αντικειμένων και τιμών κατωφλίου.

Στις γραμμές 83 και 84 της εικόνας 4.6 δημιουργούνται τα αντικείμενα connection και statement στα οποία αποδίδεται η τιμή null εφόσον ακόμη δεν έχουμε ενεργή σύνδεση με τη βάση. Αμέσως μετά (γραμμές 88-91) στο εσωτερικό της try, εισάγεται ο οδηγός της βάσης και ύστερα δημιουργείται η σύνδεση εισάγοντας το URL, το όνομα χρήστη και τον κωδικό πρόσβασης που δηλώθηκαν προηγουμένως. Εκτός αυτού, χρησιμοποιώντας την connection αποθηκεύεται στο αντικείμενο της κλάσης Statement ο τύπος του επιθυμητού resultSet (Updatable) (γραμμές 93-94).

Έπειτα αρχικοποιούνται ορισμένες μεταβλητές (γραμμές 97-103) οι οποίες χρησιμοποιούνται στις μετέπειτα αποθηκεύσεις των μετρήσεων δικτύου. Για παράδειγμα οι μεταβλητές energyStatus αρχικοποιούνται με τις τιμές «Maximum» διότι κατά την εκκίνηση του δικτύου το ενεργειακό επίπεδο των κόμβων θεωρείται ότι είναι το υψηλότερο δυνατό. Αντίστοιχα οι μεταβλητές appStatus που αφορούν την κατάσταση των εφαρμογών που χρησιμοποιούνται στο δίκτυο παίρνουν τιμές «Starting», εφόσον πριν την εκκίνηση του ON δε θα ήταν δυνατό να παρεχόταν κάποια υπηρεσία.

```
82
83          Connection connection = null;
84          Statement statement = null;
85
86          try
87          {
88            Class.forName( JDBC_DRIVER );
89
90            connection = DriverManager.getConnection( DATABASE_URL, USERNAME,
91                      PASSWORD );
92
93            statement = connection.createStatement(
94                      ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE );
95
96            // Initializing energy & application variables
97            int energy1 = 5, energy2 = 5, energy3 = 5, energy4 = 5, app5 = 4,
98                    app6 = 4, app7 = 4;
99            String energyStatus1 = "Maximum", energyStatus2 = "Maximum",
100                     energyStatus3 = "Maximum", energyStatus4 = "Maximum";
101           String app5Status = "Starting", app6Status = "Starting",
102                     app7Status = "Starting";
103           int rowcount;
```

Εικ. 4.6 Εγκατάσταση σύνδεσης και αρχικοποίηση μεταβλητών.

Συνοψίζοντας λοιπόν περί της μεθόδου setVariables παρατηρούνται οι εξής διεργασίες:

- Ορισμός του τύπου διατάξεων γραφικού περιβάλλοντος.
- Δημιουργία οχτώ αντικειμένων αντιπροσώπων των κόμβων του δικτύου.
- Δήλωση τιμών κατωφλίου για κάθε μεταβλητή.
- Σύνδεση με τη βάση δεδομένων.
- Αρχικοποίηση μεταβλητών μετρήσεων δικτύου.

Το κομμάτι κώδικα που ακολουθεί αποτελεί το σημαντικότερο τμήμα της μεθόδου αυτής αφού εδώ λαμβάνει χώρα η καταγραφή των τιμών στις κατάλληλες μεταβλητές και έπειτα πραγματοποιείται έλεγχος τερματισμού του OppNet. Στο σημείο αυτό θα γίνει αναφορά περί των μεταβλητών αυτών που ουσιαστικά αποτελούν στοιχεία του δικτύου τα οποία τίθενται υπό μελέτη μέχρις ότου να τερματιστεί. Τα στοιχεία αυτά είναι πιθανό να έχουν αναφερθεί πολλές φορές στο περιεχόμενο τις εργασίας αλλά ποτέ συγκεντρωμένα όλα μαζί. Αυτά είναι:

- Κατάσταση της εφαρμογής (application status)
- Κέρδος δικτύου (gain)
- Επίπεδο ενέργειας (energy level)
- Φάσμα συχνοτήτων (frequency spectrum)
- Ενεργές διεπαφές (interfaces)
- Ρυθμός bits (Bit-Rate)
- Καθυστέρηση (Delay)

- Ρυθμός εσφαλμένων bits (BER, Bit Error Rate)

Κατά την υλοποίηση του σεναρίου αυτού, στις μεταβλητές έχει αποδοθεί συγκεκριμένο πεδίο τιμών. Τα πεδία αυτά είναι ενδεικτικά και χρησιμοποιούνται μόνο για τη διεξαγωγή του πειράματος. Συνεπώς σε πραγματικά περιβάλλοντα είναι δυνατό να αποκλίνουν.

- Application Status : [Starting, Established, Running, Finished]
- Network Gain : [0-100%]
- Energy Level : [Maximum, High. Medium, Low]
- Frequency Spectrum : [10-100 MHz]
- Interface : [0-4]
- Bit Rate : [10-500 Kbps]
- Delay : [0-30 ms]
- BER : [0.5-5%]

Η διαδικασία καταγραφής και ελέγχου των μετρήσεων ξεκινά μέσα σε μία while (Εικ. 4.7 γραμμή 105). Ο compiler θα εξέλθει αυτής της επανάληψης μόνο όταν μία μεταβλητή key ισούται με 1. Προφανώς η key αρχικοποιείται με την τιμή 0 (Εικ. 4.5 γραμμή 78). Ο μόνος τρόπος να τροποποιηθεί αυτή η τιμή φρουρός και να μετατραπεί σε 1 είναι κάποια μέτρηση να παρατηρηθεί κάτω του επιτρεπτού ορίου, με λίγα λόγια να ξεκινήσει η διαδικασία τερματισμού του ON.

Στην εικόνα 4.7 πραγματοποιείται καταγραφή των μετρήσεων για τον κόμβο πύλης 1. Από τα οχτώ στοιχεία που αναφέρθηκαν παραπάνω, ένας κόμβος πύλης ενδιαφέρεται μόνο για τα έξι από αυτά. Αυτό είναι απόλυτα δικαιολογημένο καθώς δεν είναι δυνατό να μας ενδιαφέρει το application status για έναν κόμβο πύλης ο οποίος δεν κάνει χρήση των υπηρεσιών. Επίσης το spectrum είναι μεταβλητή δικτύου και δεν αφορά συγκεκριμένους κόμβους, με αποτέλεσμα η τιμή του να αναφέρεται στο δίκτυο.

Συνεπώς χρησιμοποιώντας το αντικείμενο node1 της κλάσης variableChoice καλούνται οι αντίστοιχες μέθοδοι για την καταγραφή των ενδιαφερόμενων στοιχείων. Ο τρόπος επιλογής των τιμών και η λειτουργία των μεθόδων αυτών θα μελετηθεί αργότερα. Πρωτεύον ζήτημα αύτη τη στιγμή είναι ότι στις γραμμές 112-130 ο καθορισμός της τιμής επιπέδου ενέργειας έχει κάτι το διαφορετικό σε σχέση με την απλή απόδοση τιμών στις υπόλοιπες μεταβλητές. Με τους δύο μηχανισμούς ελέγχου που έχουν παρατεθεί (if else) εξασφαλίζεται ότι το επίπεδο ενέργειας του κόμβου θα παραμείνει στάσιμο ή θα μειώνεται σταδιακά. Δηλαδή δεν είναι δυνατόν η ενέργεια να αυξηθεί από «Medium» σε «High». Αντιθέτως, το πιο πιθανό σενάριο είναι να ελαττώνεται με την πάροδο του χρόνου. Αυτό είναι απόρροια της ασύρματης διάστασης των συσκευών που χρησιμοποιούν τα OppNets και της μη δυνατότητας επαναφόρτισής τους κατά τη διάρκεια συμμετοχής τους σε αυτά.

```
104
105         while(key==0)
106         {
107
108           System.out.print("\n Insert new network data measurements : Done\n");
109           //NODE 1
110           float gain1 = node1.gainValue();
111
112           if ( energy1 == 4 )
113               energy1 = 3+node1.energyValue();
114           else if ( energy1 == 3 )
115               energy1 = 2+node1.energyValue();
116           else if ( energy1 == 2 )
117               energy1 = 1+node1.energyValue();
118           else if ( energy1 == 5 )
119               energy1 = 3+node1.energyValue();
120           else
121               energy1 = 1;
122
123           if ( energy1 == 1 )
124               energyStatus1 = "Low";
125           else if ( energy1 == 2 )
126               energyStatus1 = "Medium";
127           else if ( energy1 == 3 )
128               energyStatus1 = "High";
129           else if ( energy1 == 4 )
130               energyStatus1 = "Maximum";
131
132           int interface1 = node1.numberOfInterfaces();
133           int bitrate1 = 10*(1+node1.bitRateValue());
134           int delay1 = node1.delayValue();
135           double ber1 = node1.bitErrorRateValue();
136
137           rowcount=statement.executeUpdate( "INSERT INTO netData ( nodeID,"
138                     + " gain, energyLevel, Interfaces, bitRate, delay, ber,"
139                     + " attemptNum ) VALUES ( 1, "+gain1+","
140                     + " '"+energyStatus1+"', "+interface1+", "+bitrate1+","
141                     + " "+delay1+", "+ber1+", "+attempt+" )" );
142           //NODE 1
```

Εικ. 4.7 Εκκίνηση επανάληψης και καταχώρηση τιμών κόμβου 1.

Μετά την αποθήκευση των τιμών στις αντίστοιχες μεταβλητές, χρησιμοποιείται η μέθοδος executeUpdate της κλάσης Statement για την άμεση καταχώρηση στη βάση δεδομένων (γραμμές 137-141). Ταυτόχρονα στη βάση περνάει και η χρονική στιγμή που καταγράφηκε το σύνολο των στοιχειών μέσω της attempt (γραμμή 141).

Η ίδια διαδικασία θα ακολουθηθεί και για τον κόμβο 2 (Εικ. 4.8) αφού και εκείνος είναι τύπου πύλης. Όμως παρόμοια διαδικασία καταγραφής χρησιμοποιείται και από τους κόμβους αναμετάδοσης 3 και 4 (Εικ. 4.9 και 4.10 αντίστοιχα) διότι και εκείνοι συλλέγουν στοιχεία τα οποία

σχετίζονται με το επίπεδο ενέργειας, το κέρδος δικτύου, το πλήθος των διεπαφών και τα χαρακτηριστικά της ποιότητας των υπηρεσιών (QoS).

```
144             //NODE 2
145             float gain2 = node2.gainValue();
146
147             if( energy2 == 5 )
148                 energy2 = 3+node2.energyValue();
149             else if ( energy2 == 4 )
150                 energy2 = 3+node2.energyValue();
151             else if ( energy2 == 3 )
152                 energy2 = 2+node2.energyValue();
153             else if ( energy2 == 2 )
154                 energy2 = 1+node2.energyValue();
155             else
156                 energy2 = 1;
157
158             if ( energy2 == 1 )
159                  energyStatus2 = "Low";
160             else if ( energy2 == 2 )
161                  energyStatus2 = "Medium";
162             else if ( energy2 == 3 )
163                  energyStatus2 = "High";
164             else if ( energy2 == 4 )
165                  energyStatus2 = "Maximum";
166
167             int interface2 = node2.numberOfInterfaces();
168             int bitrate2 = 10*(1+node2.bitRateValue());
169             int delay2 = node2.delayValue();
170             double ber2 = node2.bitErrorRateValue();
171
172             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
173                       + " gain, energyLevel, Interfaces, bitRate, delay, ber,"
174                       + " attemptNum ) VALUES ( 2, "+gain2+","
175                       + " '"+energyStatus2+"', "+interface2+", "+bitrate2+","
176                       + " "+delay2+", "+ber2+", "+attempt+" )" );
177             //NODE 2
```

Εικ. 4.8 Καταχώρηση τιμών κόμβου 2.

```
179             //NODE 3
180             float gain3 = node3.gainValue();
181
182             if ( energy3 == 5 )
183                 energy3 = 3+node3.energyValue();
184             else if ( energy3 == 4 )
185                 energy3 = 3+node3.energyValue();
186             else if ( energy3 == 3 )
187                 energy3 = 2+node3.energyValue();
188             else if ( energy3 == 2 )
189                 energy3 = 1+node3.energyValue();
190             else
191                 energy3 = 1;
192
193             if ( energy3 == 1 )
194                 energyStatus3 = "Low";
195             else if ( energy3 == 2 )
196                 energyStatus3 = "Medium";
197             else if ( energy3 == 3 )
198                 energyStatus3 = "High";
199             else if ( energy3 == 4 )
200                 energyStatus3 = "Maximum";
201
202             int interface3 = node3.numberOfInterfaces();
203             int bitrate3 = 10*(1+node3.bitRateValue());
204             int delay3 = node3.delayValue();
205             double ber3 = node3.bitErrorRateValue();
206
207             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
208                       + " gain, energyLevel, Interfaces, bitRate, delay, ber,"
209                       + " attemptNum ) VALUES ( 3, "+gain3+","
210                       + " '"+energyStatus3+"', "+interface3+", "+bitrate3+","
211                       + " "+delay3+", "+ber3+", "+attempt+" )" );
212             //NODE 3
```

Εικ. 4.9 Καταχώρηση τιμών κόμβου 3.

```
214             //NODE 4
215             float gain4 = node4.gainValue();
216
217             if ( energy4 == 5 )
218                 energy4 = 3+node4.energyValue();
219             else if ( energy4 == 4 )
220                 energy4 = 3+node4.energyValue();
221             else if ( energy4 == 3 )
222                 energy4 = 2+node4.energyValue();
223             else if ( energy4 == 2 )
224                 energy4 = 1+node4.energyValue();
225             else
226                 energy4 = 1;
227
228             if ( energy4 == 1 )
229                 energyStatus4 = "Low";
230             else if ( energy4 == 2 )
231                 energyStatus4 = "Medium";
232             else if ( energy4 == 3 )
233                 energyStatus4 = "High";
234             else if ( energy4 == 4 )
235                 energyStatus4 = "Maximum";
236
237             int interface4 = node4.numberOfInterfaces();
238             int bitrate4 = 10*(1+node4.bitRateValue());
239             int delay4 = node4.delayValue();
240             double ber4 = node4.bitErrorRateValue();
241
242             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
243                       + " gain, energyLevel, Interfaces, bitRate, delay, ber,"
244                       + " attemptNum ) VALUES ( 4, "+gain4+","
245                       + " '"+energyStatus4+"', "+interface4+", "+bitrate4+","
246                       + " "+delay4+", "+ber4+", "+attempt+" )" );
247             //NODE 4
```

Εικ. 4.10 Καταχώρηση τιμών κόμβου 4.

Όσον αφορά τους κόμβους εφαρμογής 5,6 και 7 είναι προφανές ότι παρακολουθείται το επίπεδο εφαρμογής τους το οποίο καθολικά έχει ως τιμή εκκίνησης την «Starting». Με την πάροδο του χρόνου λοιπόν, η τιμή αυτή αλλάζει μονοσήμαντα προς τον τερματισμό της εφαρμογής (Starting→Established→Running→Finished). Το γεγονός αυτό σημαίνει τη λήξη της υπηρεσίας στην τιμή «Finished». Άπαξ και ο κόμβος χαρακτηριστεί από την τιμή αυτή, σταματά να κάνει χρήση των υπηρεσιών και εισέρχεται σε μία κατάσταση αδράνειας. Όμως αυτό δεν περιορίζει τη συνεισφορά του στο έργο της προώθησης πακέτων προς άλλους κόμβους, καθώς μετατρέπεται σε κόμβο αναμετάδοσης.

Το επίπεδο ενέργειας δε θεωρείται απαραίτητη μέτρηση προς παρακολούθηση για τους κόμβους εφαρμογών, αν η πιθανή απόσυρσή τους από το δίκτυο δεν πυροδοτεί αλλαγές στην παροχή υπηρεσιών των εναπομεινάντων κόμβων. Βέβαια αυτό προϋποθέτει την απευθείας σύνδεση των

υπόλοιπων κόμβων εφαρμογής με λοιπούς κόμβους αναμετάδοσης. Με λίγα λόγια δε θα πρέπει να υφίσταται μονοπωλιακή εξάρτηση ενός κόμβου με άλλον. Σε αντίθετη περίπτωση ή σε δίκτυο με περισσότερα σημεία σε σύγκριση με το υπό μελέτη σενάριο, η παρακολούθηση του ενεργειακού επιπέδου των κόμβων εφαρμογής θα είναι αναγκαία.

Συμπερασματικά οι τιμές που καταγράφονται για αυτούς τους τύπους κόμβων είναι η κατάσταση εφαρμογής, το κέρδος δικτύου, ο ρυθμός μετάδοσης, η καθυστέρηση καναλιών και ο ρυθμός εσφαλμένων bits (Εικ. 4.11 γραμμές 250-273).

```
249             //NODE 5
250             if ( app5 == 4 )
251                 app5 = 0;
252             else if ( app5 == 0 )
253                 app5 = node5.applicationStatusLevel();
254             else if ( app5 == 1 )
255                 app5 = 1+node5.applicationStatusLevel();
256             else if ( app5 == 2 )
257                 app5 = 2+node5.applicationStatusLevel();
258             else
259                 app5 = 3;
260
261             if ( app5 == 0 )
262                 app5Status = "Starting";
263             else if ( app5 == 1 )
264                 app5Status = "Established";
265             else if ( app5 == 2 )
266                 app5Status = "Running";
267             else if ( app5 == 3 )
268                 app5Status = "Finished";
269
270             float gain5 = node5.gainValue();
271             int bitrate5 = 10*(1+node5.bitRateValue());
272             int delay5 = node5.delayValue();
273             double ber5 = node5.bitErrorRateValue();
274             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
275                       + " appStatus, gain, bitRate, delay, ber, attemptNum )"
276                       + " VALUES ( 5, '"+app5Status+"', "+gain5+", "+bitrate5+","
277                       + " "+delay5+", "+ber5+", "+attempt+" )" );
278             //NODE 5
```

Εικ. 4.11 Καταχώρηση τιμών κόμβου 5.

Στις εικόνες 4.12 και 4.13 απεικονίζονται οι καταχωρήσεις των τιμών για τους κόμβους εφαρμογής 5 και 6.

```
280             //NODE 6
281             if ( app6 == 4 )
282                 app6 = 0;
283             else if ( app6 == 0 )
284                 app6 = node6.applicationStatusLevel();
285             else if ( app6 == 1 )
286                 app6 = 1+node6.applicationStatusLevel();
287             else if ( app6 == 2 )
288                 app6 = 2+node6.applicationStatusLevel();
289             else
290                 app6 = 3;
291
292             if ( app6 == 0 )
293                 app6Status = "Starting";
294             else if ( app6 == 1 )
295                 app6Status = "Established";
296             else if ( app6 == 2 )
297                 app6Status = "Running";
298             else if ( app6 == 3 )
299                 app6Status = "Finished";
300
301             float gain6 = node6.gainValue();
302             int bitrate6 = 10*(1+node6.bitRateValue());
303             int delay6 = node6.delayValue();
304             double ber6 = node6.bitErrorRateValue();
305
306             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
307                       + " appStatus, gain, bitRate, delay, ber, attemptNum )"
308                       + " VALUES ( 6, '"+app6Status+"', "+gain6+", "+bitrate6+","
309                       + " "+delay6+", "+ber6+", "+attempt+" )" );
310             //NODE 6
```

Εικ. 4.12 Καταχώρηση τιμών κόμβου 6.

```
312             //NODE 7
313             if ( app7 == 4 )
314                 app7 = 0;
315             else if ( app7 == 0 )
316                 app7 = node7.applicationStatusLevel();
317             else if ( app7 == 1 )
318                 app7 = 1+node7.applicationStatusLevel();
319             else if ( app7 == 2 )
320                 app7 = 2+node7.applicationStatusLevel();
321             else
322                 app7 = 3;
323
324             if ( app7 == 0 )
325                 app7Status = "Starting";
326             else if ( app7 == 1 )
327                 app7Status = "Established";
328             else if ( app7 == 2 )
329                 app7Status = "Running";
330             else if ( app7 == 3 )
331                 app7Status = "Finished";
332
333             float gain7 = node7.gainValue();
334             int bitrate7 = 10*(1+node7.bitRateValue());
335             int delay7 = node7.delayValue();
336             double ber7 = node7.bitErrorRateValue();
337
338             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
339                       + " appStatus, gain, bitRate, delay, ber, attemptNum )"
340                       + " VALUES ( 7, '"+app7Status+"', "+gain7+", "+bitrate7+","
341                       + " "+delay7+", "+ber7+", "+attempt+" )" );
342             //NODE 7
```

Εικ. 4.13 Καταχώρηση τιμών κόμβου 7.

Το τελευταίο σημείο του κώδικα στο οποίο καταγράφονται τιμές προς τη βάση δεδομένων αφορά το όγδοο αντικείμενο με την ονομασία networkValues. Αρχικά υπολογίζεται η μέση τιμή ήδη συλλεγμένων τιμών (Εικ. 4.14 γραμμές 346-356) όπως της καθυστέρησης και του κέρδους. Η μέση τιμή κάθε είδους μεταβλητής είναι απαραίτητη για τη μετέπειτα χρήση της στην απόφαση τερματισμού. Έπειτα αποδίδεται τιμή στη μεταβλητή του φάσματος συχνοτήτων spectrum (γραμμή 358). Τελικώς ανανεώνεται η βάση με τη χρήση της executeUpdate.

Συμπερασματικά η δημιουργία του 8^{ου} κόμβου είναι απόρροια της ανάγκης για καταγραφή των συνολικών μετρήσεων δικτύου. Το γεγονός αυτό επιτρέπει την ευκολότερη μελέτη τους καθώς και τη χρήση τους στο κρίσιμο σημείο της απόφασης τερματισμού.

```
344             //NODE 8
345
346             float mtGain = (gain1+gain2+gain3+gain4+gain5+gain6+gain7)/7;
347
348             double mtBitRate = ( bitrate1 + bitrate2 + bitrate3 + bitrate4 +
349                       bitrate5 + bitrate6 + bitrate7 )/7;
350             int mtIntBitRate = (int)mtBitRate;
351
352             double mtDelay = ( delay1 + delay2 + delay3 + delay4 + delay5 +
353                       delay6 + delay7 )/7;
354             int mtIntDelay = (int)mtDelay;
355
356             double mtBER = ( ber1 + ber2 +ber3 + ber4 + ber5 +ber6 +ber7 )/7;
357
358             int spectrumBnwdth = networkValues.spectrumValue();
359
360             rowcount = statement.executeUpdate( "INSERT INTO netData ( nodeID,"
361                     + " gain, spectrum , bitRate, delay, ber, attemptNum)"
362                     + " VALUES ( 8, "+mtGain+", "+spectrumBnwdth+","
363                     + " "+mtIntBitRate+", "+mtIntDelay+", "+mtBER+","
364                     + " "+attempt+" )" );
365             //NODE 8
```

Εικ. 4.14 Καταγραφή συνολικών μετρήσεων δικτύου.

Μετά την ενημέρωση της βάσης περί των μετρήσεων που ελήφθησαν σε συγκεκριμένο χρονικό στιγμιότυπο (timestamp) ακολουθεί ο έλεγχος απόφασης τερματισμού. Στο σημείο αυτό χρησιμοποιούνται οι τιμές thresholds που δηλώθηκαν παραπάνω (Εικ. 4.5 γραμμές 78-81). Για το σκοπό αυτό, εφαρμόζεται η εντολή ελέγχου if else. Κάθε συνθήκη αντιπροσωπεύει και έναν τύπο τερματισμού με εξαίρεση την τελευταία που υποδηλώνει τη συνέχεια λειτουργίας του δικτύου.

Ο βασικός κορμός εντολών στο εσωτερικό κάθε ελέγχου συμπεριλαμβάνει την αντίστοιχη εκτύπωση στη γραμμή εντολών και έπειτα την αναλυτική περιγραφή περί του χρονικού στιγμιότυπου, του τύπου τερματισμού και των ενεργειών στις οποίες θα προβεί το δίκτυο. Τα στοιχεία αυτά θα αποτυπωθούν στο JTextArea που είχε δημιουργηθεί στην αρχή της κλάσης με την μέθοδο append.

Η σειρά με την οποία πραγματοποιούνται οι έλεγχοι δεν είναι τυχαία, αλλά επιλεγμένη με βάση έναν αυστηρό βαθμό προτεραιότητας. Στην πρώτη συνθήκη λοιπόν, τοποθετούνται οι μεταβλητές app* διότι αποτελούν το σημαντικότερο λόγο τερματισμού. Αυτό συνεπάγεται από το εξής γεγονός: έστω ότι την ίδια χρονική στιγμή το επίπεδο ενέργειας των κόμβων είναι κατώτερο του threshold και ταυτόχρονα η παροχή υπηρεσιών ως προς τους τελικούς χρήστες ολοκληρώνεται. Κάποιος θα είχε το δικαίωμα να θέσει το ακόλουθο ερώτημα: «Γιατί να μην τερματιστεί το δίκτυο λόγω χαμηλού ενεργειακού επιπέδου αντί για τη λήξη παροχής υπηρεσιών;». Η απάντηση στο εύλογο αυτό ερώτημα είναι απλή. Το δίκτυο θα μπορούσε να τερματιστεί λόγω έλλειψης ενέργειας αλλά αυτό θα σήμαινε ότι η παροχή των υπηρεσιών έμεινε ανολοκλήρωτη. Όμως, το γεγονός αυτό δεν αληθεύει και έτσι αλλοιώνεται το πραγματικό αποτέλεσμα και δίνεται η λάθος

εντύπωση. Αυτό θα είχε ως συνέπεια το σύστημα να ξεκινήσει τη διαδικασία του handover με σκοπό να εξυπηρετήσει τους εναπομείναντες κόμβους εφαρμογών, κόμβοι οι οποίοι στην πραγματικότητα δε θα υπάρχουν. Είναι λοιπόν προτιμότερο ο τερματισμός να αποδοθεί στην ολοκλήρωση των εφαρμογών παρόλο που το δίκτυο παρουσιάζει χαμηλή ποσότητα ενέργειας.

Χρησιμοποιώντας την ίδια λογική, η σειρά ελέγχου που έχει επιλεχθεί με τη χρήση της εντολής if else είναι η εξής:

1. Application Status
2. Network Gain
3. Energy Level
4. Frequency Spectrum
5. Interfaces
6. Bit Rate
7. Delay
8. BER

Σε κάθε εικόνα που ακολουθεί από την 4.15 έως την 4.22 πραγματοποιείται ένας έλεγχος απόφασης τερματισμού. Αν η συνθήκη ελέγχου είναι αληθής τότε με τις εντολές print και append τυπώνονται τα αντίστοιχα αποτελέσματα. Έπειτα η τιμή-φρουρός key μετατρέπεται σε 1, υποδηλώνοντας τον τερματισμό του OppNet.

```
366
367             System.out.print(" Update Database : Done\n");
368
369             if ( app5 == 3 && app6 == 3 && app7 == 3)
370             {
371                 System.out.print(" Measurements check : Done\n");
372                 System.out.print(" Network termination triggered!");
373                 textArea.append("\n_____\n");
374                 textArea.append(" Timestamp : "+attempt+"\n");
375                 textArea.append(" Type      : Cessation of Application "
376                         + "provision!\n");
377                 textArea.append(" Actions   : Release of Nodes!");
378                 textArea.append("\n_____\n");
379                 key = 1;
380             }
```

Εικ. 4.15 Έλεγχος κατάστασης εφαρμογών.

```
381
382             else if ( mtGain < gainThreshold )
383             {
384                 System.out.print(" Measurements check : Done\n");
385                 System.out.print(" Network termination triggered!");
386                 textArea.append("\n_____________________\n");
387                 textArea.append(" Timestamp : "+attempt+"\n");
388                 textArea.append(" Type      : Inadequate gains!\n");
389                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
390                         + " Process!");
391                 textArea.append("\n_____________________\n");
392                 key = 1;
393             }
```

Εικ. 4.16 Έλεγχος κέρδους δικτύου.

```
394
395             else if ( energy1==1 && energy2==1 && energy3==1 && energy4==1 )
396             {
397                 System.out.print(" Measurements check : Done\n");
398                 System.out.print(" Network termination triggered!");
399                 textArea.append("\n_____________________\n");
400                 textArea.append(" Timestamp : "+attempt+"\n");
401                 textArea.append(" Type      : Forced Termination due to lack of "
402                         + "resourses: Low Energy Level!\n");
403                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
404                         + " Process!");
405                 textArea.append("\n_____________________\n");
406                 key = 1;
407             }
```

Εικ. 4.17 Έλεγχος επιπέδου ενέργειας.

```
408
409             else if ( spectrumBnwdth <= spectrumThreshold )
410             {
411                 System.out.print(" Measurements check : Done\n");
412                 System.out.print(" Network termination triggered!");
413                 textArea.append("\n_____________________\n");
414                 textArea.append(" Timestamp : "+attempt+"\n");
415                 textArea.append(" Type      : Forced Termination due to lack of "
416                         + "resourses: Minimum Spectrum Availability!\n");
417                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
418                         + " Process!");
419                 textArea.append("\n_____________________\n");
420                 key = 1;
421             }
```

Εικ. 4.18 Έλεγχος φάσματος συχνοτήτων.

```
422
423             else if ( interface1 <= interfaceThreshold &&
424                     interface2 <= interfaceThreshold &&
425                     interface3 <= interfaceThreshold &&
426                     interface4 <= interfaceThreshold )
427             {
428                 System.out.print(" Measurements check : Done\n");
429                 System.out.print(" Network termination triggered!");
430                 textArea.append("\n_____________________\n");
431                 textArea.append(" Timestamp : "+attempt+"\n");
432                 textArea.append(" Type      : Forced Termination due to lack of "
433                         + "resourses: Not Enough Interfaces!\n");
434                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
435                         + " Process!");
436                 textArea.append("\n_____________________\n");
437                 key = 1;
438             }
```

Εικ. 4.19 Έλεγχος πλήθους διεπαφών.

```
439
440             else if ( mtIntBitRate <= bitRateThreshold )
441             {
442                 System.out.print(" Measurements check : Done\n");
443                 System.out.print(" Network termination triggered!");
444                 textArea.append("\n_____________________\n");
445                 textArea.append(" Timestamp : "+attempt+"\n");
446                 textArea.append(" Type      : Forced Termination due to bad QoS: "
447                         + "Low Bit-Rate!\n");
448                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
449                         + " Process!");
450                 textArea.append("\n_____________________\n");
451                 key = 1;
452             }
```

Εικ. 4.20 Έλεγχος ταχύτητας καναλιού.

```
453
454             else if ( mtIntDelay >= delayThreshold)
455             {
456                 System.out.print(" Measurements check : Done\n");
457                 System.out.print(" Network termination triggered!");
458                 textArea.append("\n_____________________\n");
459                 textArea.append(" Timestamp : "+attempt+"\n");
460                 textArea.append(" Type      : Forced Termination due to bad QoS: "
461                         + "Inappropriate Delay!\n");
462                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
463                         + " Process!");
464                 textArea.append("\n_____________________\n");
465                 key = 1;
466             }
```

Εικ. 4.21 Έλεγχος καθυστέρησης καναλιού.

```
467
468             else if ( mtBER >= berThreshold )
469             {
470                 System.out.print(" Measurements check : Done\n");
471                 System.out.print(" Network termination triggered!");
472                 textArea.append("\n_____________________\n");
473                 textArea.append(" Timestamp : "+attempt+"\n");
474                 textArea.append(" Type      : Forced Termination due to bad QoS: "
475                         + "High BER**\n");
476                 textArea.append(" Actions   : Release of Nodes  -  Start handover"
477                         + " Process!");
478                 textArea.append("\n_____________________\n");
479                 key = 1;
480             }
```

Εικ. 4.22 Έλεγχος ρυθμού εσφαλμένων bits.

Στην περίπτωση που καμία από τις παραπάνω συνθήκες δεν είναι αληθής, η τελευταία εναλλακτική λύση που προσφέρει η εντολή ελέγχου if else προμηνύει ότι το δίκτυο λειτουργεί κανονικά χωρίς προβλήματα (Εικ. 4.23 γραμμές 487-489). Αν λοιπόν ο μεταγλωττιστής εισέλθει στο βρόγχο της, είναι βέβαιο πως το σύστημα θα παραμένει ενεργό τουλάχιστον μέχρι τον επόμενο έλεγχό του, στέλνοντας νέες μετρήσεις προς τη βάση. Εφόσον λοιπόν η key δεν αλλάξει την τιμή της σε 1 η while θα ξαναεκτελεστεί. Η διαδικασία αυτή θα συνεχίζεται μέχρις ότου ο μεταγλωττιστής να εισέλθει στο βρόγχο ενός εκ των συνθηκών. Τότε η key θα παραμετροποιηθεί με την τιμή 1 δίνοντας το έναυσμα για έξοδο από την while και ουσιαστικά τον τερματισμό του OppNet.

```
481
482             else
483             {
484                 System.out.print(" Measurements check : Done\n\n");
485                 textArea.append("\n_____________________\n");
486                 textArea.append(" Timestamp : "+attempt+"\n");
487                 textArea.append(" The Opportunistic Network is still working "
488                         + "properly!\n");
489                 textArea.append(" No need for termination!");
490                 textArea.append("\n_____________________\n");
491             }
492
493           }//End of While
494         }//End of Try
```

Εικ. 4.23 Το δίκτυο δεν τερματίζεται και λειτουργεί κανονικά.

Μετά το τέλος της try ορίζονται οι εξαιρέσεις που αφορούν σφάλματα κατά την εύρεση του οδηγού της ή κατά την εγκατάσταση της σύνδεσης με τη βάση δεδομένων, τυπώνοντας και στις

δύο περιπτώσεις αντίστοιχο μήνυμα (Εικ. 4.24 γραμμές 497 και 503 αντίστοιχα). Τελευταία διεργασία της μεθόδου setVariables αποτελεί η αποσύνδεση από τη βάση (γραμμές 510-511).

```
495         catch ( ClassNotFoundException classNotFound )
496         {
497             JOptionPane.showMessageDialog( null, "MySQL driver not found",
498                     "Driver not found", JOptionPane.ERROR_MESSAGE );
499             System.exit( 1 );
500         }
501         catch ( SQLException sqlException )
502         {
503             JOptionPane.showMessageDialog( null, sqlException.getMessage(),
504               "Database error", JOptionPane.ERROR_MESSAGE );
505             System.exit( 1 );
506         }
507
508         try
509         {
510             statement.close();
511             connection.close();
512         }
513         catch ( SQLException sqlException )
514         {
515             sqlException.printStackTrace();
516         }
517     }//end of SetVariables
518
```

Εικ. 4.24 Εξαιρέσεις και αποσύνδεση από τη βάση.

4.3.2.2 Η μέθοδος DisplayResults

Η μέθοδος DisplayResults ακολουθεί αμέσως μετά τη setVariables στη ροή των εντολών της main. Ο ρόλος της είναι τριπλός αφού η πρώτη της διεργασία αφορά την ολοκλήρωση του γραφικού περιβάλλοντος της επερχόμενης διεπαφής. Κατά δεύτερον επανασυνδέει το πρόγραμμα με τη βάση η οποία τώρα πλέον, συμπεριλαμβάνει όλες τις μετρήσεις που προηγήθηκαν. Τρίτη και σπουδαιότερη έρχεται η δημιουργία κατάλληλων συστατικών διεπαφής τα οποία εκτελούν διάφορες ενέργειες για την πλήρη και ουσιαστική παρακολούθηση των αποτελεσμάτων.

Στην γραμμή 521 της εικόνας 4.25 ξεκινά μία try η οποία φτάνει μέχρι το τέλος της μεθόδου περιέχοντας και τις τρεις διεργασίες που παρουσιάστηκαν. Αρχικά λοιπόν, στις γραμμές 525-527 δηλώνονται τρία αντικείμενα της κλάσης JButton καθένα από τα οποία αποτελεί και ένα κουμπί διεπαφής. Στην παραμετροποίησή τους αποδίδεται το επιθυμητό όνομα. Επί της συνέχειας ορίζονται δύο ακόμη συστατικά (γραμμές 529-530), αυτή τη φορά αντικείμενα της κλάσης JTextField. Ο ρόλος των πεδίων αυτών είναι να δέχονται κάποια είσοδο από το πληκτρολόγιο.

Ακολούθως δημιουργούνται δύο ετικέτες (labels), οι οποίες συνοδεύουν τα πεδία TextFields για την προτροπή κατάλληλης εισόδου δεδομένων (γραμμές 532-533).

Έπειτα γίνεται χρήση της κλάσης JPanel για τη δημιουργία τριών πάνελ, εκ των οποίων το ένα χαρακτηρίζεται ως bigPanel γιατί θα συμπεριλάβει τα υπόλοιπα δύο στο εσωτερικό του (γραμμές 535-537). Συνεπώς το bigPanel θα πρέπει να χαρακτηρίζεται από έναν τύπο διάταξης για τη διαχείριση των συστατικών του. Αυτός θα είναι τύπου BorderLayout (γραμμή 539), ίδιος δηλαδή με τον τύπο που χρησιμοποιείται σε ολόκληρο το γραφικό περιβάλλον. Έτσι τοποθετούνται τα smallPanel1 και smallPanel2 στο δυτικό και ανατολικό κομμάτι αντίστοιχα (γραμμές 541-542).

Η εμφώλευση όμως δεν τελειώνει εδώ, αφού και τα μικρά πάνελ δημιουργούν και επιλέγουν το δικό τους τύπο διάταξης με την προοπτική να χρησιμοποιήσουν τα συστατικά που ορίστηκαν παραπάνω (κουμπιά, πεδία, ετικέτες). Αυτή τη φορά θα γίνει χρήση της διάταξης GridLayout, η οποία παίρνει ως ορίσματα το πλήθος γραμμών, το πλήθος στηλών και την απόσταση μεταξύ τους (γραμμές 544-545).

Συνεπώς στο smallPanel1 προσθέτονται τα συστατικά label1, textField1, label2 και textField2 ενώ στο smallPanel2 όλα τα κουμπιά. Όμως όλα αυτά τα συστατικά βρίσκονται πάνω στο bigPanel το οποίο με τη σειρά του θα τοποθετηθεί στο κεντρικό σημείο του γραφικού περιβάλλοντος (γραμμή 556) κάτω από το JTextArea το οποίο βρίσκεται από πριν στη βόρεια θέση.

```
521   public void DisplayResults() {
522
523       try
524       {
525         JButton button1 = new JButton( "Database" );
526         JButton button2 = new JButton( "Node ID" );
527         JButton button3 = new JButton( "Exit" );
528
529         textField1 = new JTextField();
530         textField2 = new JTextField();
531
532         JLabel label1 = new JLabel( "Enter attempt Number:" );
533         JLabel label2 = new JLabel( "Enter node Number:" );
534
535         JPanel bigPanel = new JPanel();
536         JPanel smallPanel1 = new JPanel();
537         JPanel smallPanel2 = new JPanel();
538
539         bigPanel.setLayout( new BorderLayout( 1, 1) );
540
541         bigPanel.add( smallPanel1, BorderLayout.WEST );
542         bigPanel.add( smallPanel2, BorderLayout.EAST );
543
544         smallPanel1.setLayout( new GridLayout( 2, 2 ,2 ,1 ) );
545         smallPanel2.setLayout( new GridLayout( 1, 3 ,2 ,1 ) );
546
547         smallPanel1.add( label1 );
548         smallPanel1.add( textField1 );
549         smallPanel1.add( label2 );
550         smallPanel1.add( textField2 );
551
552         smallPanel2.add( button1 );
553         smallPanel2.add( button2 );
554         smallPanel2.add( button3 );
555
556         add( bigPanel, BorderLayout.CENTER );
```

Εικ. 4.25 Δημιουργία συστατικών του γραφικού περιβάλλοντος.

Επί της συνέχειας δημιουργείται το αντικείμενο tableModel της κλάσης TableModelTermination διοχετεύοντας πέντε παραμέτρους στη συνάρτηση δημιουργίας της (Εικ. 4.26 γραμμή 558). Αυτές είναι ο οδηγός βάσης δεδομένων, το url της βάσης δεδομένων, το όνομα χρήστη, ο κωδικός πρόσβασης καθώς και το προεπιλεγμένο ερώτημα προς τη βάση. Με αυτόν τον τρόπο το πρόγραμμα θα επανασυνδεθεί με τη βάση δεδομένων.

Η γραμμή 561 της εικόνας 4.26 δημιουργεί το αντικείμενο resultTable της κλάσης Jtable και διοχετεύει ένα αντικείμενο TableModelTermination στη συνάρτηση δημιουργίας της, που δηλώνει κατόπιν το JTable ως αναμονή συμβάντος για TableModelEvent προκαλούμενα από την TableModelTermination.

Η υλοποίηση ενός πάνελ κύλισης (scrollable panel) tablePanel και η τοποθέτησή του στη νότια πλευρά της διάταξης BorderLayout (γραμμή 567) σηματοδοτεί την ολοκλήρωση του γραφικού περιβάλλοντος το οποίο εν τέλει προκύπτει με τα εξής συστατικά:

- JTextArea, BorderLayout.NORTH
- BigPanel, BorderLayout.CENTER
- JTable, BorderLayout.SOUTH

```
558         tableModel = new TableModelTermination( JDBC_DRIVER, DATABASE_URL,
559                 USERNAME, PASSWORD, DEFAULT_QUERY );
560
561         JTable resultTable = new JTable( tableModel );
562
563         JScrollPane tablePanel = new JScrollPane( resultTable,
564                 ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
565                 ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER );
566
567         add( tablePanel, BorderLayout.SOUTH );
```

Εικ. 4.26 Επανασύνδεση με τη βάση και δημιουργία tablePanel.

Οι γραμμές 569-587 της εικόνας 4.27, δηλώνουν ένα χειρισμό συμβάντος για το button1 στο οποίο ο χρήστης κάνει κλικ για να υποβάλλει ένα ερώτημα στη βάση δεδομένων. Όταν ο χρήστης κάνει κλικ στο κουμπί αυτό, η μέθοδος actionPerformed (γραμμή 574) εμπλέκει την TableModelTermination μέθοδο setQuery για να εκτελέσει ένα ερώτημα. Το ερώτημα που τίθεται προς τη βάση μέσω του πλήκτρου button1 είναι η επιλογή όλων των στοιχείων του πίνακα netData (γραμμή 578). Στην περίπτωση, κατά την οποία για κάποιο λόγο δεν υπάρχουν τα στοιχεία υπό ζήτηση, εμφανίζεται στην οθόνη ένα μήνυμα διαλόγου ενημερώνοντας ότι υπήρξε σφάλμα στη βάση.

```
569          button1.addActionListener(
570
571            new ActionListener()
572            {
573
574              public void actionPerformed( ActionEvent event )
575              {
576                try
577                {
578                   tableModel.setQuery( "SELECT * FROM netData" );
579                }
580                catch ( SQLException sqlException )
581                {
582                   JOptionPane.showMessageDialog( null, sqlException.getMessage(),
583                           "Database Error", JOptionPane.ERROR_MESSAGE );
584                }
585              }
586            }
587          );
```

Εικ. 4.27 Αναμονή συμβάντος για κουμπί 1.

Αντίστοιχα, οι γραμμές 589-607 της εικόνας 4.28 και 609-629 της εικόνας 4.29 δηλώνουν δύο χειρισμούς συμβάντος για τα button2 και button3. Όταν ο χρήστης επιλέξει να κάνει κλικ στο button2 τότε στέλνεται ένα ερώτημα στη βάση που ζητά όλα τα στοιχεία του πίνακα netNodes. Στην ίδια διαδικασία, για το button3, εμπλέκεται η TableModelTermination μέθοδος UpdateQuery η οποία διαγράφει όλα τα δεδομένα του πίνακα netData. Ύστερα καλεί τη μέθοδο disconnectFromDatabase η οποία τερματίζει τη σύνδεση με τη βάση. Η εντολή System.exit(1) (Εικ. 4.29 γραμμή 620), τερματίζει το σύστημα.

```
589          button2.addActionListener(
590
591            new ActionListener()
592            {
593
594              public void actionPerformed( ActionEvent event )
595              {
596                try
597                {
598                       tableModel.setQuery( "SELECT * FROM netNodes" );
599                }
600                catch ( SQLException sqlException )
601                {
602                   JOptionPane.showMessageDialog( null, sqlException.getMessage(),
603                          "Database Error", JOptionPane.ERROR_MESSAGE );
604                }
605              }
606            }
607          );
```

Εικ. 4.28 Αναμονή συμβάντος για κουμπί 2.

```
609          button3.addActionListener(
610
611            new ActionListener()
612            {
613
614              public void actionPerformed( ActionEvent event )
615              {
616                try
617                {
618                   tableModel.UpdateQuery("DELETE FROM netData WHERE nodeID < 9");
619                   tableModel.disconnectFromDatabase();
620                   System.exit(1);
621                }
622                catch ( SQLException sqlException )
623                {
624                   JOptionPane.showMessageDialog( null, sqlException.getMessage(),
625                          "Database Error", JOptionPane.ERROR_MESSAGE );
626                }
627              }
628            }
629          );
```

Εικ. 4.29 Αναμονή συμβάντος για κουμπί 3.

Στις γραμμές 631-657 της εικόνας 4.30 και 659-686 της εικόνας 4.31 δηλώνονται δύο νέοι χειρισμοί συμβάντων, αυτή τη φορά των συστατικών textField1 και textField2. Στο εσωτερικό της actionPerformed, αρχικά αποθηκεύεται σε μία μεταβλητή τύπου string η είσοδος από το πληκτρολόγιο (Εικ. 4.30 γραμμή 640 ,Εικ 4.31 γραμμή 668 αντίστοιχα). Αν αυτή είναι έγκυρη τότε για ακόμη μία φορά γίνεται χρήση της setQuery, η οποία ζητά από τη βάση όλα τα στοιχεία του

πίνακα netData που εκπληρώνουν τις προϋποθέσεις της συνθήκης «where». Σε διαφορετική περίπτωση εμφανίζεται μήνυμα σφάλματος προτρέποντας την έγκυρη εισαγωγή τιμής (Εικ. 4.30 γραμμές 649-651, Εικ. 4.31 γραμμές 677-679).

```
631         textField1.addActionListener(
632 
633           new ActionListener()
634           {
635 
636             public void actionPerformed( ActionEvent event )
637             {
638               try
639               {
640                   String attemptNumber = textField1.getText();
641 
642                   tableModel.setQuery( "SELECT * FROM netData WHERE attemptNum = "
643                           +attemptNumber+"" );
644 
645                   textField1.setText(" ");
646               }
647               catch ( SQLException sqlException )
648               {
649                   JOptionPane.showMessageDialog( null, "Invalid value!! Please "
650                           + "insert an attempt number", "Input Error" ,
651                           JOptionPane.ERROR_MESSAGE );
652 
653                   textField1.setText(" ");
654               }
655             }
656           }
657         );
```

Εικ. 4.30 Αναμονή συμβάντος για πεδίο κειμένου 1.

```
659         textField2.addActionListener(
660
661            new ActionListener()
662            {
663
664              public void actionPerformed( ActionEvent event )
665              {
666                try
667                {
668                   String nodeIdNum = textField2.getText();
669
670                   tableModel.setQuery( "SELECT * FROM netData WHERE nodeID ="
671                           + " "+nodeIdNum+"" );
672
673                   textField2.setText(" ");
674                }
675                catch ( SQLException sqlException )
676                {
677                   JOptionPane.showMessageDialog( null, "Invalid value!! Please "
678                           + "insert an nodeId number", "Input Error" ,
679                           JOptionPane.ERROR_MESSAGE );
680
681                   textField2.setText(" ");
682                }
683              }
684            }
685          );
686         }//End of Try
```

Εικ. 4.31 Αναμονή συμβάντος για πεδίο κειμένου 2.

Παρόμοια με το τέλος της μεθόδου setVariables, έτσι και στη DisplayResults ορίζονται οι εξαιρέσεις που αφορούν σφάλματα κατά την εύρεση του οδηγού της ή κατά την εγκατάσταση της σύνδεσης με τη βάση δεδομένων, τυπώνοντας και στις δύο περιπτώσεις ανάλογο μήνυμα (Εικ 4.32 γραμμές 691 και 697 αντίστοιχα).

```
688
689         catch ( ClassNotFoundException classNotFound )
690         {
691            JOptionPane.showMessageDialog( null, "MySQL driver not found",
692                    "Driver not found", JOptionPane.ERROR_MESSAGE );
693            System.exit(1);
694         }
695         catch ( SQLException sqlException )
696         {
697            JOptionPane.showMessageDialog( null, sqlException.getMessage(),
698               "Database error", JOptionPane.ERROR_MESSAGE );
699
700            tableModel.disconnectFromDatabase();
701            System.exit(1);
702         }
703      }//End of DisplayResults
704   }//End of Class TerminationModel
705
```

Εικ. 4.32 Εξαιρέσεις μεθόδου DisplayResults.

4.3.3 Η κλάση VariableChoice

Η κλάση VariableChoice δημιουργήθηκε για τη διεκπεραίωση ενός ιδιαίτερου σκοπού, αυτού της γέννησης τιμών για το δίκτυο. Έτσι ο κορμός της αποτελείται μόνο από μεθόδους οι οποίες επιστρέφουν τιμές επιλεγμένες μέσα από ένα καλά ορισμένο σύνολο. Όπως αναφέρθηκε στην TerminationModel μέθοδο setVariables, οι τιμές αυτές αποδίδονται στις αντίστοιχες μεταβλητές δικτύου (appStatus, gain κλπ), με τη χρήση αντικειμένων της.

Είναι χαρακτηριστικό λοιπόν ότι για τη γέννηση των τιμών δε χρησιμοποιείται άλλη παρά η κλάση Random (Εικ. 4.33 γραμμή 3). Με τη δημιουργία ενός αντικειμένου generator (γραμμή 11) καλούνται οι μέθοδοι nextInt και nextFloat. Κατά την πρώτη, ορίζεται ένα επιθυμητό σύνολο μέσα από το οποίο το σύστημα επιλέγει έναν τυχαίο ακέραιο αριθμό. Αντίθετα, με τη χρήση της δεύτερης, το σύνολο αυτό είναι προεπιλεγμένο μεταξύ του [0-1] και δεν παρέχεται η δυνατότητα μετατροπής του.

Είναι εμφανές λοιπόν ότι το σύνολο των μεταβλητών εκτός από τις gain και spectrum χρησιμοποιεί τη μέθοδο nextInt ενώ οι υπόλοιπες δύο τη nextFloat. Όμως, αξίζει να σημειωθεί ότι η επιλογή μεταξύ ακεραίων ή δεκαδικών αφορά μόνο τον τυχαίο τρόπο συλλογής τιμών και όχι την τελική μορφή με την οποία θα παρουσιαστούν στην οθόνη. Για παράδειγμα, η VariableChoice μέθοδος applicationStatusLevel επιστρέφει τις τιμές 0 ή 1, αλλά το σύνολο των τιμών που χρησιμοποιεί η μεταβλητή appStatus εν τέλει είναι το: [Starting, Established, Running, Finished].

```java
package TerminationExecute;

import java.util.Random;

/**
 *
 * @author Papadopoulos Sokratis
 */
public class VariableChoice {

    Random generator = new Random();

    public int applicationStatusLevel()
    {
        int statusLevel = generator.nextInt(2);
        return statusLevel;
    }

    public float gainValue()
    {
        float gain = generator.nextFloat();
        return gain;
    }

    public int energyValue()
    {
        int energyHigh = generator.nextInt(2);
        return energyHigh;
    }

    public int spectrumValue()
    {
        int spectrumBnwdth;
        float spectrum = generator.nextFloat();
            if ( spectrum <= 0.05 )
                spectrumBnwdth = 10;
            else if ( spectrum > 0.05 && spectrum <= 0.15 )
                spectrumBnwdth = 20;
            else if ( spectrum > 0.15 && spectrum <= 0.3 )
                spectrumBnwdth = 30;
            else if ( spectrum > 0.3 && spectrum <= 0.4 )
                spectrumBnwdth = 40;
            else if ( spectrum > 0.4 && spectrum <= 0.5 )
                spectrumBnwdth = 50;
            else if ( spectrum > 0.5 && spectrum <= 0.6 )
                spectrumBnwdth = 60;
            else if ( spectrum > 0.6 && spectrum <= 0.7 )
                spectrumBnwdth = 70;
            else if ( spectrum > 0.7 && spectrum <= 0.8 )
                spectrumBnwdth = 80;
```

Εικ. 4.33 Κλάση VariableChoice μέρος 1ο.

```
51              else if ( spectrum > 0.8 && spectrum <= 0.9 )
52                  spectrumBnwdth = 90;
53              else
54                  spectrumBnwdth = 100;
55          return spectrumBnwdth;
56      }
57
58      public int numberOfInterfaces()
59      {
60          int interfaces = generator.nextInt(5);
61          return interfaces;
62      }
63
64      public int bitRateValue()
65      {
66          int bitrate = generator.nextInt(50);
67          return bitrate;
68      }
69
70      public int delayValue()
71      {
72          int delay = generator.nextInt(31);
73          return delay;
74      }
75
76      public double bitErrorRateValue()
77      {
78          double bertest;
79          int ber = generator.nextInt(20);
80
81          if( ber<=4 )
82               return bertest = 0.005;
83          else if ( ber>4 && ber<=14 )
84               return  bertest = 0.01;
85          else if ( ber>14 && ber<=17 )
86               return bertest = 0.03;
87          else
88               return bertest = 0.05;
89      }
90  }
```

Εικ. 4.34 Κλάση VariableChoice μέρος 2°.

4.3.4 Η κλάση TableModelTermination

Η κλάση TableModelTermination εκτελεί τη σύνδεση στη βάση δεδομένων και διατηρεί το ResultSet το οποίο είναι απαραίτητο για την προβολή των αποτελεσμάτων στον πίνακα JTable. Χρησιμοποιώντας το αντικείμενο tableModel παρέχονται τα δεδομένα ResultSet στο JTable. Για να αποθηκευτούν τα δεδομένα στο ResultSet εκτελούνται ερωτήματα προς τη βάση, ερωτήματα τα

οποία πυροδοτούνται μέσω κατάλληλων συστατικών διεπαφής όπως περιγράφηκαν στην TerminationModel μέθοδο DisplayResults (buttons, textFields). Αφού πραγματοποιηθεί η εισαγωγή των αναγκαίων κλάσεων (Εικ 4.35 γραμμές 7-13) η TableModelTermination επεκτείνει την κλάση AbstractTableModel, που υλοποιεί το περιβάλλον TableModel (γραμμή 15). Μετά τη δήλωση ορισμένων αντικειμένων και μεταβλητών ακολουθεί η συνάρτηση δημιουργίας της κλάσης η οποία αφού εκτελέσει τη σύνδεση με τη βάση (γραμμή 32), θέτει το πρώτο ερώτημα με τη χρήση της setQuery (γραμμή 39).

```
1  package TerminationExecute;
2
3  /**
4   *
5   * @author Papadopoulos Sokratis
6   */
7  import java.sql.Connection;
8  import java.sql.Statement;
9  import java.sql.DriverManager;
10 import java.sql.ResultSet;
11 import java.sql.ResultSetMetaData;
12 import java.sql.SQLException;
13 import javax.swing.table.AbstractTableModel;
14
15 public class TableModelTermination extends AbstractTableModel {
16
17    private Connection connection;
18    private Statement statement;
19    private ResultSet resultSet;
20    private ResultSetMetaData metaData;
21    private int numberOfRows;
22
23    private boolean connectedToDatabase = false;
24
25    public TableModelTermination( String driver, String url, String username,
26            String password, String query ) throws SQLException,
27            ClassNotFoundException
28    {
29
30       Class.forName( driver );
31
32       connection = DriverManager.getConnection( url, username, password );
33
34       statement = connection.createStatement( ResultSet.TYPE_SCROLL_SENSITIVE,
35               ResultSet.CONCUR_READ_ONLY );
36
37       connectedToDatabase = true;
38
39       setQuery( query );
40    }
```

Εικ. 4.35 Κλάση TableModelTermination και η συνάρτηση δημιουργίας της.

Η κατάθεση ερωτημάτων προς τη βάση επιτυγχάνεται με τις setQuery και UpdateQuery μεθόδους. Η πρώτη (Εικ. 4.36) αφού ελέγξει τη σύνδεση με τη βάση, διοχετεύει το ερώτημα στην executeQuery και αποθηκεύει τα αποτελέσματα στην resultSet. Ύστερα η getRow, επιστρέφει το πλήθος των γραμμών που συμπεριελάμβαναν τα εμπλεκόμενα στοιχεία. Απεναντίας η μέθοδος UpdateQuery (Εικ 4.37) δε θέτει κάποιο ερώτημα στη βάση, αλλά τροποποιεί και ανανεώνει τα συστατικά της μέσω της executeUpdate (γραμμή 63).

```
42      public void setQuery(String query) throws SQLException,IllegalStateException
43      {
44         if ( !connectedToDatabase )
45            throw new IllegalStateException( "Not Connected to Database" );
46
47         resultSet = statement.executeQuery( query );
48
49         metaData = resultSet.getMetaData();
50
51         resultSet.last();
52         numberOfRows = resultSet.getRow();
53
54         fireTableStructureChanged();
55      }
```

Εικ. 4.36 Η μέθοδος SetQuery.

```
57      public void UpdateQuery( String query ) throws SQLException,
58              IllegalStateException
59      {
60         if ( !connectedToDatabase )
61            throw new IllegalStateException( "Not Connected to Database" );
62
63         statement.executeUpdate( query );
64      }
```

Εικ. 4.37 Η μέθοδος UpdateQuery.

Στη γραμμή 73 της εικόνας 4.38 η μέθοδος getColumnCount παίρνει το πλήθος των στηλών και έπειτα το επιστρέφει στο resultSet, ενώ αντίστοιχα η getRowCount (Εικ. 4.39 γραμμή 87) επιστρέφει το πλήθος των σειρών. Αλλά και ο ρόλος της getValueAt (Εικ. 4.40) είναι πολύ σημαντικός αφού παίρνει την τιμή σε συγκεκριμένη σειρά και στήλη με απώτερο σκοπό την επιστροφή και αποθήκευσή της (Εικ. 4.40 γραμμές 97-98).

```
66      public int getColumnCount() throws IllegalStateException
67      {
68        if ( !connectedToDatabase )
69          throw new IllegalStateException( "Not Connected to Database" );
70
71        try
72        {
73           return metaData.getColumnCount();
74        }
75        catch ( SQLException sqlException )
76        {
77           sqlException.printStackTrace();
78        }
79        return 0;
80      }
```

Εικ. 4.38 Η μέθοδος getColumnCount.

```
82      public int getRowCount() throws IllegalStateException
83      {
84          if ( !connectedToDatabase )
85           throw new IllegalStateException( "Not Connected to Database" );
86
87          return numberOfRows;
88      }
```

Εικ. 4.39 Η μέθοδος getRowCount.

```
90      public Object getValueAt( int row, int column ) throws IllegalStateException
91      {
92          if ( !connectedToDatabase )
93           throw new IllegalStateException( "Not Connected to Database" );
94
95          try
96          {
97              resultSet.absolute( row +1 );
98              return resultSet.getObject( column + 1 );
99          }
100         catch ( SQLException sqlException )
101         {
102             sqlException.printStackTrace();
103         }
104         return "";
105     }
```

Εικ. 4.40 Η μέθοδος getValueAt.

Η μέθοδος getColumnName (Εικ. 4.41) που ακολουθεί, διοχετεύει το όνομα μίας συγκεκριμένης στήλης στο resultSet (γραμμή 114). Σε περίπτωση που υπάρξει κάποιο πρόβλημα επιστρέφει κενή string για όνομα στήλης (γραμμή 120). Στη συνέχεια η GetColumnClass (Εικ. 4.42) παίρνει την κλάση που αναπαριστά τύπο στήλης. Τελευταία δηλώνεται η μέθοδος disconnectFromDatabase

(Εικ. 4.43), η οποία τερματίζει τη σύνδεση θέτοντας τη μεταβλητή-φρουρό connectedToDatabase ως ψευδή (γραμμή 157).

```
107     public String getColumnName( int column ) throws IllegalStateException
108     {
109       if ( !connectedToDatabase )
110          throw new IllegalStateException( "Not Connected to Database" );
111
112       try
113       {
114          return metaData.getColumnName( column + 1 );
115       }
116       catch ( SQLException sqlException )
117       {
118          sqlException.printStackTrace();
119       }
120       return "";
121     }
```

Εικ. 4.41 Η μέθοδος getColumnName.

```
123     public Class getColumnClass( int column ) throws IllegalStateException
124     {
125        if ( !connectedToDatabase )
126          throw new IllegalStateException( "Not Connected to Database" );
127
128        try
129        {
130           String className = metaData.getColumnClassName( column + 1 );
131
132           return Class.forName( className );
133        }
134        catch ( Exception exception )
135        {
136          exception.printStackTrace();
137        }
138        return Object.class;
139      }
```

Εικ. 4.42 Η μέθοδος getColumnClass.

```
141     public void disconnectFromDatabase()
142     {
143         if ( !connectedToDatabase )
144            return;
145
146         try
147         {
148             statement.close();
149             connection.close();
150         }
151         catch ( SQLException sqlException )
152         {
153              sqlException.printStackTrace();
154         }
155         finally
156         {
157              connectedToDatabase = false;
158         }
159     }
160   }//end of Class TableModelTermination
```

Εικ. 4.43 Η μέθοδος disconnectFromDatabase.

4.4 Εκτέλεση και αποτελέσματα σεναρίου

Στην ενότητα αυτή περιγράφονται τα στάδια εκτέλεσης του προγράμματος που αναλύθηκε παραπάνω μέχρι και την εμφάνιση των αποτελεσμάτων του στο γραφικό περιβάλλον. Κατά το έναυσμα της εκτέλεσης ενεργοποιείται ο MySQL Community Server στη θύρα (port) 3306 με την εντολή mysqld - -console (Εικ. 4.44).

```
C:\Users\inZi>cd c:/mysql/bin

c:\mysql\bin>mysqld --console
111006 17:12:46 [Note] Plugin 'FEDERATED' is disabled.
111006 17:12:46 InnoDB: The InnoDB memory heap is disabled
111006 17:12:46 InnoDB: Mutexes and rw_locks use Windows interlocked functions
111006 17:12:46 InnoDB: Compressed tables use zlib 1.2.3
111006 17:12:46 InnoDB: Initializing buffer pool, size = 128.0M
111006 17:12:46 InnoDB: Completed initialization of buffer pool
111006 17:12:47 InnoDB: highest supported file format is Barracuda.
InnoDB: Log scan progressed past the checkpoint lsn 5012282
111006 17:12:47  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Restoring possible half-written data pages from the doublewrite
InnoDB: buffer...
InnoDB: Doing recovery: scanned up to log sequence number 5017571
111006 17:12:48  InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percents: 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
InnoDB: Apply batch completed
111006 17:12:48  InnoDB: Waiting for the background threads to start
111006 17:12:49 InnoDB: 1.1.5 started; log sequence number 5017571
111006 17:12:50 [Note] Event Scheduler: Loaded 0 events
111006 17:12:50 [Note] mysqld: ready for connections.
Version: '5.5.10'  socket: ''  port: 3306  MySQL Community Server (GPL)
```

Εικ. 4.44 Ενεργοποίηση MySQL Community Server στη θύρα 3306.

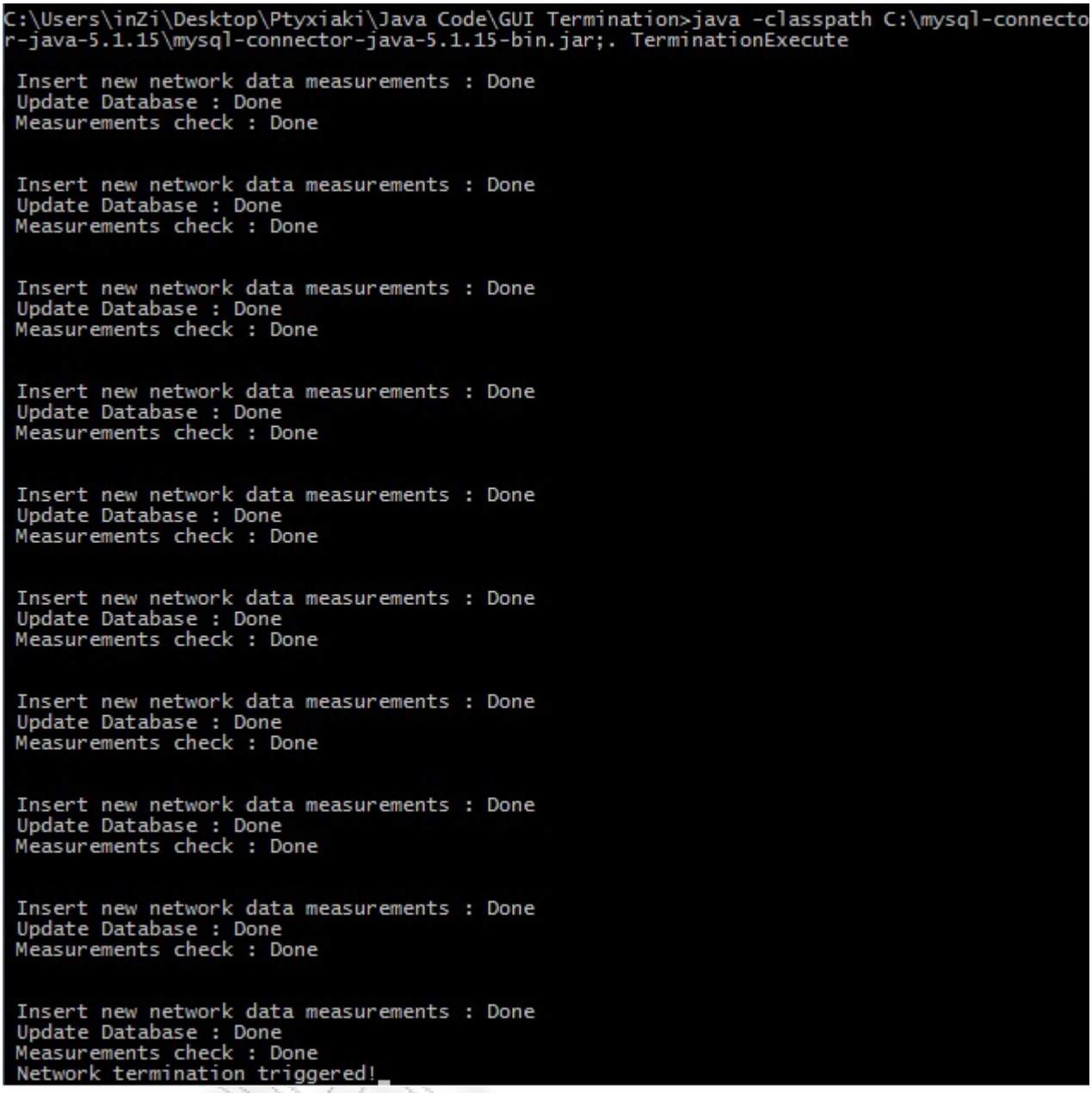
Εφόσον τώρα ο server είναι ενεργός, όλα είναι έτοιμα για την εκτέλεση του προγράμματος και τη σύνδεση με τη βάση δεδομένων. Η εντολή που παρατίθεται (Εικ. 4.45) θέτει το classpath στο μονοπάτι του connector έτσι ώστε να εντοπιστεί η τοποθεσία του στο σκληρό δίσκο. Ο java connector όπως έχει προαναφερθεί είναι απαραίτητο στοιχείο για τη συνεργασία της java με τον MySQL Server. Τέλος επιλέγεται το java αρχείο προς εκτέλεση το οποίο φυσικά περιλαμβάνει και τη μέθοδο main (TerminationExecute).

```
C:\Users\inZi>cd C:\Users\inZi\Desktop\Ptyxiaki\Java Code\GUI Termination

C:\Users\inZi\Desktop\Ptyxiaki\Java Code\GUI Termination>java -classpath C:\mysql-connecto
r-java-5.1.15\mysql-connector-java-5.1.15-bin.jar;. TerminationExecute
```

Εικ. 4.45 Εντολή εκτέλεσης προγράμματος Java.

Μόλις εκτελεστεί η παραπάνω εντολή το πρόγραμμα ξεκινά δημιουργώντας σύνδεση με τη βάση δεδομένων oNetResults. Παράλληλα παρακολουθεί τις τιμές του δικτύου κάθε χρονική στιγμή. Όσο η παρακολούθηση αυτή δε διακόπτεται από κάτι απρόοπτο οι ενδείξεις «Done» στη γραμμή εντολών σηματοδοτούν την ορθή λειτουργία του OppNet για κάθε στιγμιότυπο (Εικ. 4.46).

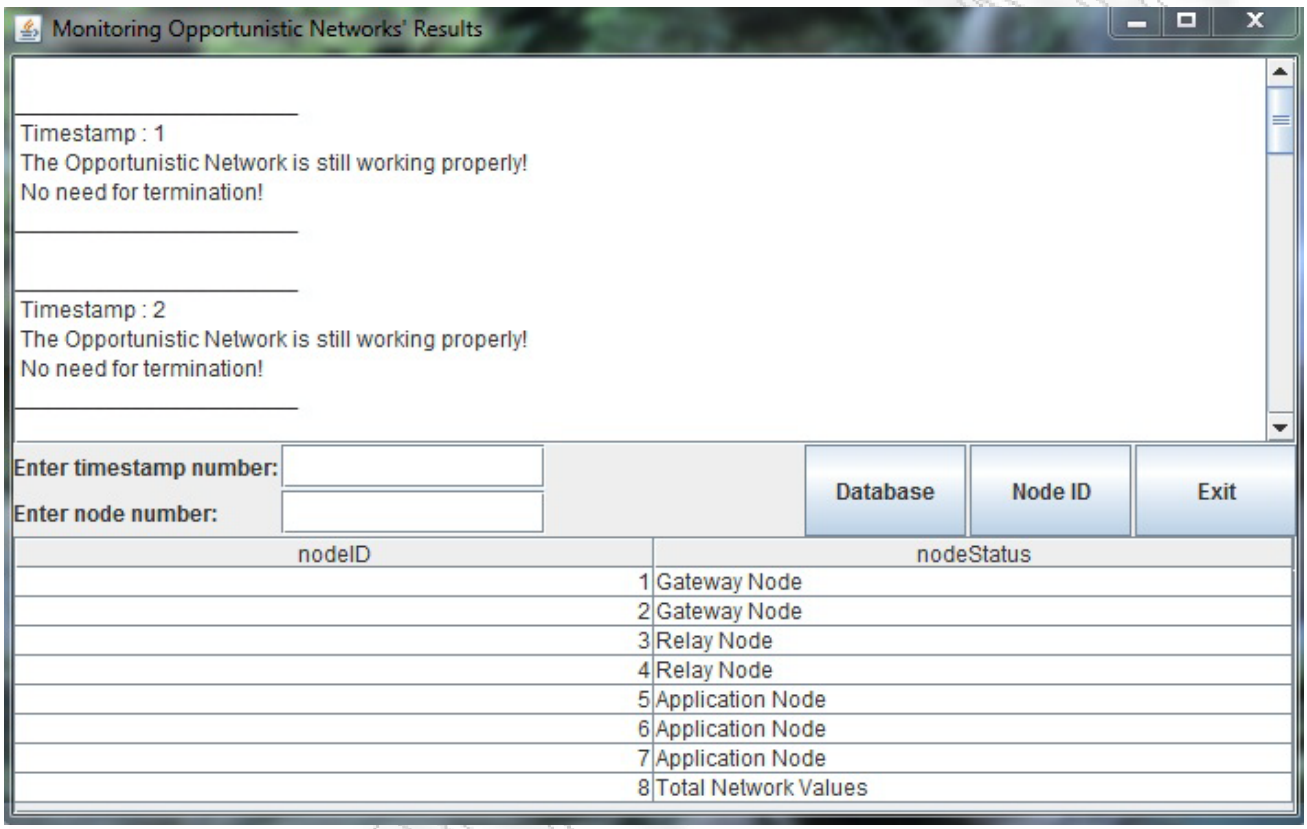


Εικ. 4.46 Από τη γραμμή εντολών παρακολουθείται το δίκτυο.

Όταν όμως εμφανιστεί το μήνυμα «Network Termination Triggered» τότε η λήξη του OppNet είναι γεγονός και αμέσως εμφανίζεται το γραφικό περιβάλλον java της εικόνας 4.47.

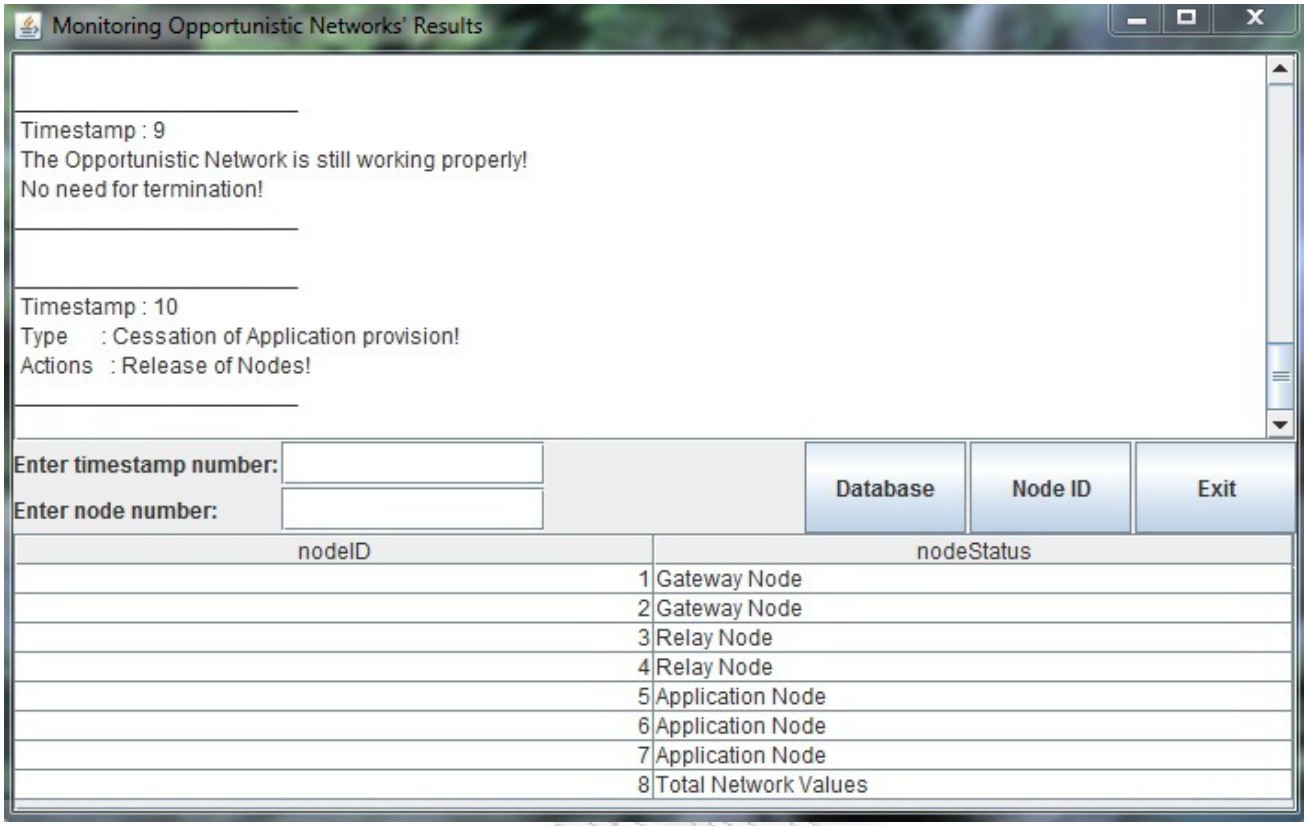
Στο βόρειο κομμάτι του γραφικού περιβάλλοντος καταγράφονται στην περιοχή κειμένου τα χρονικά στιγμιότυπα (timestamp 1, timestamp 2) κατά τα οποία το δίκτυο ελέγχθηκε. Ακριβώς από κάτω και αριστερά είναι τοποθετημένα δύο πεδία κειμένου συνοδευόμενα από ετικέτες, τα οποία προτρέπουν την εισαγωγή μίας τιμής, ενώ στα δεξιά τρία πλήκτρα με τις ονομασίες τους εγγεγραμμένες. Στο κάτω μέρος της οθόνης σχηματίζονται οι πίνακες της βάσης, προβάλλοντας τα δεδομένα όπως ακριβώς αυτά αποθηκεύονται στο εσωτερικό της. Προεπιλεγμένος είναι ο πίνακας NetNodes, ο οποίος αποτελείται από δύο στήλες και οχτώ γραμμές, αφού πρόκειται για τον

πίνακα των κόμβων (7 κόμβοι και ένας για τις συνολικές τιμές δικτύου). Η προεπιλογή αυτή, πηγάζει από την ανάγκη άμεσης παρουσίασης της αντιστοίχισης μεταξύ ταυτότητας και τύπου κόμβων κατά την εμφάνιση του γραφικού περιβάλλοντος (δηλαδή των nodeID και nodeStatus). Έτσι ο επιβλέπων του συστήματος, διαθέτει μία γρήγορη εικόνα περί του ρόλου των κόμβων.



Εικ. 4.47 Το γραφικό περιβάλλον της java αμέσως μετά τη λήξη του OppNet.

Μία από τις δυνατότητες που προσφέρει το γραφικό περιβάλλον στο διαχειριστή του συστήματος είναι η ταχεία αναγνώριση του τύπου τερματισμού. Αν στην περιοχή κειμένου κάνει κύλιση προς τα κάτω, είναι εμφανές ότι στην προκειμένη περίπτωση το δίκτυο ολοκλήρωσε τον κύκλο λειτουργίας του κατά το δέκατο έλεγχο λόγω λήξης της παροχής υπηρεσιών (Εικ. 4.48). Έτσι συνεπάγεται η απελευθέρωση των πόρων ως ακόλουθη πράξη.



Εικ. 4.48 Άμεση παρατήρηση του λόγου τερματισμού.

Όμως η απλή εμφάνιση ενός συμβάντος τερματισμού δεν είναι ικανοποιητική σε τέτοιο επίπεδο. Στο διαχειριστή θα πρέπει να δίνεται η δυνατότητα λεπτομερέστερης αναπαράστασης των μετρήσεων του OppNet. Για αυτό το λόγο, έχει τοποθετηθεί το πλήκτρο «Database» το οποίο προβάλλει όλα τα δεδομένα που καταγράφονταν όσο το δίκτυο ήταν ενεργό (Εικ 4.49). Τα δεδομένα αυτά συνθέτουν έναν πίνακα netData ο οποίος διαθέτει δέκα στήλες εκ των οποίων η πρώτη αφορά την ταυτότητα των κόμβων και η τελευταία το χρόνο που εισήχθησαν σε αυτή. Οι υπόλοιπες αφορούν τις μεταβλητές του δικτύου στα πλαίσια των τιμών όπως αυτά παρουσιάστηκαν στην ενότητα 4.3.2.1.

Monitoring Opportunistic Networks' Results

Timestamp : 9
The Opportunistic Network is still working properly!
No need for termination!

Timestamp : 10
Type : Cessation of Application provision!
Actions : Release of Nodes!

Enter timestamp number:
Enter node number:
Database | Node ID | Exit

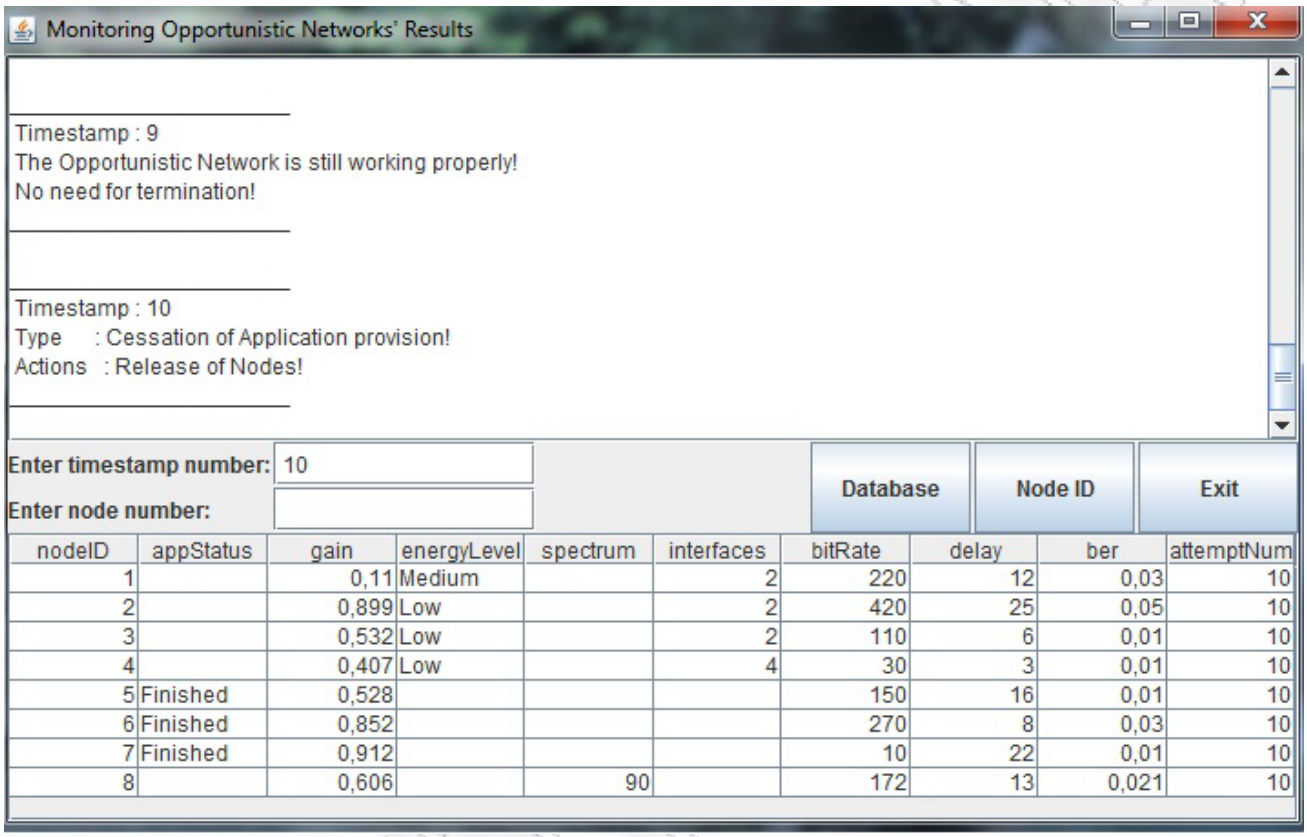
nodeID	appStatus	gain	energyLevel	spectrum	interfaces	bitRate	delay	ber	attemptNum
1		0,103	Maximum		2	190	22	0,01	1
2		0,822	Maximum		1	10	19	0,005	1
3		0,256	Maximum		3	140	0	0,05	1
4		0,445	Maximum		2	60	12	0,01	1
5	Starting	0,142				140	23	0,05	1
6	Starting	0,239				500	26	0,01	1
7	Starting	0,452				400	7	0,03	1
8		0,351		70		205	15	0,024	1
1		0,121	Maximum		1	360	29	0,01	2
2		0,765	High		4	190	4	0,01	2
3		0,272	High		4	190	15	0,01	2
4		0,749	High		4	30	2	0,01	2
5	Established	0,154				440	24	0,005	2
6	Starting	0,613				200	27	0,01	2
7	Established	0,639				10	5	0,005	2
8		0,473		30		202	15	0,009	2
1		0,228	Maximum		0	340	20	0,01	3
2		0,152	High		3	170	30	0,05	3
3		0,508	Medium		3	240	2	0,005	3
4		0,947	Medium		4	470	15	0,01	3
5	Established	0,344				100	6	0,01	3
6	Established	0,595				480	21	0,01	3
7	Established	0,17				70	7	0,01	3
8		0,421		60		267	14	0,015	3
1		0,503	Maximum		1	250	18	0,005	4

Εικ. 4.49 Αναλυτική παρουσίαση των μετρήσεων του δικτύου.

Στις περιπτώσεις που το OppNet αποτελείται από πολλαπλάσιους σε σχέση με το σενάριο αυτό κόμβους, η εμφάνιση όλων των αποτελεσμάτων δε θα είναι αποδοτική. Συνεπώς παρουσιάζεται η ανάγκη φιλτραρίσματος με βάση ορισμένα στοιχεία. Το συγκεκριμένο γραφικό περιβάλλον υποστηρίζει τέτοιους μηχανισμούς φιλτραρίσματος με τη χρήση των πεδίων κειμένου (textFields). Το φιλτράρισμα πραγματοποιείται είτε με βάση το χρονικό στιγμιότυπο είτε με βάση τον κόμβο.

Αν λοιπόν ο διαχειριστής εισάγει από το πληκτρολόγιο την τιμή «10» στο πεδίο «timestamp number» και πατήσει enter, αυτόματα θα εμφανιστεί πίνακας που περιλαμβάνει όλες τις τιμές που ελήφθησαν το χρονικό στιγμιότυπο (attemptNum στον πίνακα) 10 (Εικ 4.50). Εφόσον αυτό

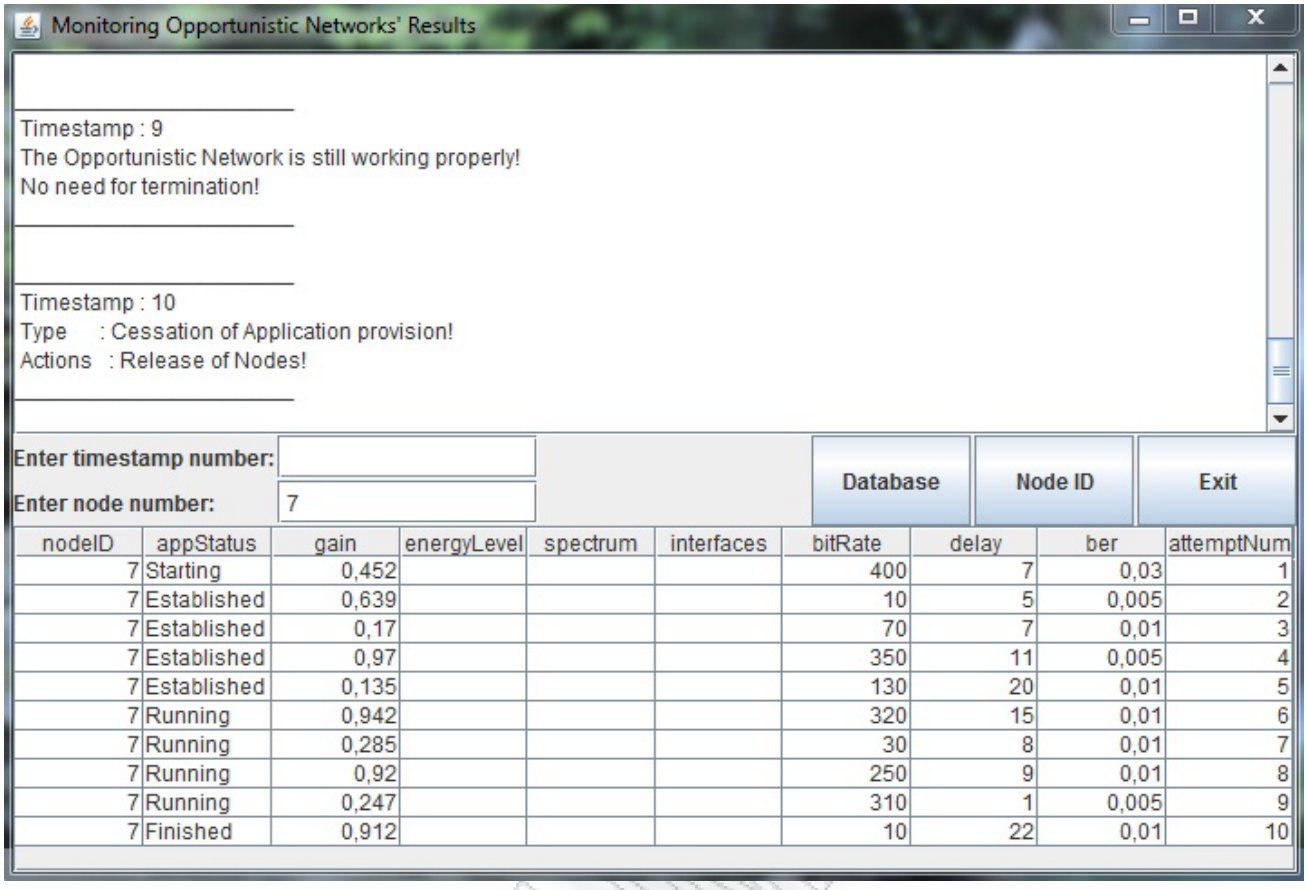
είναι και το τελευταίο κατ' αυτή την εκτέλεση, ο επιβλέπων θα παρατηρήσει και το λόγο τερματισμού του OppNet, που δεν είναι άλλος παρά οι τρεις ενδείξεις «Finished» στο πεδίο appStatus. Σε αντίθεση με αυτές, όλες οι υπόλοιπες μετρήσεις του δικτύου φαίνονται φυσιολογικές και άνω των επιτρεπτών ορίων. Για παράδειγμα το energyLevel των κόμβων 1 έως 4 είναι επιτρεπτό, εφόσον ο κόμβος 1 έχει ακόμη τιμή Medium.



nodeID	appStatus	gain	energyLevel	spectrum	interfaces	bitRate	delay	ber	attemptNum
1		0,11	Medium		2	220	12	0,03	10
2		0,899	Low		2	420	25	0,05	10
3		0,532	Low		2	110	6	0,01	10
4		0,407	Low		4	30	3	0,01	10
5	Finished	0,528				150	16	0,01	10
6	Finished	0,852				270	8	0,03	10
7	Finished	0,912				10	22	0,01	10
8		0,606		90		172	13	0,021	10

Εικ. 4.50 Φιλτράρισμα με βάση το χρονικό στιγμιότυπο.

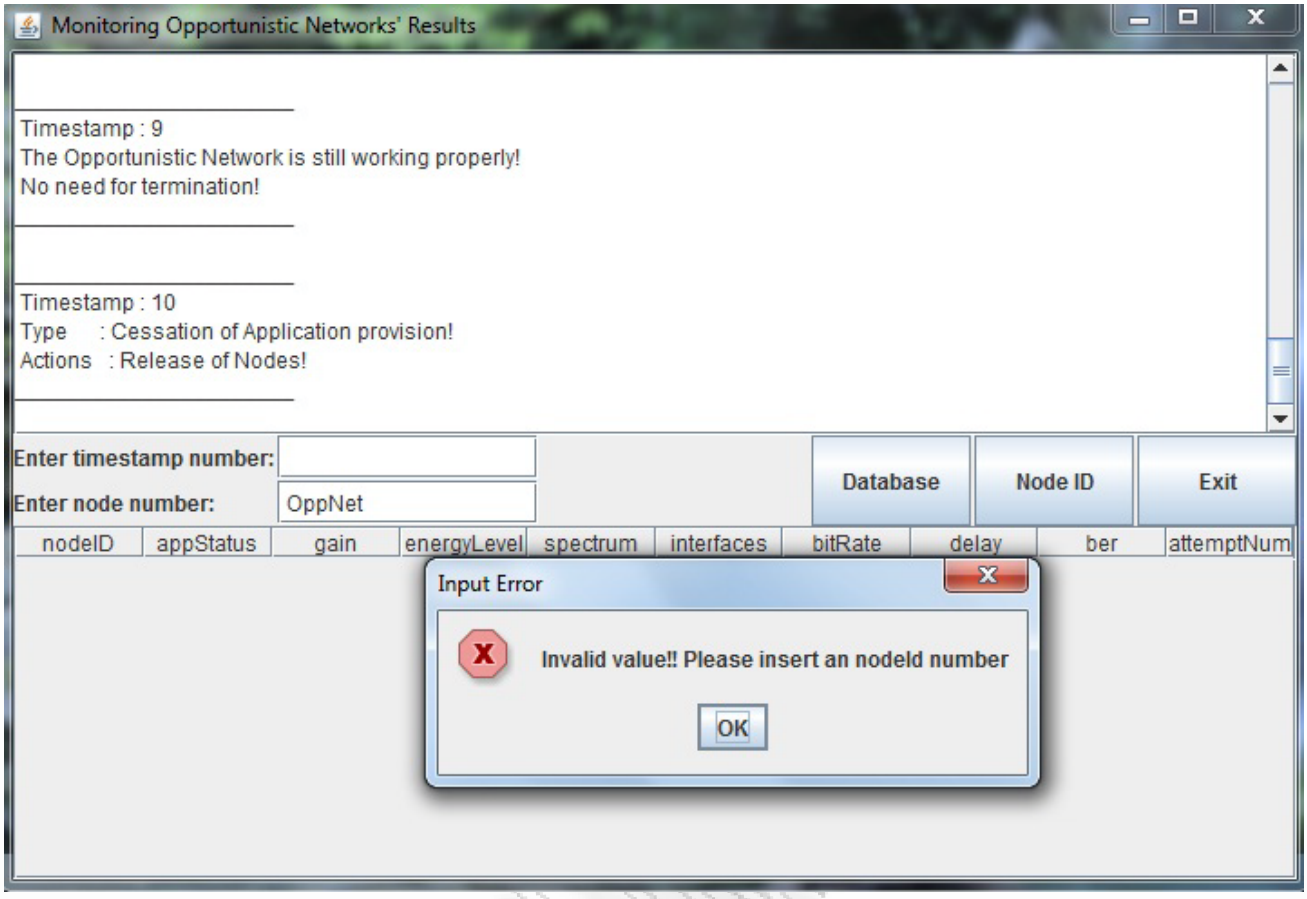
Το επόμενο φιλτράρισμα λαμβάνει χώρα στο δεύτερο πεδίο της διεπαφής, όταν ο χρήστης εισάγει τη ταυτότητα του κόμβου. Αν λοιπόν η τιμή αυτή είναι το «7», οι τιμές που εμφανίζονται στην οθόνη αφορούν μόνο εκείνες του έβδομου κόμβου. Έτσι παρατηρείται για παράδειγμα η κλιμακωτή πτώση της κατάστασης εφαρμογής, η οποία κατά την έναρξη του δικτύου βρισκόταν στο «Starting» και με το τέλος της λειτουργίας του χαρακτηρίζεται ως «Finished» (Εικ. 4.51).



nodeID	appStatus	gain	energyLevel	spectrum	interfaces	bitRate	delay	ber	attemptNum
7	Starting	0,452				400	7	0,03	1
7	Established	0,639				10	5	0,005	2
7	Established	0,17				70	7	0,01	3
7	Established	0,97				350	11	0,005	4
7	Established	0,135				130	20	0,01	5
7	Running	0,942				320	15	0,01	6
7	Running	0,285				30	8	0,01	7
7	Running	0,92				250	9	0,01	8
7	Running	0,247				310	1	0,005	9
7	Finished	0,912				10	22	0,01	10

Εικ. 4.51 Φιλτράρισμα με βάση τον κόμβο.

Αξίζει να σημειωθεί ότι το πρόγραμμα έχει προβλέψει τυχόν εσφαλμένους τύπους ή τιμές εισόδου από το πληκτρολόγιο και εμφανίζει τις ανάλογες προειδοποιήσεις. Αν η τιμή εισαγωγής στο πρώτο πεδίο είναι «11» τότε η βάση δε θα επιστρέψει κανένα αποτέλεσμα εφόσον δεν υπάρχει τέτοια τιμή. Επίσης αν εισαχθεί τιμή χαρακτήρων όπως η «OppNet» τότε θα ακολουθήσει το προειδοποιητικό μήνυμα της εικόνας 4.52.



Εικ. 4.52 Προειδοποιητικό μήνυμα εσφαλμένης τιμής εισόδου.

Τα εναπομείναντα δύο πλήκτρα «Node ID» και «Exit» αναμένουν συμβάντα, το πρώτο για την επαναφορά του πίνακα netNodes και το δεύτερο για την έξοδο από το γραφικό περιβάλλον. Όμως, ο ρόλος του «Exit» είναι τριπλός αφού πέρα από τη βασική του διεργασία, εκτελεί ακόμη δύο ενέργειες. Αρχικά διαγράφει όλα τα δεδομένα της βάσης και έπειτα τερματίζει τη σύνδεση. Με την επιλογή του «Exit» το πρόγραμμα φτάνει στο πέρας του.

Κεφάλαιο 5
Προσομοίωση Δικτύου

5.1 Προσομοίωση δικτύου μέσω του Omnet++

Η εκτέλεση του προγράμματος της java βασιζόμενη στον αλγόριθμο τερματισμού ενός δικτύου OppNet, προσφέρει σημαντικά αποτελέσματα αφού παρέχει τη δυνατότητα παρατήρησης όλων των μεταβλητών δικτύου μέχρι τη λήξη του. Παρόλα αυτά, ο κώδικας της java δεν είναι τόσο ισχυρός ώστε να δημιουργήσει ένα πραγματικό εικονιζόμενο δίκτυο. Αυτό είναι φυσικό, αφού ο λόγος δημιουργίας του αφορά τον έλεγχο λειτουργικότητας του αλγορίθμου και όχι τη γέννηση μίας δικτυακής τοπολογίας.

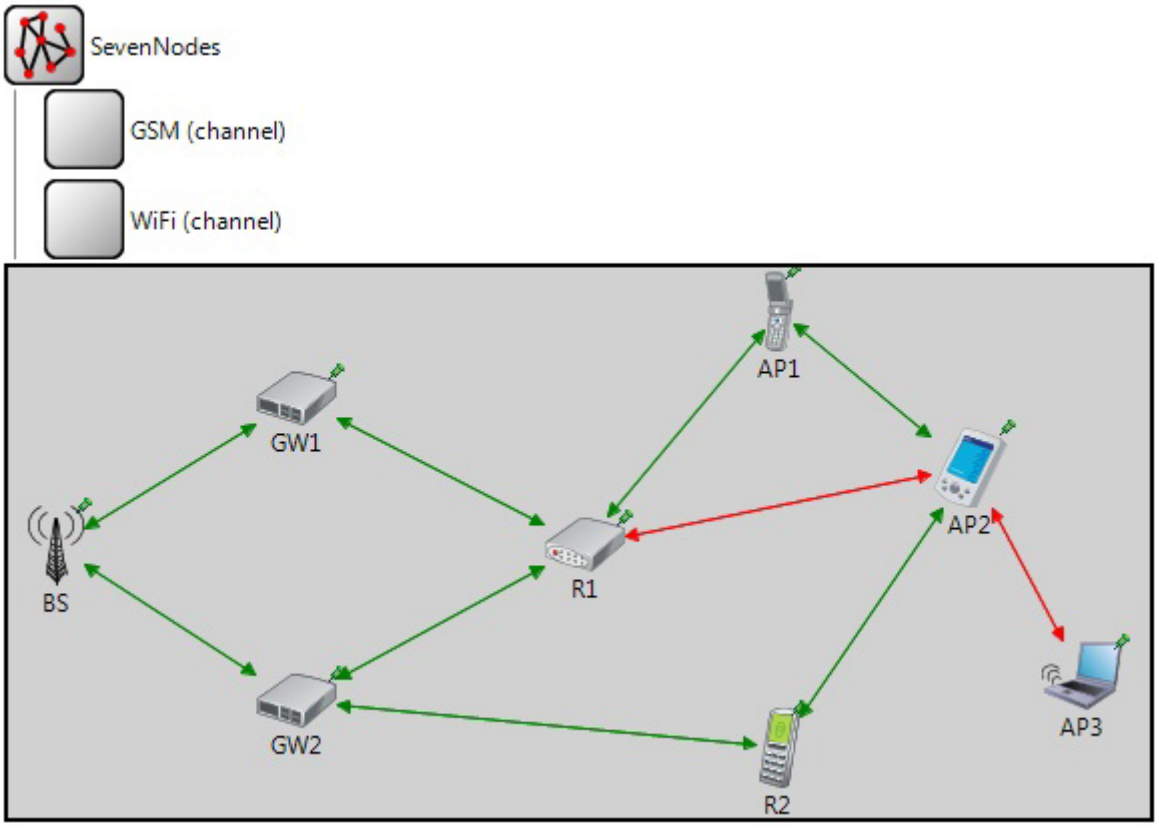
Συνεπώς κρίνεται αναγκαία η παρουσία μίας ζωντανής τοπολογίας η οποία θα τονίζει τη δομή του OppNet καθώς και τον τύπο των κόμβων του, υποστηρίζοντας με αυτόν τον τρόπο το ρόλο του καθενός. Η ιδέα αυτή καθίσταται υλοποιήσιμη με τη χρήση του προσομοιωτή Omnet++.

Με τον προσομοιωτή Omnet++ θα δοθεί έμφαση στο πιο σημαντικό χαρακτηριστικό του δικτύου, το επίπεδο ενέργειας των κόμβων. Το γεγονός αυτό οφείλεται στον ασύρματο χαρακτήρα τους και στη διασπορά που παρουσιάζουν οι μετρήσεις με τη χρήση διαφορετικών σεναρίων. Για αυτό το λόγο στις ενότητες που ακολουθούν γίνεται μία εμπεριστατωμένη μελέτη σε τρία διαφορετικά σενάρια OppNets.

Τα σενάρια αυτά καταγράφουν το ενεργειακό επίπεδο όλων των κόμβων ανεξαρτήτως τύπου, σε αντίθεση με τον κώδικα java. Αυτό συμβαίνει διότι στις τοπολογίες που θα «χτιστούν» υπάρχουν κόμβοι εφαρμογής οι οποίοι δεν είναι τελικοί χρήστες, και έτσι προωθούν πακέτα. Από την άλλη πλευρά στο πρόγραμμα που παρουσιάστηκε, θεωρείται ότι όλοι οι κόμβοι εφαρμογής ήταν τελικοί χρήστες με αποτέλεσμα να μην είναι αναγκαία η παρακολούθηση των τιμών ενέργειάς τους.

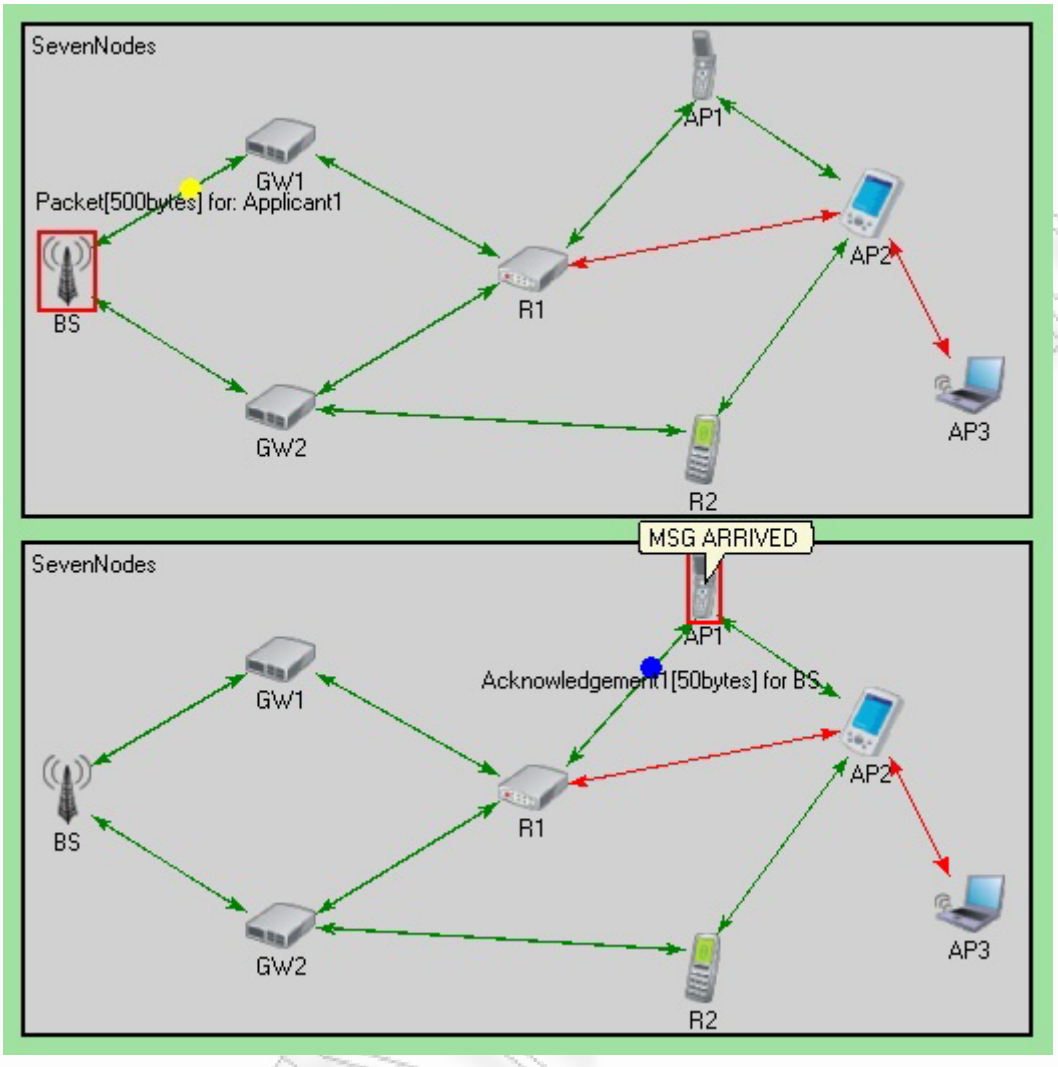
5.2 Σενάριο 1 – Δίκτυο 7 κόμβων

Το σενάριο που ακολουθεί δημιουργεί μία τοπολογία δικτύου OppNet με την ονομασία SevenNodes (Εικ. 5.1) περιλαμβάνοντας 2 κόμβους πύλης, 2 κόμβους αναμετάδοσης και 3 κόμβους εφαρμογών. Ακόμη διαθέτει δύο τύπους καναλιών, ένα GSM και ένα Wi-Fi. Η τεχνολογία Wi-Fi χρησιμοποιείται για τις συνδέσεις μεταξύ του δρομολογητή R1 και του smartphone AP2 ο οποίος με τη σειρά του συνδέεται με το laptop AP3. Οι υπόλοιπες συσκευές αποτελούν κινητά τηλέφωνα, μεταγωγείς, δρομολογητές και χρησιμοποιούν όλες συνδέσεις GSM. Τα στοιχεία των καναλιών χρησιμοποιούν ίδιες τιμές στο σύνολο των χαρακτηριστικών τους (Delay, BER, Bit-Rate), έτσι ώστε να μην επηρεάζουν το χρόνο προσομοίωσης ως προς τα τελικά συμπεράσματα.



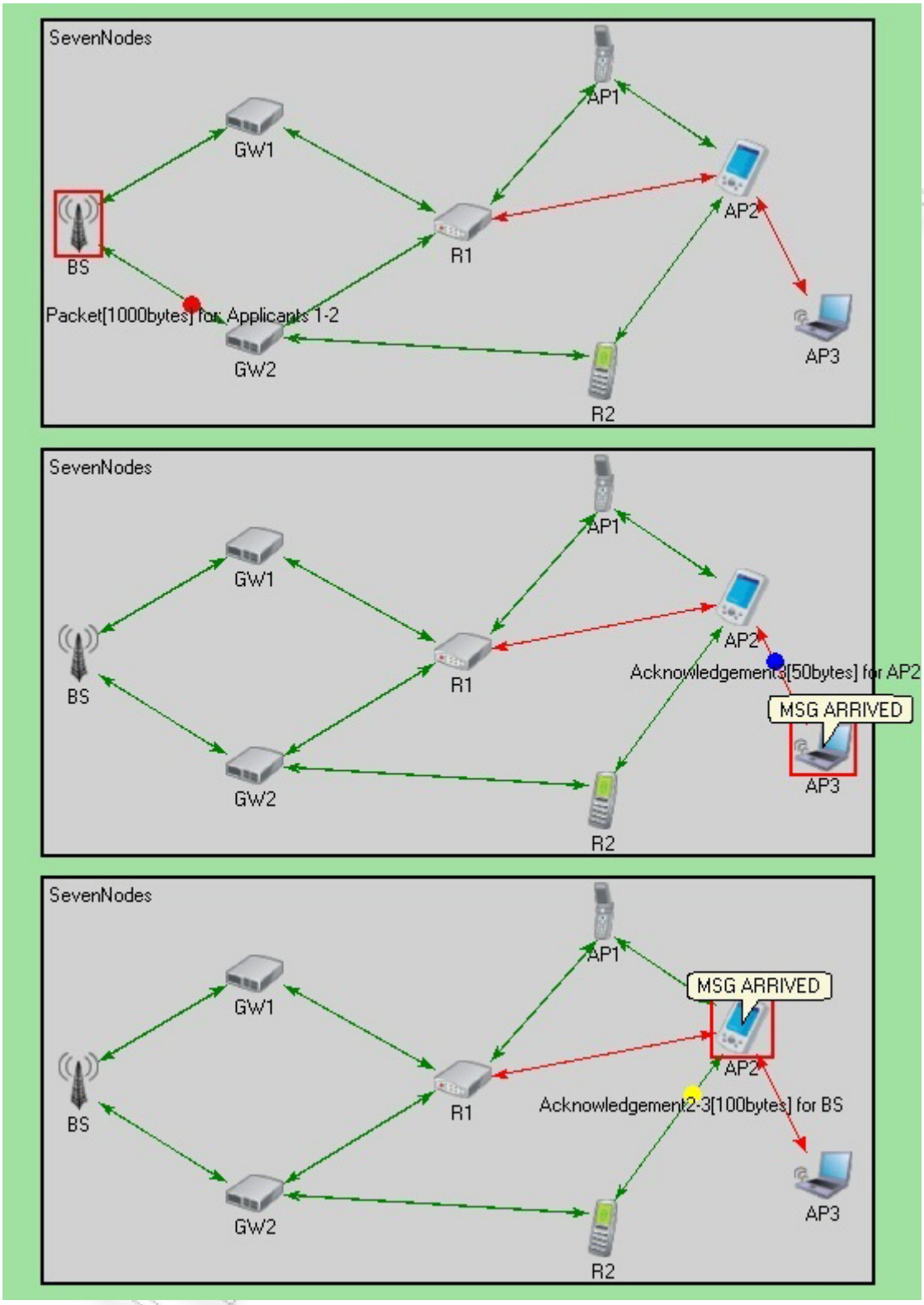
Εικ. 5.1 Τοπολογία σεναρίου 7 κόμβων.

Κατά την εκκίνηση του OppNet θεωρείται ότι οι κόμβοι AP1, AP2 και AP3 έχουν κάνει αίτηση για μία εφαρμογή video streaming από τη βασική υποδομή (infrastructure) του δικτύου. Συνεπώς στην εικόνα 5.2 απεικονίζεται ο BS να στέλνει ένα πακέτο μεγέθους 500 bytes προς τον κόμβο AP1. Ακολούθως το κινητό AP1 στέλνει μήνυμα ACK μεγέθους 50 bytes προς τον BS, επιβεβαιώνοντας την άφιξη του αρχικού πακέτου.



Εικ. 5.2 Εκκίνηση εφαρμογής σεναρίου 7 κόμβων.

Έπειτα ο BS αποστέλλει δεύτερο πακέτο, αυτή τη φορά προς τους κόμβους AP2 και AP3 (Εικ. 5.3). Όταν αυτό φτάσει στο smartphone AP2 προωθείται προς τον AP3 χωρίς ο πρώτος να στείλει πακέτο ACK πίσω στον BS. Ύστερα, αφού το laptop λάβει το μήνυμα, επιστρέφει ένα πακέτο ACK προς τον AP2 και μόνο τότε δημιουργείται ένα νέο πακέτο επιβεβαίωσης 100 bytes (διπλάσιου μεγέθους γιατί προέρχεται από 2 κόμβους) προς το σταθμό βάσης, εφόσον και οι δύο έχουν λάβει το αρχικό.



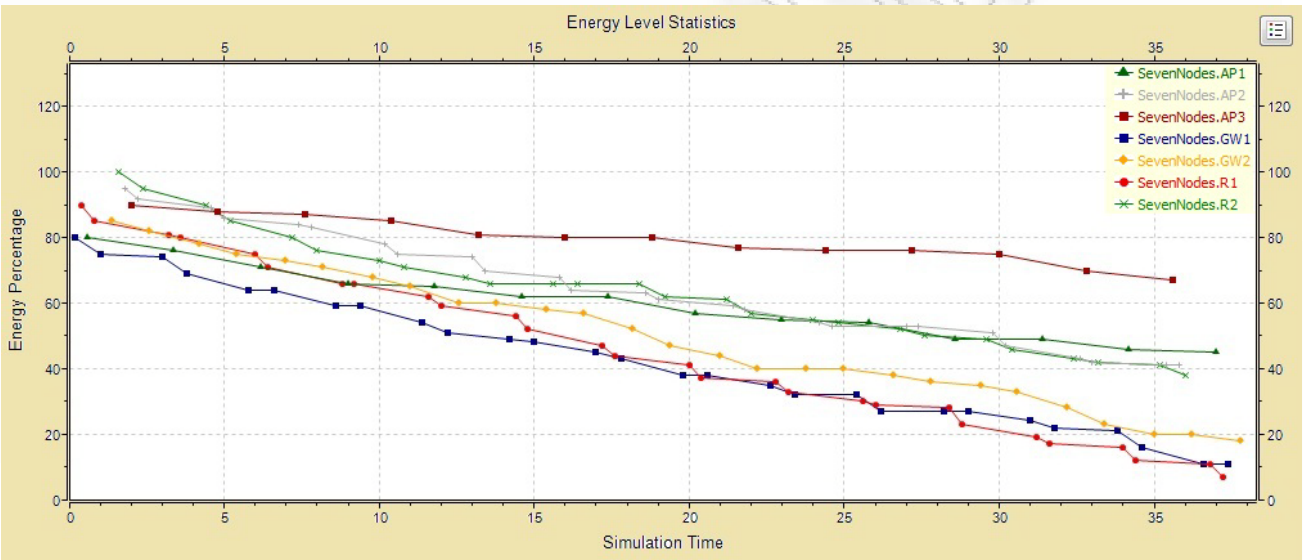
Εικ. 5.3 Χρήση της υπηρεσίας και από τους υπόλοιπους κόμβους εφαρμογής.

Η συγκεκριμένη δρομολόγηση των πακέτων συνεχίζεται ανελλιπώς μέχρις ότου τρεις εκ των κόμβων να διαθέτουν μη επιτρεπτά επίπεδα ενέργειας. Το κατώτερο επιτρεπόμενο επίπεδο ενέργειας ανά κόμβο είναι 20%. Μόλις τρεις οποιοιδήποτε κόμβοι αγγίξουν το 19% ή και λιγότερο,

το δίκτυο αυτόματα διαγράφει τα μηνύματα που κυκλοφορούν και τυπώνει αντίστοιχο μήνυμα αναφέροντας χαμηλή ενέργεια.

Η επιλογή του 20% χρησιμοποιείται για την ορθή διεξαγωγή της προσομοίωσης. Το ποσοστό αυτό καθώς και το πλήθος των κόμβων που είναι απαραίτητοι για τερματισμό σε πραγματικά δίκτυα ενδέχεται να καθορίζεται με εντελώς διαφορετικά κριτήρια.

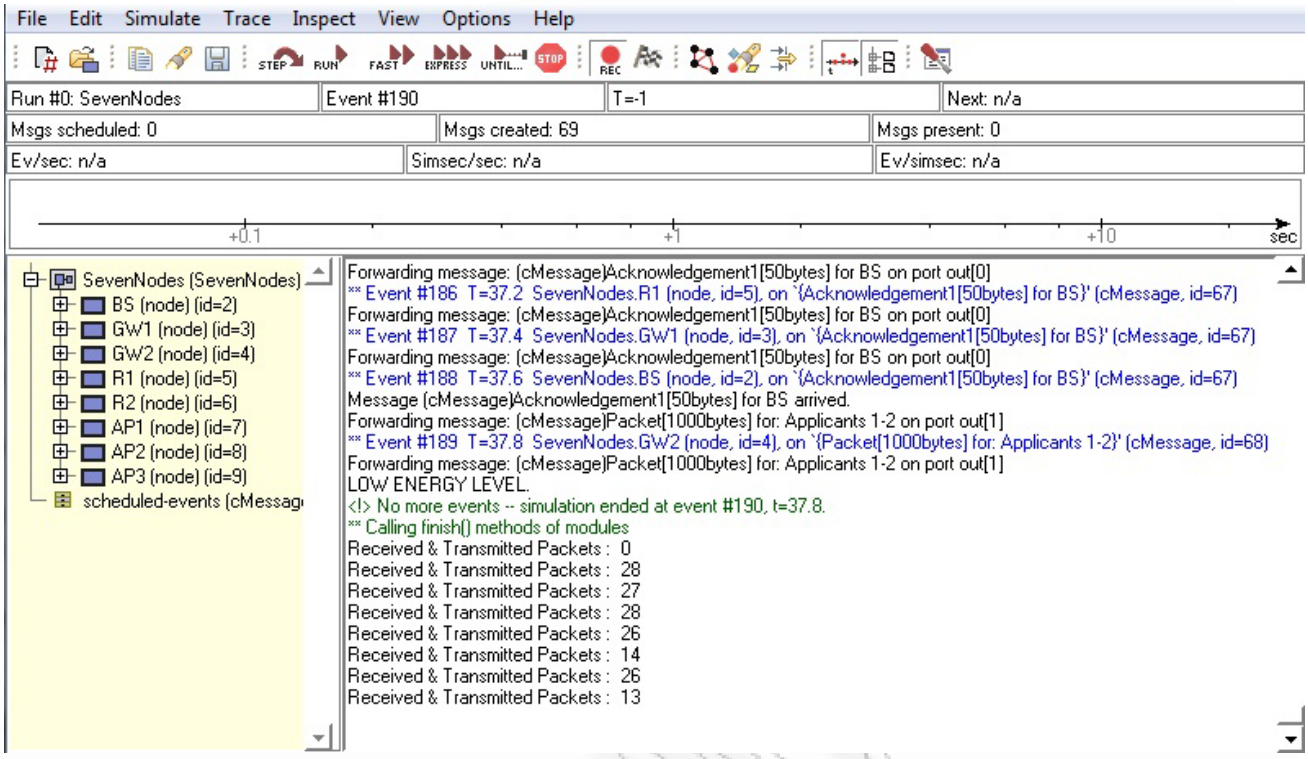
Η εικόνα 5.4 απεικονίζει το ποσοστό ενέργειας κάθε κόμβου στη διάρκεια χρόνου προσομοίωσης. Κάθε κόμβος χαρακτηρίζεται από μοναδική γραμμή χρώματος και σχήματος. Είναι εμφανές λοιπόν ότι τη χρονική στιγμή 31,2 το ποσοστό ενέργειας του κόμβου R1 πέφτει στο 19% με ακόλουθο τον κόμβο GW1 τη χρονική στιγμή 34,6 με ποσοστό 16%. Ο κόμβος που σχηματίζει την τελική τριάδα δίνοντας το έναυσμα για τερματισμό, είναι ο GW2, με ποσοστό 18% στο χρονικό σημείο με τιμή 37,8 χρονικών μονάδων προσομοίωσης.



Εικ. 5.4 Διάγραμμα επιπέδου ενέργειας 7 κόμβων.

Τα συμπεράσματα που προκύπτουν από το σενάριο 1, είναι ότι το δίκτυο, ολοκλήρωσε τη λειτουργία του λόγω χαμηλού ποσοστού ενέργειας των κόμβων R1, GW1 και GW2 σε διάρκεια 37,8 μονάδων προσομοίωσης. Ακόμη, σημαντικό στοιχείο αποτελεί ο αριθμός των πακέτων που επιτυχώς ελήφθησαν από τους κόμβους, με τη τιμή να φτάνει τα 69.

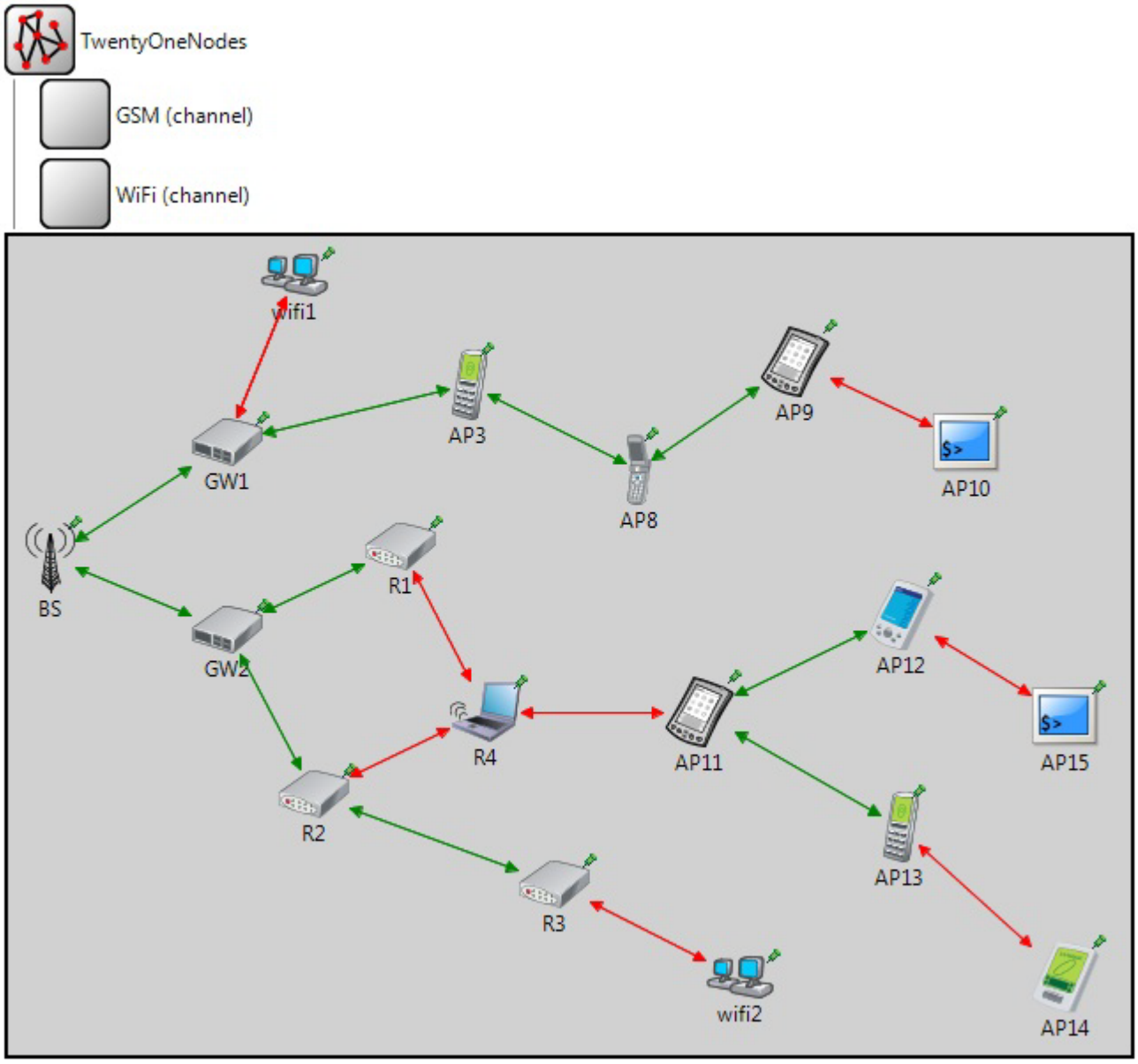
Στην εικόνα 5.5 απεικονίζεται το module output μετά το τέλος διεξαγωγής της προσομοίωσης. Σε αυτό απεικονίζονται διάφορες τιμές όπως ο χρόνος, το πλήθος των γεγονότων (events), τα μηνύματα (πακέτα) που δημιουργήθηκαν, τα μηνύματα που είναι ενεργά κλπ. Το πλήθος των γεγονότων αυξάνεται κάθε φορά που συμβαίνει μία ενέργεια σε έναν κόμβο όπως για παράδειγμα η αποστολή ενός πακέτου. Στη γραμμή εντολών είναι εμφανές ότι μετά το #189 γεγονός το δίκτυο τερματίζεται λόγω χαμηλού επιπέδου ενέργειας («Low Energy Level») τη χρονική στιγμή 37,8, τιμή η οποία συμφωνεί με τη γραφική παράσταση της εικόνας 5.4.



Εικ. 5.5 Module Output 7 κόμβων.

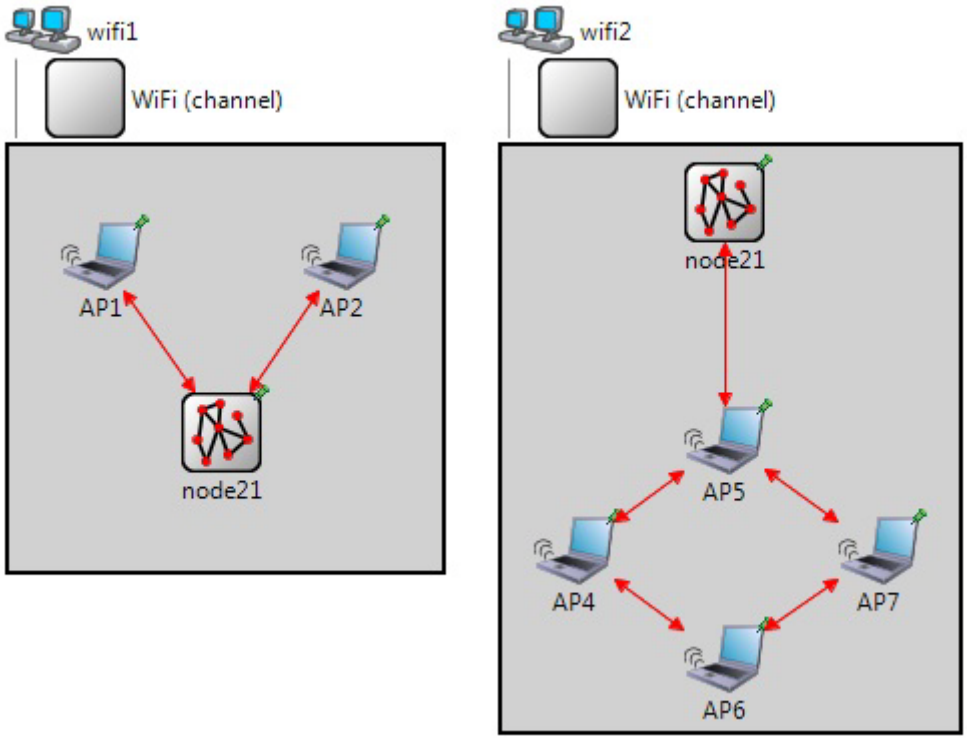
5.3 Σενάριο 2 – Δίκτυο 12 κόμβων

Στο σενάριο 2 της εικόνας 5.6, το δίκτυο που εγκαθίσταται αποτελείται από 21 κόμβους εκ των οποίων όμως οι 12 είναι ενεργοί και συμμετέχουν στην προώθηση των πακέτων. Από αυτούς οι 2 είναι πύλες, οι 3 είναι κόμβοι αναμετάδοσης και οι υπόλοιποι 7 αποτελούν τους κόμβους εφαρμογών. Όπως και στο σενάριο υπ' αριθμόν 1 χρησιμοποιούνται δύο τύποι καναλιών, ένα GSM και ένα Wi-Fi. Τα κανάλια χρησιμοποιούν ίδιες τιμές με το προηγούμενο σενάριο για την αποφυγή αποκλίσεων στις μετρήσεις. Επίσης στο σενάριο 2, στους κόμβους εφαρμογών εκτός από κινητά τηλέφωνα και smartphones έχουν προστεθεί δύο tablets καθώς και μία ιατρική συσκευή παρακολούθησης.



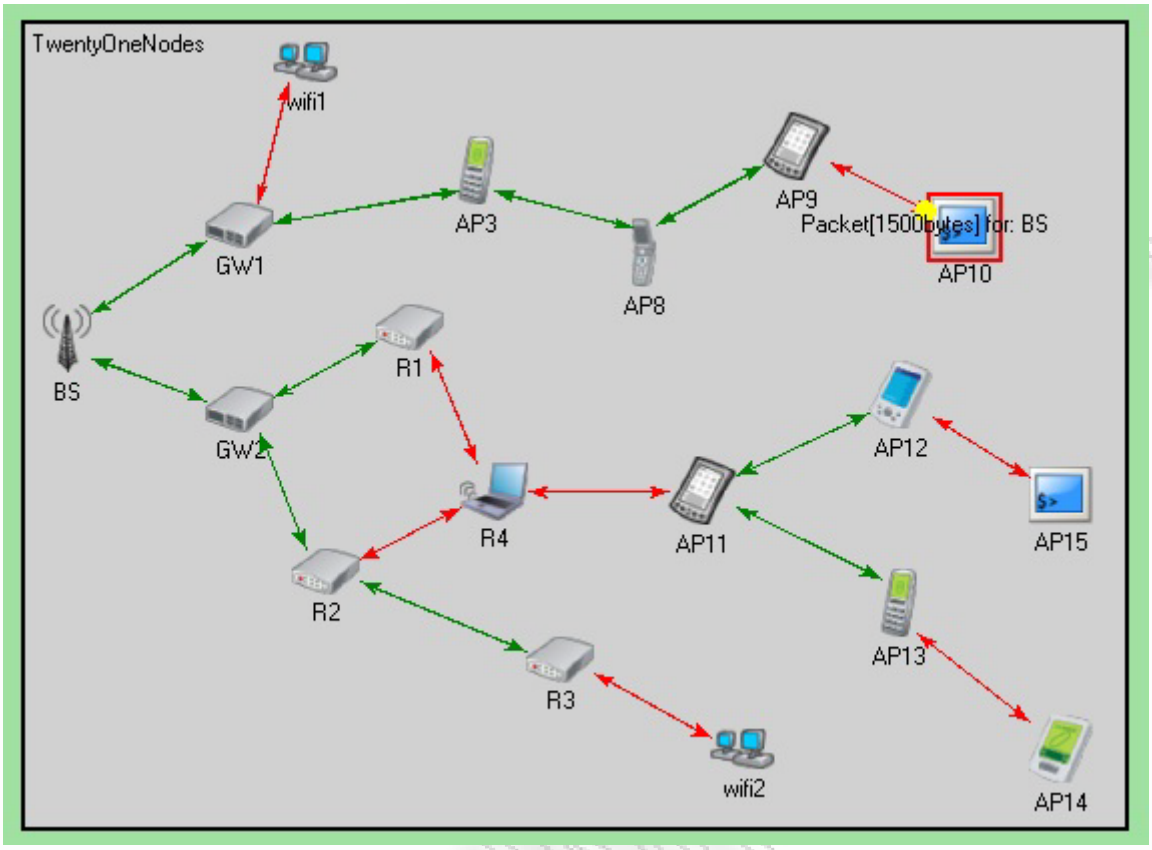
Εικ. 5.6 Τοπολογία σεναρίου 12 κόμβων.

Επιπλέον το δίκτυο περιέχει δύο τοπικά υποδίκτυα Wi-Fi1 και Wi-Fi2, στα οποία συμπεριλαμβάνονται συνολικά 6 κόμβοι (2 και 4 αντίστοιχα), όντας όμως ανενεργοί στο συγκεκριμένο σενάριο (Εικ. 5.7).

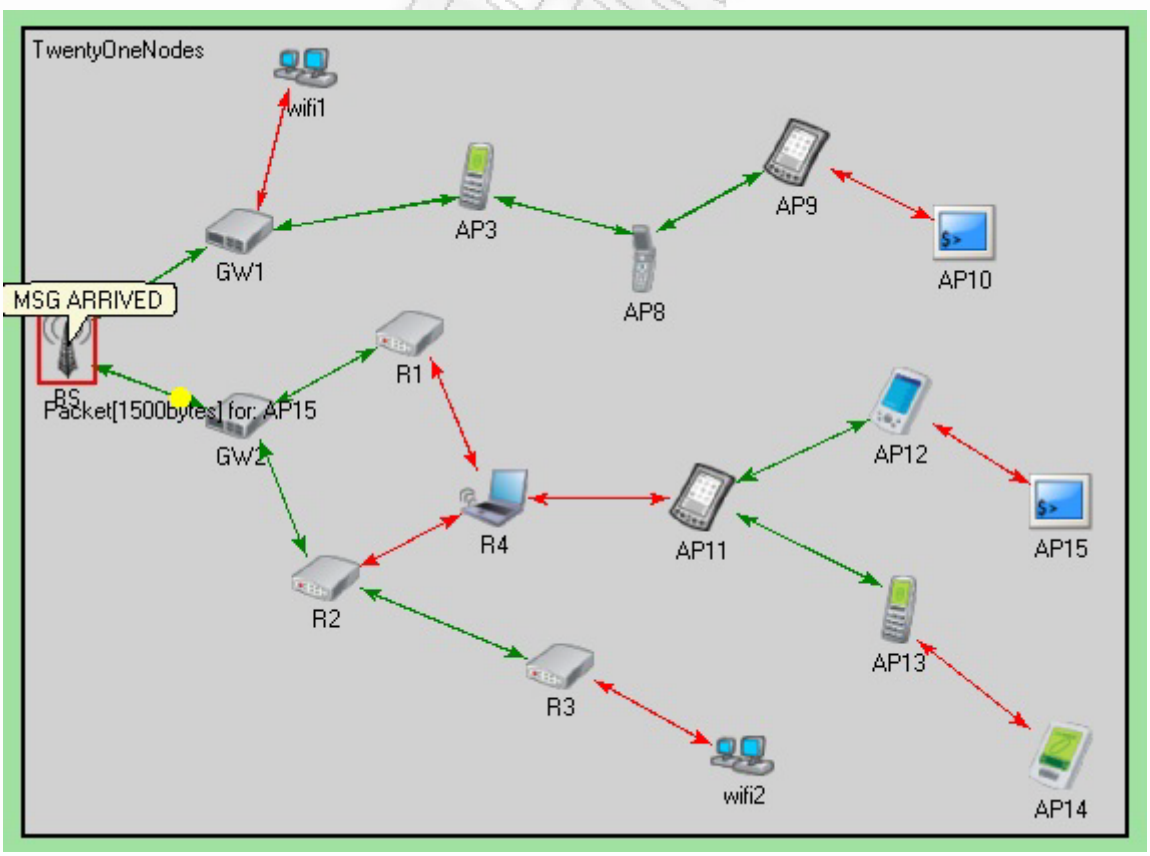


Εικ. 5.7 Υποδίκτυα Wi-Fi1 και Wi-Fi2

Στην περίπτωση του σεναρίου 2 επιθυμείται η επικοινωνία μεταξύ των κόμβων AP10 και AP15 μέσω video κλήσης. Για το λόγο αυτό το πρώτο πακέτο 1500 bytes ξεκινά από το tablet AP10 κατευθυνόμενο προς τον BS ο οποίος θα είναι υπεύθυνος για τη μετέπειτα δρομολόγησή του (Εικ. 5.8). Ύστερα εκείνος θα προωθήσει το πακέτο προς τον GW2 όντας και η μοναδική πύλη επιλογής μέχρις ότου το πακέτο να φτάσει στον προορισμό του (Εικ. 5.9).

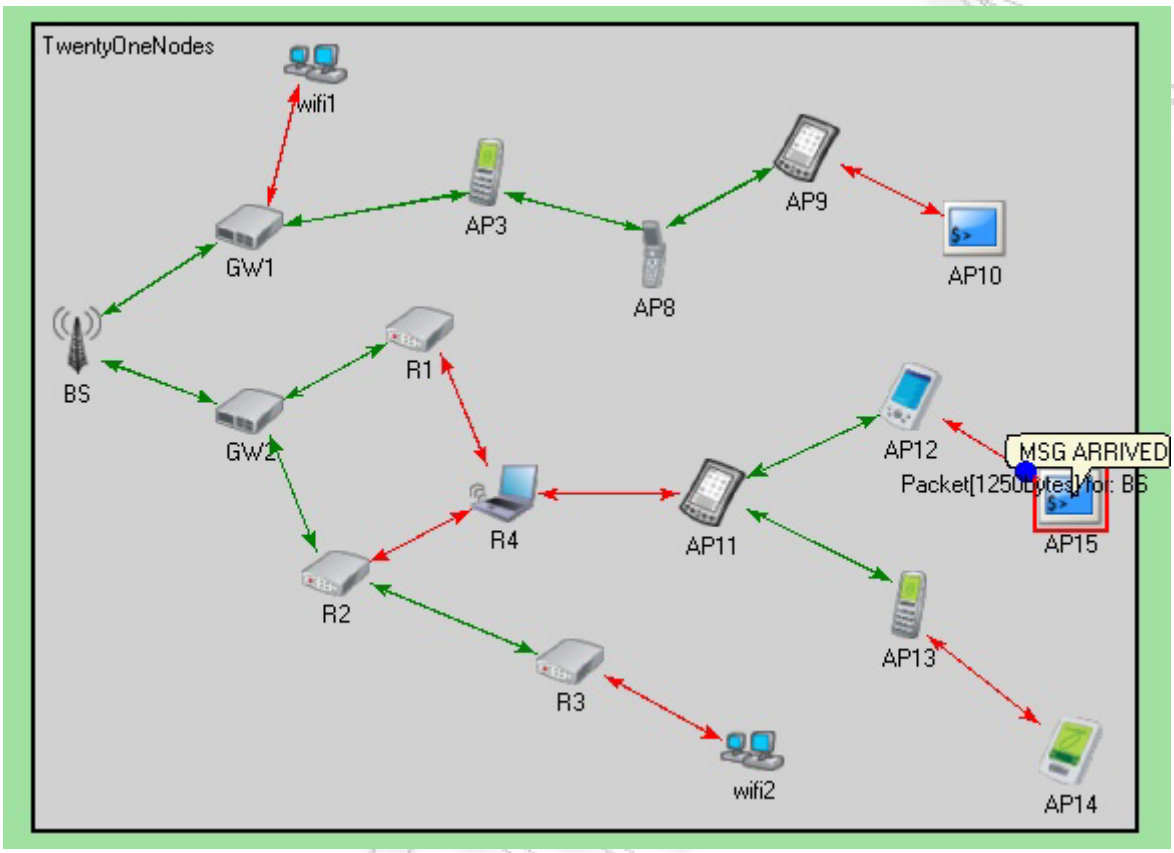


Εικ. 5.8 Εκκίνηση εφαρμογής σεναρίου 12 κόμβων.



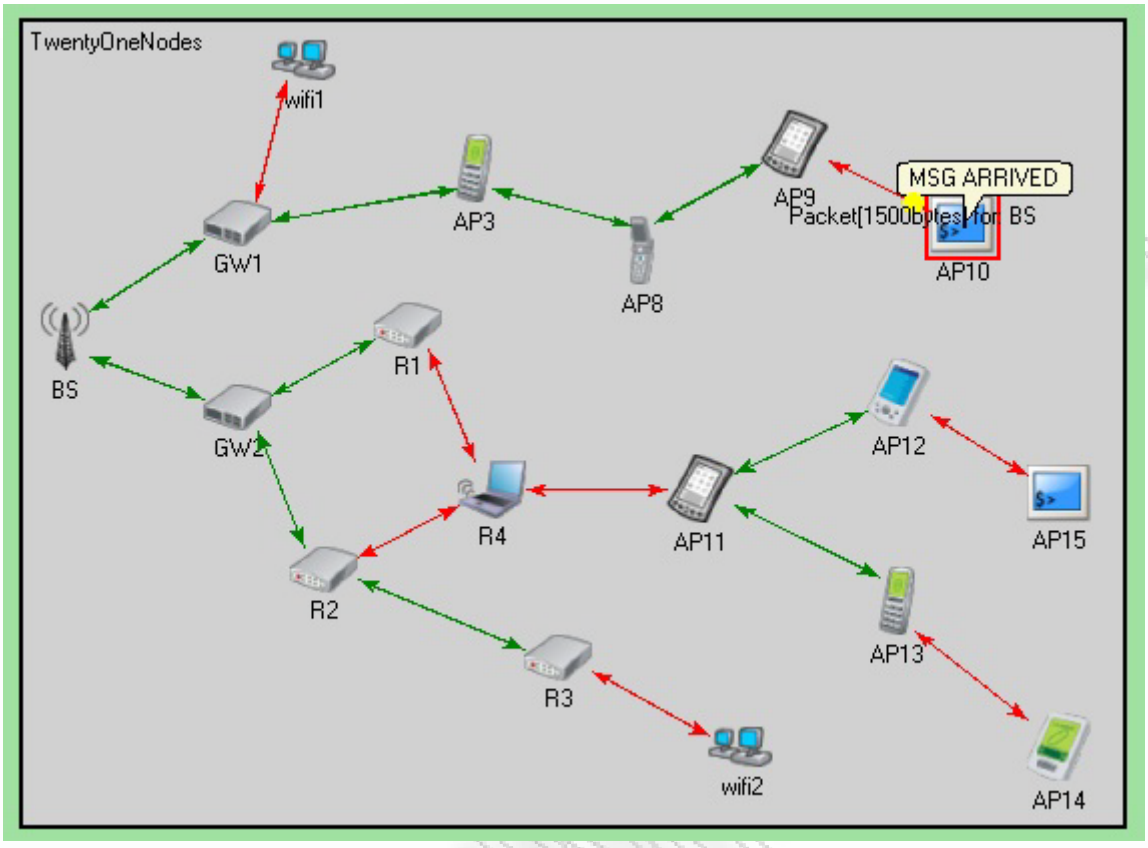
Εικ. 5.9 Δρομολογημένο πακέτο προς κόμβο AP15.

Όταν ο AP15 λάβει το προερχόμενο πακέτο από τον AP10 δημιουργεί το δικό του 1250 bytes (Εικ. 5.10) και το αποστέλλει προς το σταθμό βάσης ενώ εκείνος με τη σειρά του έπειτα το δρομολογεί προς την πύλη GW1 με σκοπό να φτάσει στο tablet AP10.



Εικ. 5.10 Δρομολογημένο πακέτο προς κόμβο AP10.

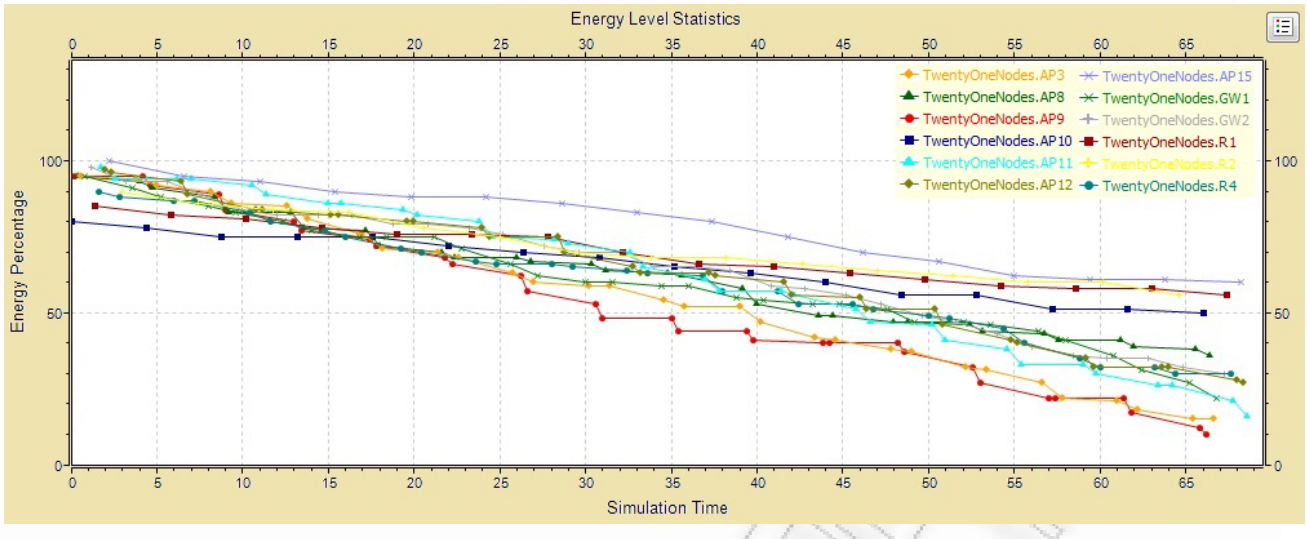
Ο κόμβος εφαρμογής AP10, αφού το λάβει δημιουργεί και στέλνει στο δίκτυο το επόμενο μήνυμα (Εικ. 5.11).



Εικ. 5.11 Ολοκλήρωση του κύκλου επικοινωνίας και νέο δρομολογημένο πακέτο προς κόμβο AP15.

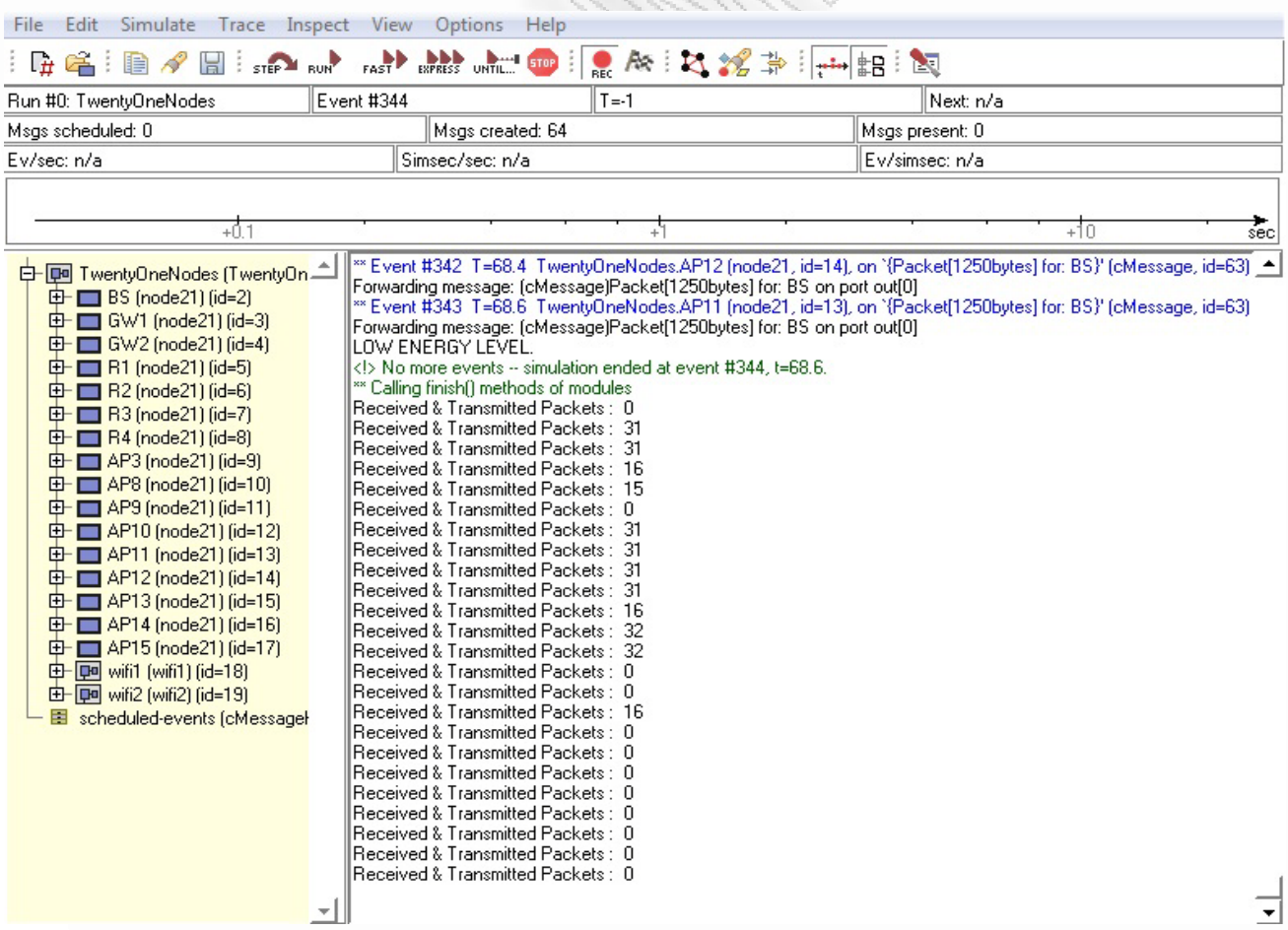
Με αυτόν τον τρόπο οι δύο τελικοί χρήστες επικοινωνούν αδιάκοπα μέχρις ότου να συμβεί κάτι απρόσμενο. Σε αντίθεση με το σενάριο 1, εδώ δεν αποστέλλονται πακέτα ACK γιατί η εφαρμογή που χρησιμοποιείται είναι ζωντανή video κλήση. Το γεγονός αυτό συμβάλλει στην ταχύτερη ανταλλαγή πληροφορίας που είναι επιθυμητό για τέτοιου είδους επικοινωνίες. Η πιθανή απώλεια πακέτων θα έχει ως αντίκτυπο τη χαμηλότερη ποιότητα ήχου και εικόνας για κάποια κλάσματα του δευτερολέπτου, το οποίο είναι αποδεκτό και πολλές φορές μη παρατηρήσιμο.

Παρόμοια με το προηγούμενο σενάριο χρησιμοποιείται η ίδια τιμή κατωφλίου για το επίπεδο ενέργειας. Έτσι, μόλις οι 3 πρώτοι κόμβοι σημειώσουν τιμή ενέργειας κατώτερη του 20% ο καθένας, το OppNet θα τερματιστεί. Στο διάγραμμα της εικόνας 5.12 παρατηρείται ότι ο κόμβος AP9 είναι ο πρώτος του οποίου η ενέργεια κατρακυλά κάτω από το επιτρεπτό όριο παίρνοντας την τιμή 17%. Αμέσως μετά, στο χρονικό σημείο 62,2 ακολουθεί ο AP3 με ποσοστό 18%. Ο κόμβος που ολοκληρώνει την ομάδα είναι ο AP11 τη χρονική στιγμή 68,6 με την τιμή του ενεργειακού επιπέδου να πέφτει στο 16% από το 21%.



Εικ. 5.12 Διάγραμμα επιπέδου ενέργειας 12 κόμβων.

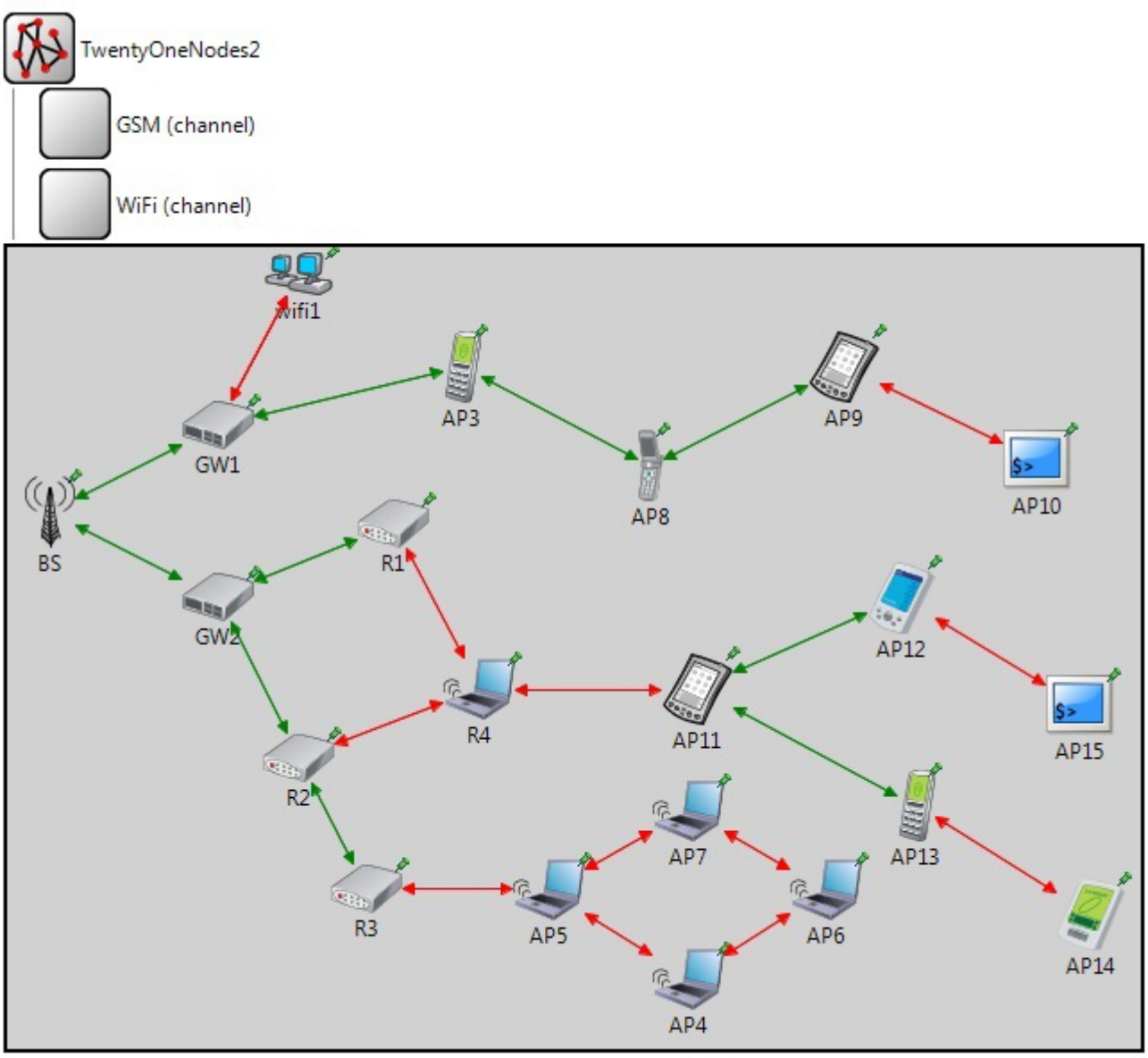
Είναι γεγονός λοιπόν ότι το δίκτυο ολοκλήρωσε τη λειτουργία του στο χρονικό σημείο 68,6 λόγω ελλιπούς ενέργειας στους κόμβους AP9, AP3 και AP11. Τα πακέτα που επιτυχώς ελήφθησαν από όλους τους κόμβους κατά τη διάρκεια λειτουργίας του είναι 64. Τα στοιχεία της εικόνας 5.13 επιβεβαιώνουν τα αποτελέσματα αυτά αφού μετά το γεγονός #343 με χρόνο 68.6 διατυπώνεται το μήνυμα χαμηλής ενέργειας.



Εικ. 5.13 Module Output 12 κόμβων.

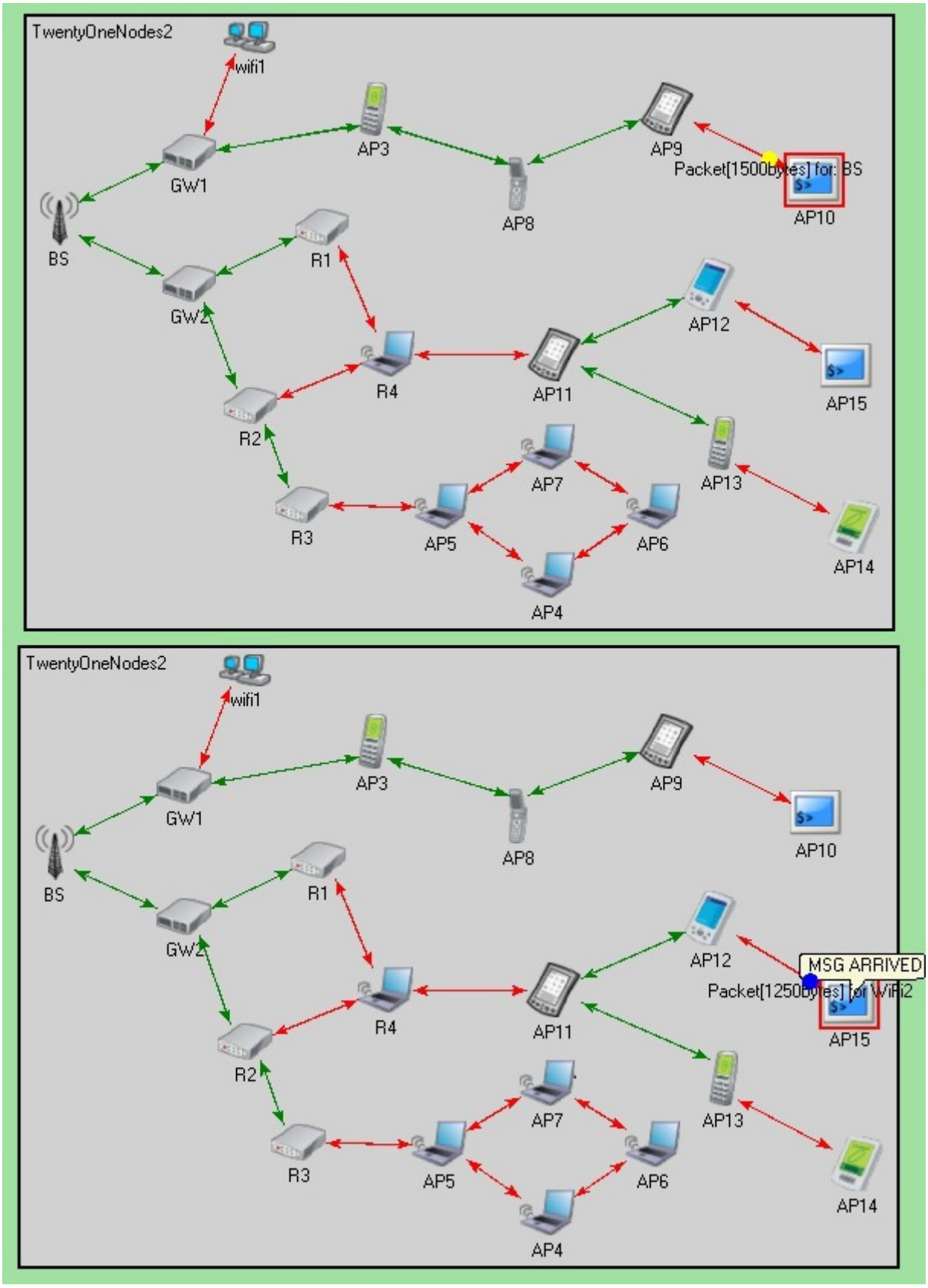
5.4 Σενάριο 3 – Δίκτυο 17 κόμβων

Το σενάριο 3 χρησιμοποιεί την ίδια τοπολογία δικτύου με αυτό της προηγούμενης ενότητας με τη διαφορά ότι οι κόμβοι που συμμετέχουν στη δρομολόγηση των μηνυμάτων είναι 17 (Εικ. 5.14). Οι 4 επιπλέον κόμβοι που εντάσσονται στο παιχνίδι είναι τα τερματικά του υποδικτύου Wi-Fi2, και 5^{ος} ο κόμβος αναμετάδοσης R3. Συνεπώς οι κόμβοι εφαρμογών αυξάνονται στους 11.



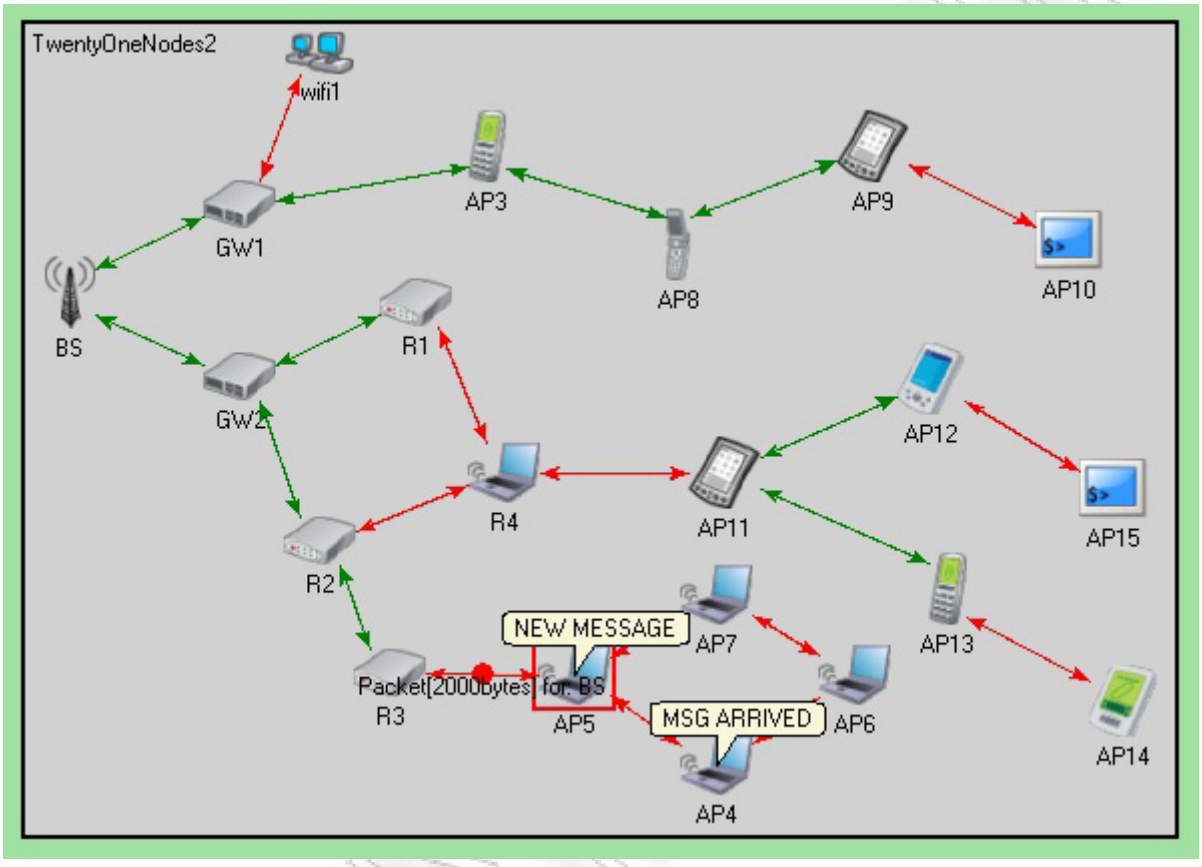
Εικ. 5.14 Τοπολογία σεναρίου 17 κόμβων.

Η υπηρεσία που θα χρησιμοποιηθεί στο OppNet αυτή τη φορά για την επικοινωνία των κόμβων, είναι η τηλεδιάσκεψη (video conference). Σε αυτή θα συμμετέχουν οι κόμβοι AP10 και AP15 όπως και προηγουμένως, καθώς και όλα τα συστατικά στοιχεία του Wi-Fi2. Συνεπώς το tablet AP10 ξεκινά την τηλεδιάσκεψη στέλνοντας πακέτο 1500 bytes με ετικέτα προορισμού AP15. Έπειτα ο AP15 αφού λάβει το πακέτο δημιουργεί και στέλνει το δικό του 1250 bytes προς το Wi-Fi2.



Εικ. 5.15 Εκκίνηση εφαρμογής σεναρίου 17 κόμβων.

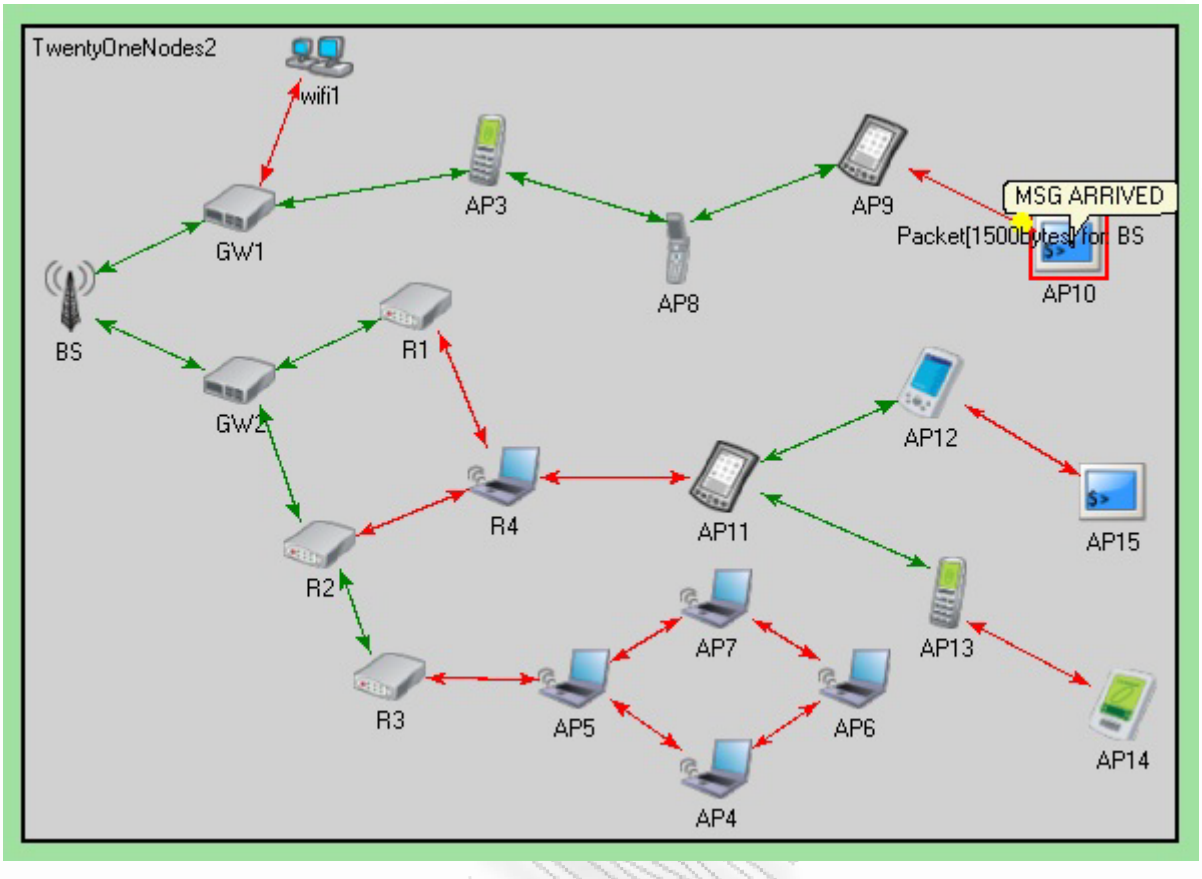
Στο συγκρότημα του Wi-Fi2, το πρώτο τερματικό που λαμβάνει το πακέτο είναι ο AP5 και ύστερα ακολουθούν οι υπόλοιποι σειριακά, μέχρις ότου να ξαναφτάσει στον ίδιο (AP5 → AP7 → AP6 → AP4 → AP5). Τότε, το μήνυμα διαγράφεται και δημιουργείται ένα νέο 2000 bytes με προορισμό τον AP10 (Εικ. 5.16).



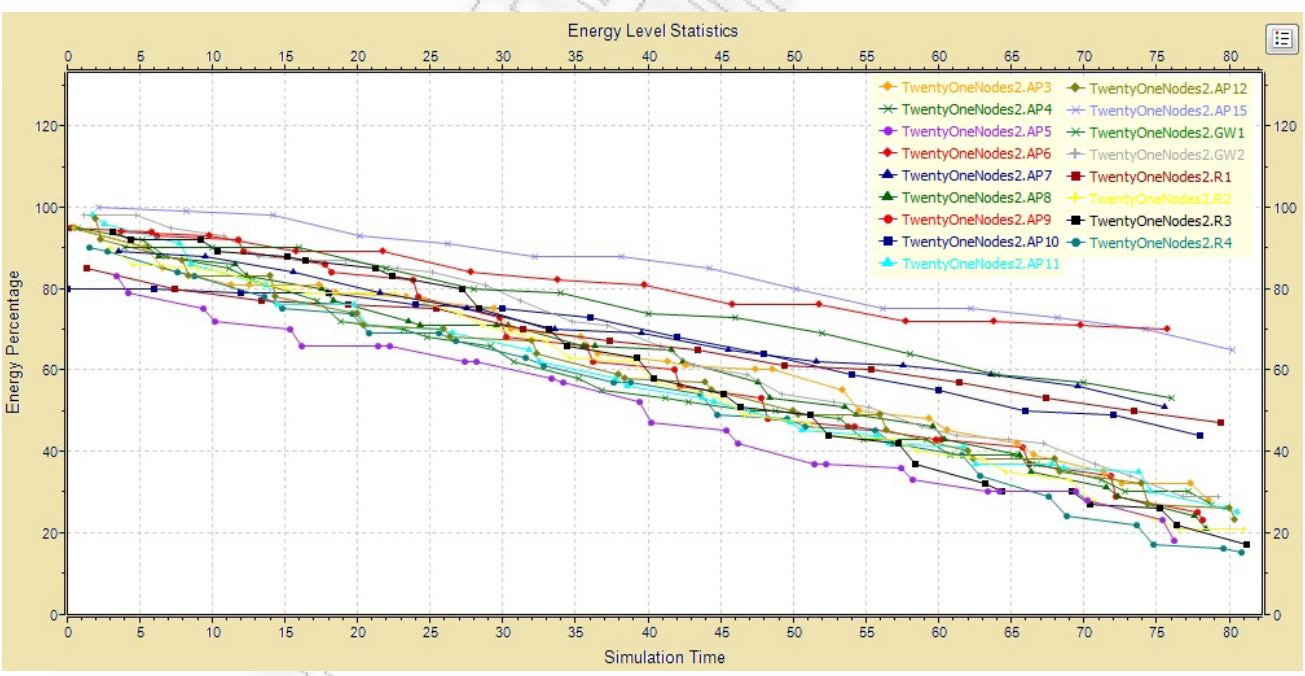
Εικ. 5.16 Άφιξη πακέτου στο συγκρότημα Wi-Fi2 και αποστολή στον AP10.

Ο κύκλος της τηλεδιάσκεψης ολοκληρώνεται μόλις το πακέτο του Wi-Fi2 φτάσει στον προορισμό του, με τον κόμβο AP10 να ξεκινά ένα νέο κύκλο μεταφοράς δεδομένων (Εικ. 5.17).

Στην εικόνα 5.18 απεικονίζεται το διάγραμμα του ποσοστού ενέργειας σε σχέση με το χρόνο. Τη χρονική στιγμή 74,8 ο κόμβος R4 φτάνει στο 17% ενώ ακολουθεί ο AP5 με 18% στο χρονικό στιγμιότυπο με τιμή 76,2. Τελευταίος έρχεται ο R3 στο 17% του ενεργειακού ποσοστού τερματίζοντας το δίκτυο σε διάρκεια 81,2 χρονικών μονάδων προσομοίωσης.



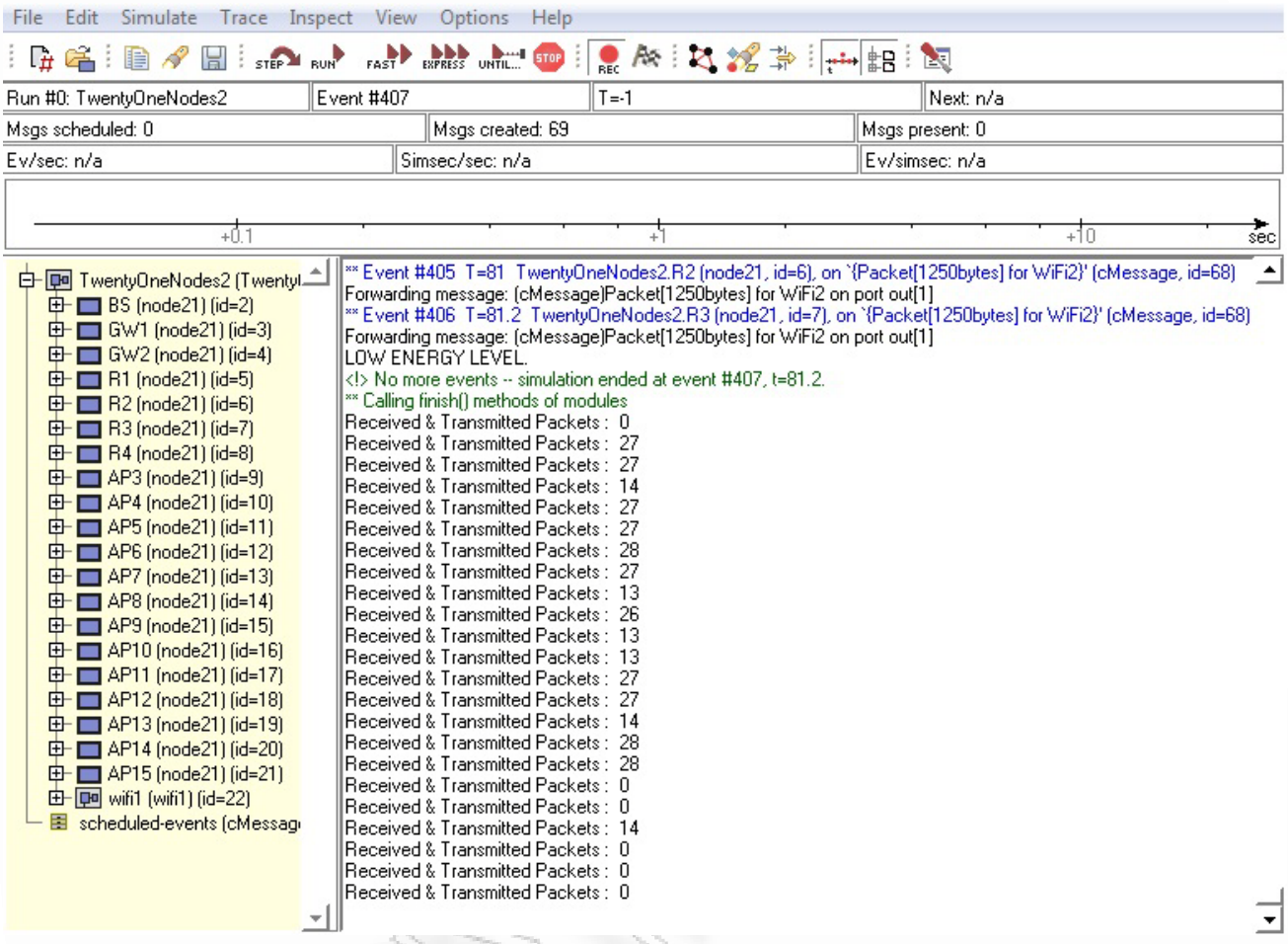
Εικ. 5.17 Ολοκλήρωση του κύκλου επικοινωνίας και νέο δρομολογημένο πακέτο προς κόμβο AP15.



Εικ. 5.18 Διάγραμμα επιπέδου ενέργειας 17 κόμβων.

Στη εικόνα 5.19 του module output αναγράφεται ο χρόνος διάρκειας λειτουργίας του δικτύου (81,2 χρονικές μονάδες προσομοίωσης) καθώς και το πλήθος των πακέτων που δημιουργήθηκαν (69). Επίσης στο αριστερό πάνελ η ταυτότητα των κόμβων και στο κεντρικό το σύνολο των

πακέτων που έλαβε και προώθησε ο καθένας απ' αυτούς (με σειρά σύμφωνη με το αριστερό πάνελ).



Εικ. 5.19 Module Output 17 κόμβων.

5.5 Σύγκριση αποτελεσμάτων σεναρίων 1,2 και 3

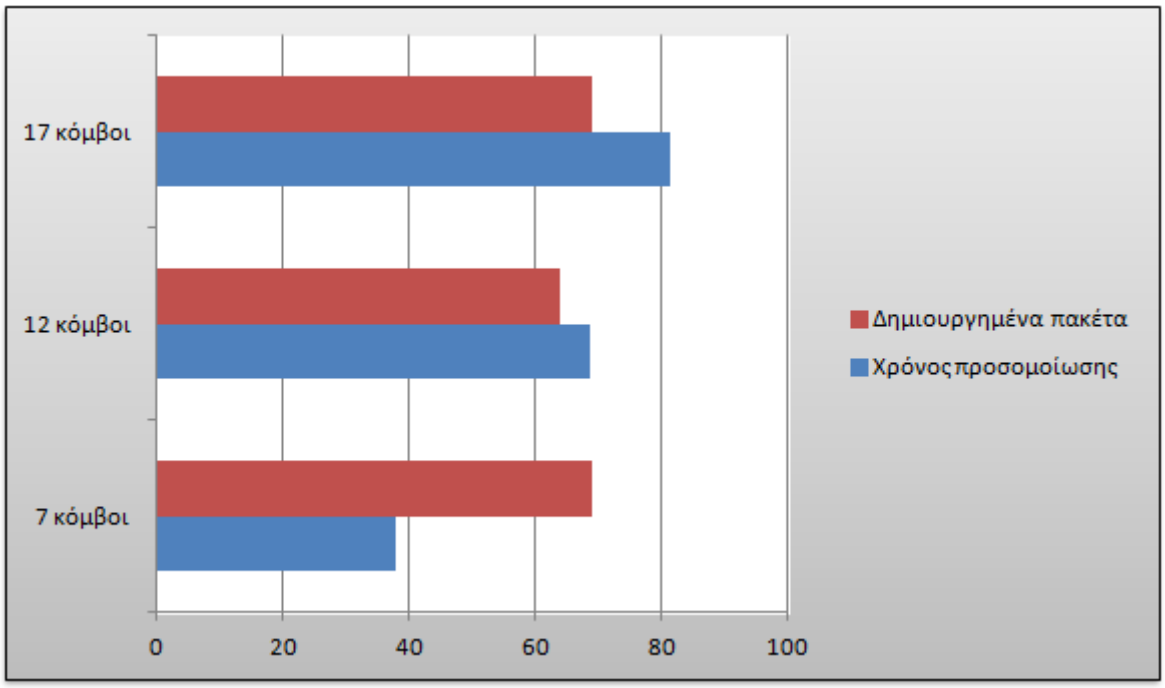
Τα σενάρια που περιγράφηκαν και υλοποιήθηκαν μέσω του προγράμματος προσομοίωσης επέφεραν σημαντικά αποτελέσματα προς μελέτη. Τα στοιχεία που φανερώνουν ιδιαίτερο ενδιαφέρον, αφορούν το χρόνο λειτουργίας του OppNet αλλά και το πλήθος των δημιουργημένων πακέτων σε σχέση με το μέγεθός του.

Στο διάγραμμα της εικόνας 5.20, απεικονίζονται τα σενάρια αναγραφόμενα σε πλήθος κόμβων. Είναι φανερό λοιπόν, ότι η διάρκεια ζωής του δικτύου είναι μακροβιότερη όσο αυξάνεται ο αριθμός των κόμβων του. Το γεγονός αυτό είναι απόλυτα φυσιολογικό αφού με τη χρήση περισσότερων στοιχείων δικτύου, το φορτίο διαμοιράζεται και έτσι οι κόμβοι διατηρούν για μεγαλύτερη διάρκεια την ενέργεια τους.

Στο σημείο αυτό, πρέπει να τονιστεί η σταθερότητα της τιμής κατωφλίου (οι 3 πρώτοι κόμβοι με ενέργεια κατώτερη του 20%) και στα 3 σενάρια. Αυτή η τιμή-φρουρός θα μπορούσε να αποτελεί το

worst-case-scenario για τις περιπτώσεις των 12 και 17 κόμβων, αφού θα ήταν δυνατό να αυξηθεί στους 4 ή και στους 5 κόμβους. Σε διαφορετικές συνθήκες λοιπόν, ο χρόνος λειτουργίας του δικτύου θα σκαρφάλωνε σε ακόμη πιο υψηλές τιμές.

Το δεύτερο στοιχείο που αντλείται από το διάγραμμα και προαναφέρθηκε είναι τα πακέτα που δημιουργήθηκαν κατά τη λειτουργία του OppNet. Παρατηρείται λοιπόν ότι οι τιμές, κυμαίνονται από 64 μέχρι 69, με τις τοπολογίες των 7 και 17 κόμβων να έχουν ακριβώς ίσες τιμές. Αυτό οφείλεται στις διαφορετικές δρομολογήσεις των σεναρίων σε σχέση με το χρόνο λειτουργίας τους. Δηλαδή, στο σενάριο 1 το οποίο περιλαμβάνει μόνο 7 κόμβους, τα μονοπάτια δρομολόγησης που χρησιμοποιούνται είναι σύντομα και μικρών αποστάσεων. Συνεπώς στις 37,8 μονάδες προσομοίωσης προλαβαίνει να δημιουργήσει 69 πακέτα σε αντίθεση με το σενάριο 3 των 17 κόμβων, στο οποίο δημιουργούνται σε 81,2 μονάδες λόγω των μεγαλύτερων αποστάσεων δρομολόγησης.



Εικ. 5.20 Διάγραμμα σύγκρισης αποτελεσμάτων.

Απόρροια αυτής της σύγκρισης λοιπόν, είναι ότι το πλήθος των πακέτων δεν είναι ανάλογο της διάρκειας λειτουργίας του δικτύου, αλλά εξαρτάται από την απόσταση δρομολόγησης καθώς και τον τύπο υπηρεσίας που ο κόμβος χρησιμοποιεί. Αντιθέτως, το πλήθος των κόμβων επηρεάζει και αυξάνει το χρόνο ζωής του δικτύου γιατί διοχετεύεται σε αυτό περισσότερη ενέργεια.

Συμπέρασμα

Τα Opportunistic Networks αποδεικνύεται ότι θα αποτελούν αναπόσπαστο κομμάτι του Μελλοντικού Διαδικτύου για όλους τους φορείς του τομέα της πληροφορικής. Η υποστήριξη και η εξέλιξή τους θα πρέπει να είναι καθολική και εντατική έτσι ώστε να παρέχονται όσο το δυνατό περισσότερα προνόμια στους τελικούς χρήστες.

Με την υλοποίηση του προαναφερθέντος αλγορίθμου παρέχεται μία ενδεικτική λύση στο πρόβλημα εντοπισμού τερματισμού του δικτύου. Ο αλγόριθμος κατηγοριοποιεί τρεις τύπους τερματισμού οι οποίοι βασίζονται σε διαφορετικά κριτήρια και η πυροδότησή τους θέτει σε λειτουργία διαφορετικές διεργασίες. Οι διεργασίες αυτές είναι εφικτό να πραγματοποιηθούν ταχύτατα, λόγω της άμεσης ενημέρωσης προς το διαχειριστή του δικτύου.

Τέλος, η εφαρμογή αυτού του μηχανισμού σε σύνολο διαφορετικών σεναρίων προσομοίωσης, επιφέρει αξιόπιστα αποτελέσματα τα οποία αναδεικνύουν την ορθότητά του. Αναμένεται λοιπόν ότι η υιοθέτηση της λύσης αυτής θα αποδειχθεί προνομιακή για τις μετέπειτα μελέτες και έρευνες στο πλαίσιο της λειτουργίας των Opportunistic Networks.

Βάση Δεδομένων oNetResults

```
CREATE DATABASE IF NOT EXISTS oNetResults;

USE oNetResults;

DROP TABLE IF EXISTS netNodes;
DROP TABLE IF EXISTS netData;

CREATE TABLE netNodes (
	nodeID INT NOT NULL AUTO_INCREMENT,
	nodeStatus varchar (25) NOT NULL,
	PRIMARY KEY (nodeID)
)
;

CREATE TABLE netData (
	nodeID int NOT NULL,
	appStatus varchar(15) default NULL,
	gain float (2) default NULL,
	energyLevel varchar(12) default NULL,
	spectrum int default NULL,
	interfaces int default NULL,
	bitRate int default NULL,
	delay int default NULL,
	ber float(2) default NULL,
	attemptNum int NOT NULL,
	FOREIGN KEY (nodeID) REFERENCES netNodes(nodeID)
)
;

insert into netNodes (nodeStatus) values ('Gateway Node');
insert into netNodes (nodeStatus) values ('Gateway Node');
insert into netNodes (nodeStatus) values ('Relay Node');
insert into netNodes (nodeStatus) values ('Relay Node');
insert into netNodes (nodeStatus) values ('Application Node');
insert into netNodes (nodeStatus) values ('Application Node');
insert into netNodes (nodeStatus) values ('Application Node');
insert into netNodes (nodeStatus) values ('Total Network Values');
```

Βάση Δεδομένων oNetResults

Πηγές και Βιβλιογραφία

1. ICT-OneFit Project, http://www.ict-onefit.eu/, 2011.
2. P. Demestichas, K. Tsagkaris, V. Stavroulaki, Y. Kritikou, A. Georgakopoulos, *"Technical Challenges for Merging Opportunistic Networks with Respective Cognitive Management Systems in the Future Internet*", 21st IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Istanbul, Turkey, 26-27 August 2010.
3. P. Demestichas, D. Boscovic, V. Stavroulaki, A. Lee, J. Strassner, *"m@ANGEL: Autonomic Management Platform for Seamless Cognitive Connectivity to the Mobile Internet"*, IEEE Communications Magazine. , Vol. 44, No. 6, pp. 118-127, June 2006.
4. L. Lilien, Z. H. Kamal, V. Bhuse, A. Gupta, "*Opportunistic Networks: The Concept and Research Challenges in Privacy and Security*", Department of Computer Science, Western Michigan University, Kalamazoo, USA, 2006.
5. L. Lilien, Z. H. Kamal, A. Gupta**, "***Opportunistic Networks: Challenges in Specializing the P2P Paradigm*", 17th International Conference on Database and Expert System Applications (DEXA), Krakow, 16 October 2006.
6. A. Georgakopoulos, K. Tsagkaris, V. Stavroulaki, P. Demestichas, *"Specification and Assessment of a Fitness Function for the Creation of Opportunistic Networks"*, Future Networks and Mobile Summit, Warsaw, Poland, June 2011.
7. L. Pelusi, A. Passarella, M. Conti, *"Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad hoc Networks"*, IEEE Communications Magazine, Vol. 44, No. 11, pp. 134-141, 20 November 2006.
8. A. Pentland, R. Fletcher, A. Hasson, "*DaKNet Rethinking Connectivity in Developing Nations"*, Computer Magazine, Vol. 37, No. 1, pp. 78–83, January 2004.
9. A. Doria, M. Uden, D. P. Pandey, *"Providing Connectivity to the Saami Nomadic Community"*, 2nd Int. Conference on Open Collaborative Design for Sustainable Innovation, 2002.
10. P. Juang, H Oki, Y. Wang, M. Martonosi, Li-S. Peh, D. Rubenstein, *"Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet"*, 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), San Jose, California, 5-9 October 2002.
11. K. R. Fall, *"A Delay-Tolerant Network Architecture for Challenged Internets"*, ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 27-34, Karlsurhe, Germany, 25-29 August 2003.
12. A. Heinemann, *"Collaboration in Opportunistic Networks"*, VDM Verlag, Saarbrücken, Germany, 2007.
13. A. Georgakopoulos, V. Stavroulaki, J. Gebert, O. Moreno, O. Sallent, M. Matinmikko, P. Demestichas, *et al.*, *"Opportunistic Networks for Efficient Application Provisioning in the Future Internet: Business Scenarios and Technical Challenges",* Future Networks and Mobile Summit, pp. 44-51, Warsaw, Poland, June 2011.
14. N. Mittal, K. L. Phaneesha, F. C. Freiling, *"Safe Termination Detection in an Asynchronous Distributed System when Processes may Crash and Recover"*, Theoretical Computer Science, Vol. 410, No. 6-7, Essex, UK, February, 2009.
15. J. C. Cavouras, "*Programming with Java"*, Kleidarithmos, pp.493-534, Athens, 2003.
16. H. M. Deitel, P. J. Deitel, "*Java How to Program Sixth Edition*", M. Giurdas, pp. 510-581 & 1189-1227, Athens, 2006.

17. MySQL, http://www.mysql.com.
18. Dev. MySQL, http://dev.mysql.com/downloads/connector/j/5.1.