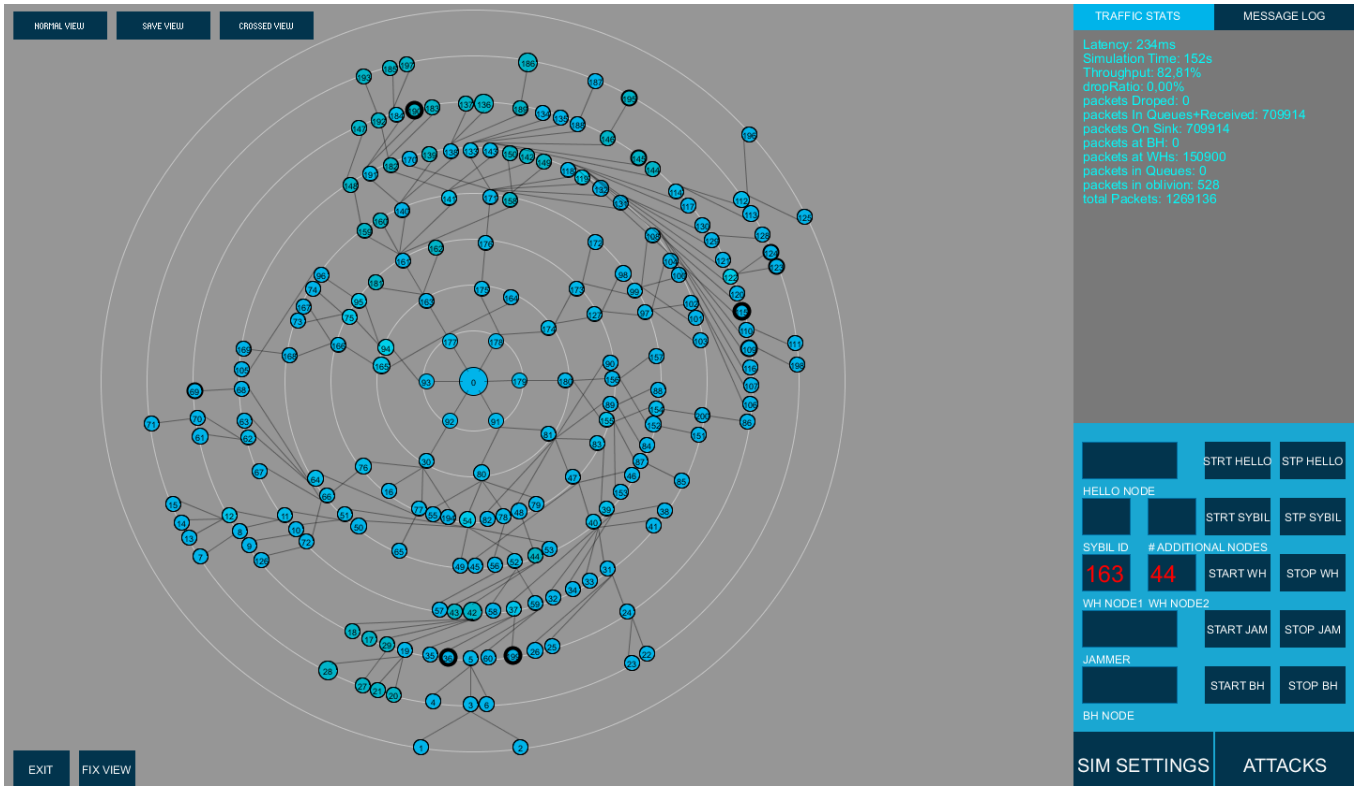
UNIVERSITY OF WESTERN MACEDONIA
DEPT. OF INFORMATICS AND TELECOMMUNICATIONS ENGINEERING

«Σχεδιασμός και Απεικόνιση Επιθέσεων Ασφάλειας σε Ασύρματα Δίκτυα Αισθητήρων με το Εργαλείο Processing»

Διπλωματική Εργασία

ΤΟΥ

ΠΑΝΑΓΙΩΤΗ Ι. ΠΑΠΑΔΟΠΟΥΛΟΥ



Επιβλέποντες Καθηγητές:
Παναγιώτης Σαρρηγιανίδης (Λέκτορας)
Μαλαματή Λούτα (Αναπληρώτρια Καθηγήτρια)

Κοζάνη, Ιούλιος 2015

Περιεχόμενα

Περίληψη

Η Γραφική Απεικόνιση Προβλημάτων Ασφαλείας στα δίκτυα τηλεπικοινωνιών(Security Visualization – Γ.Α.Π.Α) είναι ένας πολύ νέος όρος. Πιο συγκεκριμένα η Γ.Α.Π.Α στα ασύρματα δίκτυα αισθητήρων είναι ένα θέμα με το οποίο έχουν ασχοληθεί λίγοι ερευνητές. Τα Ασύρματα Δίκτυα Αισθητήρων βρίσκουν πολλές χρήσεις σε διάφορες εφαρμογές, αλλά ταυτόχρονα είναι ευάλωτα σε επιθέσεις. Οι επιθέσεις στα ΑΔΑ είναι ικανές να προκαλέσουν σοβαρότατα προβλήματα στην λειτουργία τους όπως αλλοίωση δεδομένων, εισαγωγή ψευδών δεδομένων και διακοπή επικοινωνίας. Ο σκοπός αυτής της Διπλωματικής εργασίας είναι η δημιουργία ενός προσομοιωτή ασύρματου δικτύου αισθητήρων, η γραφική αναπαράσταση, καθώς και η ανίχνευση των πιο διαδεδομένων επιθέσεων σε αυτό. Ο προσομοιωτής διαθέτει τέσσερις διαφορικές απεικονίσεις του δικτύου, με την κάθε μία να συμβάλει στην αναγνώριση των επιθέσεων. Ο χρήστης θα είναι σε θέση να εκτελέσει πέντε επιθέσεις και να παρακολουθήσει ζωντανά τις επιπτώσεις τους στο δίκτυο. Τέλος σε όλη την διάρκεια της προσομοίωσης εμφανίζονται αναλυτικά στατιστικά από την κίνηση των δεδομένων μεταξύ των κόμβων. Το εργαλείο έχει αναπτυχθεί στην γλώσσα Java με την βοήθεια των βιβλιοθηκών για γραφικά του Processing.

Η εφαρμογή που δημιουργήσαμε είναι ανεξάρτητη πλατφόρμας και μπορεί να εκτελεστεί σε μηχανήματα με Windows, Linux ή Mac OSX. Σε σύγκριση με άλλες εφαρμογές δεν είναι προσανατολισμένη σε ένα συγκεκριμένο είδος αισθητήρων, ο χρήστης έχει τη δυνατότητα να τροποποιήσει αυτός τις παραμέτρους του δικτύου ανάλογα με την εφαρμογή που τον ενδιαφέρει. Τέλος με το πάτημα ενός κουμπιού παρέχεται η δυνατότητα στον χρήστη να εξάγει σε ένα csv αρχείο τα στατιστικά που καταγράφηκαν κατά τη διάρκεια της προσομοίωσης. Το αρχείο αυτό μπορεί να στη συνέχεια να ανοιχτεί από το excel ή κάποιο παρόμοιο πρόγραμμα και ο χρήστης να αναλύσει τα δεδομένα.

Στο πρώτο κεφάλαιο παρουσιάζεται το πρότυπο IEEE 802.15.4 που χρησιμοποιείται στα ΑΔΑ. Ένας κακόβουλος χρήστης έχει τη δυνατότητα να εκμεταλλευτεί τα χαρακτηριστικά του προτύπου αυτού και να προκαλέσει ζημιά στο δίκτυο.

Στο δεύτερο κεφάλαιο περιγράφεται τι είναι τα ασύρματα δίκτυα αισθητήρων και αναφέρονται τα χαρακτηριστικά και οι εφαρμογές τους.

Στο τρίτο κεφάλαιο πραγματοποιείται μία ανάλυση των σημαντικότερων επιθέσεων που μπορούν να προσβάλλουν τα ΑΔΑ και αναφέρονται συνοπτικά τρόποι ανίχνευσης και αντιμετώπισης από την βιβλιογραφία.

Το τέταρτο κεφάλαιο είναι αφιερωμένο στην σημασία της οπτικής αναπαράστασης στον τομέα της ασφάλειας τον δικτύων.

Στο πέμπτο κεφάλαιο παρουσιάζεται συνοπτικά το εργαλείο Processing και οι μέθοδοι του που θα χρησιμοποιηθούν περισσότερο στην εφαρμογή μας.

Στο έκτο κεφάλαιο πραγματοποιείται η αναλυτική παρουσίαση της εφαρμογής. Στην αρχή γίνεται σαφής ο στόχος της εφαρμογής και αναφέρονται τα τεχνικά χαρακτηριστικά του υπολογιστή που την προγραμματίστηκε και εκτελέστηκε. Στη συνέχεια περιγράφεται η τοπολογία του δικτύου που θα προσομοιωθεί. Έπειτα γίνεται ανάλυση βήμα προς βήμα του κώδικα της εφαρμογής σε όλα τα στάδια δημιουργίας της. Τέλος εκτελείται η εφαρμογή και παρουσιάζονται τα αποτελέσματα της εκτέλεσης.

Στο έβδομο κεφάλαιο γίνεται ένα evaluation της εφαρμογής και παρουσιάζονται τα αποτελέσματα.

Στο τελευταίο κεφάλαιο γίνεται μία ανακεφαλαίωση των όσων παρουσιάστηκαν στην παρούσα Διπλωματική εργασία και παρατίθενται μελλοντικές κατευθύνσεις και επεκτάσεις..

Ευχαριστίες

Η δουλειά αυτή πραγματοποιήθηκε στα πλαίσια διπλωματικής εργασίας για την απόκτηση του διπλώματος Μηχανικού Πληροφορικής και Τηλεπικοινωνιών από το Πανεπιστήμιο Δυτικής Μακεδονίας. Θα ήθελα να ευχαριστήσω τον κ. Παναγιώτη Σαρηγιαννίδη που σαν υπεύθυνος καθηγητής ήταν πάντα παρών για να με καθοδηγήσει όποτε χρειάστηκα βοήθεια στην εκπόνηση αυτής της Διπλωματικής εργασίας. Θα ήθελα επίσης να ευχαριστήσω όλους τους υπόλοιπους καθηγητές του Πανεπιστημίου, που χωρίς τις γνώσεις που μου μετέδωσαν τίποτα από όλο αυτό δεν θα μπορούσε να πραγματοποιηθεί.

Έπειτα θα ήθελα να ευχαριστήσω την κοινότητα του stackoverflow.com που βοήθησε αισθητά στην αντιμετώπιση των προγραμματιστικών προβλημάτων. Τέλος το μεγαλύτερο ευχαριστώ το οφείλω στους γονείς μου που με στήριξαν καθ’ όλη την διάρκεια των δύσκολων χρόνων των σπουδών μου και μου προσέφεραν εγκάρδια ότι βοήθεια χρειάστηκα.

Το πρότυπο IEEE 802.15.4

Το Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (Institute of Electrical and Electronics Engineers – IEEE) υποστηρίζει πολλές ομάδες εργασίας για να αναπτύξουν και να διατηρήσουν ενσύρματα και ασύρματα πρότυπα επικοινωνίας. Για παράδειγμα το πρότυπο 802.3 είναι το ενσύρματο Ethernet και το πρότυπο 802.11 είναι για ασύρματα τοπικά δίκτυα (ΑΤΔ - WLANs) ή τα γνωστά σε όλους μας Wi-Fi. Στην ομάδα προτύπων 802.15 καθορίζονται τα Προσωπικά Ασύρματα Δίκτυα (ΠΑΔ – WPANs). Για παράδειγμα το 802.15.1 είναι το Bluetooth, το 802.15.3 είναι μία υψηλού ρυθμού δεδομένων κατηγορία για Υπέρ Ευρείας Ζώνης (ΥΕΖ) τεχνολογίες και το 802.15.6 είναι για δίκτυα σώματος (ΔΣ - BAN). Πέρα από αυτά υπάρχουν και άλλα.

Η κατηγορία 802.15.4 [44] είναι πιθανώς το μεγαλύτερο πρότυπο για Χαμηλού Ρυθμού δεδομένων ΠΑΔ (ΧΡ-ΠΑΔ). Ένα ΧΡ-ΠΑΔ είναι ένα απλό χαμηλού κόστους δίκτυο επικοινωνιών που επιτρέπει ασύρματη σύνδεση σε εφαρμογές με περιορισμένη ισχύ και χαλαρές απαιτήσεις ρυθμαπόδοσης. Οι βασικοί στόχοι ενός ΧΡ-ΠΑΔ είναι η ευκολία εγκατάστασης, η αξιόπιστη μεταφορά δεδομένων, το υπερβολικά χαμηλό κόστος και η ικανοποιητική διάρκεια ζωής μπαταρίας, διατηρώντας ταυτόχρονα ένα απλό και ευέλικτο πρωτόκολλο. Το βασικό πρότυπο με τις πιο πρόσφατες ενημερώσεις και βελτιώσεις είναι το 802.15.4a/b με το 802.15.4c για την Κίνα, το 802.15.4d για την Ιαπωνία, το 802.15.4e για βιομηχανικές εφαρμογές, το 802.15.4f για chips Αναγνώρισης ΡαδιοΣυχνότητας (ΑΡΣ – RFID), και το 802.15.4g για Έξυπνα Δίκτυα Κοινής Ωφέλειας (ΕΔΚΩ) και για παρακολούθηση του Έξυπνου δικτύου ενέργειας (Smart Grid). Όλες αυτές οι ειδικές εκδόσεις χρησιμοποιούν την ίδια βασική ραδιοφωνική τεχνολογία και το πρωτόκολλο που καθορίζεται στο 802.15.4a/b

Μερικές από τις δυνατότητες αυτού του προτύπου είναι οι παρακάτω:

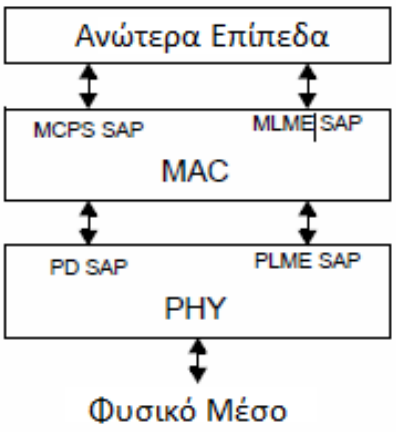
- Λειτουργία αστέρα ή peer-to-peer
- Μοναδική 64-bit εκτεταμένη διεύθυνση ή 16-bit κατανεμημένη σύντομη διεύθυνση
- Προαιρετική κατανομή εγγυημένων χρονοθυρίδων (ΕΧ - guaranteed time slots – GTSs)
- CSMA-CA ή ALOHA πρόσβαση καναλιού
- Πλήρως αναγνωρισμένο πρωτόκολλο για αξιοπιστία μετάδοσης
- Μικρή κατανάλωση ισχύος.
- Ανίχνευση Ενέργειας (ΑΕ)
- Ένδειξη Ποιότητας Ζεύξης (ΕΠΖ)

Υπάρχουν δύο ειδών συσκευές που μπορούν να συμμετέχουν σε ένα 802.15.4 δίκτυο: μία Πλήρως Λειτουργική Συσκευή (ΠΛΣ) και μία Μερικώς Λειτουργική Συσκευή (ΜΛΣ). Μία ΠΛΣ μπορεί να λειτουργεί σαν συντονιστής προσωπικού δικτύου (ΠΑΔ) ή σαν συντονιστής. Μία ΜΛΣ συσκευή δεν μπορεί να λειτουργήσει σαν συντονιστής ή συντονιστής προσωπικού δικτύου. Οι ΜΛΣ προορίζονται για υπερβολικά απλές

εφαρμογές, όπως διακόπτες για φώτα ή παθητικοί αισθητήρες υπερύθρων, που δεν χρειάζεται να στέλνουν μεγάλες ποσότητες δεδομένων και συνεργάζονται με μόνο μία ΠΛΣ την φορά. Κατά συνέπεια η ΜΛΣ μπορεί να υλοποιηθεί χρησιμοποιώντας τους ελάχιστους πόρους και χωρητικότητα μνήμης.

Αρχιτεκτονική

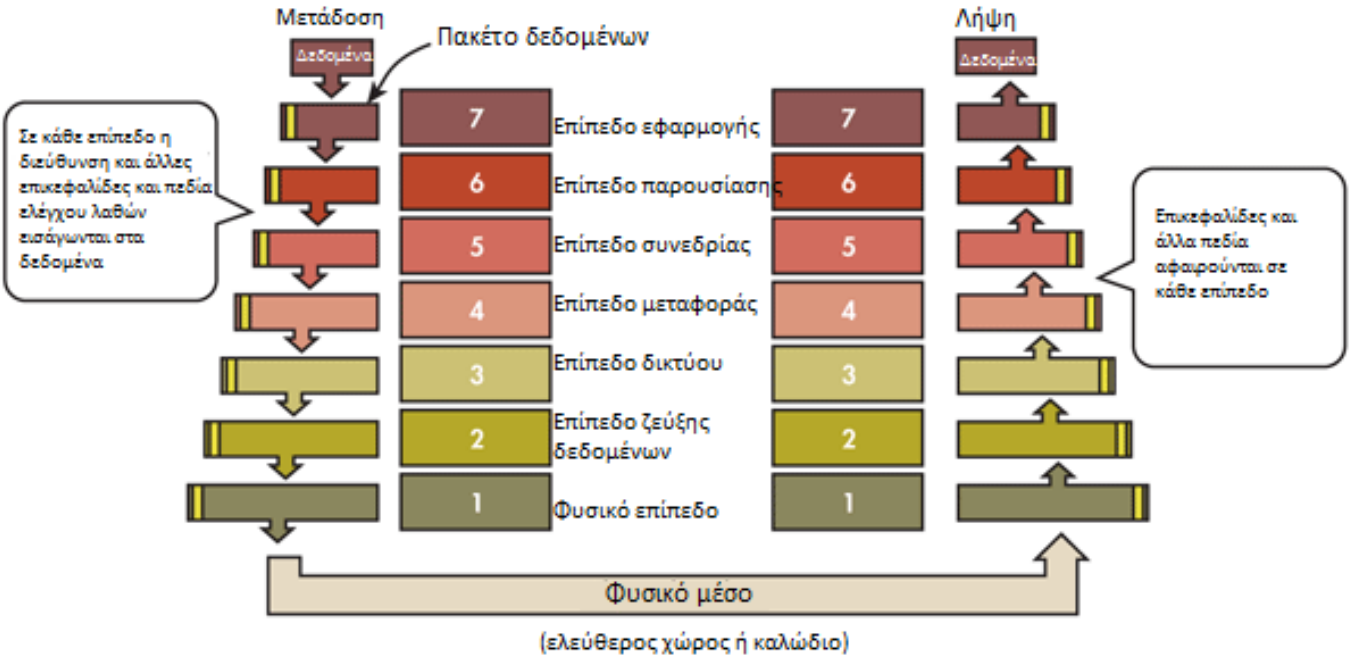
Η αρχιτεκτονική του IEEE 802.15.4 διαχωρίζεται σε επίπεδα για να απλοποιηθεί το πρότυπο. Κάθε επίπεδο είναι υπεύθυνο για ένα κομμάτι του προτύπου και προσφέρει υπηρεσίες στα υψηλοτέρα επίπεδα. Οι διεπαφές μεταξύ των επιπέδων καθορίζουν τις λογικές ζεύξεις που περιγράφονται σε αυτό το πρότυπο. Μία συσκευή ΧΡ-ΠΑΔ αποτελείται από τουλάχιστον ένα PHY επίπεδο, που περιέχει τον πομποδέκτη ραδιοσυχνοτήτων μαζί με τον χαμηλού επιπέδου μηχανισμό ελέγχου, και ένα υποεπίπεδο MAC που παρέχει πρόσβαση στο φυσικό κανάλι για όλους τους τύπους μεταφορών. Τα ανώτερα επίπεδα που φαίνονται στην Εικόνα 1 αποτελούνται από το επίπεδο δικτύου, το οποίο παρέχει την διαμόρφωση του δικτύου, χειραγώγηση και δρομολόγηση μηνυμάτων, και το επίπεδο εφαρμογής που παρέχει την προβλεπόμενη λειτουργία της συσκευής. Όπως μπορούμε εύκολα να διαπιστώσουμε, το πρότυπο αυτό καθορίζει μόνο 2 από τα επίπεδα του OSI (Εικόνα 2), ενώ τα περισσότερα συστήματα χρησιμοποιούν τουλάχιστον τέσσερα.



Εικόνα.1: Τα επίπεδα του προτύπου IEEE 802.15.4 *[44]*

Φυσικό επίπεδο (PHY)

Το PHY παρέχει δύο υπηρεσίες: την υπηρεσία δεδομένων PHY και την υπηρεσία διαχείρισης PHY. Η πρώτη επιτρέπει την μετάδοση και λήψη των μονάδων δεδομένων πρωτοκόλλου PHY (ΜΔΠ PHY) διαμέσου του φυσικού ραδιοφωνικού καναλιού. Τα χαρακτηριστικά του PHY είναι η ενεργοποίηση και απενεργοποίηση του πομποδέκτη, η ΑΕ, ΕΠΖ, επιλογή καναλιού, εκτίμηση ελεύθερου καναλιού (ΕΕΚ) και μετάδοση και λήψη πακέτων διαμέσου του φυσικού μέσου.



Εικόνα.2: Επίπεδα του OSI *[45]*

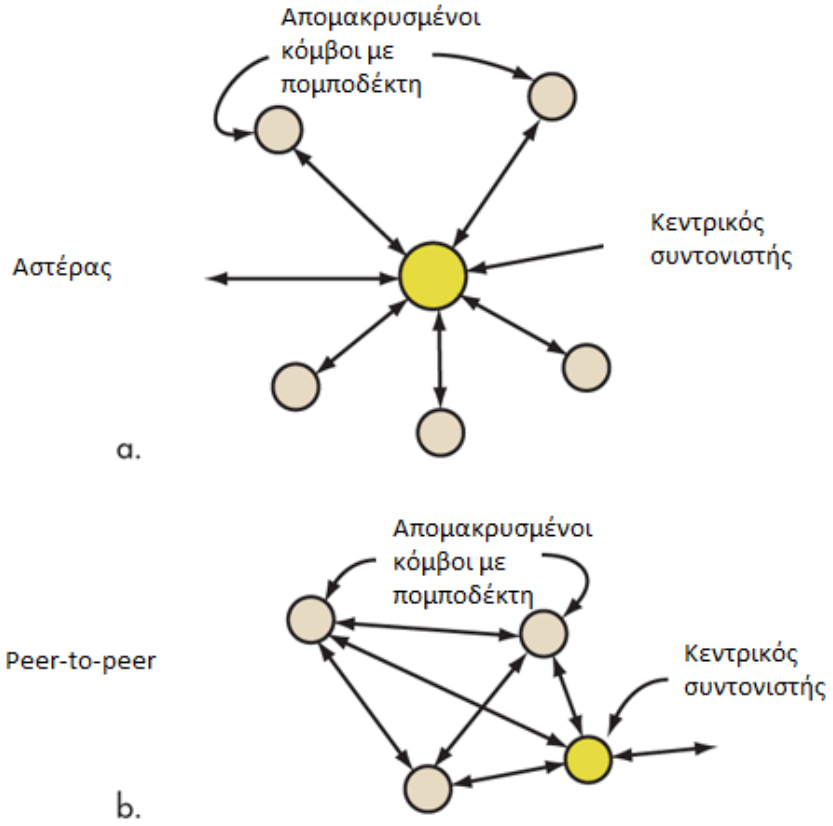
Υποεπίπεδο MAC

Το υποεπίπεδο MAC παρέχει και αυτό δύο υπηρεσίες: την υπηρεσία δεδομένων MAC και την υπηρεσία διαχείρισης MAC που λειτουργεί ως διεπαφή για την *«Οντότητα Διαχείρισης Υποεπιπέδου MAC» (ΟΔΥΜ) - σημείο πρόσβασης υπηρεσίας (ΣΠΥ)»* γνωστή ως ΟΔΥΜ-ΣΠΥ. Η υπηρεσία δεδομένων MAC επιτρέπει την μετάδοση και λήψη των ΜΔΠ MAC (ΜΔΠΜ) διαμέσου της υπηρεσίας δεδομένων του PHY. Τα χαρακτηριστικά του MAC περιλαμβάνουν: διαχείριση αναγνωριστικών, πρόσβαση στο κανάλι, διαχείριση ΕΧ, επιβεβαίωση πλαισίων, αναγνωρισμένη παράδοση πλαισίων, σύνδεση και αποσύνδεση. Επιπλέον το υποεπίπεδο MAC παρέχει άγκιστρα για την υλοποίηση μηχανισμών ασφαλείας κατάλληλων για εφαρμογές.

Τοπολογίες δικτύου

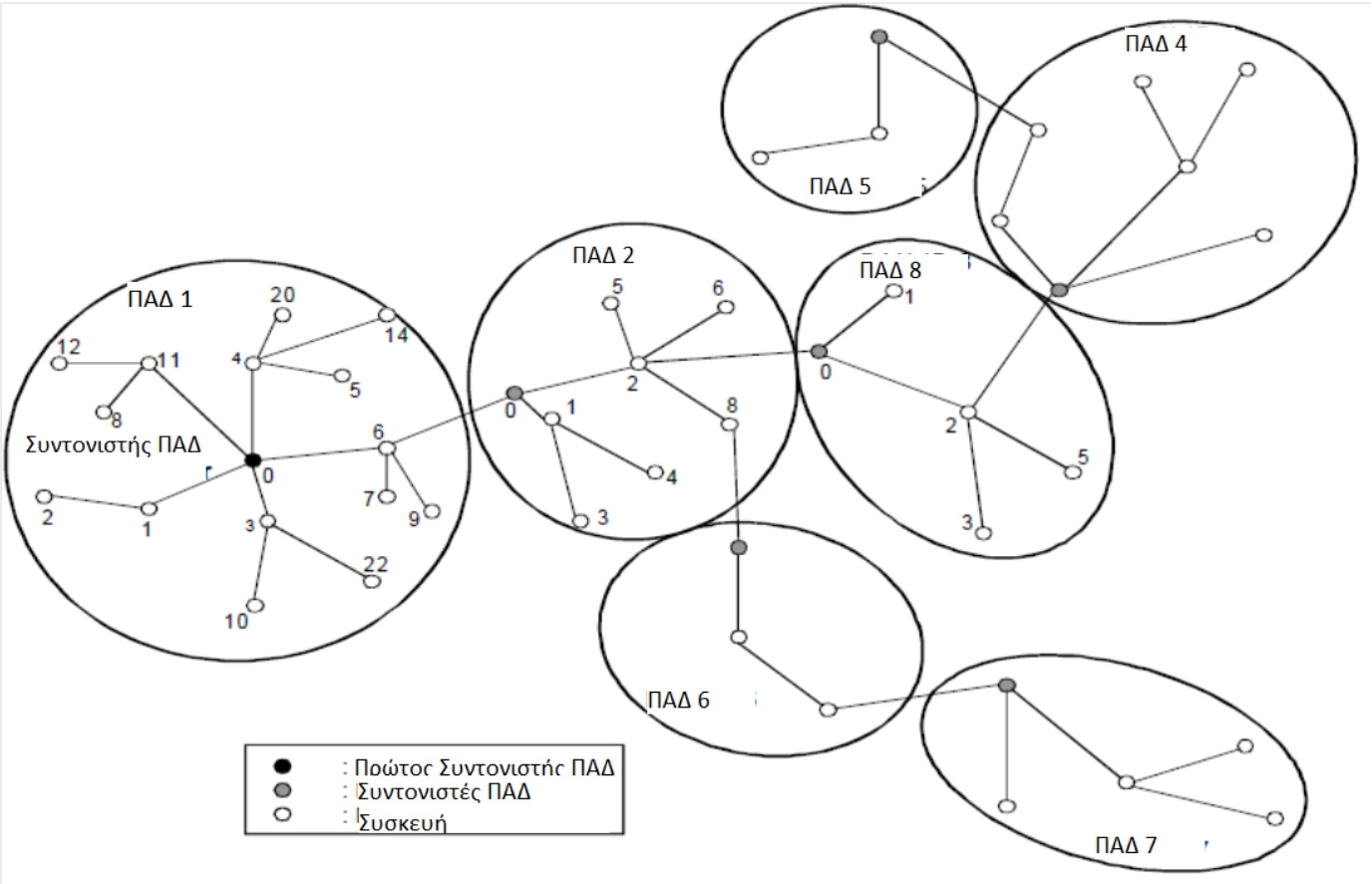
Ανάλογα με τις απαιτήσεις του δικτύου ένα ΧΡ-ΠΑΔ λειτουργεί σε μία από τις δύο τοπολογίες: τοπολογία αστέρα ή τοπολογία peer-to-peer (Εικόνα 3). Στην τοπολογία αστέρα η επικοινωνία εκτελείται μεταξύ των συσκευών και ενός κεντρικού ελεγκτή, που ονομάζεται συντονιστής ΠΑΔ. Η συσκευή συνήθως εκτελεί κάποια εφαρμογή και είναι είτε η πηγή είτε ο προορισμός της επικοινωνίας. Ο συντονιστής ΠΑΔ είναι ο βασικός ελεγκτής του ΠΑΔ και μπορεί να εκτελεί μία συγκεκριμένη εφαρμογή, αλλά χρησιμοποιείται κυρίως για να εκκινήσει, να τερματίσει ή να δρομολογήσει τις επικοινωνίες του δικτύου. Όλες οι συσκευές του δικτύου οποιασδήποτε τοπολογίες έχουν μοναδικές διευθύνσεις που ονομάζονται εκτεταμένες διευθύνσεις (extended addresses). Η συσκευή θα χρησιμοποιήσει την εκτεταμένη διεύθυνση για επικοινωνίες μέσα στο ΠΑΔ ή την σύντομη διεύθυνση (short address) που της δόθηκε από τον συντονιστή όταν συνδέθηκε. Ο συντονιστής ΠΑΔ είναι συνήθως συνδεδεμένος σε πρίζα

ενώ οι υπόλοιπες συσκευές τροφοδοτούνται με μπαταρία. Οι εφαρμογές που ωφελούνται από την τοπολογία αστέρα περιλαμβάνουν αυτοματισμούς σπιτιών, προσωπικούς υπολογιστές, περιφερειακά, παιχνίδια και προσωπική υγεία.



Εικόνα.3:Τοπολογίες a)Αστέρα, b)Peer-to-peer *[45]*

Η τοπολογία peer-to-peer έχει και αυτή έναν συντονιστή ΠΑΔ, όμως διαφέρει από την τοπολογία αστέρα επειδή κάθε συσκευή μπορεί να επικοινωνήσει απευθείας με οποιαδήποτε άλλη, αρκεί να βρίσκεται εντός της εμβέλειάς της. Αυτή η τοπολογία επιτρέπει την δημιουργία πιο πολύπλοκων σχηματισμών δικτύου, όπως η τοπολογία πλέγματος (mesh). Εφαρμογές όπως ασύρματα δίκτυα αισθητήρων, βιομηχανικός έλεγχος, παρακολούθηση κεφαλαίων και εμπορευμάτων, έξυπνη γεωργία και ασφάλεια χρησιμοποιούν την τοπολογία peer-to-peer. Ένα παράδειγμα peer-to-peer τοπολογίας είναι το δέντρο συστάδων (cluster tree - Εικόνα 4). Στο δίκτυο δέντρου συστάδων οι περισσότερες συσκευές είναι ΠΛΣ με τις ΜΛΣ να συνδέονται ως φύλλα στο τέλος του κλαδιού επειδή δεν επιτρέπουν σε άλλες συσκευές να συνδεθούν πάνω τους. Κάθε ΠΛΣ μπορεί να λειτουργήσει ως συντονιστής και να παρέχει υπηρεσίες συγχρονισμού στις άλλες συσκευές ή συντονιστές. Μόνο ένας όμως από τους συντονιστές είναι ο κύριος συντονιστής ΠΑΔ, συνήθως επειδή έχει την μεγαλύτερη υπολογιστική ισχύ από τους υπόλοιπους.



Εικόνα.4: Δίκτυο δέντρου συστάδων *[44]*

Ο συντονιστής ΠΑΔ δημιουργεί την πρώτη συστάδα (cluster) επιλέγοντας ένα αχρησιμοποίητο αναγνωριστικό ΠΑΔ και εκπέμποντας αναγνωριστικά πλαίσια στους γείτονες του. Μία συσκευή που θα λάβει ένα από τα πλαίσια θα αιτηθεί από τον συντονιστή ΠΑΔ να εισέλθει στο δίκτυο. Εάν η αίτηση γίνει αποδεκτή ο συντονιστής εισάγει την συσκευή σαν παιδί του στην λίστα γειτόνων. Έπειτα η νέα συσκευή στο δίκτυο προσθέτει τον συντονιστή ΠΑΔ σαν πατέρα της στην λίστα γειτόνων και εκπέμπει αναγνωριστικά πλαίσια προς στους δικούς της γείτονες. Όσοι λάβουν αυτά τα πλαίσια μπορούν να εισέλθουν στο δίκτυο μέσω αυτής της συσκευής. Αν η πρώτη συσκευή δεν καταφέρει να μπει στο δίκτυο του συντονιστή ΠΑΔ θα προσπαθήσει να βρει κάποια άλλη συσκευή για πατέρα της.

Διαχείριση συχνοτήτων [45]

Ο σκοπός του προτύπου είναι να παρέχει τη βασική διαμόρφωση πάνω στην οποία θα μπορούν να προστεθούν άλλα πρωτόκολλα και λειτουργίες από τα ανώτερα επίπεδα (3 έως 7). Ενώ υπάρχουν τρεις αναθέσεις συχνοτήτων (Εικόνα 4), αυτή των 2.4GHz είναι η πιο κοινή. Οι περισσότερες συσκευές χρησιμοποιούν την δημοφιλή ISM band.

Το πρότυπο χρησιμοποιεί την διαμόρφωση άμεσης ακολουθίας εκτεταμένου φάσματος (*ΑΑΕΦ)*. Έχει υψηλές ανοχές σε θόρυβο και παρεμβολές και προσφέρει απολαβή κωδικοποίησης για να βελτιώσει την αξιοπιστία της ζεύξης. Το *Standard*

binary phase-shift keying (BPSK) χρησιμοποιείται στις δύο εκδόσεις χαμηλών ταχυτήτων και το *offset-quadrature phase-shift keying (O-QPSK)* χρησιμοποιείται στην έκδοση υψηλότερου ρυθμού δεδομένων. Το O-QPSK έχει σταθερή περιβάλλουσα κύματος που σημαίνει ότι μπορούν να χρησιμοποιηθούν πιο αποδοτικές μη-γραμμικές τεχνικές ενίσχυσης ισχύος για την ελαχιστοποίηση κατανάλωσης ενέργειας.

ΕΠΙΛΟΓΕΣ ΑΝΑΘΕΣΗΣ ΣΥΧΝΟΤΗΤΑΣ			
Γεωγραφικές Περιοχές	**Ευρώπη**	**Αμερική**	**Παγκοσμίως**
Ανάθεση συχνότητας	868 με 868.6MHz	902 με 928 MHz	2.4 με 2.4835GHz
Αριθμός καναλιών	1	10	16
Εύρος ζώνης	600 KHz	2 MHz	5 MHz
Ρυθμός συμβόλων	20 ksymbols/s	40 ksymbols/s	62.5 ksymbols/s
Ρυθμός δεδομένων	20 kbit/s	40 kbit/s	250 kbit/s
Διαμόρφωση	BPSK	BPSK	Q-QPSK

Πίνακας. 1: Επιλογές ανάθεση συχνότητας του προτύπου IEEE 802.15.4 *[45]*

Όσον αφορά την πρόσβαση στο κανάλι, το 802.15.4 χρησιμοποιεί *αίσθηση του μεταφορέα πολλαπλής πρόσβασης με αποφυγή των συγκρούσεων (*ΑΜΠΠ-ΑΣ*).* Αυτού του είδους η πολυπλεξία επιτρέπει σε πολλούς κόμβους να έχουν πρόσβαση στο ίδιο κανάλι σε διαφορετικές χρονικές στιγμές χωρίς παρεμβολές. Οι περισσότερες μεταδόσεις αποτελούνται από μικρές ριπές πακέτων που όμως δεν συμβαίνουν καθόλου συχνά (<1%), ελαχιστοποιώντας έτσι την κατανάλωση ισχύος. Το ελάχιστο επίπεδο ισχύος καθορίζεται στα -3dBm ή 0,5mW. Οι περισσότερες συσκευές χρησιμοποιούν 0dBm ή 1mW, όμως υπάρχουν και άλλες με 20dBm ή 100mW.

Η εμβέλεια μετάδοσης εξαρτάται πολύ από το είδος της διαδρομής και πρέπει στο μεγαλύτερο κομμάτι να υπάρχει οπτική επαφή. Μεγάλο ρόλο παίζουν επίσης η ισχύς του πομπού αλλά και η ευαισθησία του δέκτη. Υπό τις βέλτιστες προϋποθέσεις σε μία εξωτερική περιοχή χωρίς εμπόδια η εμβέλεια μπορεί να φτάσει το 1km. Στις περισσότερες εφαρμογές όμως η εμβέλεια είναι πολύ μικρότερη, της τάξης των 10 με 75 μέτρων.

Εισαγωγή στα ασύρματα δίκτυα αισθητήρων

Ένα ασύρματο δίκτυο αισθητήρων (ΑΔΑ - Wireless Sensor Network) αποτελείται από αυτόνομους αισθητήρες κατανεμημένους σε μία περιοχή με σκοπό την απόκτηση μετρήσεων για φυσικές ή περιβαλλοντολογικές καταστάσεις. Οι μετρήσεις αυτές περιλαμβάνουν την θερμοκρασία, τον ήχο, ατμοσφαιρική πίεση, ένταση φωτός κ.α. Οι αισθητήρες είναι συνδεδεμένοι μεταξύ τους σε ένα ad-hoc δίκτυο και μεταδίδουν τα δεδομένα τους σε έναν κεντρικό κόμβο. Τα περισσότερα σύγχρονα δίκτυα είναι δύο κατευθύνσεων (bi-directional) επιτρέποντας έτσι τον απομακρυσμένο έλεγχο των αισθητήρων. Η ανάπτυξη των ΑΔΑ ξεκίνησε από στρατιωτικές εφαρμογές, όπως παρακολούθηση πεδίου μάχης, σήμερα όμως τέτοια δίκτυα χρησιμοποιούνται ευρέως στην βιομηχανία, αλλά και σε εμπορικές εφαρμογές.

Ένα ΑΔΑ αποτελείται από κόμβους, με τον αριθμό τους να κυμαίνεται από δεκάδες μέχρι και μερικές χιλιάδες. Ο κάθε κόμβος συνδέεται πάντα με έναν ή πιο σπάνια και με περισσότερους κόμβους. Οι κόμβοι αποτελούνται από διάφορα μέρη: έναν πομποδέκτη με ενσωματωμένη ή εξωτερική κεραία, έναν μικροεπεξεργαστή που αποτελεί την διεπαφή με τους αισθητήρες και μία πηγή ενέργειας, συνήθως μπαταρία ή κάποιας μορφής συλλέκτη ενέργειας. Το μέγεθος των αισθητήρων ποικίλει από αυτό ενός κουτιού παπουτσιών μέχρι ενός κόκκου σκόνης. Το κόστος τους είναι και αυτό μεταβλητό, συνήθως μερικές εκατοντάδες ευρώ, και εξαρτάται από την πολυπλοκότητα, το μέγεθος και το είδος των μετρήσεων που λαμβάνει. Οι περιορισμοί του κόστους και του μεγέθους είναι ανάλογοι των περιορισμών των πόρων του αισθητήρα, όπως η ενέργεια, η μνήμη, η επεξεργαστική ισχύς και το εύρος ζώνης. Υπάρχουν διάφορες τοπολογίες ΑΔΑ όπως αυτή ενός απλού αστέρα ή ενός δικτύου πλέγματος πολλαπλών ανακλάσεων (multi-hop mesh network). Η μετάδοση δεδομένων μέσα στο δίκτυο μπορεί να γίνεται με τη μέθοδο της πλημμύρας ή με διάφορους αλγορίθμους δρομολόγησης. Τέλος στα ασύρματα δίκτυα αισθητήρων γίνεται συνεχής έρευνα από επιστήμονες πληροφορικής και τηλεπικοινωνιών και οργανώνονται κάθε χρόνο workshops και συνέδρια. Κάποια από αυτά είναι το IPSN, SenSys και EWSN.

Εφαρμογές

Παρακολούθηση Περιοχής

Η παρακολούθηση μίας περιοχής είναι από τις πιο κοινές εφαρμογές των ΑΔΑ. Ένα ασύρματο δίκτυο αισθητήρων τοποθετείται σε μία περιοχή για την παρακολούθηση κάποιου φαινομένου. Μία στρατιωτική εφαρμογή είναι η χρήση αισθητήρων για την ανίχνευση εχθρών και μία εμπορική εφαρμογή είναι η χαρτογράφηση αγωγών αερίου ή πετρελαίου.

Ιατρικές εφαρμογές

Υπάρχουν δύο είδη ιατρικών εφαρμογών, συσκευές που φοριούνται εξωτερικά και συσκευές που εμφυτεύονται στο σώμα του ασθενούς.

Περιβαλλοντικές εφαρμογές

- Ατμοσφαιρική ρύπανση
- Ανίχνευση πυρκαγιάς σε δάση
- Ανίχνευση κατολισθήσεων
- Παρακολούθηση της ποιότητας του νερού
- Πρόληψη φυσικών καταστροφών (πχ πλημμύρες)
- Ανίχνευση επικίνδυνων χημικών ουσιών

Βιομηχανικές Εφαρμογές

- Παρακολούθηση «υγείας» μηχανημάτων
- Παρακολούθηση υδάτινων αποβλήτων
- Παρακολούθηση «υγείας» κτηρίων

Χαρακτηριστικά

Τα βασικά χαρακτηριστικά ενός ΑΔΑ περιλαμβάνουν:

- Ενεργειακούς περιορισμούς για τους κόμβους που τροφοδοτούνται με μπαταρία
- Δυνατότητα αντιμετώπισης αποτυχίας κάποιου κόμβου
- Κινητικότητα των κόμβων
- Ετερογένεια των κόμβων
- Δυνατότητα κλιμάκωσης σε μεγάλη κλίμακα ανάπτυξης
- Αντοχή σε δυσμενείς περιβαλλοντικές συνθήκες
- Ευκολία στην χρήση
- Διαστρωματικός σχεδιασμός (Cross-layer design)

Ο διαστρωματικός σχεδιασμός γίνεται ολοένα και πιο σημαντικό αντικείμενο έρευνας στον τομέα των ασύρματων επικοινωνιών. Η παραδοσιακή προσέγγιση με επίπεδα παρουσιάζει τρία σημαντικά προβλήματα

1) Οι πληροφορίες δεν μπορούν να διαμοιραστούν ανάμεσα στα διάφορα επίπεδα, με αποτέλεσμα η προσέγγιση αυτή να μην μπορεί να εγγυηθεί την βελτιστοποίηση όλου του δικτύου.
2) Δεν έχει τη δυνατότητα να προσαρμόζεται σε περιβαλλοντικές αλλαγές
3) Εξαιτίας της παρεμβολής διάφορων χρηστών, αλληλοεπικαλύψεων πρόσβασης και την αλλαγή του περιβάλλοντος στα ασύρματα δίκτυα αισθητήρων, η παραδοσιακή προσέγγιση δεν μπορεί να εφαρμοστεί.

Επομένως ο διαστρωματικός σχεδιασμός μπορεί να χρησιμοποιηθεί για να κάνει την βέλτιστη διαμόρφωση για να βελτιώσει τις επιδόσεις μετάδοσης όπως ρυθμός δεδομένων, ενεργειακή απόδοση, ποιότητα της υπηρεσίας (ΠτΥ) κλπ.

Επιθέσεις στα ασύρματα δίκτυα αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων αποκτούν όλο και περισσότερο ενδιαφέρον στην επιστημονική κοινότητα λόγω των μοναδικών τους χαρακτηριστικών. [2] Εκτός από τους ενεργειακούς περιορισμούς η ασφάλεια παίζει εξίσου σημαντικό ρόλο στον σχεδιασμό του δικτύου. Ο σχεδιασμός όμως πρωτοκόλλων ασφαλείας είναι ένα δύσκολο εγχείρημα για τους εξής λόγους: [16]

- Τα ασύρματα κανάλια είναι ανοιχτά σε όλους και έχουν την ασύρματη διεπαφή τους ρυθμισμένη στην ίδια συχνότητα. Έτσι όλοι μπορούν να παρακολουθήσουν και να συμμετέχουν στην επικοινωνία που διεξάγεται στο κανάλι, παρέχοντας στους κακόβουλους χρήστες έναν βολικό τρόπο να εισχωρήσουν στο δίκτυο.
- Όπως και στο Internet, έτσι και στα ΑΔΑ τα περισσότερα πρωτόκολλα δεν έχουν λάβει υπόψη τους απαραίτητους μηχανισμούς ασφαλείας κατά τη διάρκεια του σχεδιασμού. Επίσης τα περισσότερα πρωτόκολλα είναι ευρέως γνωστά λόγω των αναγκών για τυποποίηση. Για τους λόγους αυτούς μπορούν εύκολα να πραγματοποιηθούν επιθέσεις εκμεταλλευόμενες τις τρύπες ασφαλείας στα πρωτόκολλα αυτά.
- Οι περιορισμένοι πόροι των αισθητήρων είναι ένα εμπόδιο στην ανάπτυξη ισχυρών αλγορίθμων ασφαλείας εξαιτίας της πολυπλοκότητάς τους. Επιπλέον μεγάλοι αριθμοί κόμβων έχουν ανάγκη από απλά, ευέλικτα και επεκτάσιμα πρωτόκολλα ασφαλείας.
- Ένα ισχυρό πρωτόκολλο ασφαλείας καταναλώνει πολλούς πόρους του αισθητήρα και μπορεί να οδηγήσει σε πτώση της απόδοσης του. Στις περισσότερες περιπτώσεις είναι αναγκαίο να γίνει ένας συμβιβασμός ανάμεσα στην ασφάλεια και στην απόδοση. Τα «ελαφριά» όμως πρωτόκολλα ασφαλείας δεν είναι ισχυρά και μπορούν να παραβιαστούν.
- Τα ΑΔΑ συνήθως στήνονται σε εχθρικές περιοχές χωρίς καμία σταθερή υποδομή. Είναι συνεπώς πολύ δύσκολο να πραγματοποιείται συνεχής παρακολούθηση του δικτύου

Απαιτήσεις ασφαλείας

Ένα ΑΔΑ πρέπει να πληροί τις παρακάτω απαιτήσεις ασφαλείας *[17] με [22]*:

Απόρρητό των δεδομένων είναι η εξασφάλιση εξουσιοδοτημένης πρόσβασης στις πληροφορίες. Είναι η ικανότητα του δικτύου να αποκρύπτει μηνύματα από έναν παθητικό επιτιθέμενο έτσι ώστε κάθε μήνυμα που δρομολογείται να μένει απόρρητο. Έτσι εξασφαλίζεται η προστασία ευαίσθητων πληροφοριών ενάντια σε υποκλοπές.

Αυθεντικότητα των δεδομένων είναι η εξασφάλιση της ταυτότητας των κόμβων που βρίσκονται σε επικοινωνία. Στα ΑΔΑ δρομολογούνται ευαίσθητα δεδομένα που βοηθούν στην λήψη σημαντικών αποφάσεων. Συνεπώς κρίνεται απαραίτητο κάθε κόμβος να γνωρίζει ότι τα πακέτα που λαμβάνει προέρχονται από έναν πραγματικό

αποστολέα. Σε αντίθετη περίπτωση ο κόμβος μπορεί να ξεγελαστεί και να πραγματοποιήσει λανθασμένες ενέργειες.

Ακεραιότητα των δεδομένων είναι η εξασφάλιση ότι η πληροφορία δεν έχει αλλοιωθεί κατά την μετάδοση λόγω κάποιας επίθεσης είτε κατά λάθος. Αυτή είναι μία από τις βασικές απαιτήσεις κατά την επικοινωνία γιατί ο παραλήπτης πρέπει να γνωρίζει ακριβώς τι θέλει να του πει ο αποστολέας. Στις ασύρματες επικοινωνίες όμως κάτι τέτοιο δεν είναι πάντα εύκολο να εξασφαλιστεί.

Φρεσκάδα των δεδομένων είναι η εξασφάλιση ότι κάθε πακέτο που μεταδίδεται είναι πρόσφατο και ότι δεν επαναλαμβάνονται παλαιότερα μηνύματα. Όλες οι πληροφορίες μου μεταδίδονται σε ένα δίκτυο περιγράφουν μία προσωρινή κατάσταση ενός αντικειμένου και είναι έγκυρες για ένα περιορισμένο χρονικό διάστημα. Έτσι κάθε κόμβος πρέπει να είναι βέβαιος ότι το πακέτο που έλαβε είναι πρόσφατο, διαφορετικά το πακέτο είναι άχρηστο στην καλύτερη, ή επικίνδυνο στην χειρότερη περίπτωση. Η επανάληψη παλιών πακέτων είναι μία σοβαρή απειλή που μπορεί να προκαλέσει σύγχυση στο δίκτυο και να οδηγήσει σε λανθασμένες ενέργειες.

Διαθεσιμότητα είναι η εξασφάλιση της ικανότητας να μπορούν να παρέχονται οι αναμενόμενες υπηρεσίες όπως αυτές έχουν σχεδιαστεί αρχικά. Είναι μία πολύ ολοκληρωμένη αντίληψη, με την έννοια ότι σχετίζεται με όλες τις πτυχές του δικτύου. Η διαθεσιμότητα του δικτύου μπορεί να εκτεθεί σε περίπτωση που τελειώσει η μπαταρία κάποιων κόμβων, λόγο μία επίθεσης παρεμβολών (jamming attack) ή άρνησης υπηρεσίας (ΑτΥ).

Απειλές Ασφάλειας

Μία απειλή είναι μία συγκυρία ή ένα γεγονός που έχει υψηλή πιθανότητα να επιδράσει αρνητικά σε ένα δίκτυο μέσω μίας παραβίασης ασφαλείας. Η πιθανότητα ένας κακόβουλος χρήστης να εκμεταλλευτεί αυτή την απειλή και να προκαλέσει ζημιά στο δίκτυο ονομάζεται ρίσκο. Μπορούν να υπάρξουν πολλές απειλές για ένα ΑΔΑ, για παράδειγμα εξάντληση της πηγής ενέργειας, φυσική παραβίαση, καταστροφή αμέσως μετά το στήσιμο λόγω εχθρικού περιβάλλοντος ή εσκεμμένες προσπάθειες παραβίασης της ασφάλειας των κόμβων. Οι απειλές μπορούν να διαχωριστούν σε κατηγορίες: α) Παθητική συλλογή πληροφοριών, β) Υπονόμευση ενός κόμβου ή εισαγωγή κακόβουλου κόμβου, γ) Διακοπή λειτουργίας κόμβου, δ) Άρνηση υπηρεσίας, ε) Διαφθορά μηνύματος, στ) Ανάλυση κίνησης [37]. Ο εντοπισμός και η αντιμετώπιση των απειλών είναι πολύ σημαντικό να γίνει εγκαίρως και αποτελεσματικά, σε διαφορετική περίπτωση μπορεί να προκληθούν πολύ σοβαρά προβλήματα. Υπάρχουν εφαρμογές ΑΔΑ όπου η μη ορθή λειτουργία τους εξαιτίας μίας απειλής μπορεί να κοστίσει μέχρι και σε ανθρώπινες ζωές.

Σύμφωνα με τον Karlof et.al [28] οι επιθέσεις στα ΑΔΑ μπορούν να κατηγοριοποιηθούν ως εξής:

- **Εξωτερικές και εσωτερικές επιθέσεις:** Οι εξωτερικές επιθέσεις πραγματοποιούνται από κόμβους που δεν ανήκουν στο δίκτυο. Ο εξωτερικός επιτιθέμενος δεν έχει πρόσβαση στο μεγαλύτερο μέρους το κρυπτογραφικού υλικού ενός ΑΔΑ. Οι εσωτερικές επιθέσεις συμβαίνουν όταν νόμιμοι κόμβοι συμπεριφέρονται με μη αναμενόμενο ή μη εξουσιοδοτημένο τρόπο. Ο εσωτερικός επιτιθέμενος μπορεί να κατέχει μερικά από τα μυστικά κλειδιά και να έχει την εμπιστοσύνη των άλλων κόμβων. Οι εσωτερικές επιθέσεις είναι πολύ δυσκολότερο να ανιχνευτούν. Οι εξωτερικοί επιτιθέμενοι μπορούν να κρυφακούσουν μηνύματα μεταξύ κόμβων, να εισάγουν εσφαλμένα δεδομένα στο δίκτυο και να προκαλέσουν επιθέσεις AtY. Οι εσωτερικές επιθέσεις μπορούν να πραγματοποιηθούν είτε από παραβιασμένους κόμβους που εκτελούν κακόβουλο λογισμικό ή από επιτιθέμενους που έχουν κλέψει τα κλειδιά, τον κώδικα και τα δεδομένα από νόμιμους κόμβους και έπειτα χρησιμοποιούν μία ή περισσότερες συσκευές τύπου laptop για να επιτεθούν στο δίκτυο.

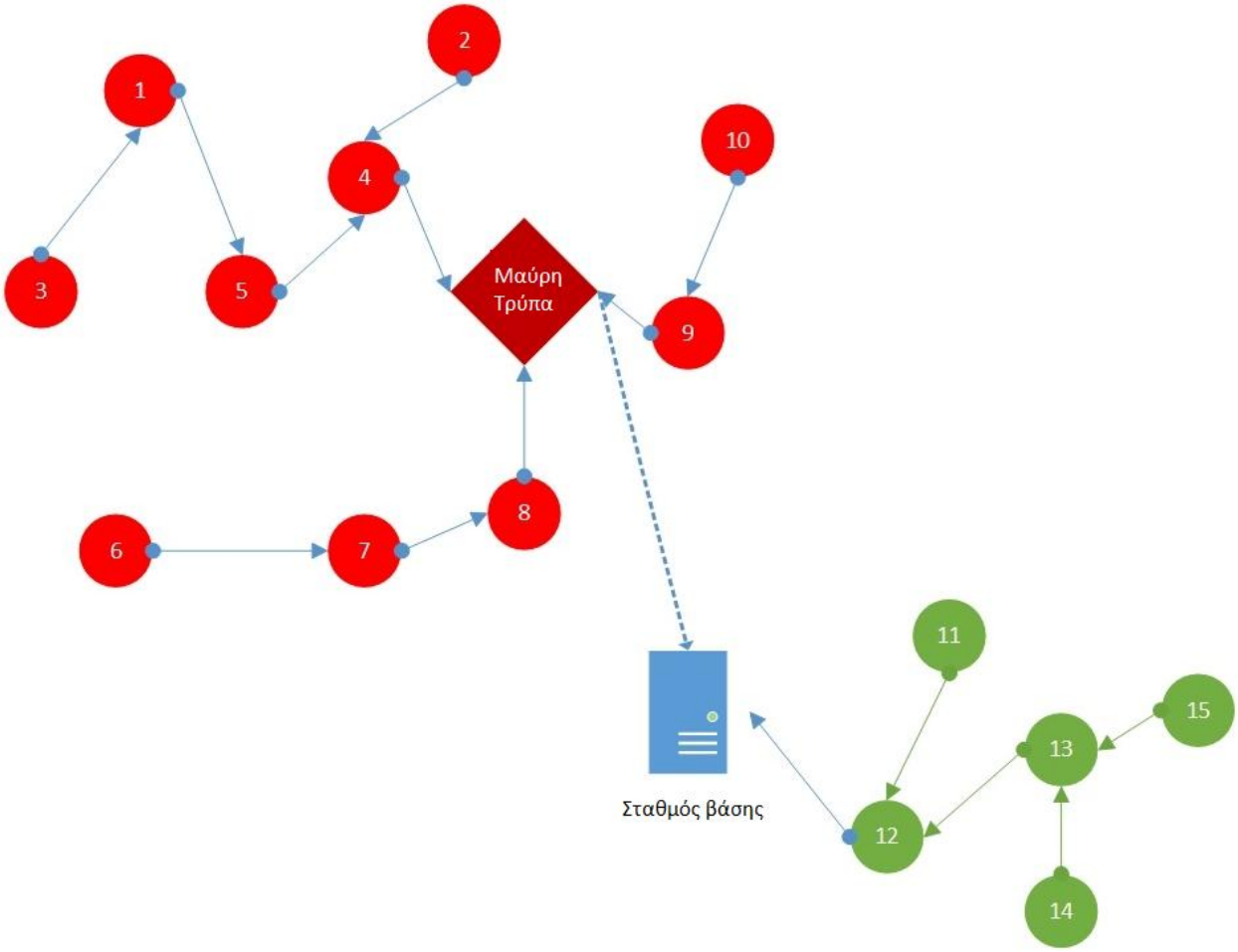
- **Παθητικές και ενεργητικές επιθέσεις:** Οι παθητικές επιθέσεις είναι της μορφής υποκλοπών ή παρακολούθησης της ανταλλαγής πακέτων σε ένα ΑΔΑ. Οι ενεργητικές επιθέσεις περιλαμβάνουν τροποποιήσεις δεδομένων ή εισαγωγή εσφαλμένων δεδομένων στο δίκτυο.

- **Επιθέσεις τύπου κόμβου (Node-class) και τύπου laptop (Laptop-class):** Στις επιθέσεις τύπου κόμβου ο κακόβουλος χρήστης επιτίθεται στο δίκτυο χρησιμοποιώντας κόμβους με παρόμοιες δυνατότητες με αυτούς του δικτύου. Στις επιθέσεις τύπου laptop γίνεται χρήση μίας πολύ ισχυρότερης συσκευής, όπως ένα laptop, και μπορεί να προκληθεί πολύ μεγαλύτερη ζημιά στο δίκτυο. Αυτού του είδους οι επιθέσεις μπορούν να προκαλέσουν παρεμβολές σε ολόκληρο το δίκτυο. Επίσης ένας επιτιθέμενος τύπου laptop έχει μεγαλύτερη ισχύ μπαταρίες, πολύ πιο δυνατή CPU, μεγαλύτερης ισχύος πομπό και πιο ευαίσθητο δέκτη και επομένως μπορούν να επηρεάσουν το δίκτυο πολύ περισσότερο από ότι οι απλοί κόμβοι. Τέλος ένας και μόνο επιτιθέμενος με laptop μπορεί να έχει τη δυνατότητα να κρυφακούσει τις επικοινωνίες όλων των κόμβων του δικτύου.

Στην εργασία αυτή θα ασχοληθούμε με τις πέντε πιο διαδεδομένες και σοβαρές επιθέσεις στα ασύρματα δίκτυα αισθητήρων:

1) Επίθεση μαύρης τρύπας (MT - Black hole Attack – BH),
2) Επίθεση σκουληκότρυπας (ΣT - Wormhole Attack – WH)
3) Επίθεση παρεμβολών (Jamming Attack)
4) Επίθεση Sybil
5) Επίθεση Hello.

Επίθεση Μαύρης Τρύπας



Εικόνα.6: Σχηματική απεικόνιση επίθεσης μαύρης τρύπας

Μία από τις κυριότερες και πιο σοβαρές επιθέσεις σε ένα ΑΔΑ είναι η επίθεση μαύρης τρύπας. Οι κακόβουλοι κόμβοι στο δίκτυο εκμεταλλεύονται το πρωτόκολλο δρομολόγησης διαφημίζοντας τον εαυτό τους ως τον κόμβο με το πιο σύντομο μονοπάτι προς τον κεντρικό κόμβο όπου πρέπει να σταλούν τα δεδομένα (ΚΚ - sink node). Η ΜΤ αποτελεί πολύ σοβαρή απειλή για το δίκτυο, καθώς οι κόμβοι που την πραγματοποιούν δεν προωθούν κανένα πακέτο από αυτά που λαμβάνουν, με αποτέλεσμα ο κεντρικός κόμβος να μπορεί να λάβει δεδομένα από τους κόμβους που έχουν επηρεαστεί. Η επίθεση αυτή ανήκει στην κατηγορία των επιθέσεων Άρνησης της Υπηρεσίας (ΑτΥ). Όταν ένας κόμβος πηγή θέλει να μεταδώσει δεδομένα προς τον ΚΚ, στέλνει ένα Route REQuest (RREQ) μήνυμα στους γειτονικούς του κόμβους. Οι γείτονες μόλις λάβουν το RREQ απαντούν με ένα Route REPly (RREP) μήνυμα, το οποίο περιέχει πληροφορίες για το κόστος και την ποιότητα της διαδρομής προς τον ΚΚ μέσω αυτών. Οι κακόβουλοι κόμβοι που πραγματοποιούν την ΜΤ, απαντούνε και αυτοί με RREP μηνύματα, τα οποία όμως περιέχουν ψευδείς πληροφορίες και παρουσιάζουν ένα πολύ μικρό κόστος που τους κάνει ελκυστικούς. Έτσι οι υπόλοιποι κόμβοι της περιοχής τους προτιμούν και δρομολογούν τα δεδομένα τους μέσω αυτών.

Τρόποι ανίχνευσης και αντιμετώπισης

Έχουν προταθεί διάφοροι τρόποι ανίχνευσης των κακόβουλων κόμβων που πραγματοποιούν επίθεση μαύρης τρύπας. Θα αναφέρουμε μερικούς από αυτούς.

Ο Yuanming WU et. Al [4], συζήτησαν τις ευπάθειες ασφαλείας του μηχανισμού watchdog [5], του μηχανισμού εμπιστοσύνης (trust mechanism) και τον τρόπο που οι επιτιθέμενοι εκ των έσω μπορούν να τους εκμεταλλευτούν. Οι δουλειά τους βασίστηκε στην ανίχνευση των επιτιθέμενων εκ των έσω και ο μηχανισμός εμπιστοσύνης τους περιλαμβάνει 3 στάδια. 1) Συμπεριφορά κόμβων, 2) Μέτρηση εμπιστοσύνης, 3) Ανίχνευση εσωτερικών επιθέσεων

Στο [5] ο Marti Guili et. Al. πρότειναν μία τεχνική μηχανισμού watchdog η οποία βασίζεται στην έννοια των υποκλοπών. Ο κόμβος μπορεί να κρυφακούσει όλες τις μεταδόσεις που γίνονται εντός της εμβέλειας του. Ο μηχανισμός αυτός όμως είχε πολλά μειονεκτήματα εξαιτίας της απλής μεθόδου υποκλοπών που χρησιμοποιούσε. Ο μηχανισμός βελτιώθηκε αργότερα από τον A. Forootaniniaand [6], όπου προστέθηκε ένας επικεφαλής κάθε συστάδας και χρησιμοποιήθηκαν buffers για την αποθήκευση των πακέτων που στέλνονται από τους κόμβους.

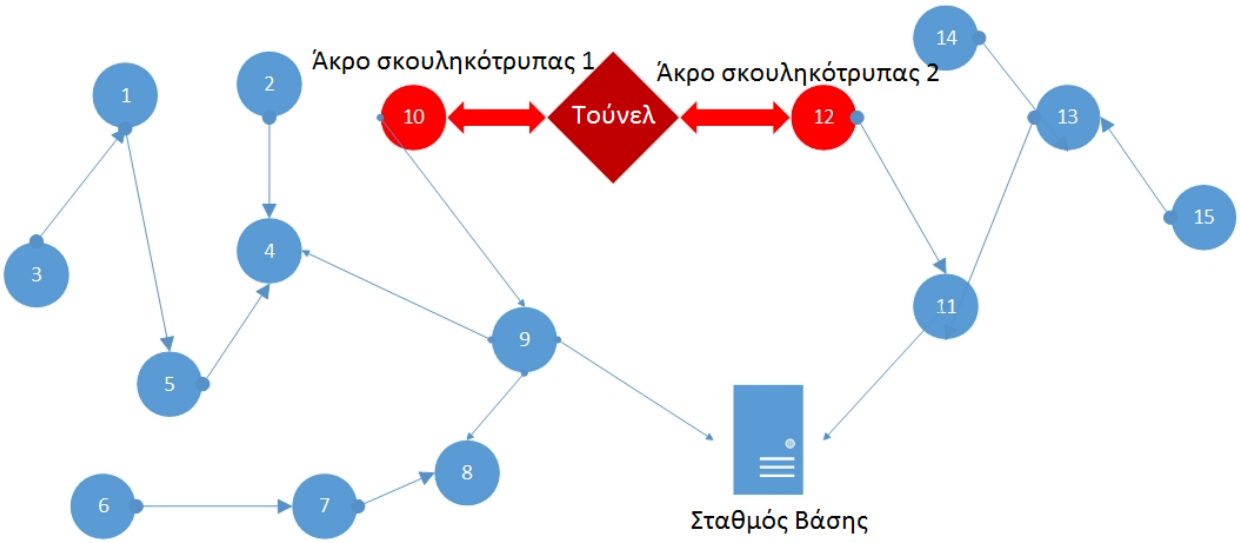
Ο S.Banerjee et. al. [7] πρότειναν έναν αλγόριθμο για την αντιμετώπιση των MT και keyhole επιθέσεων. Σύμφωνα με τον αλγόριθμο τα ολοκληρωμένα δεδομένα διασπούνται σε μικρότερα κομμάτια και βασίζονται στο δίκτυο να ανακαλύψει και να εξαλείψει τους κακόβουλους κόμβους. Οι προσκείμενοι κόμβοι συνεργάζονται στην παράδοση των μικρότερων κομματιών από την πηγή στον προορισμό. Η αναφορά παράδοσης από τον κόμβο προορισμού βοηθάει στην ανίχνευση του κακόβουλου κόμβου.

Στο [8] ο Jian yin et. al. πρότειναν ένα ιεραρχικό ασφαλές πρωτόκολλο που χρησιμοποιεί κρυπτογραφία συμμετρικών κλειδιών για να βρει μία ασφαλή διαδρομή όταν υπάρχουν επιθέσεις μαύρης τρύπας. Στο πρωτόκολλο αυτό το δίκτυο χωρίζεται σε ομάδες οργανωμένες σαν ένα δέντρο, με τον αρχηγό κάθε ομάδας να αποτελεί την ρίζα. Αφού καθιερωθούν τα εσωτερικά και τα εξωτερικά κλειδιά για κάθε ομάδα οι επιθέσεις MT εντοπίζονται τοπικά. Για την ανίχνευση των συνεταιριστικών επιθέσεων μαύρης τρύπας προτείνεται ένα τυχαιοποιημένο καθεστώς αναγνώρισης δεδομένων.

Στο [3] προτείνεται ο μηχανισμός Εκθετικής Εμπιστοσύνης (Exponential Trust Based). Ο μηχανισμός χρησιμοποιεί έναν μετρητή που αυξάνεται όταν υπάρχουν διαδοχικές απώλειες πακέτων (packet drops). Ο συγγραφέας ισχυρίζεται ότι ο μηχανισμός αυτός δουλεύει καλύτερα από τις άλλες μεθόδους γιατί ο παράγοντας εμπιστοσύνης μειώνεται εκθετικά όταν υπάρχουν διαδοχικές απώλειες πακέτων. Κάθε κόμβος διατηρεί έναν πίνακα με τον παράγοντα εμπιστοσύνης (ΠΕ) για κάθε άλλον κόμβο, ο οποίος είναι αρχικά 100. Υπάρχει επίσης ένας μετρητής (streak counter) ο οποίος είναι αρχικά 0 και αυξάνεται κατά 1 για κάθε διαδοχική απώλεια πακέτου. Όταν ο κόμβος τελικά προωθήσει ένα πακέτο ο μετρητής μηδενίζεται. Ο TF για κάθε κόμβο υπολογίζεται από την εξίσωση $100*(x^n)$, όπου x=0.95 και n είναι ο streak counter. Τέλος

ο μηχανισμός στηρίζεται στην υπόθεση ότι οι κόμβοι-μαύρες τρύπες απορρίπτουν όλα τα πακέτα που λαμβάνουν.

Επίθεση σκουληκότρυπας



Εικόνα.7: Σχηματική απεικόνιση επίθεσης σκουληκότρυπας

Η επίθεση σκουληκότρυπας είναι μία επικίνδυνη επίθεση στην οποία ο επιτιθέμενος καταγράφει τα πακέτα σε ένα μέρος του δικτύου και δημιουργώντας ένα τούνελ τα στέλνει σε ένα άλλο απομακρυσμένο μέρος του δικτύου, παρακάμπτοντας τους ενδιάμεσους κόμβους. Έτσι δημιουργείται μία εσφαλμένη εντύπωση ότι ο αρχικός αποστολέας βρίσκεται στην γειτονιά της απομακρυσμένης περιοχής. Ο επιτιθέμενος έχει επίσης τη δυνατότητα να αναμεταδώσει πακέτα επιλεκτικά. Η επίθεση αυτή μπορεί να μετατραπεί σε επίθεση μαύρης τρύπας αν ο επιτιθέμενος ισχυρίζεται ότι μπορεί να παρέχει μίας υψηλής ποιότητας γραμμή προς τον σταθμό βάσης, έτσι η κίνηση του δικτύου θα αναδρομολογηθεί μέσω αυτού αν οι υπόλοιπες διαδρομές είναι λιγότερο ελκυστικές. Με αυτόν τον τρόπο οι επιτιθέμενοι προσελκύουν πολλούς κόμβους με αποτέλεσμα τα πακέτα τους να χάνονται στην σκουληκότρυπα και η επίθεση αυτή να συγκαταλέγεται στις επιθέσεις AτΥ. Υπάρχει επίσης το ενδεχόμενο να δημιουργηθούν περισσότερες από μία σκουληκότρυπες οδηγώντας σε ακόμα μεγαλύτερη υποβάθμιση του δικτύου. Έχει δειχθεί ότι με μία στρατηγική τοποθέτηση μίας σκουληκότρυπας μπορεί να διακόψει κατά μέσω όρο το 32% της συνολικής κίνησης του δικτύου [9].

Τρόποι ανίχνευσης και αντιμετώπισης

Υπάρχουν διάφοροι πιθανοί τρόποι άμυνας ενάντια σε επιθέσεις ΣΤ, καθένας από τους οποίους εκμεταλλεύεται κάποιο μοναδικό χαρακτηριστικό που παρουσιάζει μία σκουληκότρυπα. Αυτοί οι μέθοδοι μπορούν να κατηγοριοποιηθούν σε δύο ευρείες κατηγορίες.

Αυτόματες προσεγγίσεις:

Οι περισσότερες από τις υπάρχουσες μεθόδους εκμεταλλεύονται το μη φυσιολογικό μήκος της σκουληκότρυπας. Όπως ειπώθηκε προηγουμένως η σκουληκότρυπα είναι ουσιαστικά μία ζεύξη 2 κόμβων που βρίσκονται σε μεγάλη απόσταση μεταξύ τους. Ο απλούστερος τρόπος άμυνας ενάντια σε μία τέτοια επίθεση είναι να αποτραπούν οι κόμβοι από την επίτευξη ζεύξης σκουληκότρυπας. Αυτό μπορεί να επιτευχθεί εξοπλίζοντας τους κόμβους με GPS και επαληθεύοντας την θέση του κόμβου με τον οποίο πρόκειται να συνδεθούν. Η μέθοδος αυτή είναι αποτελεσματική ενάντια σε σκουληκότρυπες εκτός του δικτύου, αλλά δεν μπορεί να αποτρέψει τις Byzantine ΣΤ [11],[12] καθώς οι κακόβουλοι κόμβοι είναι πραγματικοί κόμβοι του δικτύου.

Ένα άλλο μοναδικό χαρακτηριστικό μίας σκουληκότρυπας είναι ότι αυξάνεται πλασματικά η γειτονιά του κόμβου. Την ιδιαιτερότητα αυτή χρησιμοποιεί ο συγγραφέας του [13] για να ανιχνεύσει κρυμμένες σκουληκότρυπες. Ας υποθέσουμε ότι ο W_1 είναι ένας WH Node και έχει δημιουργήσει ένα τούνελ με τον απομακρυσμένο κόμβο W_2. Ο W_1 μπορεί να μεταδώσει πληροφορίες για τη γειτονιά του στον W_2 και να ξεγελάσει τους γείτονες του W_2 στο να πιστεύουν ότι μοιράζονται την ίδια γειτονιά με τους γείτονες του W_1. Αυτή η ανωμαλία αυξάνει τον αριθμό των γειτόνων ενός κόμβου που βρίσκεται στην ίδια γειτονιά με έναν WH Node.

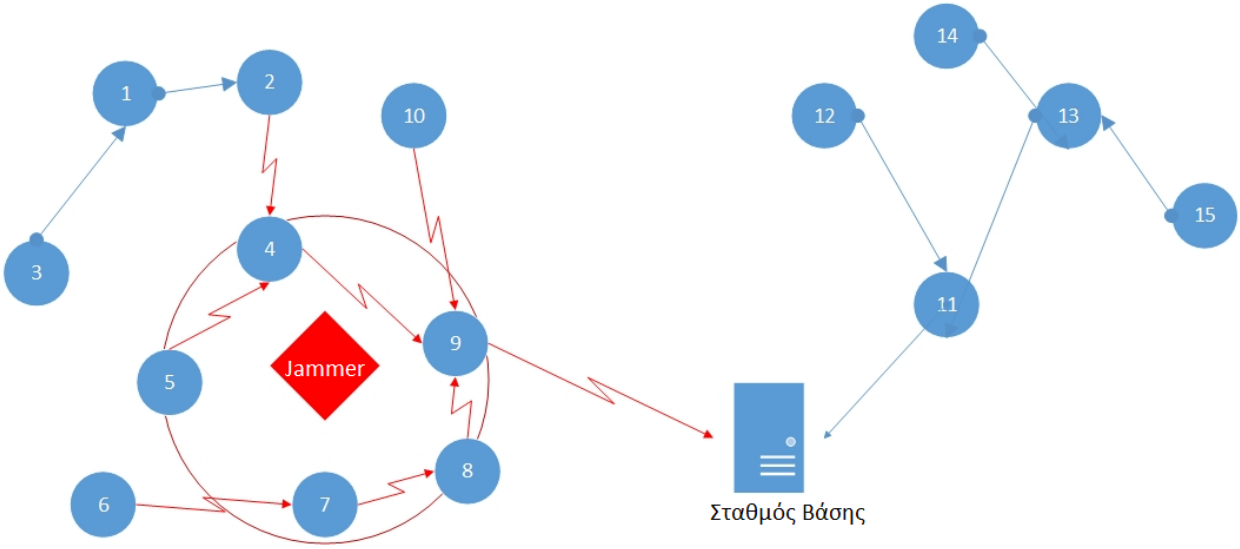
Προσεγγίσεις οπτικής αναπαράστασης:

Έχουν προταθεί 2 σχήματα για την ανίχνευση σκουληκότρυπας με οπτική αναπαράσταση. Στις αρχές του 2004 οι Wang και Bhargava [14] πρότειναν το MDS-VOW, το οποίο έχει τη δυνατότητα να αναγνωρίζει μία επίθεση σκουληκότρυπας σε στατικά ασύρματα δίκτυα αισθητήρων. Με την χρήση της πολυδιάστατης κλιμάκωσης (ΠΚ) και μίας στρατηγικής εξομάλυνσης επιφανειών υπολογίζεται μια εικονική αναπαράσταση του δικτύου. Αν υπάρχει σκουληκότρυπα το σχήμα του δικτύου θα λυγίσει και θα καμπυλωθεί προς αυτήν, αλλιώς το δίκτυο εμφανίζεται επίπεδο.

Αργότερα οι Wang και Lu [15] επέκτειναν το MDS-VOW προτείνοντας έναν βελτιωμένο μηχανισμό ανίχνευσης που ονόμασαν Διαδραστική Οπτικοποίηση Σκουληκότρυπας (Interactive Visualization of Wormhole - IVoW). Το IVoW ενσωματώνει αυτόματους αλγορίθμους ανίχνευσης επιθέσεων με οπτική αναπαράσταση και αλληλεπίδραση του χρήστη για να υποστηρίξει απεικόνιση πολλών σκουληκοτρύπων σε μεγάλης έκτασης δυναμικά ΑΔΑ.

Τέλος το 2014 αναπτύχθηκε το VA-WAD [2] από Έλληνες καθηγητές-ερευνητές. Το VA-WAD είναι ένα σύστημα το οποίο εκμεταλλεύεται πλήρως την ισχύ της οπτικής αναπαράστασης και ανίχνευσης ανωμαλιών με σκοπό να οδηγήσει τον χρήστη στην γρήγορη και ακριβή αντιμετώπιση επιθέσεων σκουληκότρυπας σε μεγάλης κλίμακας ΑΔΑ. Το σύστημα αυτό αποτελείται από δύο βασικά μέρη, τη μηχανή αναγνώρισης ανωμαλιών σκουληκότρυπας (wormhole anomaly detection engine - WAD) και την μηχανή απεικόνισης (visualization engine). Το WAD αναπαριστά την αυτοματοποιημένη λογική ανίχνευσης του συστήματος ενώ η μηχανή απεικόνισης είναι το εργαλείο προβολής.

Επίθεση παρεμβολών (Jamming)



Εικόνα.8: Σχηματική απεικόνιση επίθεσης παρεμβολών

Η επίθεση παρεμβολών είναι ένα είδος Άρνησης της Υπηρεσίας (ΑτΥ) η οποία εμποδίζει την επικοινωνία των κόμβων του συστήματος καταλαμβάνοντας την χωρητικότητα του καναλιού μετάδοσης. Η πιο διαδεδομένη υπόθεση είναι ότι ο Jammer εκπέμπει συνεχόμενα ραδιοφωνικά σήματα για να γεμίσει το κανάλι έτσι ώστε η κανονική κίνηση να διακοπεί. Υπάρχουν όμως και άλλες διαφορετικές συμπεριφορές που μπορεί να υιοθετήσει ένας jammer. [23] Για παράδειγμα ο jammer μπορεί να παραμένει ανενεργός όταν δεν υπάρχει κίνηση στο κανάλι και να ξεκινάει τις παρεμβολές μόλις ανιχνεύσει μετάδοση. Το κοινό χαρακτηριστικό όλων των jamming επιθέσεων είναι ότι δε συμβαδίζουν με το MAC πρωτόκολλο. Επομένως ο jammer ορίζεται ως *μία οντότητα που ηθελημένα προσπαθεί να παρέμβει στην φυσική μετάδοση και λήψη δεδομένων στις ασύρματες επικοινωνίες*.

Επομένως ο σκοπός του jammer είναι να διακόψει τις ασύρματες επικοινωνίες. Για να επιτύχει αυτόν τον σκοπό μπορεί είτε να αποτρέψει έναν κόμβο πηγή από το να μεταδώσει πακέτα ή να παρέμβει στην λήψη των πακέτων. Ας υποθέσουμε ότι οι Α και Β είναι 2 κανονικοί κόμβοι που επικοινωνούν ασύρματα μεταξύ τους και πως ο Χ είναι ο jammer. Ο Χ μπορεί να παρέμβει στην επικοινωνία των Α και Β με διάφορους τρόπους. Ένας από αυτούς είναι να εκπέμπει συνεχώς ένα σήμα στο κανάλι με αποτέλεσμα ο Α να βλέπει το κανάλι πάντα κατειλημμένο. Ένας ακόμη τρόπος είναι ο Χ να στέλνει συνεχώς άχρηστα πακέτα στον Α αναγκάζοντας τον έτσι να μένει σε λειτουργία λήψης και να μην μπορεί να μεταδώσει αυτά που θέλει. Ακόμη και αν ο Α καταφέρει να στείλει πακέτα προς τον Β, ο Χ είναι ικανός να μεταδώσει ένα ισχυρό σήμα στο κανάλι και να καταστρέψει τα πακέτα αυτά πριν τα λάβει ο Β.

Μοντέλα επιθέσεων jamming

Υπάρχουν διάφορα ήδη jammers που χρησιμοποιούνται από κακόβουλους χρήστες. Παρακάτω θα αναλύσουμε μερικά από αυτά.

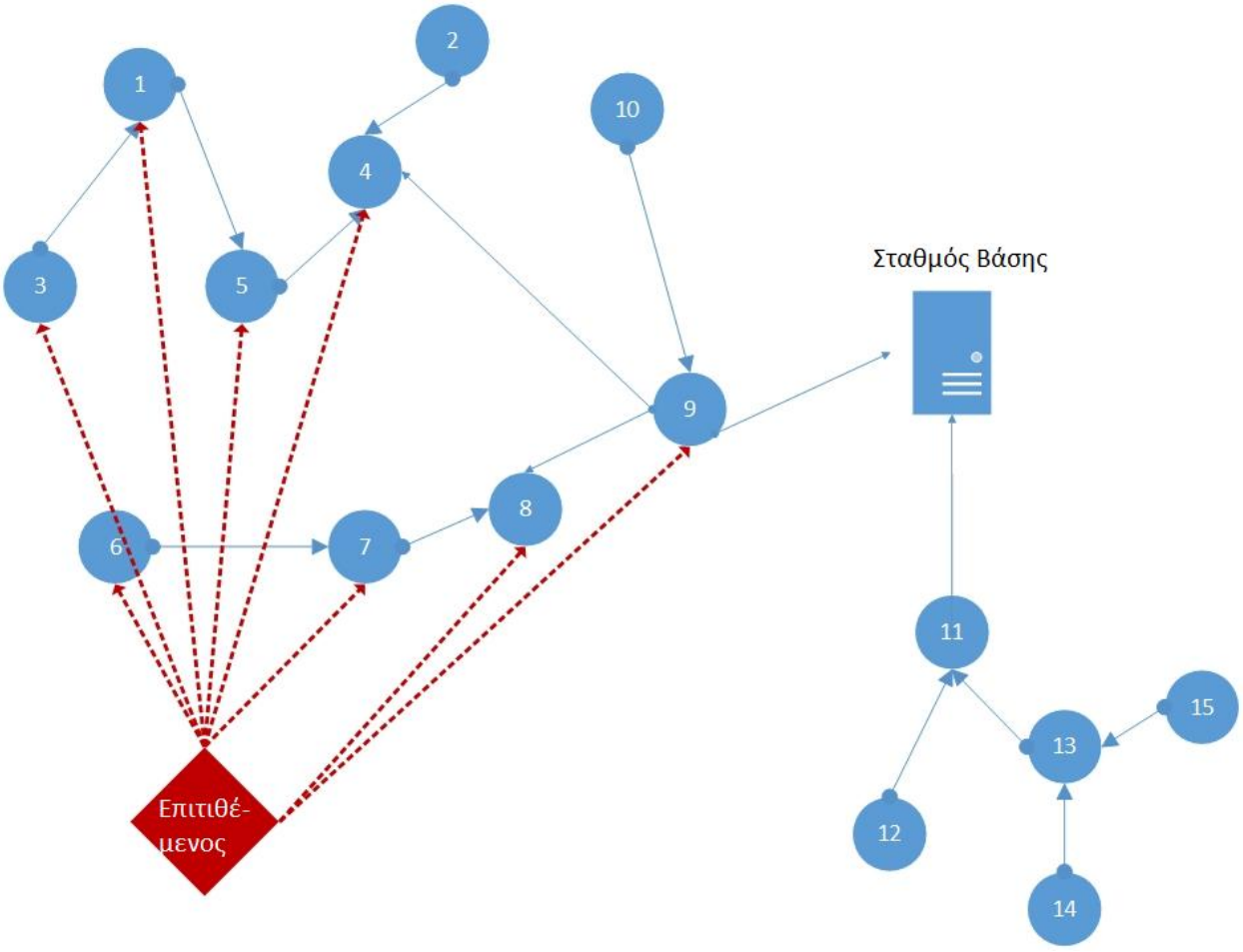
- **Συνεχής jammer (constant jammer):** αυτό το είδος jammer εκπέμπει συνεχώς ένα σήμα στο κανάλι που αποτελείται από τυχαία bits χωρίς να ακολουθεί το MAC πρωτόκολλο. Πιο συγκεκριμένα ο συνεχής jammer δεν περιμένει το κανάλι να γίνει αδρανές πριν μεταδώσει. Το MAC πρωτόκολλο ορίζει πως για να μεταδώσει ένας κόμβος, το κανάλι πρέπει να είναι αδρανές, συνεπώς η επικοινωνία των κόμβων που χρησιμοποιούν το κανάλι παρεμποδίζεται.
- **Παραπλανητικός jammer (deceptive jammer):** Σε αντίθεση με τον συνεχή jammer, ο παραπλανητικός δεν στέλνει τυχαία bits στο κανάλι, αλλά στέλνει άχρηστα πακέτα στους κόμβους που βρίσκονται στην εμβέλεια του, χωρίς κανένα κενό ανάμεσα στις μεταδόσεις. Αυτό έχει ως αποτέλεσμα οι κόμβοι να νομίζουν πως τα πακέτα είναι πραγματικά και τους οδηγεί να παραμείνουν σε λειτουργία λήψης. Στο TinyOS για παράδειγμα αν ένας κόμβος λαμβάνει πακέτα παραμένει σε λειτουργία λήψης ακόμα και αν έχει ο ίδιος πακέτα προς μετάδοση. Συνεπώς άμα υπάρχει συνεχείς ροή πακέτων από τον jammer ο κόμβος δεν θα καταφέρει ποτέ να μεταδώσει τα δεδομένα του.
- **Τυχαίος jammer (random jammer):** Αντί να στέλνει συνεχώς σήματα, ο τυχαίος jammer εναλλάσσεται μεταξύ 2 καταστάσεων, κατάσταση jamming και κατάσταση ύπνου. Πιο συγκεκριμένα θα βρίσκεται στην κατάσταση jamming για t_j χρονικές μονάδες, έπειτα θα απενεργοποιεί την ασύρματη κάρτα του και θα εισέρχεται σε κατάσταση ύπνου. Αφού περάσουν t_s χρονικές μονάδες σε κατάσταση ύπνου θα επανέρχεται πάλι σε κατάσταση jamming. Ο τυχαίος jammer όταν είναι ενεργός μπορεί να συμπεριφέρεται είτε ως συνεχής είτε ως παραπλανητικός jammer. Η εναλλαγή αυτή των καταστάσεων βοηθάει στην εξοικονόμηση ενέργειας του κόμβου, καθώς η συνεχείς μετάδοση σημάτων εξαντλεί γρήγορα την -συνήθως- μικρή μπαταρία του κόμβου..
- **Αντιδραστικός jammer (reactive jammer):** Τα 3 προηγούμενα μοντέλα είναι ενεργητικοί jammers δεδομένου ότι προσπαθούν να ανακόψουν την επικοινωνία χωρίς να λαμβάνουν υπόψη αν υπάρχει κίνηση στο κανάλι. Οι ενεργητικοί jammers είναι συνήθως πολύ αποτελεσματικοί επειδή κρατάνε το κανάλι συνεχώς απασχολημένο, όμως είναι αρκετά εύκολο να ανιχνευθούν. Οι αντιδραστικοί jammers αντιθέτως, παρακολουθούν το κανάλι και μένουν αδρανείς όσο δεν υπάρχει κίνηση. Μόλις όμως αντιληφθούν την ύπαρξη κάποιας μετάδοσης ξεκινάνε να εκπέμπουν σήματα στο κανάλι με σκοπό να αλλοιώσουν ή να καταστρέψουν το μήνυμα. Αυτή η συμπεριφορά έχει ως αποτέλεσμα οι αντιδραστικοί jammers να στοχεύουν στην αποτροπή λήψης των μηνυμάτων. Αξίζει να σημειωθεί ότι ο τρόπος λειτουργίας αυτών των jammers δεν συμβάλει στην εξοικονόμηση ενέργειας επειδή απαιτείται η συνεχής παρακολούθηση του καναλιού. Το βασικό πλεονέκτημα των αντιδραστικών jammers είναι η δυσκολία στην ανίχνευση τους.

Τρόποι ανίχνευσης

Υπάρχουν διάφορες μετρικές και στατιστικά που μπορεί κάποιος να χρησιμοποιήσει για να ανιχνεύσει την ύπαρξη ενός jammer σε ένα ΑΔΑ. Αποδείχθηκε όμως [23] ότι η χρήση της ισχύος σήματος, του χρόνου ανίχνευσης φορέα ή του ποσοστού παράδοσης

πακέτων δεν επαρκούν για την αναγνώριση όλων των ειδών jammers. Για την βελτίωση της ανίχνευσης προτάθηκε η μέθοδος του ελέγχου συνέπειας (consistency check). Σύμφωνα με αυτήν το ποσοστό παράδοσης πακέτων χρησιμοποιείται για να κατηγοριοποιήσει αν μία ασύρματη ζεύξη είναι κακής ποιότητας και έπειτα ο έλεγχος συνέπειας έρχεται για να αναγνωρίσει αν η κακή ποιότητα οφείλεται στην ύπαρξη κάποιου jammer. Αναπτύχθηκαν 2 αλγόριθμοι ελέγχου συνέπειας, ο ένας κάνει χρήση της ισχύς σήματος και ο άλλος εκμεταλλεύεται πληροφορίες τοποθεσίας. Με τον συνδυασμό αυτών τον μεθόδων είναι δυνατή η αναγνώριση όλων των παραπάνω τύπων jammers.

Επίθεση Hello



Εικόνα.9: Σχηματική απεικόνιση επίθεσης Hello

Κάποια πρωτόκολλα δρομολόγησης για ΑΔΑ απαιτούνε από τους κόμβους να εκπέμπουν μηνύματα προς όλους τους άλλους κόμβους στην εμβέλεια τους, ανακοινώνοντας την ύπαρξη τους. Όταν ένας κόμβος λαμβάνει ένα μήνυμα τέτοιου είδους υποθέτει ότι ο κόμβος που το έστειλε βρίσκεται μέσα στην εμβέλεια του. Σε κάποιες περιπτώσεις όμως η υπόθεση αυτή είναι λανθασμένη. [25] Ένας επιτιθέμενος με έναν laptop το οποίο έχει μεγάλη ισχύ μετάδοσης μπορεί να μεταδώσει HELLO πακέτα σχεδόν προς όλους τους κόμβους ενός δικτύου, οδηγώντας τους στην εσφαλμένη εντύπωση ότι είναι γείτονας τους. Για παράδειγμα ο επιτιθέμενος μπορεί να

διαφημίσει ότι έχει μία πολύ καλή ποιότητα γραμμής προς τον σταθμό βάσης και να προκαλέσει τους κόμβους του δικτύου που έλαβαν το μήνυμα να προσπαθήσουν να μεταδώσουν τα πακέτα τους μέσω αυτού. Όσοι κόμβοι όμως επηρεάστηκαν θα στέλνουν τα πακέτα τους στο κενό γιατί δεν θα έχουν αρκετή ισχύ για να καλύψουν την μεγάλη απόσταση. Έτσι οι κόμβοι θα σπαταλούν άδικα την μπαταρία τους προσπαθώντας να στείλουν ξανά και ξανά πακέτα τα οποία δεν θα φτάσουν ποτέ, προκαλώντας έτσι μία γενική σύγχυση στο δίκτυο. Τα πρωτόκολλα που εξαρτώνται από την τοπική ανταλλαγή πληροφοριών μεταξύ των γειτόνων για την διατήρηση της τοπολογίας επηρεάζονται κατά κύριο λόγο από την επίθεση Hello. [24]

Τρόποι ανίχνευσης και αντιμετώπισης

Όπως στις περισσότερες επιθέσεις, έτσι και στην Hello έχουν προταθεί πολλοί τρόποι αντιμετώπισης, άλλοι λιγότερο και άλλοι περισσότερο αποτελεσματικοί.

Στο [24] προτείνεται μία τεχνική δρομολόγησης με πολλαπλά μονοπάτια και πολλαπλούς σταθμούς βάσης, όπου κάθε κόμβος διατηρεί έναν αριθμό από διαφορετικά μυστικά κλειδιά σε ένα πολλαπλό δέντρο. Οι κόμβοι μεταδίδουν τις πληροφορίες τους κάνοντας χρήση αυτών των κλειδιών. Υπάρχουν πολλαπλοί σταθμοί βάσης στο δίκτυο, που έχουν τον έλεγχο ενός συγκεκριμένου αριθμού κόμβων και επίσης χρησιμοποιούνται σαν μέσω επικοινωνίας ανάμεσα στους σταθμούς βάσης. Κάθε σταθμός βάσης έχει όλα τα μυστικά κλειδιά όλων των κόμβων υπό την επίβλεψη του σύμφωνα με ένα πρωτόκολλο ανάθεσης κλειδιών. Δεδομένου του μυστικού κλειδιού και ενός νέου κλειδιού που δημιουργείται για κάθε ζευγάρι κόμβων, η λειτουργία δρομολόγησης απαιτεί μεγάλη επεξεργαστική ισχύ και χαρακτηρίζεται ως ανεπαρκής.

Στο [26] ο συγγραφέας ισχυρίζεται ότι η επίθεση Hello μπορεί να αντιμετωπιστεί με την χρήση του πρωτοκόλλου επιβεβαίωσης ταυτότητας (identity verification protocol). Το πρωτόκολλο επιβεβαιώνει την διακετευθυντικότητα (bi-directionality) της γραμμής με έναν κρυπτογραφημένο echo-back μηχανισμό πριν γίνει οποιαδήποτε ενέργεια. Η άμυνα αυτή δεν είναι αποτελεσματική όταν ο επιτιθέμενος διαθέτει πολύ ευαίσθητο δέκτη και πολύ ισχυρό πομπό.

Στο [28] χρησιμοποιείται μία κρυπτογραφική τεχνική για την πρόληψη της επίθεσης hello. Κάθε ζεύγος κόμβων μοιράζεται το ίδιο μυστικό κλειδί. Κάθε νέο κλειδί δημιουργείται κατά τη διάρκεια της επικοινωνίας. Αυτό το φαινόμενο διαβεβαιώνει ότι μόνο οι κόμβοι που βρίσκονται στην εμβέλεια μπορούν να αποκρυπτογραφήσουν και να επιβεβαιώσουν το μήνυμα, αποτρέποντας έτσι την επίθεση. Το βασικό μειονέκτημα αυτής της τεχνικής είναι ότι ο επιτιθέμενος μπορεί πρώτα να αλλάξει την ταυτότητα του και έπειτα να ξεκινήσει την επίθεση.

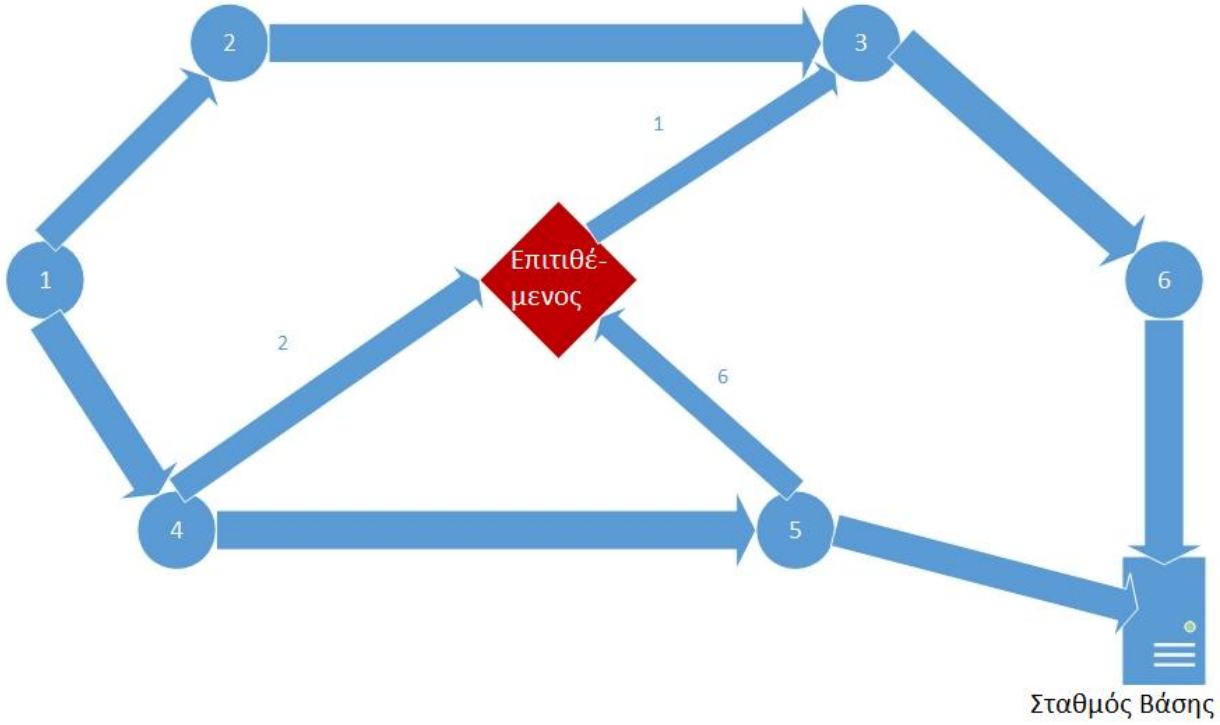
Στο [29] προτείνεται μία λύση ασφαλείας για την προστασία του σταθμού βάσης ενάντια σε επιθέσεις AtY. Μετά την ανίχνευση της επίθεσης AtY ο σταθμός βάσης αναθέτει στους πελάτες του την επίλυση κρυπτογραφημάτων για να προστατευτεί από επιθέσεις. Η δυσκολία του κάθε κρυπτογραφήματος εξαρτάται από την τιμή εμπιστοσύνης κάθε κόμβου. Έτσι οι κακόβουλοι κόμβοι τιμωρούνται περισσότερο χωρίς να επιβαρύνονται οι κανονικοί.

Ένας μηχανισμός ασφαλείας που βασίζεται στην ισχύ σήματος και σε γεωγραφικές πληροφορίες προτείνεται στο [30] για την ανίχνευση κόμβων που πραγματοποιούν επιθέσεις hello και ΣΤ. Η ιδέα είναι να συγκριθεί η ισχύς μίας μετάδοσης με την αναμενόμενη τιμή που υπολογίζεται βάση γεωγραφικών πληροφοριών και των προδιαγραφών του πομποδέκτη. Το ποσοστό ανίχνευσης εξαρτάται από διάφορες παραμέτρους όπως η πυκνότητα του δικτύου, ο πολλαπλασιαστής ισχύος μετάδοσης του κακόβουλου κόμβου, η πιθανότητα ελέγχου μηνύματος κ.α.

Στο [31] παρουσιάζεται ένα παραβιασμένο δίκτυο όπου ο επιτιθέμενος που έχει ευαίσθητο δέκτη πραγματοποιεί μία επίθεση hello με μεγάλη ισχύ μετάδοσης. Πολλοί κόμβοι λαμβάνουν το σήμα και προσπαθούν να απαντήσουν με ένα πρωτόκολλο διπλής ή πολλαπλής χειραψίας για να ανακοινώσουν την παρουσία τους. Όμως οι υγιείς κόμβοι έχουν μικρή ισχύ μετάδοσης και ακτίνα ανίχνευσης φορέα. Έτσι όσοι βρίσκονται μακρύτερα από την ακτίνα ανίχνευσης φορέα τους θα προσπαθήσουν να στείλουν μηνύματα ταυτόχρονα. Η βασική ιδέα είναι να τροποποιηθεί η πρόσβαση στο κανάλι και οι παράμετροι μετάδοσης έτσι ώστε οι απαντήσεις αυτών των κόμβων να συγκρούονται λόγω της υψηλής πυκνότητας χρόνου άφιξης και ο επιτιθέμενος να μην μπορεί να αποκρυπτογραφήσει τα μηνύματα. Με αυτόν τον τρόπο ο κακόβουλος χρήστης δεν έχει δυνατότητα να ακούσει τις απαντήσεις των θυμάτων και υποχρεώνεται να μειώσει την ισχύ του και να συμπεριφερθεί σαν απλός κόμβος.

Τέλος μερικές ακόμη λύσεις για την πρόληψη και αντιμετώπισης της hello επίθεσης είναι: Κάθε κόμβος ελέγχει ανά σταθερά χρονικά διαστήματα τα hello μηνύματα που λαμβάνει με τη βοήθεια ενός μετρητή. Έπειτα προσπαθεί να απαντήσει πρώτα στους κόμβους που έστειλαν να λιγότερα μηνύματα και τελευταία σε αυτούς με τα περισσότερα. Μία άλλη λύση βασίζεται σε χρονικό κατώφλι. Όταν ένας κόμβων δεν λάβει μήνυμα απάντησης για ένα καθορισμένο χρονικό διάστημα θεωρεί τον αποστολέα του ως κακόβουλο και η πληροφορία αυτή μεταδίδεται στους υπόλοιπους κόμβους του δικτύου.

Επίθεση Sybil



Εικόνα.10: Σχηματική αναπαράσταση επίθεσης Sybil

Η επίθεση Sybil είναι μία πολύ σοβαρή επίθεση όπου ο επιτιθέμενος έχει την δυνατότητα να παίρνει πολλαπλές ταυτότητες και να εμφανίζεται σε κάθε κόμβο ως διαφορετικός κόμβος. Οι επιπλέον ταυτότητες του επιτιθέμενου ονομάζονται Sybil nodes. Για την ευκολότερη κατανόηση αυτής της επίθεσης μπορούμε να την χωρίσουμε σε 3 κατηγορίες ανάλογα με την μορφή της [32]: 1) Άμεση και έμμεση επικοινωνία, 2) πλαστές και κλεμμένες ταυτότητες, 3) Ταυτοχρονισμός.

Μορφές επίθεσης Sybil

1) Άμεση και έμμεση επικοινωνία

Άμεση επικοινωνία. Ένας τρόπος να πραγματοποιήσουμε μία Sybil επίθεση είναι οι Sybil nodes να επικοινωνούν απευθείας με τους νόμιμους κόμβους. Όταν ένας νόμιμος κόμβος στέλνει μήνυμα σε έναν Sybil node, τότε ο επιτιθέμενος ακούει αυτό το μήνυμα. Με την ίδια λογική, τα μηνύματα που στέλνονται από τους Sybil nodes, στην πραγματικότητα στέλνονται από τον επιτιθέμενο.

Έμμεση επικοινωνία. Σε αυτή την περίπτωση οι νόμιμοι κόμβοι δεν μπορούν να επικοινωνήσουν απευθείας με τους Sybil nodes. Αντιθέτως η κόμβος του επιτιθέμενου ισχυρίζεται πως έχει μονοπάτι που φτάνει τους Sybil nodes. Τα μηνύματα προς έναν Sybil node δρομολογούνται μέσω του επιτιθέμενου ο οποίος προσποιείται ότι τα παραδίδει στον Sybil node.

2) Πλαστές και κλεμμένες ταυτότητες

Ένας Sybil node μπορεί να αποκτήσει ταυτότητα με 2 τρόπους, είτε να κατασκευάσει μία είτε να κλέψει την ταυτότητα ενός νόμιμου κόμβου.

Πλαστές ταυτότητες. Σε ορισμένες περιπτώσεις ο επιτιθέμενος μπορεί αυθαίρετα να κατασκευάσει τις ταυτότητες των Sybil nodes του. Αν για παράδειγμα κάθε κόμβος έχει ένα 32-bit αναγνωριστικό, ο επιτιθέμενος μπορεί να δημιουργήσει τυχαίους 32-bit αριθμούς και να τους αναθέσει στους Sybil nodes του.

Κλεμμένες ταυτότητες. Αν υπάρχει κάποιος μηχανισμός για να αναγνωρίζει τους νόμιμους κόμβους τότε ο επιτιθέμενος δεν μπορεί να δημιουργήσει αυθαίρετα ταυτότητες. Για παράδειγμα σε κάποια ΑΔΑ ο χώρος ονομάτων είναι εσκεμμένα περιορισμένος και δεν επιτρέπει στον επιτιθέμενο να εισάγει καινούριες ταυτότητες. Στην περίπτωση αυτή πρέπει οι Sybil nodes να πάρουν την ταυτότητα ήδη υπαρχόντων νόμιμων κόμβων. Η κλοπή της ταυτότητας μπορεί να μην αναγνωριστεί αν ο επιτιθέμενος καταστρέψει ή θέσει εκτός λειτουργίας τον νόμιμο κόμβο.

3)Ταυτοχρονισμός

Ταυτόχρονα. Ο επιτιθέμενος μπορεί να προσπαθήσει να κάνει τους Sybil nodes του να συμμετέχουν στο δίκτυο ταυτοχρόνως. Στην πραγματικότητα μία οντότητα υλικού (πχ ασύρματη κάρτα δικτύου) μπορεί συμπεριφέρεται ως μία ταυτότητα κάθε χρονική στιγμή, όμως υπάρχει η δυνατότητα να αλλάζει κυκλικά τις ταυτότητες της, ώστε να φαίνεται ότι όλες υπάρχουν ταυτόχρονα.

Μη ταυτόχρονα. Εναλλακτικά ο επιτιθέμενος μπορεί να παρουσιάζει έναν μεγάλο αριθμό ταυτοτήτων σε μία χρονική περίοδο, αλλά να ενεργεί ως ένας μικρότερος αριθμός ταυτοτήτων κάθε δεδομένη χρονική στιγμή. Αυτό μπορεί να το επιτύχει παρουσιάζοντας ότι μία ταυτότητα αποχωρεί από το δίκτυο και μία άλλη μπαίνει στην θέση της. Μία συγκεκριμένη ταυτότητα μπορεί να βγει και να επανέλθει στο δίκτυο πολλές φορές ή ο επιτιθέμενος να χρησιμοποιεί κάθε ταυτότητα μόνο μία φορά. Μια άλλη περίπτωση είναι ο επιτιθέμενος να διαθέτει πολλές συσκευές στο δίκτυο οι οποίες εναλλάσσουν ταυτότητες μεταξύ τους. Παρόλο που ο αριθμός των ταυτοτήτων που μπορεί να έχει ο επιτιθέμενος είναι ίσος με τον αριθμό των συσκευών στην κατοχή του, κάθε συσκευή μπορεί να παίρνει διαφορετικές ταυτότητες κάθε φορά.

Εφαρμογές επιθέσεων

Κατανεμημένη Αποθήκευση. Ο Deouceur παρατήρησε στο [33] ότι η επίθεση Sybil μπορεί να νικήσει τους μηχανισμούς αντιγραφής και κατακερματισμού σε peer-to-peer συστήματα αποθήκευσης. Το ίδιο πρόβλημα υπάρχει και στην κατανεμημένη αποθήκευση στα ΑΔΑ. Για παράδειγμα, μία επίθεση Sybil μπορεί εύκολα να νικήσει την αντιγραφή και τον κατακερματισμό σε έναν κατανεμημένο hash table, όπως το GHT[34]. Παρόλο που το σύστημα σχεδιάστηκε για να αποθηκεύει δεδομένα σε διάφορους κόμβους, με μία επίθεση Sybil τα δεδομένα αυτά μπορούν να αποθηκευτούν στους Sybil nodes ενός επιτιθέμενου.

Δρομολόγηση. Ο Karlof και ο Wagner επισημαίνουν ότι η επίθεση Sybil μπορεί να χρησιμοποιηθεί ενάντια σε αλγορίθμους δρομολόγησης των ΑΔΑ [28]. Ένας ευάλωτος μηχανισμός είναι η δρομολόγηση πολλαπλών διαδρομών (*multipath)* ή διασπαρσιμότητας *(dispersity)*, όπου φαινομενικά ασύνδετα μονοπάτια μπορούν να περνούν μέσω ενός κακόβουλου κόμβου που παρουσιάζει πολλαπλές ταυτότητες. Ένας ακόμη ευάλωτος μηχανισμός είναι η γεωγραφική δρομολόγηση, όπου αντί να έχει σταθερές συντεταγμένες, ένας Sybil node μπορεί να εμφανίζεται σε πολλά σημεία την ίδια χρονική στιγμή

Συνάθροιση Δεδομένων (*Data Aggregation).* Κάποια αποδοτικά πρωτόκολλα υπολογίζουν σύνολα από τις μετρήσεις των κόμβων με σκοπό την εξοικονόμηση ενέργειας αντί να επιστρέφουν μεμονωμένες μετρήσεις. Ένας μικρός αριθμός κακόβουλων κόμβων που μεταδίδουν εσφαλμένες μετρήσεις μπορεί να μην είναι σε θέση να επηρεάσει σημαντικά το υπολογιζόμενο σύνολο. Όμως με την επίθεση Sybil ένας κακόβουλος κόμβος μπορεί να «συνεισφέρει» στο σύνολο πολλές φορές με τις πολλαπλές του ταυτότητες και να αλλάξει σημαντικά το τελικό σύνολο.

Ψηφίσματα. Στα ΑΔΑ οι κόμβοι μπορεί να χρειαστεί να ψηφίσουν για διάφορα θέματα. Η επίθεση Sybil μπορεί να επηρεάσει την ψηφοφορία. Ανάλογα με τον αριθμό των Sybil nodes ο επιτιθέμενος μπορεί να καθορίσει το αποτέλεσμα κάθε ψηφοφορίας. Για παράδειγμα μπορεί να χρησιμοποιηθεί για την πραγματοποίηση επίθεσης εκβιασμού, όπου ο επιτιθέμενος ισχυρίζεται ότι ένας υγιής κόμβος δεν συμπεριφέρεται σωστά. Ακόμα και να υπάρξει ψηφοφορία για το αν είναι ή όχι έγκυρες οι ταυτότητες του επιτιθέμενου, αυτός μπορεί να χρησιμοποιήσει τους Sybil nodes του για να εγγυηθεί η μία για την άλλη.

Δίκαιη κατανομή των πόρων. Κάποιοι πόροι του δικτύου μπορεί να κατανέμονται ανά κόμβο. Για παράδειγμα, κοντινοί κόμβοι μπορεί να μοιράζονται το ίδιο κανάλι με τον καθένα να έχει ορισμένο χρόνο για να μεταδώσει σε κάθε κύκλο. Η επίθεση Sybil μπορεί να επιτρέψει τον επιτιθέμενο να χρησιμοποιεί περισσότερους πόρους από ότι του αναλογούν. Αυτό από τη μία προκαλεί άρνηση υπηρεσίας στους νόμιμους κόμβους, και από την άλλη δίνει περισσότερους πόρους στον επιτιθέμενο για να πραγματοποιήσει άλλες επιθέσεις.

Ανίχνευση ανάρμοστης συμπεριφοράς. Ας υποθέσουμε ότι το δίκτυο μπορεί να ανιχνεύσει κάποιους τύπους ανάρμοστης συμπεριφοράς. Είναι πολύ συνηθισμένο ένα τέτοιο σύστημα να παράγει ψευδώς θετικά αποτελέσματα (false positives). Επομένως έχει προγραμματιστεί να μην λαμβάνει δράση αν δεν υπάρχουν επαναλαμβανόμενες ενδείξεις ανάρμοστης συμπεριφοράς για κάποιον κόμβο. Ο επιτιθέμενος που θα έχει πολλούς Sybil nodes μπορεί να μοιράσει το φταίξιμο στις πολλαπλές του ταυτότητες μη επιτρέποντας τες να συμπεριφερθούν αρκετά ανάρμοστα ώστε το σύστημα να λάβει δράση. Ακόμα και αν το σύστημα λάβει δράση και αποκλείσει την πρόσβαση σε κάποιον Sybil node, ο επιτιθέμενος μπορεί να συνεχίσει αυτό που κάνει χρησιμοποιώντας καινούριες ταυτότητες κάθε φορά.

Τρόποι ανίχνευσης και αντιμετώπισης

Θα αναφέρουμε περιληπτικά κάποιους τρόπους άμυνας ενάντια σε επιθέσεις Sybil.

Δοκιμή πόρων ραδιοφώνου - Radio Resource Testing. Βασική υπόθεση σε αυτή τη μέθοδο είναι ότι κάθε συσκευή έχει έναν πομποδέκτη και πως δεν έχει τη δυνατότητα να στέλνει ή να λαμβάνει ταυτόχρονα σε περισσότερα από ένα κανάλια. Όταν ένας κόμβος θέλει να εξακριβώσει αν κάποιοι από τους n γείτονες του είναι Sybil nodes, αναθέτει σε κάθε γείτονα ένα κανάλι και εκπέμπει κάποιο μήνυμα σε αυτό. Έπειτα επιλέγει ένα τυχαίο κανάλι στο οποίο «ακούει». Αν ο γείτονας που του ανατέθηκε το κανάλι αυτό είναι νόμιμος θα πρέπει να ακούσει το μήνυμα. Υποθέτουμε ότι s από τους n γείτονες είναι sybils. Σε αυτή την περίπτωση η πιθανότητα να επιλεγεί ένα κανάλι στο οποίο δεν υπάρχει μετάδοση (κάτι που συνεπάγεται την ανίχνευση ενός Sybil) είναι s/n. Αντιστρόφως η πιθανότητα να μην ανιχνευτεί ένας Sybil node είναι (n-s)/n. Αν το τεστ αυτό επαναληφθεί για r γύρους, τότε η πιθανότητα να μην ανιχνευτεί κανένας Sybil node είναι ((n-s)/n)^r.

Προκατανομή τυχαίου κλειδιού - Random Key Predistribution. Σε αυτή τη μέθοδο αναθέτουμε ένα τυχαίο σετ κλειδιών σε κάθε κόμβο. Στην αρχική φάση μοιράσματος κλειδιών κάθε κόμβος μπορεί να ανακαλύψει ή να υπολογίσει τα κοινά κλειδιά που μοιράζεται με τους γείτονες του. Τα κλειδιά αυτά θα χρησιμοποιηθούν σε μία μυστική συνεδρία για να εξασφαλίσουν μυστικότητα από κόμβο σε κόμβο. Οι βασικές ιδέες είναι: 1) Αντιστοίχιση μίας ταυτότητας κόμβου με τα κλειδιά που ανατέθηκαν στον κόμβο αυτόν. 2) Επικύρωση κλειδιών, δηλαδή το δίκτυο να μπορεί να επιβεβαιώσει μέρος ή όλα τα κλειδιά που κάθε ταυτότητα ισχυρίζεται ότι έχει. Κατά συνέπεια, δεδομένου ενός περιορισμένου σετ κλειδιών υπάρχει μικρή πιθανότητα μία αυθαίρετα δημιουργημένη ταυτότητα να περάσει το τεστ επικύρωσης κλειδιών και να γίνει αποδεκτή από το δίκτυο. Υπάρχουν διάφορες παραλλαγές αυτής της μεθόδου όπως: Key pool, Single-space Pairwise Key Distribution, Multi-space Pairwise Key Distribution.

Η σημασία της οπτικής απεικόνισης επιθέσεων στα δίκτυα

Μία παλιά και ευρέως γνωστή παροιμία λέει ότι «μία εικόνα αξίζει χίλιες λέξεις». Οι εικόνες χρησιμοποιούνται για την αποτελεσματική μετάδοση πληροφοριών. Μία εικόνα μπορεί να αιχμαλωτίσει όλη την ομορφιά από ένα τοπίο. Θα ήταν όμως αδύνατο να περιγράψουμε αυτή την ομορφιά με λέξεις. Όπως λέει και ο συγγραφέας του βιβλίου [35]:

- *Μία εικόνα αξίζει χίλιες καταγραφές συμβάντων.*

Αντί να δοθεί σε κάποιον ένα αρχείο καταγραφής συμβάντων που περιγράφει μία επίθεση μπορεί να του δοθεί μία εικόνα, μία οπτική αναπαράσταση των καταγραφών. Με μία ματιά η εικόνα ενημερώνει για το περιεχόμενο του αρχείου. Ο χρήστης μπορεί να αφομοιώσει τις πληροφορίες σε πολύ λιγότερο χρόνο από ότι θα του έπαιρνε να διαβάσει όλο το αρχείο καταγραφής.

Η οπτικοποίηση στον τομέα της ασφάλειας είναι η διαδικασία δημιουργίας εικόνων που βασίζονται σε καταγραφές συμβάντων. Καθορίζει το πώς οι καταγραφές συμβάντων αντιστοιχίζονται πάνω σε μία οπτική αναπαράσταση.

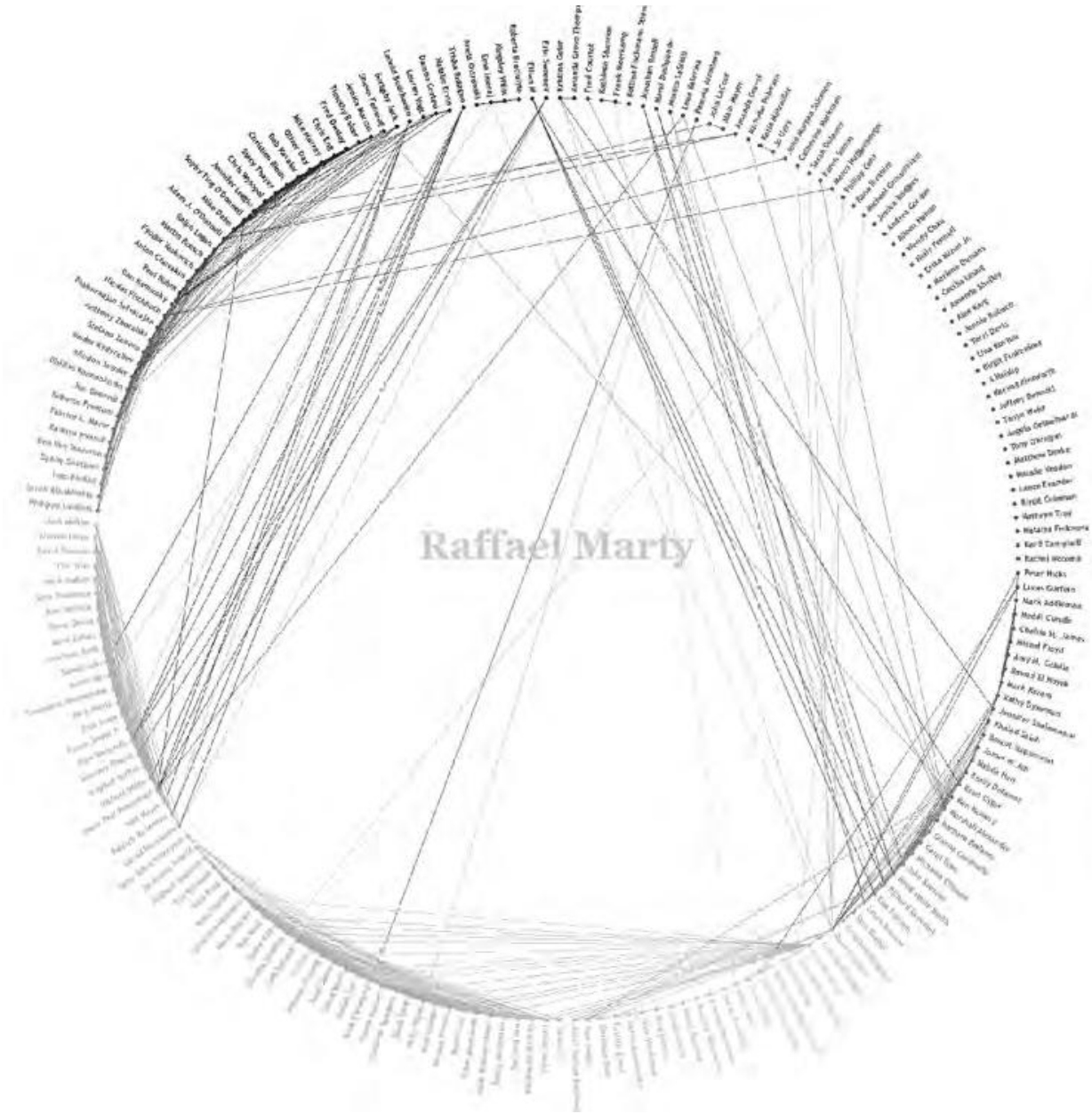
Γιατί όμως να μας ενδιαφέρει η οπτικοποίηση; Επειδή το ανθρώπινο οπτικό σύστημα είναι ένας αναζητητής μοτίβων τεράστιας ισχύος και οξύνοιας. Το μάτι και ο οπτικός φλοιός του εγκεφάλου συνθέτουν έναν μαζικό παράλληλο επεξεργαστή που αποτελεί ένα κανάλι πολύ υψηλού εύρους ζώνης προς τα γνωσιακά κέντρα. [36]

Οι οπτικές αναπαραστάσεις δεδομένων επιτρέπουν την μεταφορά τεράστιων ποσοτήτων πληροφοριών στους χρήστες. Πολύ συχνά οι πληροφορίες είναι σε μορφή κειμένου. Είναι δυσκολότερο για τον εγκέφαλο να επεξεργαστεί κείμενο από φωτογραφίες ή εικόνες. Σε μία εικόνα μπορεί να συμπεριληφθεί μεγάλος πλούτος πληροφοριών και να τις μεταδοθεί σε έναν άνθρωπο. Οι εικόνες χρησιμοποιούν σχήματα, χρώματα, μεγέθη, σχετικές τοποθεσίες κλπ για να κωδικοποιήσουν τις πληροφορίες κάνοντας τες πιο εύκολα αναγνώσιμες.

Η περιήγηση σε μεγάλες ποσότητες δεδομένων είναι κρίσιμη για την εύρεση πληροφοριών και έπειτα την αναζήτηση λεπτομερειών σε ένα αποτέλεσμα. Η αλληλεπίδραση με τις οπτικοποιήσεις είναι ένα από τα βασικά στοιχεία σε αυτή την διαδικασία. Η οπτικοποίηση δεν παρέχει μόνο ταχύτητα στην περιήγηση, αλλά σε αντίθεση με την έγγραφη αναπαράσταση, βοηθάει στην ανακάλυψη συσχετίσεων που είναι καλά κρυμμένες στον πλούτο των δεδομένων.

Ένα απλό παράδειγμα εφαρμογής της οπτικοποίησης είναι το Friend Wheel του Facebook (εικόνα 11), μία εφαρμογή που δημιουργεί μία οπτική αναπαράσταση όλων

των φίλων μας στο Facebook. Κάθε άτομο που είναι φίλος με εμάς τοποθετείται στην περίμετρο ενός κύκλου και μία γραμμή τον συνδέει με τους κοινούς μας φίλους.



Εικόνα.11: Το Friend Wheel του Facebook *[35]*

Υπάρχει η ανάγκη οπτικοποίησης σε πολλούς τομείς. Το Friend Wheel είναι ένα απλό παράδειγμα για το πώς η οπτικοποίηση έγινε mainstream. Η έκρηξη δεδομένων και η απορρέουσα ανάγκη για οπτικοποίηση επηρεάζει την ασφάλεια υπολογιστών και δικτύων περισσότερο από κάθε άλλον τομέα. Οι αναλυτές ασφαλείας αντιμετωπίζουν καθημερινά έναν - συνεχώς αυξανόμενο - αριθμό δεδομένων που χρειάζεται ανάλυση. Δεν υπάρχουν μόνο τα αρχεία συμβάντων συσκευών δικτύου όπως firewalls και συστήματα ανίχνευσης επιθέσεων. Στην εποχή μας χρειάζεται να αναλυθεί ολόκληρη η στοίβα, ξεκινώντας από το επίπεδο δικτύου και ανεβαίνοντας τέρμα επάνω στις εφαρμογές, που τείνουν να δημιουργούν απερίγραπτα μεγάλες ποσότητες δεδομένων.

Πλεονεκτήματα οπτικοποίησης

Η οπτικοποίηση προσφέρει έναν αριθμό από πλεονεκτήματα σε σχέση με εργαλεία κειμένου. Αυτά τα πλεονεκτήματα βασίζονται στην ικανότητα του ανθρώπου να επεξεργάζεται αποδοτικά τις εικόνες. Οι άνθρωποι μπορούν να εξετάζουν, να αναγνωρίζουν και να θυμούνται εικόνες αστραπιαία. Επιπλέον ο εγκέφαλος είναι ένα καταπληκτικό εργαλείο αναγνώρισης μοτίβων, μπορεί να ανιχνεύσει πολύ αποδοτικά αλλαγές σε μεγέθη, χρώματα, σχήματα, κίνηση. Παρακάτω θα αναφερθούν μερικά από τα πλεονεκτήματα της οπτικής αναπαράστασης.

- **Απαντάει σε μία ερώτηση:** Η οπτικοποίηση επιτρέπει την δημιουργία μίας εικόνας για κάθε ερώτηση σε ένα σύνολο δεδομένων. Αντί να προσπελαστούν όλα τα δεδομένα κειμένου με όλες τις συσχετίσεις μεταξύ των εγγραφών, μπορεί να χρησιμοποιηθεί μία εικόνα που μεταφέρει τα δεδομένα σε συνοπτική μορφή.
- **Θέτει νέες ερωτήσεις:** Μία ενδιαφέρουσα πτυχή των οπτικών αναπαραστάσεων είναι ότι προκαλούν τον χρήστη να θέσει νέες ερωτήσεις. Οι άνθρωποι έχουν την ικανότητα να κοιτάξουν μία οπτική αναπαράσταση δεδομένων και να δούνε μοτίβα. Συχνά αυτά τα μοτίβα δεν είναι αναμενόμενα τη στιγμή που δημιουργείται η εικόνα. Τι παράξενο υπάρχει εδώ; Γιατί αυτοί οι κόμβοι επικοινωνούν μεταξύ τους;
- **Εξερευνάει και ανακαλύπτει:** Οπτικοποιώντας τα δεδομένα έχουμε έναν νέο τρόπο προβολής και διερεύνησης αυτών. Η οπτική αναπαράσταση παρέχει νέες απόψεις σε ένα σύνολο δεδομένων. Διαφορετικά γραφήματα και ρυθμίσεις επισημάνουν διαφορετικές ιδιότητες σε ένα σύνολο δεδομένων και βοηθούν στην αναγνώριση προηγουμένως άγνωστων πληροφοριών. Αν οι ιδιότητες και οι συσχετίσεις ήταν γνωστές εκ των προτέρων, θα ήταν δυνατό να ανιχνευτούν αυτά τα γεγονότα χωρίς οπτικοποίηση. Όμως θα πρέπει πρώτα να έχουν ανακαλυφθεί, και τα οπτικά εργαλεία είναι κατάλληλα για αυτή τη δουλειά. Διαδραστικές οπτικοποιήσεις παρέχουν ακόμα πιο πλούσιες διερευνήσεις και βοηθούν στην ανακάλυψη κρυφών ιδιοτήτων ενός συνόλου δεδομένων.
- **Υποστηρίζει αποφάσεις:** Η οπτικοποίηση βοηθάει στην ανάλυση έναν μεγάλου αριθμό δεδομένων πολύ γρήγορα. Αποφάσεις μπορούν να βασιστούν σε έναν μεγάλο αριθμό δεδομένων επειδή η οπτικοποίηση βοήθησε να μετατραπούν σε κάτι που έχει νόημα. Περισσότερα δεδομένα βοηθούν επίσης στην υποστήριξη των αποφάσεων μας. Η επίγνωση της κατάστασης είναι ένα σημαντικό εργαλείο στην υποστήριξη αποφάσεων.
- **Κοινοποιεί πληροφορίες:** Οι γραφικές αναπαραστάσεις δεδομένων είναι πιο αποτελεσματικές στην επικοινωνία από τα αρχεία κειμένου. Μία ιστορία μπορεί να ειπωθεί πιο αποδοτικά και ο χρόνος που χρειάζεται για την κατανόηση μίας φωτογραφίας είναι πολύ λιγότερος από αυτόν που χρειάζεται για την κατανόηση κειμένου.
- **Αυξάνει την αποδοτικότητα:** Παρατηρώντας ένα γράφημα μπορούν να διακριθούν πολύ πιο αποδοτικά οι τάσεις και οι ακραίες τιμές. Ο χρόνος που χρειάζεται για την ανάλυση αρχείων καταγραφής μειώνεται δραστικά. Έτσι ελευθερώνεται χρόνος και επιτρέπει τους ανθρώπους να ασχοληθούν με άλλα

πράγματα, όπως τα μοτίβα και οι συσχετίσεις των δεδομένων. Επίσης επιταχύνει την αναγνώριση και τον χρόνο αντίδρασης σε νέα γεγονότα.

- **Εμπνέει:** Οι εικόνες μας εμπνέουν. Η ανάλυση οπτικών δεδομένων μπορεί να εμπνεύσει για να αναπτυχτούν νέες μέθοδοι οπτικοποίησης που βοηθούν περισσότερο σε κάτι συγκεκριμένο από τις προηγούμενες. Μερικές φορές οι εμπνεύσεις μπορεί να καταλήξουν σε αδιέξοδο, αλλά τις περισσότερες οδηγούν σε νέες ανακαλύψεις και βοηθούν στην καλύτερη κατανόηση των δεδομένων μας.

Οπτικοποίηση Ασφάλειας

Ο τομέας της οπτικοποίησης ασφάλειας είναι ακόμη στα σπάργανα. Μέχρι και σήμερα έχει γίνει πολύ περιορισμένη πρόοδος σε αυτόν τον τομέα. Έχοντας έναν πολύ μεγάλο αριθμό δεδομένων που χρειάζονται ανάλυση για να αναγνωριστούν προβλήματα ασφαλείας, η οπτικοποίηση φαίνεται να είναι η σωστή προσέγγιση.

- Ο συνεχώς αυξανόμενος αριθμός δεδομένων που συλλέγεται σε IT περιβάλλοντα χρειάζεται νέες μεθόδους ανάλυσης.
- Η ανάλυση καταγραφών και γεγονότων γίνεται ένα από τα κύρια εργαλεία των αναλυτών ασφαλείας για να διερευνήσουν και να κατανοήσουν τα δίκτυα τους, τους servers, τις εφαρμογές και τις επαγγελματικές διεργασίες. Όλες αυτές οι εργασίες έχουν να κάνουν με έναν τεράστιο αριθμό δεδομένων που χρειάζεται ανάλυση.
- Η συμμόρφωση με τους κανονισμούς ζητά συχνή ανάλυση καταγραφών. Οι αναλυτές χρειάζονται καλύτερα και πιο αποδοτικά εργαλεία για να ανταπεξέλθουν.
- Το τοπίο του ηλεκτρονικού εγκλήματος αλλάζει. Οι επιθέσεις πλέον έχουν ξεφύγει από το επίπεδο δικτύου. Οι επιθέσεις σήμερα κινούνται στο επίπεδο εφαρμογών: Web 2.0, επιθέσεις σε εφαρμογές άμεσων μηνυμάτων, ηλεκτρονικές απάτες, υποκλοπές πληροφοριών, crimeware είναι μόνο μερικά παραδείγματα από τους νέους τύπους επιθέσεων οι οποίες παράγουν πολλά δεδομένα που χρειάζονται ανάλυση.
- Τη σήμερον ημέρα οι επιθέσεις που πρέπει να προστατευτούμε είναι στοχευόμενες. Δεν είμαστε πλέον τυχαία θύματα όπως συνέβαινε παλιά. Οι επιτιθέμενοι γνωρίζουν σε ποιον θέλουν να κάνουν κακό. Πρέπει να είμαστε προετοιμασμένοι και να αναλύουμε προληπτικά τα αρχεία καταγραφών μας. Οι επιτιθέμενοι είναι προσεκτικοί και δεν θα ενεργοποιήσουν συναγερμούς.

Εξαιτίας του μεγάλου αριθμού δεδομένων προς ανάλυση τα κλασικά εργαλεία ασφαλείας όπως τείχη προστασίας και συστήματα ανίχνευσης εισβολών έχουν αρχίσει να κάνουν χρήση γραφημάτων και γραφικών και να αναφέρουν τα ευρήματά τους. Αυτές οι οπτικοποιήσεις όμως δεν είναι διαδραστικές και δεν επιτρέπουν την εξερεύνηση των δεδομένων. Επιπλέον είναι ακόμα σε πολύ βασικό στάδιο και παρουσιάζουν τα ευρήματα τους εκ των υστέρων. Η κατάσταση όμως έχει αρχίσει να βελτιώνεται. Οι εταιρίες προϊόντων ασφαλείας έχουν αρχίσει να συνειδητοποιούν ότι η

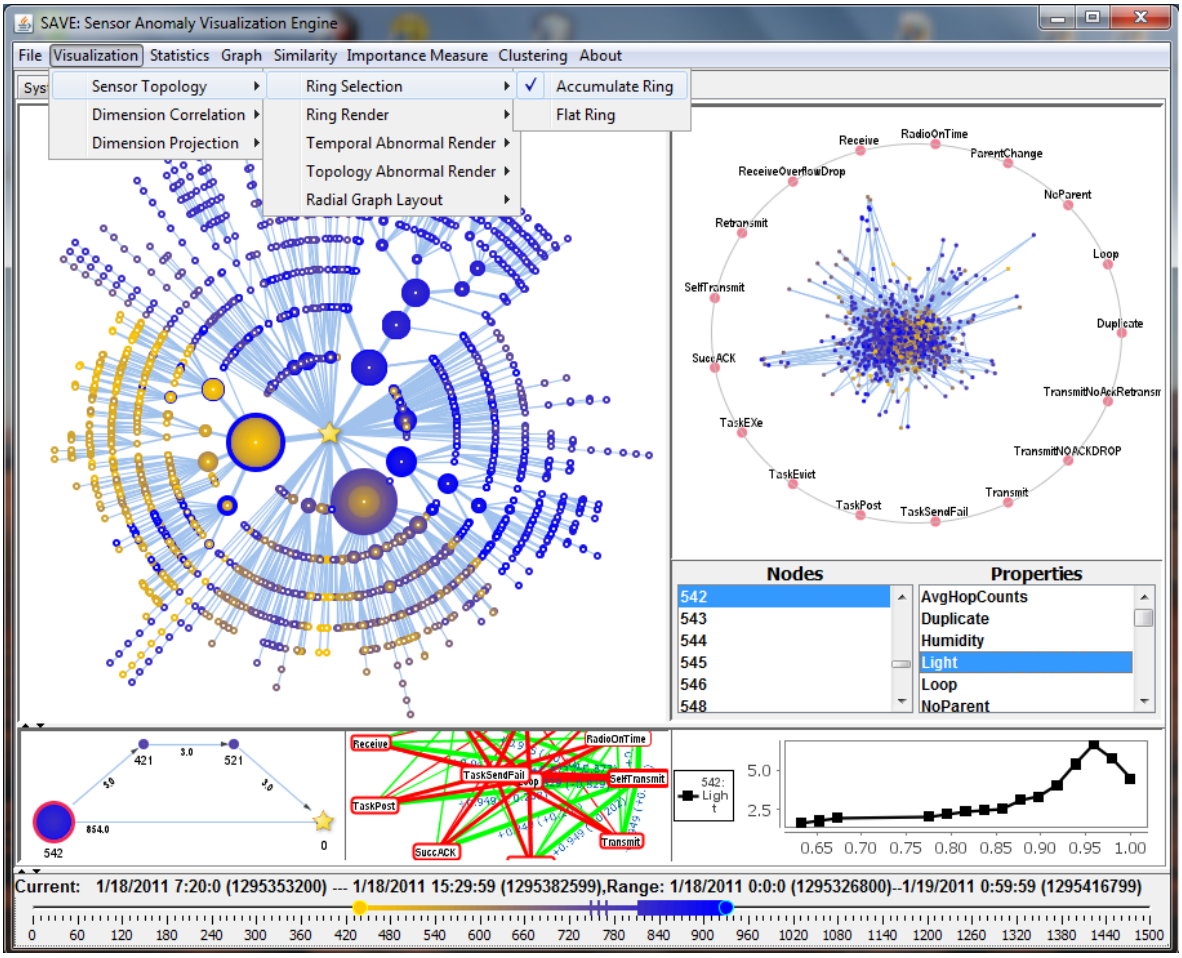
οπτική αναπαράσταση είναι ένα ανταγωνιστικό πλεονέκτημα για αυτές και ότι οι εργασίες των χρηστών θα απλοποιηθούν σημαντικά με οπτικά βοηθήματα.

Συστήματα απεικόνισης ανωμαλιών στα ΑΔΑ.

Παραδοσιακά η ανάλυση δικτύων αισθητήρων έχει βασιστεί είτε σε γενικούς (ns-2) είτε σε in-domain (TOSSIM [38]) προσομοιωτές δικτύων. Για παράδειγμα το MOTE_VIEW [39] παρακολουθεί την υγεία και την κατάσταση των κόμβων σε μικρά προσομοιωμένα δίκτυα όπως επίσης απεικονίζει τις μετρήσεις των αισθητήρων από την κίνηση ανθρώπων ή οχημάτων σε μία προσομοιωμένη επικίνδυνη περιοχή. Το NetTopo [40] παρέχει προσομοίωση αλλά και συναρτήσεις οπτικής αναπαράστασης και την δοκιμασία και επαλήθευση αλγορίθμων στα ΑΔΑ. Πιο συγκεκριμένα το TOSSIM, ένας διακριτικός προσομοιωτής συμβάντων για δίκτυα με αισθητήρες TinyOS περιλαμβάνει το TinyViz [38], ένα περιβάλλον οπτικής αναπαράστασης γραμμένο σε Java που αλληλεπιδράει με το TOSSIM μέσω του δικτυακού του πρωτοκόλλου. Παρόλο που εργαλεία οπτικοποίησης όπως τα προηγούμενα επιτρέπουν στους χρήστες να επιλέξουν και να διαχειριστούν προσομοιωμένους κόμβους μέσω ενός GUI, περιορίζονται μόνο σε προσομοιώσεις και δεν μπορούν να διαχειριστούν πειραματικά δεδομένα ενός πραγματικού δικτύου. Αυτό έχει ως αποτέλεσμα αυτά τα εργαλεία να μην μπορούν να ανιχνεύουν απρόβλεπτες ανωμαλίες που συμβαίνουν στα πραγματικά δίκτυα.

Προϊόντα όπως το Sensor Network Analyzer (SNA) [41] είναι διαθέσιμα ως εμπορικές λύσεις για την ανάπτυξη, αποκωδικοποίηση, διόρθωση σφαλμάτων και τοποθέτηση (deploying) ασύρματων ενσωματωμένων δικτύων. Υποστηρίζει διάφορα πρωτόκολλα, συμπεριλαμβανομένων των IEEE 802.15.4 και ZigBee. Το SNA υποστηρίζει φιλτράρισμα, επισήμανση και χρωματική κωδικοποίηση για να διευκολύνει την εύρεση πακέτων ενδιαφέροντος και την μέτρηση απόδοσης. Το Surge Network Viewer [42] είναι μία εφαρμογή που συμπεριλαμβάνεται στην διανομή των TinyOS Tools και είναι χρήσιμο για την παρακολούθηση και ανάλυση της απόδοσης των ασυρμάτων δικτύων αισθητήρων. Επιπλέον το SpyGlass [43] απεικονίζει την τοπολογία του δικτύου και την κατάσταση των αισθητήρων του μέσω βασικών γράφων και επισήμανσης κόμβων. Στο Sensor Network Analysis and Management Platform (SNAMP) [44], τα δεδομένα που εκπέμπονται από τους κόμβους συλλέγονται από έναν πολλαπλό καταγραφέα πακέτων και μεταδίδονται σε ένα ευέλικτο μηχανισμό οπτικοποίησης με πολλές οπτικές. Με μία έννοια οι συναρτήσεις debugging που περιλαμβάνονται στο SNAMP οπτικοποιούν την ανάπτυξη ΑΔΑ εφαρμογών. Παρόλο που προϊόντα όπως το SNA επιτρέπουν τους χρήστες να οπτικοποιήσουν πακέτα σε επίπεδο byte, τους λείπει η έξυπνη αναλυτική δυνατότητα να ανιχνεύσουν και να αναλύσουν τις ανωμαλίες των ασύρματων κόμβων-αισθητήρων.

Sensor Anomaly Visualization Engine (SAVE)



Εικόνα 12: Γραφικό περιβάλλον του SAVE *[42]*

Στα προβλήματα των παραπάνω εφαρμογών έρχεται να δώσει λύσει το Sensor Anomaly Visualization Engine [42], μία ολοκληρωμένη μηχανή ανίχνευσης και οπτικής αναπαράστασης ανωμαλιών στα ασύρματα δίκτυα αισθητήρων, πάνω στη οποία έχει βασιστεί και μεγάλο μέρος της εφαρμογής μας. Το SAVE είναι ένα σύστημα οπτικής ανάλυσης για διάγνωση πραγματικού χρόνου αλλά και playback δεδομένων σε πραγματικού κόσμου δίκτυα αισθητήρων. Το σύστημα αυτό δοκιμάστηκε στο GreenOrbs [43], το οποίο είναι ένα μακροχρόνιο και μεγάλου μεγέθους ασύρματο δίκτυο αισθητήρων τοποθετημένο σε ένα δάσος.

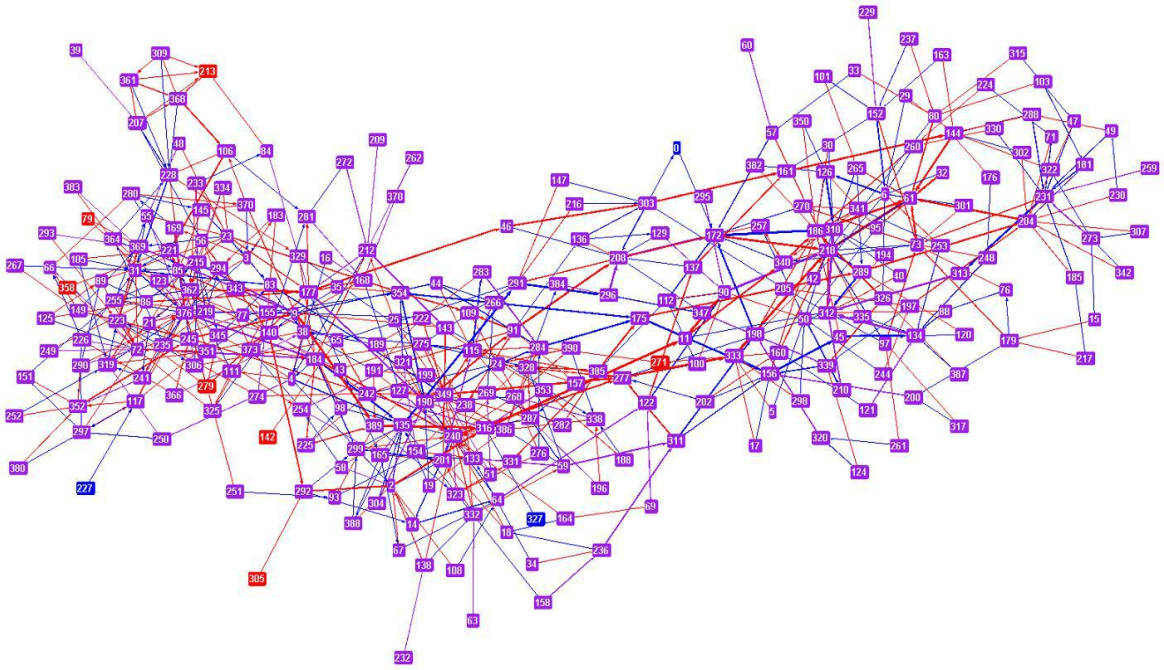
Framework του SAVE

Το σύστημα SAVE σχεδιάστηκε σε τρία στάδια: προεπεξεργασία δεδομένων, ανίχνευση ανωμαλιών και multi-view στάδια οπτικοποίησης. Στο πρώτο στάδιο τα ανεπεξέργαστα δεδομένα προεπεξεργάζονται και η χρονική τοπολογία του δικτύου και οι συσχετίσεις διαστάσεων «χτίζονται» από κατανεμημένα κομμάτια δεδομένων. Έπειτα ανωμαλίες όπως ακραίες τιμές μετρήσεων και αλλαγές συσχετίσεων υπολογίζονται και αποθηκεύονται online. Τέλος, στο βασικό στάδιο, οι πτυχές των δεδομένων και τα προετοιμασμένα analytics ενώνονται και απορροφούνται από τα εργαλεία οπτικής αναπαράστασης του SAVE.

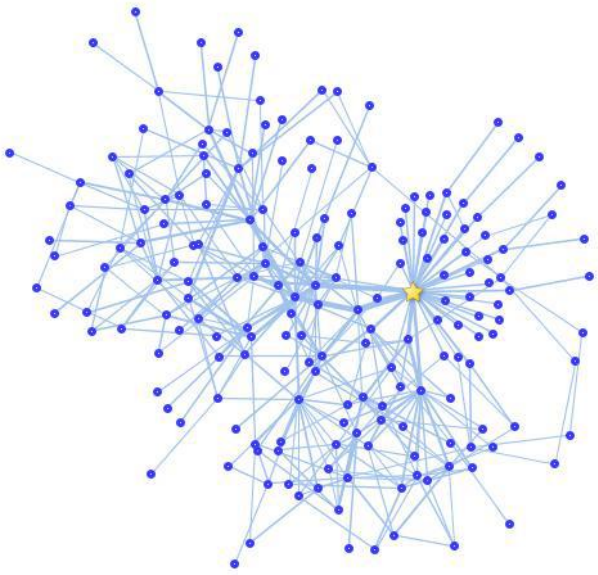
Από την οπτική του χρήστη, αποκτούν πρόσβαση στο SAVE επιλέγοντας μία συλλογή από ίχνη δεδομένων στην λειτουργία offline ανάλυσης. Έπειτα το SAVE τους παρουσιάζει μία άποψη όλου του σετ δεδομένων με πολλαπλές συντονισμένες απεικονίσεις που αναπαριστούν διαφορετικές πτυχές των δεδομένων, συμπεριλαμβανομένου της τοπολογίας, συσχέτισης και απόψεις προβολής διαστάσεων (dimension projection views). Παρέχεται επίσης χρονικός έλεγχος επιτρέποντας στον χρήστη να ορίσει την χρονική περίοδο που τον ενδιαφέρει και να δει τα δεδομένα της. Με αυτούς τους τρόπους ο χρήστης μπορεί να βρει την ενδιαφέρουσα παρτίδα των δεδομένων από διάφορες προοπτικές όπως χρονική, τοπική και διαστατική και να διεξάγει περεταίρω ανάλυση των λεπτομερειών με εργαλεία όπως γραμμές τάσεων (trend lines)

Analytics τοπολογίας

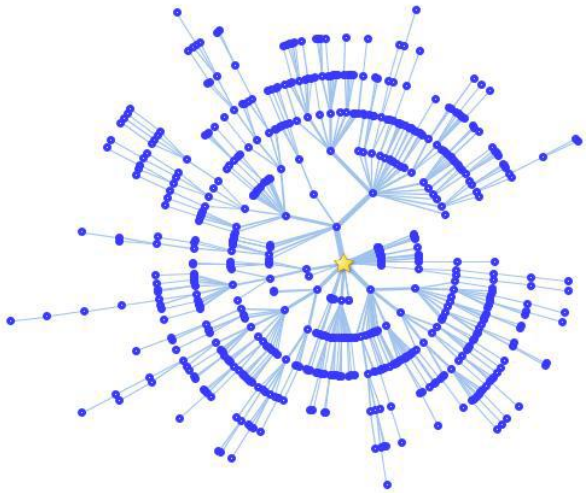
Οι δυναμικές της τοπολογίας δρομολόγησης είναι ο κύριος δείκτης για την απόδοση ενός δικτύου αισθητήρων. Από την φύση της η τοπολογία ένας γράφος μεγάλης έκτασης και μεταβλητού χρόνου λόγω της αστάθειας των ad-hoc επικοινωνιών και του συστήματος δρομολόγησης που εφαρμόζεται. Οι παραδοσιακές τεχνικές απεικόνισης που φαίνεται στην εικόνα 13α) και 13β) δεν είναι αποτελεσματικές γιατί η τοπολογία αλλάζει συνεχώς και σε μία χαοτική εικόνα σαν και αυτές οι αλλαγές δύσκολα γίνονται αντιληπτές. Οι δημιουργοί του SAVE πρότειναν το Χρονικό Μοντέλο Ανάπτυξης (Temporal Expansion Model - ΧΜΑ), έναν πιο διαισθητικό γράφο για το στάδιο οπτικοποίησης (εικόνα 13γ). Το ΧΜΑ εκμεταλλεύεται το κύριο χαρακτηριστικό των δικτύων αισθητήρων που μελετώνται – όλοι οι κόμβοι στέλνουν τα πακέτα τους στον κεντρικό κόμβο (ΚΚ). Η βασική ιδέα είναι να διαχωρίσουμε έναν φυσικό κόμβο σε πολλαπλούς λογικούς κόμβους ανάλογα με τα διαφορετικά μονοπάτια τους προς τον ΚΚ. Τα πλεονεκτήματα σε αυτό είναι δύο: α) οι γράφοι που δημιουργούνται είναι κατευθυνόμενα δέντρα που είναι πολύ καλύτερα για οπτικοποίηση και πλοήγηση και β) οι χρονικές αλλαγές στο δίκτυο εμφανίζονται στον γράφο παρέχοντας πληροφορίες για περαιτέρω ανάλυση. Μία ακόμη διαφορά του ΧΜΑ είναι ότι κρατάει την χρονική σειρά των γεγονότων για κάθε κόμβο αντί να διατηρεί μόνο ένα αριθμητικώς συσσωρευμένο βάρος.



α)Γεωγραφική απεικόνιση



β)Απεικόνιση Force-directed



γ) ΑπεικόνισηTEM

Εικόνα. 13: Διαφορετικές απεικονίσεις ΑΔΑ *[42]*

Το εργαλείο Processing ως Μέθοδος Οπτικοποίησης

Το Processing [1] είναι μία γλώσσα προγραμματισμού, ένα περιβάλλον ανάπτυξης εφαρμογών και μία online κοινότητα. Αρχικά δημιουργήθηκε ως ένα sketchbook για προγράμματα και για να διδάξει τις αρχές του προγραμματισμού σε ένα οπτικό περιβάλλον, στη συνέχεια όμως το Processing εξελίχθηκε σε ένα εργαλείο ανάπτυξης εφαρμογών για επαγγελματίες.

Το Processing είναι λογισμικό ανοιχτού κώδικα και παρέχεται δωρεάν για Windows, Mac OS X και GNU/Linux. Υποστηρίζει την ανάπτυξη διαδραστικών προγραμμάτων με έξοδο σε 2D, 3D ή PDF. Επίσης μπορούν να χρησιμοποιηθούν οι βιβλιοθήκες OpenGL για 3D επιτάχυνση.

Θα αναφέρουμε τώρα μερικές από τις συναρτήσεις γραφικών του Processing που θα χρησιμοποιηθούν κατά κόρον στην εφαρμογή μας.

- `ellipse(a, b, c, d)` σχεδιάζει μία έλλειψη με a:συντεταγμένη x, b:συντεταγμένη y, c: πλάτος, d: ύψος.
- `line(x1, y1, x2, y2)` σχεδιάζει μία γραμμή, x1: συντεταγμένη x πρώτου σημείου, y1: συντεταγμένη y πρώτου σημείου, x2: συντεταγμένη x δεύτερου σημείου, y2: συντεταγμένη y δεύτερου σημείου
- `rect(a, b, c, d) σχεδιάζει` ένα ορθογώνιο παραλληλόγραμμο, a: συντεταγμένη x της πάνω αριστερής γωνίας, b: συντεταγμένη y της πάνω αριστερής γωνίας, c: πλάτος, d: μήκος.
- `text(c, x, y)` τυπώνει ένα κείμενο c στην θέση με συντεταγμένες x και y.
- `lerp(start, stop, amt)` χρησιμοποιείται για να υπολογίσει έναν αριθμό ανάμεσα σε 2 άλλους αριθμούς. Start και stop είναι τα όρια, και amt είναι ένας float ανάμεσα στο 0 και το 1 με το 0 να αναπαριστά το αρχικό σημείο και το 1 το τελικό, οποιοδήποτε ενδιάμεση τιμή αναπαριστά έναν ενδιάμεσο αριθμό. Χρησιμοποιείται να τραβήξουμε γραμμές με κουκίδες ή για να αναπαραστήσουμε κίνηση σε μία ευθεία
- `fill(rgb)` δέχεται ένα RGB χρώμα και γεμίζει σχήματα με αυτό το χρώμα. Καλείται πριν από την συνάρτηση του σχήματος.
- `noFill()` αναιρεί την λειτουργία της `fill(rgb)`
- `background(rgb)` χρωματίζει το φόντο του παραθύρου.
- `stroke(rgb)` χρωματίζει το περίγραμμα των σχημάτων που ακολουθούν την κλήση της
- `noStroke()` αναιρεί την λειτουργία της `stroke(rgb)`.

Αναλυτική παρουσίαση της εφαρμογής

Ο στόχος

Ο σκοπός της εργασίας αυτής είναι η κατασκευή με την βοήθεια του Processing μίας εφαρμογής που να είναι σε θέση να απεικονίζει τα 5 βασικά είδη επιθέσεων σε ένα ασύρματο δίκτυο αισθητήρων. Για την εφαρμογή έχει αναπτυχθεί ένας προσομοιωτής ασύρματου δικτύου αισθητήρων. Ο χρήστης έχει τη δυνατότητα να τροποποιήσει τις παραμέτρους του δικτύου πριν ξεκινήσει την προσομοίωση. Κατά την διάρκεια της προσομοίωσης ο χρήστης μπορεί να επιλέξει μέσα από πέντε διαφορετικά είδη επιθέσεων για να επιτεθεί στο δίκτυο. Έχουν επίσης αναπτυχθεί τέσσερεις διαφορετικές απεικονίσεις του δικτύου, οι οποίες μπορούν να χρησιμοποιηθούν για να παρακολουθήσουμε τις επιπτώσεις των επιθέσεων. Τέλος η εφαρμογή κρατάει στατιστικά για την κίνηση ολόκληρου του δικτύου αλλά και κάθε κόμβου ξεχωριστά. Άμα συγκριθούν τα στατιστικά από διαφορετικές επιθέσεις ή καταστάσεις του δικτύου ο χρήστης μπορεί να έχει ακόμη μία εικόνα για το τι συμβαίνει στο δίκτυο.

Τεχνικά χαρακτηριστικά

Η εφαρμογή έχει προγραμματιστεί σε Java DK 1.8 στο IDE του Processing v2.2.1. Για σχεδίαση γραφικών 2D χρησιμοποιήθηκαν οι βιβλιοθήκες του Processing και για το GUI η βιβλιοθήκη Control P5 [43].

Ο υπολογιστής όπου έγινε ο πραγματισμός και η εκτέλεση της εφαρμογής είναι ένα laptop με Intel i3-2328M@ 2.20GHz, 4GB DDR3 RAM, Intel 3000 GPU με ανάλυση οθόνης 1440x900. Το λειτουργικό σύστημα είναι Windows 7 Pro x64.

Τοπολογία δικτύου προσομοίωσης

Η εφαρμογή προσομοιώνει ένα ΑΔΑ τοπολογίας peer-to-peer. Υπάρχουν διάσπαρτοι στον χώρο πολλοί ομότιμοι κόμβοι-αισθητήρες που όλοι στέλνουν τα δεδομένα τους σε έναν κεντρικό κόμβο, τον sink node. Η δρομολόγηση των πακέτων γίνεται από κόμβο σε κόμβο με τη βοήθεια ενός δυναμικού αλγορίθμου δρομολόγησης. Ο χρήστης έχει τη δυνατότητα να επιλέξει μία προκαθορισμένη διάταξη κόμβων ή να δημιουργήσει την δική του η οποία μετά αποθηκεύεται σε αρχείο που μπορεί να φορτωθεί και να χρησιμοποιηθεί ξανά.

Οι κλάσεις

Σε αυτή την ενότητα θα ξεκινήσουμε την ανάλυση της εφαρμογής περιγράφοντας τις κλάσεις της. Κάθε ασύρματο δίκτυο αισθητήρων αποτελείται από κόμβους-αισθητήρες, έτσι η βασική κλάση μας είναι ο `Node`.

```
public class Node {
  //General
  public int id;
  public float range;
  public int xpos;
  public int ypos;
  public int saveX;
  public int saveY;
  public float radius;
  public float costToSink=9999;
  public int routeToSink=-1;
  public float routingScore = 100;
  public String type="Normal";
  public float strokeWeight=1;
  //Save view
  public float angle;
  public float saveRadius=10;
  boolean angleChanged=false;
  public float rightend=556677;
  public float leftend=556677;
  public color saveColor=color(0,180,234);
  //Traffic Related
  public int packetsDropped=0;
  public int packetsSent=0;
  public int packetsTransmited=0;
  public int packetsOnSink=0;
  public boolean isConnected=true;
  public int missedWindows=1;
  public int currDelay=0;
  public double battery=100;
  public double oblivion=0;
  public ArrayList<Integer> NeighborList = new ArrayList<Integer>();
  public ArrayList<Integer> HopList = new ArrayList<Integer>();
  public ArrayList<Integer> ChildrenList = new ArrayList<Integer>();
  public ArrayList<Integer> AllDelays = new ArrayList<Integer>();
  public ArrayList<Packet> queue = new ArrayList<Packet>();
  public Node SybilParent = null;
```

Όπως φαίνεται και από τη δήλωση μεταβλητών κάθε κόμβος έχει πολλά χαρακτηριστικά, απαραίτητα για την λειτουργία της εφαρμογής. Κάθε κόμβος έχει ένα μοναδικό αναγνωριστικό `id`, μία θέση στο δίκτυο που καθορίζεται από 2 συντεταγμένες, `xpos` και `ypos` την εμβέλεια `range` και την ακτίνα του `radius`. Για τον αλγόριθμο δρομολόγησης οι κόμβοι χρειάζεται να γνωρίζουν τους γείτονες τους `NeighborList`, το επόμενο τους hop προς το sink `routeToSink`, την διαδρομή προς το sink `HopList`, το κόστος τους `costToSink` και το `routingScore` που χρησιμοποιείται για την επιλογή του επόμενου hop όταν υπάρχει ισοπαλία κόστους. Κατά την διάρκεια της προσομοίωσης χρειάζεται να αποθηκεύουμε κάποια στατιστικά που αφορούν την κίνηση των πακέτων από κόμβο σε κόμβο: `packetsDropped`, `packetsSent`, `packetsTransmited`, `packetsOnSink`, `currDelay`, `AllDelays`. Όλοι οι κόμβοι είναι απαραίτητο να έχουν μία ουρά (`queue`) όπου αποθηκεύονται προσωρινά τα πακέτα προς μετάδοση και μία ένδειξη μπαταρίας (`battery`) που μειώνεται με την αποστολή πακέτων.

Υπάρχουν επίσης μεταβλητές που μας βοηθούν να τοποθετήσουμε τους κόμβους στο SaveView (`angle`, `saveRadius`, `angleChanged`, `rightend`, `leftend`, `saveColor`,

ChildrenList) οι οποίες όμως θα αναλυθούν αργότερα μαζί με αυτή την απεικόνιση. Τέλος μεταβλητές όπως οι SybilParent, oblivion, missedWindows έχουν να κάνουν με τις επιθέσεις που πραγματοποιούνται στους κόμβους. Η κλάση Node εκτός από τις setters και getters μεθόδους για τις παραπάνω μεταβλητές έχει επίσης μεθόδους για: την τοποθέτηση των κόμβων στο Normal View, τον υπολογισμό της διαδρομής μέχρι τον KK, τον υπολογισμό των παιδιών και τον υπολογισμό του αριθμού των κόμβων του υποδέντρου του.

Η δεύτερη κλάση της εφαρμογής είναι το Packet. Αναπαριστά το πακέτο που δημιουργεί κάθε κόμβος και στέλνει μέσω της μεθόδου δρομολόγησης στον κεντρικό κόμβο.

```
public class Packet {
 public int id;
 public Node source;
 public Node nextHop;
 public Node current;
 public int size; //bytes
 public int timestamp; //seconds
 public boolean isRoutingPacket = false;
 public TreeMap<String,Float> RouteInfo = new TreeMap<String,Float>();


  Packet(int tempId, Node tempSource, Node tempnextHop, Node tempCurrent, int
tempSize, int temptimestamp, boolean TempisRoutingPacket,TreeMap<String,Float>
tempRouteInfo) {
    id=tempId;
    source=tempSource;
    nextHop=tempnextHop;
    current=tempCurrent;
    size=tempSize;
    timestamp=temptimestamp;
    isRoutingPacket=TempisRoutingPacket;
    if(tempRouteInfo!=null) {
      RouteInfo = tempRouteInfo;
    } else {
      RouteInfo.put("route",0.0);
      RouteInfo.put("cost",9999.0);
    }

     }

}
```

Κάθε πακέτο χαρακτηρίζεται από ένα μοναδικό αναγνωριστικό id, την πηγή του source, τον κόμβο προορισμού nextHop, τον κόμβο που βρίσκεται current, το μέγεθος του size, την χρονική στιγμή δημιουργίας του timestamp, εάν επρόκειτο για πακέτο δρομολόγησης isRoutingPacket και έναν πίνακα με πληροφορίες για την δρομολόγηση RouteInfo. Πέρα από τον constructor και τα setters/getters δεν υπάρχουν άλλες μέθοδοι σε αυτή την κλάση.

Η τελευταία κλάση είναι η κύρια κλάση της εφαρμογής στη οποία βρίσκονται οι εισαγωγές βιβλιοθηκών, αρχικοποιείς πολλών μεταβλητών και πινάκων, οι controllers

του GUI και οι 2 βασικές μέθοδοι του Processing, setup() και draw(). Η setup() εκτελείται μία φορά μόλις ξεκινάει το πρόγραμμα και ρυθμίζει το μέγεθος του παραθύρου, αρχικοποιεί τα γραφικά, τις γραμματοσειρές και κάποιες χρονικές μεταβλητές. Η draw() είναι η μέθοδος που περικλείει όλες τις λειτουργίες του προγράμματος, ό,τι φαίνεται εκείνη τη στιγμή στην οθόνη είναι αυτό που εκτελείται στην draw(). Η draw() είναι ένας ατέρμον βρόγχος, η «πλοήγηση» μέσα στα διάφορα κομμάτια του και η επιλογή του τι θα εκτελεστεί κάθε χρονική στιγμή γίνεται με την χρήση flags. Για παράδειγμα με το που ξεκινάει η εκτέλεση της εφαρμογής υπάρχει ένα flag που δηλώνει ότι θα φαίνεται η οθόνη υποδοχής, ανάλογα με την επιλογή λειτουργίας που θα κάνει ο χρήστης θα ενεργοποιηθεί κάποιο άλλο flag για να την εκτελέσει και το flag της οθόνης υποδοχής θα πάρει την τιμή false. Υπάρχουν flags για την κάθε απεικόνιση, επίθεση, menu, την παραγωγή κίνησης, την δρομολόγηση κλπ…

Οι κύριες μέθοδοι

Θα ξεκινήσουμε να περιγράφουμε την λογική και την λειτουργία της εφαρμογής με την ανάλυση των βασικών της μεθόδων. Αρχικά έχουμε την `initf` η οποία εκτελεί την αρχικοποίηση των κόμβων και είναι χωρισμένη σε τρία σκέλη ανάλογα με τον τύπο εισαγωγής κόμβων: α) εισαγωγή των προκαθορισμένων κόμβων, β) εισαγωγή κόμβων χειροκίνητα και γ) εισαγωγή κόμβων από αρχείο. Σε κάθε περίπτωση η λειτουργία είναι παρόμοια και περιλαμβάνει την ανάγνωση του αριθμού των κόμβων, την δημιουργία των αντικειμένων `Node` και την τοποθέτηση τους σε ένα Arraylist για άμεση πρόσβαση. Αφού δημιουργηθούν οι κόμβοι πρέπει να ανακαλύψουν τους γείτονες τους και για αυτό η `initf` καλεί την `DiscoverNeighbors()`. Η ανακάλυψη των γειτόνων γίνεται ελέγχοντας αν η απόσταση των κέντρων δύο κόμβων είναι μικρότερη από την ακτίνα του κόμβου, επομένως αν ένας κόμβος βρίσκεται μέσα στην ακτίνα του άλλου τότε θεωρείται γείτονας.

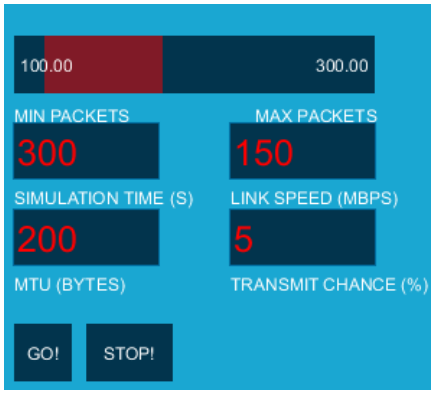
```
if(dist(Nodes.get(i).getXpos(),Nodes.get(i).getYpos(),Nodes.get(j).getXpos(),Nodes.ge
t(j).getYpos()) <= Nodes.get(i).getRange())
```

Βέβαια πρέπει να ληφθούν υπόψη και άλλες παράμετροι που αφορούν τις επιθέσεις, καθώς κάποιες από αυτές επηρεάζουν την απεικόνιση των κόμβων για το ποιους θεωρούν γείτονες.

Αφού οι κόμβοι ανακαλύψουν τους γείτονες τους πρέπει να εκτελεστεί ο αλγόριθμος δρομολόγησης για να βρεθούν οι διαδρομές των πακέτων προς τον κεντρικό κόμβο με την `routing` . Κάθε κόμβος δημιουργεί πακέτα δρομολόγησης και τα στέλνει στους γείτονες του. Κάθε πακέτο αναφέρει το κόστος διαδρομής σε hops που έχει ο κόμβος μέχρι τον KK. Αρχικά οι κόμβοι έχουν άπειρο κόστος, όταν όμως κάποιος ανακαλύψει πως έχει γείτονα τον 0 (sink node) θέτει το κόστος του σε 1 και το διαφημίζει στους γείτονες του οι οποίοι θα έχουν κόστος 2 και έτσι συνεχίζεται μέχρι όλοι οι κόμβοι να αποκτήσουν ένα έγκυρο μονοπάτι προς τον 0. Ο αλγόριθμος αυτός εκτελείται κάθε 500ms έτσι αν υπάρξει οποιαδήποτε αλλαγή στο δίκτυο πχ επίθεση ή αποτυχία κόμβου οι υπόλοιποι κόμβοι θα προσαρμοστούν στα καινούρια δεδομένα αλλάζοντας τις

διαδρομές τους. Εάν για κάποιον λόγο δεν υπάρχει διαθέσιμη διαδρομή προς τον sink ο κόμβος χαρακτηρίζεται ως αποσυνδεδεμένος και απενεργοποιείται.

Ένας προσομοιωτής δικτύου θα πρέπει να είναι σε θέση να προσομοιώσει κίνηση πακέτων μέσα στο δίκτυο. Η `StartTraffic` ξεκινάει να εκτελείται μόλις ο χρήστης εισάγει τις παραμέτρους για την προσομοίωση (χρόνο προσομοίωσης, μέγεθος πακέτων, ταχύτητα δικτύου, πιθανότητα μετάδοσης ανά κόμβο, αριθμός πακέτων ανά κόμβο) και πατήσει το “GO!”.



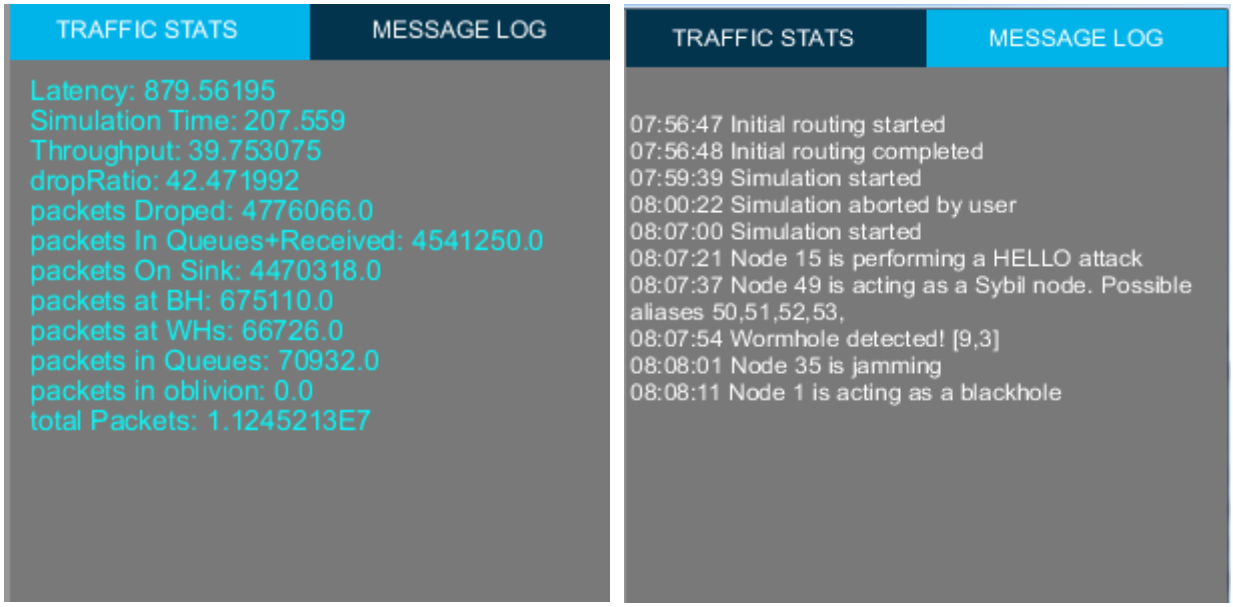
Εικόνα.14: Μενού παραμέτρων προσομοίωσης

Σε μία επανάληψη που τρέχει για κάθε κόμβο υπολογίζεται εάν αυτός θα μεταδώσει και πόσα πακέτα θα μεταδώσει, έπειτα δημιουργούνται τα πακέτα αυτά με τα αντίστοιχα χαρακτηριστικά τους και τοποθετούνται στην ουρά του κόμβου. Επίσης κρατούνται στατιστικά για το πόσα πακέτα δημιουργήθηκαν.

```
Packet tPacket = new
Packet(w,Nodes.get(t),Nodes.get(Nodes.get(t).getrouteToSink()),Nodes.get(t),MTU,Simul
ationTimer,false,null);
Nodes.get(t).queue.add(tPacket);
```

Το επόμενο βήμα είναι να υπολογιστεί πόσα πακέτα στέλνονται ανά δευτερόλεπτο (`pps`) ανάλογα με την ταχύτητα μετάδοσης και το μέγεθος του πακέτου. Μία επανάληψη που εκτελείται κάθε 66.66ms καλεί την `ProccessQueue` για κάθε κόμβο δίνοντας τις ως παράμετρο να επεξεργαστεί `pps/15` πακέτα. Η `ProccessQueue` αφαιρεί τα πακέτα αυτά από την ουρά του κόμβου και αφού εκτελέσει διάφορους ελέγχους τα τοποθετεί στην ουρά του επόμενου κόμβου στην διαδρομή προς τον sink. Εάν ένα πακέτο βρίσκεται στην ουρά για πάνω από 10s απορρίπτεται και καταγράφεται. Όταν το πακέτο τελικά φτάσει στον κεντρικό κόμβο, ενημερώνονται τα στατιστικά του κόμβου και καταστρέφεται. Επίσης στην `StartTraffic`, στην ίδια loop όπου καλείται η `ProccessQueue`, υπολογίζεται η μέση καθυστέρηση (latency) του δικτύου μετρώντας πόση ώρα κάνει να παραδοθεί το κάθε πακέτο. Σε όσους κόμβους έχουν μεγαλύτερο latency από το μέσο όρο του δικτύου γίνεται εύρεση του πρώτου κοινού τους κόμβου (από την `FindFirstCommonNode`) και το χρώμα του αλλάζει ανάλογα με το πόσο μεγαλύτερο είναι το latency από το μέσο για να δειχθεί ότι υπάρχει συμφόρηση. Στους κόμβους που έχουν να παραδώσουν πακέτο στον sink για χρόνο μεγαλύτερο των 30 δευτερολέπτων το μέγεθος τους αυξάνεται σταδιακά (σαν να φουσκώνουν) για να υποδηλωθεί αυτό το γεγονός.

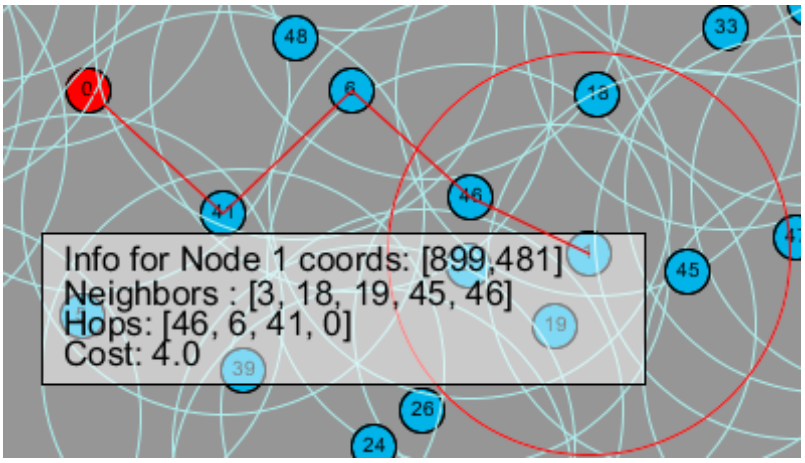
Στην δεξιά μεριά του παραθύρου της εφαρμογής υπάρχει μία περιοχή όπου εμφανίζονται ζωντανά στατιστικά και μηνύματα κατά τη διάρκεια της προσομοίωσης. Την δουλειά αυτή έχουν αναλάβει οι μέθοδοι `ShowTrafficStats` και `ShowMessages`.



Εικόνα.15

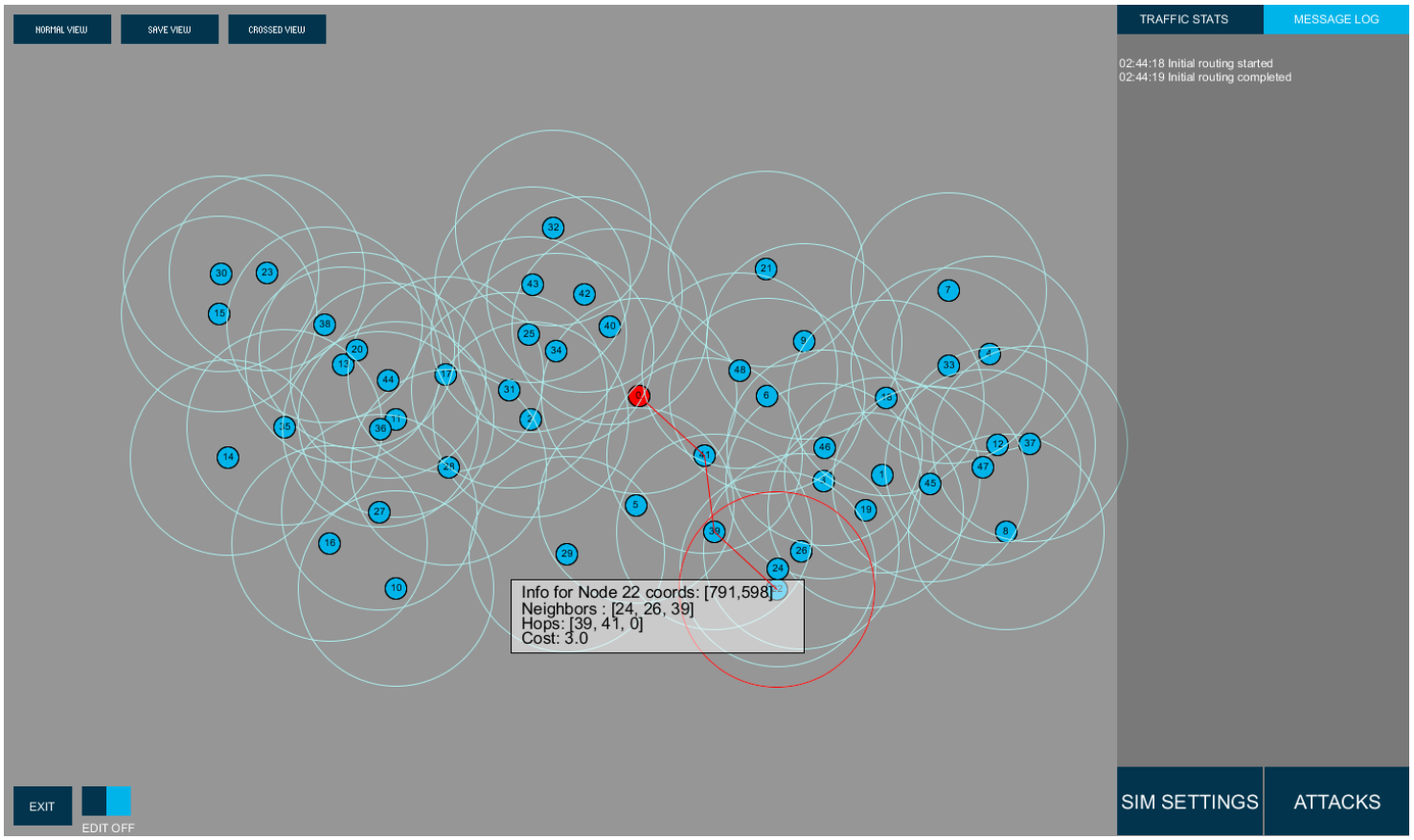
Πέρα από τις κύριες μεθόδους που αναφέρθηκαν παραπάνω υπάρχουν και άλλες μέθοδοι που μας βοηθούν σε διάφορες λειτουργίες.

- Η `CalculatePacketDelay` δέχεται ως παράμετρο ένα πακέτο και υπολογίζει τον χρόνο που έκανε να παραδοθεί στον προορισμό του, επίσης προσαρμόζει το πάχος της περιφέρειας του κύκλου του κόμβου ανάλογα με το latency, όσο μεγαλύτερο τόσο πιο παχύ. Η κλήση της γίνεται όταν κάποιο πακέτο παραδίδεται στον sink node.
- Η `FindFirstCommonNode` δέχεται σαν παραμέτρους 2 Nodes. Στη συνέχεια συγκρίνει τις διαδρομές τους και ψάχνει να βρει αν είναι μέλη του ίδιου δέντρου, αν δεν είναι τερματίζει. Αν βρίσκονται στο ίδιο δέντρο αναζητάει τον πρώτο τους κοινό κόμβο. Καλείται από την `StartTraffic`.
- Η `nodeInfo` καλείται όταν ο χρήστης κάνει mouse over σε έναν κόμβο και εμφανίζει πληροφορίες για τον κόμβο αυτό στην θέση του ποντικιού. Οι πληροφορίες αυτές περιλαμβάνουν το id του, τις συντεταγμένες του, τους γείτονες του, την διαδρομή μέχρι το sink, το κόστος και το score του. Τέλος εμφανίζει μία κόκκινη γραμμή που συνδέει τους κόμβους που βρίσκονται στο μονοπάτι προς τον sink.



Εικόνα. 16: Πληροφορίες κόμβου με mouse over

Απεικονίσεις - Views



Εικόνα. 17: Normal View με 50 κόμβους

Στην εφαρμογή έχουν αναπτυχθεί 4 διαφορετικές απεικονίσεις για να μας βοηθήσουν στην απεικόνιση των επιθέσεων και να διευκολύνουν την ανάλυσή τους. Η πρώτη απεικόνιση είναι το Normal View ή αλλιώς Geographical View στο οποίο φαίνονται οι κόμβοι στην φυσική τους θέση στον χώρο.

Normal View

Το Normal View έχει μία βασική επανάληψη που τρέχει για όλους τους κόμβους και καλεί την μέθοδο `placeNode`. Η if μέσα στην επανάληψη εκτελείται όταν θέλουμε να αλλάξουμε την θέση κάποιου κόμβου.

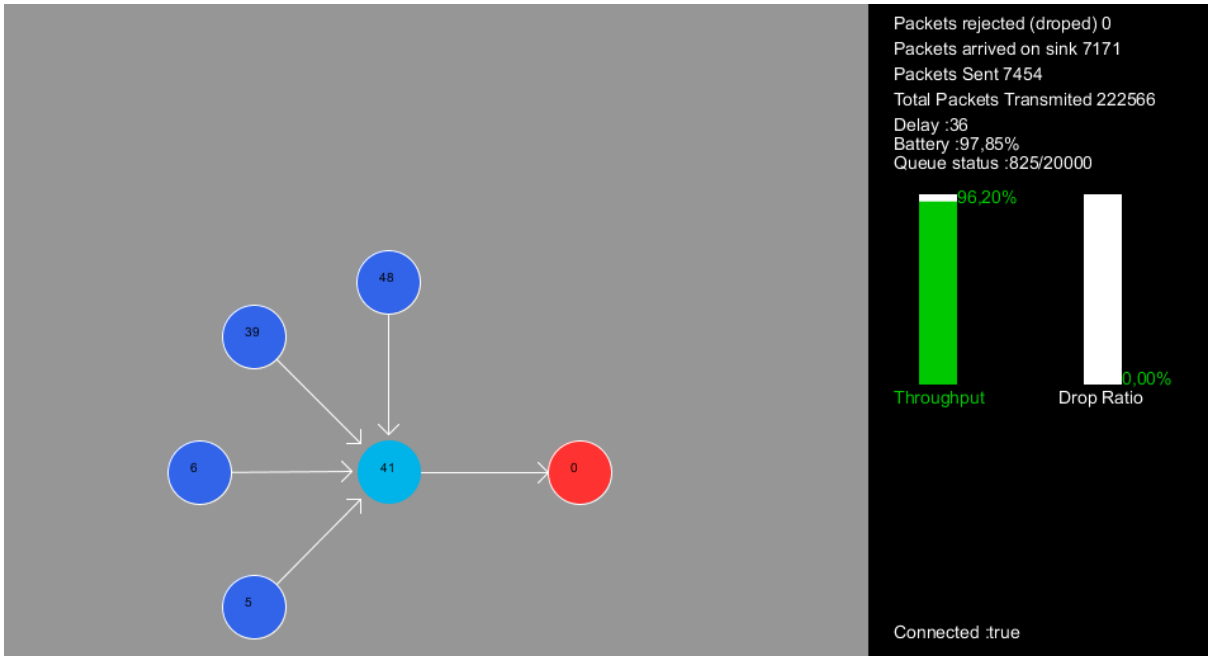
```
for (int counter = 0; counter < numberNodes; counter++) {
	if(lockednode==counter && EditModeFlag) {
		ellipse(coordx, coordy,20,20);
		Nodes.get(counter).setYpos(coordy);
		Nodes.get(counter).setXpos(coordx);
	}
	Nodes.get(counter).placeNode(true);
}
```

Η `placeNode` είναι μέθοδος της κλάσης Node και όταν καλείται ζωγραφίζει έναν κύκλο με ακτίνα την ακτίνα που έχει οριστεί για κάθε κόμβο και κέντρο τις συντεταγμένες του κόμβου. Επίσης μέσα στον κύκλο αναγράφεται το id του κόμβου. Τέλος το χρώμα του κύκλου εξαρτάται από τον τύπο του κόμβου.

Στο Normal View επίσης στο δεξί μέρος της οθόνης εμφανίζονται τα στατιστικά κίνησης των πακέτων, τα μηνύματα και το μενού της εφαρμογής που περιέχει ρυθμίσεις για την προσομοίωση και τις επιθέσεις. Τα κουμπιά πάνω αριστερά μας επιτρέπουν να πλοηγούμαστε στις οπτικές. Αυτή η οπτική μας παρέχει απλώς μία πανοραμική άποψη του δικτύου και βασικές πληροφορίες για τους κόμβους όταν κάνουμε mouse over. Σε περίπτωση επιθέσεων το Normal View θα χρωματίσει τους κακόβουλους κόμβους με το χρώμα που έχουμε επιλέξει για κάθε επίθεση αλλά δεν είναι σε θέση να μας δώσει περαιτέρω πληροφορίες για το πώς έχει αλλάξει η διάταξη των κόμβων και πως επηρεάζεται η κίνηση των πακέτων στο δίκτυο. Για να αποκτήσουμε μία καλύτερη εικόνα για το τι συμβαίνει θα πρέπει να χρησιμοποιήσουμε τις άλλες απεικονίσεις που έχουμε αναπτύξει.

Node View

Η επόμενη απεικόνιση είναι το Node View. Για να μεταφερθούμε σε αυτό αρκεί να κάνουμε click σε έναν κόμβο. Θα δούμε την άμεση οικογένεια του κόμβου με τα παιδιά αριστερά, τον πατέρα δεξιά και τον κόμβο μας στην μέση. Στην μαύρη στήλη δεξιά εμφανίζονται αναλυτικά στατιστικά για την κίνηση πακέτων, την καθυστέρηση, το μέγεθος της ουράς και την μπαταρία. Τέλος εμφανίζεται στο κάτω μέρος εάν ο κόμβος είναι συνδεδεμένος στο δίκτυο



Εικόνα.18: Node View

Για να επιτύχουμε αυτό το αποτέλεσμα χρειάζεται να γνωρίζουμε τους γείτονες, τον πατέρα και τα παιδιά του κόμβου:

```
Node parent = Nodes.get(thisNode.getrouteToSink());
      for(int j=0;j<Neighbors.size();j++){ //find the children of the given node
            if(Nodes.get(Nodes.get(Neighbors.get(j)).getrouteToSink()) == thisNode){
                  children++;
                  ChildrenList.add(Neighbors.get(j));
             }
```

```
	}
```

Έπειτα τοποθετούμε τον κόμβο στην μέση και υπολογίζουμε τις θέσεις που θα μπουν τα παιδιά περιμετρικά αυτού:

```
float startangle = PI/2;
float endangle = 3*PI/2;
float step = (3*PI/2-PI/2)/children;
float angle=startangle;
for(int i=0;i<ChildrenList.size();i++) {
	angle+=step;
	x=(int)(xSize/2 + 150 * cos(angle));
	y=(int)(ySize/2 + 150 * sin(angle));
	drawArrow(x,y,120,angle+PI);
	fill(50,100,234);
	ellipse(x,y,50,50);
	fill(0);
	text(Nodes.get(ChildrenList.get(i)).getId(),x-7,y);
}
```

Αφού τοποθετήσουμε τα παιδιά σειρά έχει να τοποθετηθεί ο πατέρας, επειδή τον θέλουμε από τα δεξιά του κόμβου η γωνία που θα χρησιμοποιηθεί είναι η 0 rad.

```
x=(int)(xSize/2 + 150 * cos(0));
y=(int)(ySize/2 + 150 * sin(0));
fill(255,50,50);
ellipse(x,y,50,50);
fill(0);
text(parent.getId(),x-7,y);
drawArrow(xSize/2+25,ySize/2,100,0);
```

Τέλος πρέπει να λάβουμε της πληροφορίες κίνησης του κόμβου, να υπολογίσουμε το throughput και drop ratio και να τα εμφανίσουμε την οθόνη.

```
int DropedPacketsCount=thisNode.GetpacketsDropped();
int SentPacketsCount = thisNode.GetpacketsTransmited();
int PacketsReceived = thisNode.GetpacketsOnSink();
int PacketsSent =  thisNode.GetpacketsSent();
int currentQueueSize = thisNode.queue.size();
boolean isConnected = thisNode.isConnected;
int delay = 0;

if(DelayPerNode.containsKey((thisNode.getId()))) {
	delay = DelayPerNode.get(thisNode.getId());
}
float DropRatio = (float)DropedPacketsCount/SentPacketsCount*100;
if(DropRatio>100) {DropRatio=100;}
	PFont pgFont = createFont("Verdana-24",50);
	textFont(pgFont, 14);
	textAlign(LEFT);
	fill(255);
	if(SentPacketsCount>0) {
		text("Total Packets Transmited "+SentPacketsCount,xSize-280,80);
		text("Drop Ratio",xSize-150,315);
		pushMatrix();
		translate(xSize-100,300);
		rotate(PI);
		fill(255);
```

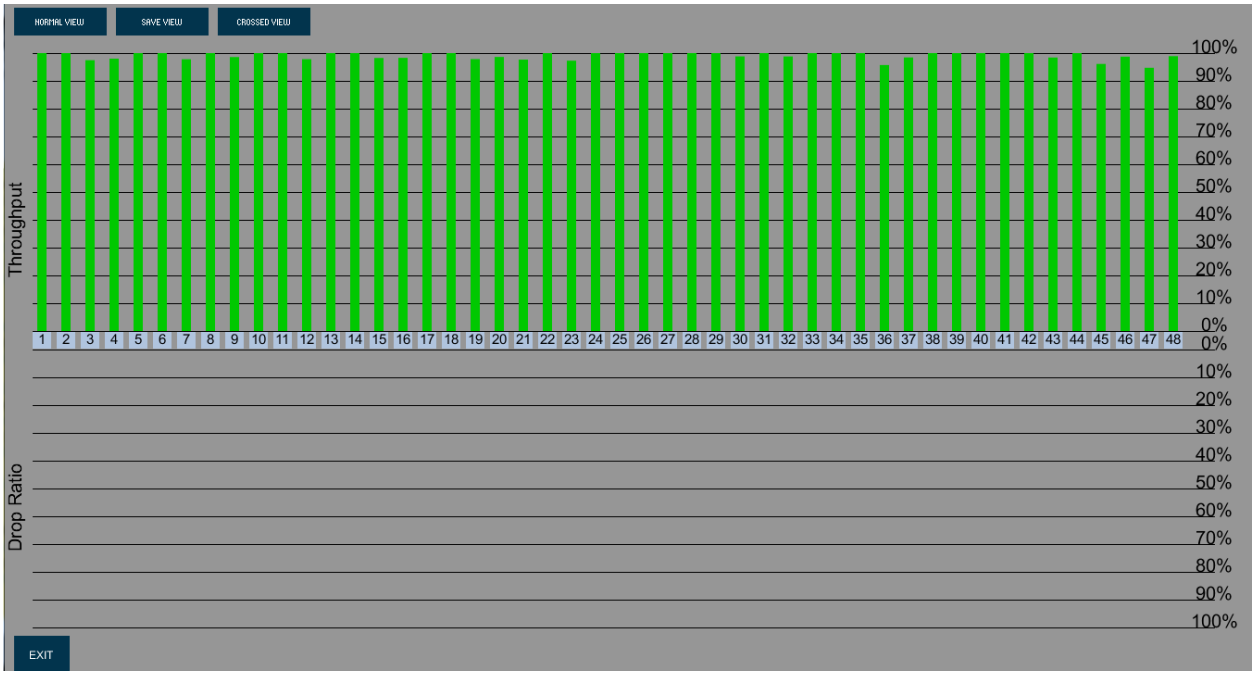
```
            rect(0,0,30,150);
            if(DropRatio >= 60) {
                  fill(255,0,0);
             }
             if (DropRatio < 60 && DropRatio >=10){
                  fill(255,255,0);
             }
             if(DropRatio < 10) {
                  fill(0,200,0);
             }
             rect(0,0,30,150*DropRatio/100);
             rotate(-PI);
             text(formatter.format(DropRatio)+"%",0,0-150*DropRatio/100);
             popMatrix();
      }
      if(PacketsSent > 0) {
            float thput = (float)PacketsReceived/PacketsSent*100;
            text("Throughput",xSize-280,315);
            pushMatrix();
            translate(xSize-230,300);
            rotate(PI);
            fill(255);
            rect(0,0,30,150);

            if(thput >= 80) {
                  fill(0,200,0);
            }
            if (thput < 80 && thput >=50){
                  fill(255,255,0);
            }
            if(thput < 50) {
                  fill(255,0,0);
            }
            rect(0,0,30,150*thput/100);
            rotate(-PI);
            text(formatter.format(thput)+"%",0,0-150*thput/100);
            popMatrix();

      }
fill(255);
text("Packets rejected (droped) "+DropedPacketsCount,xSize-280,20);
text("Packets arrived on sink "+PacketsReceived,xSize-280,40);
text("Packets Sent "+PacketsSent,xSize-280,60);
text("Queue status :"+currentQueueSize+"/"+MaxNodeQueueSize,xSize-280,130);
text("Delay :" + delay,xSize-280,100);
text("Battery :" + formatter.format(battery) +"%",xSize-280,115);
text("Connected :" +isConnected,xSize-280,500);
```

Το Node View λοιπόν είναι μία άποψη της γειτονιάς του κόμβου που μας επιτρέπει να βλέπουμε από κοντά ό,τι συμβαίνει.

Crossed View

Η τρίτη οπτική της εφαρμογής είναι το Crossed View και μας δείχνει 2 γραφήματα με μπάρες για κάθε κόμβο συγκεντρωτικά. Το πρώτο είναι το throughput, ή αλλιώς το ποσοστό τον πακέτων που έχουν φθάσει στον sink node σε σχέση με τα συνολικά πακέτα που έχουν σταλεί. Το δεύτ είναι το Drop ratio, ή αλλιώς το ποσοστό των

πακέτων που έχουν απορριφθεί για κάποιον λόγο και δεν έφτασαν στον προορισμό τους.



Εικόνα.19: Crossed View σε κανονική λειτουργία δικτύου

Αρχικά πρέπει να σχεδιάσουμε τις οριζόντιες γραμμές και να τυπώσουμε τα ποσοστά στα δεξιά και τα labels αριστερά:

```
pushMatrix();
translate(18,550);
rotate(-PI/2);
text("Drop Ratio",0,0);
text("Throughput",300,0);
rotate(+PI/2);
popMatrix();
int percentstep = 10;
for(int i=0;i<=100;i+=10) {
	text(i+"%",xSize-50,ySize/2-percentstep);
	line(xSize-50,ySize/2-percentstep,30,ySize/2-percentstep);
	percentstep+=30;
}
percentstep = 10;
for(int i=0;i<=100;i+=10) {
	textAlign(CENTER);
	text(i+"%",xSize-50,ySize/2+percentstep);
	line(xSize-50,ySize/2+percentstep,30,ySize/2+percentstep);
	percentstep+=30;
}
```

Οι εντολές `pushMatrix(); translate(18,550); rotate(-PI/2);` μας λένε να αποθηκεύσουμε το τωρινό σύστημα συντεταγμένων, να μεταφερθούμε στο σημείο (18,550) και να περιστρέψουμε το σύστημα συντεταγμένων κατά 90 μοίρες. Αφού γίνει η μετατροπή και τυπωθούν κάθετα οι λέξεις “Drop Ratio” και “Throughput” επαναφέρουμε το προηγούμενο σύστημα συντεταγμένων με την `popMatrix();`

Στη συνέχεια πρέπει να υπολογίσουμε για κάθε κόμβο το throughput και drop ratio και να τυπώσουμε το id του κόμβου:

```
int step = 0;
for(int i=1; i<numberNodes; i++) {
	Node thisNode = Nodes.get(i);
	int DropedPacketsCount=thisNode.GetpacketsDropped();
	int SentPacketsCount = thisNode.GetpacketsTransmited();
	int PacketsReceived = thisNode.GetpacketsOnSink();
	int PacketsSent =  thisNode.GetpacketsSent();
	float DropRatio = (float)DropedPacketsCount/SentPacketsCount;
	if(DropRatio>1){DropRatio=1;}
		float thput = 0;
		if(PacketsSent > 0) {
			thput = (float)PacketsReceived/PacketsSent;
			 if(thput>=1){thput=1;}
		}
		rectMode(CENTER);
		noStroke();
		fill(176,196,222);
		rect(40+step,ySize/2,20,18);
		fill(0);
		textFont(f);
		text(i,40+step,ySize/2+4);
```

Τέλος αυτό που μένει είναι να δημιουργήσουμε τις μπάρες και να τις χρωματίσουμε ανάλογα με το ποσοστό του στατιστικού. Οι μπάρες είναι ορθογώνια παραλληλόγραμμα που ξεκινάνε από το 0% (κέντρο της οθόνης) και έχουν ύψος μέχρι το 100% (κάτω και πάνω μέρος της οθόνης). Το μέγεθος τους μεταβάλλεται ανάλογα με το ποσοστό του throughput και του drop ratio.

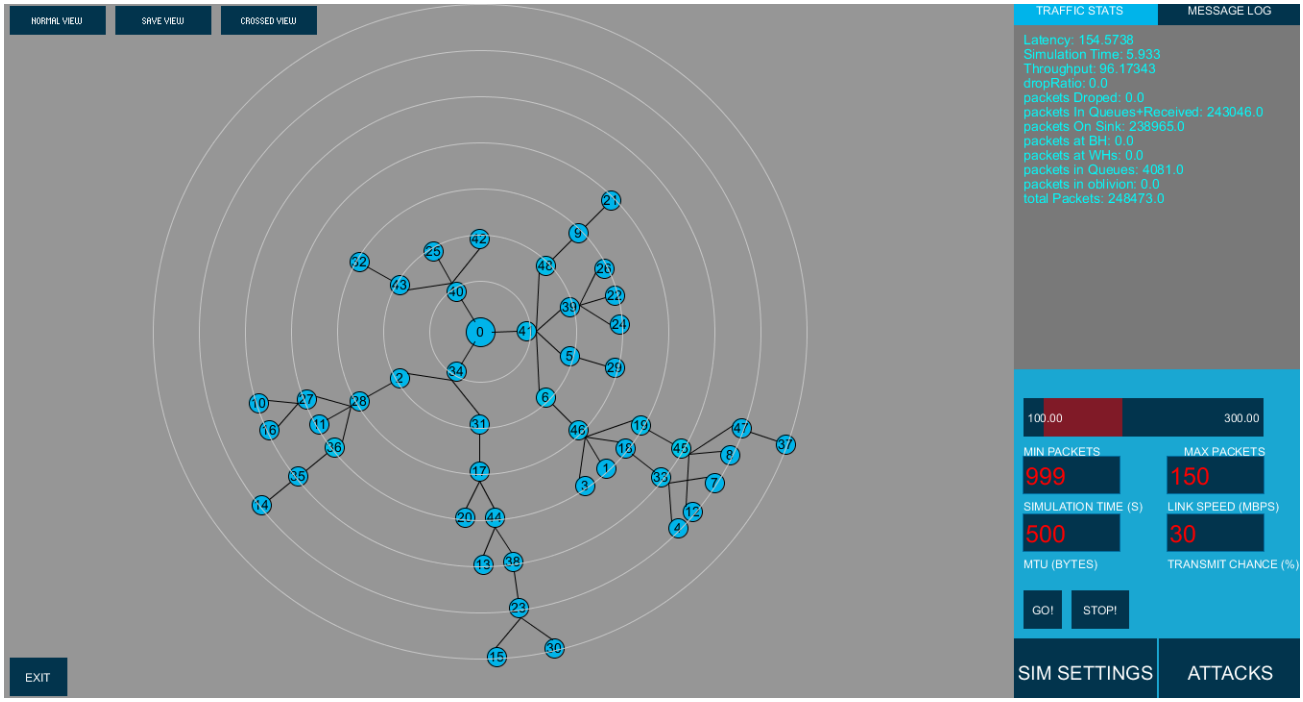
```
if(DropRatio >= 0.6) {
	fill(255,0,0);
} if (DropRatio < 0.60 && DropRatio >=0.10){
	fill(255,255,0);
} if(DropRatio < 0.10) {
	fill(0,200,0);
}
rect(35+step,ySize/2+10,10,300*DropRatio);
if(thput >= 0.80) {
	fill(0,200,0);
} if (thput < 0.80 && thput >=0.50){
	fill(255,255,0);
} if(thput < 0.50) {
	fill(255,0,0);
}
rect(35+step,ySize/2-10,10,-300*thput);
fill(0);
step+=(xSize-70)/numberNodes;
}
```

Έχοντας μία συγκεντρωτική άποψη αυτών των χαρακτηριστικών ο χρήστης μπορεί να εκτιμήσει την απόδοση του δικτύου και με μία ματιά να αναγνωρίσει ποιος κόμβος έχει πρόβλημα. Χαμηλό throughput σημαίνει ότι υπάρχει κάποιο είδος AtY επίθεσης

στον κόμβο και τα πακέτα του δεν φτάνουν στον προορισμό τους. Υψηλό drop ratio δηλώνει ότι κάποιο γεγονός έχει αναγκάσει τον κόμβο να απορρίπτει τα πακέτα που λαμβάνει από τους γείτονες του. Παρόλο που αυτά τα στατιστικά μας δίνουν μία πιο καλή εικόνα για το εάν υπάρχει πρόβλημα στο δίκτυο δεν επαρκούν για την αναγνώριση του προβλήματος.

Save View

Το Save View είναι η τελευταία και παράλληλα πιο σημαντική οπτική της εφαρμογής. Είναι εμπνευσμένο από το XMA του Sensor Anomaly Visualization Engine (SAVE). Μέσω αυτού ο χρήστης μπορεί να ανιχνεύσει αμέσως τις αλλαγές στην τοπολογία του δικτύου, το υψηλό latency κάποιου κόμβου, την ύπαρξη συμφόρησης, την έλλειψη επικοινωνίας κάποιου κόμβου κλπ. Επίσης κάθε κόμβος που πραγματοποιεί μία επίθεση χρωματίζεται με ένα χαρακτηριστικό χρώμα για την εύκολη αναγνώρισή του, κόμβοι που επηρεάζονται από την επίθεση χρωματίζονται επίσης.



Εικόνα.20: Save View με 50 κόμβους χωρίς επιθέσεις

Στο κέντρο της οθόνης τοποθετείται ο sink node (0). Οι άσπροι ομόκεντροι κύκλοι αναπαριστούν το κόστος σε hops. Οι υπόλοιποι κόμβοι τοποθετούνται στον αντίστοιχο κύκλο ανάλογα με το κόστος τους, δηλαδή οι κόμβοι 34, 40, 41 έχουν κόστος 1, οι 15, 30, 37 έχουν κόστος 7 κλπ. Όπως βλέπετε οι κόμβοι σχηματίζουν κατευθυνόμενα δέντρα με μία κοινή ρίζα, τον κεντρικό κόμβο. Οι ακμές που ενώνουν τους κόμβους υποδηλώνουν το μονοπάτι που ακολουθούν τα πακέτα του κάθε κόμβου για να φτάσουν στον sink node. Οποιαδήποτε αλλαγή στην τοπολογία θα ανιχνευτεί αμέσως καθώς θα αλλάξει κάποιος γονέας ή κάποιο παιδί και συνεπώς θα τροποποιηθεί και το αντίστοιχο δέντρο. Παρακάτω θα αναλύσουμε το πώς δημιουργήσαμε αυτή την απεικόνιση.

Αρχικά δημιουργούμε ένα table και τοποθετούμε μέσα όλους τους κόμβους του δικτύου και κάποιες απαραίτητες μεταβλητές. Ταυτόχρονα υπολογίζουμε και το μέγιστο κόστος του δικτύου:

```
Table NodeTable = new Table();
NodeTable.addColumn("id");
NodeTable.addColumn("xpos");
NodeTable.addColumn("ypos");
NodeTable.addColumn("route");
NodeTable.addColumn("cost");
NodeTable.addColumn("angle");

for(int i=0;i<numberNodes;i++) {
	if(Nodes.get(i).isConnected) {
		TableRow newRow = NodeTable.addRow();
		float cost = Nodes.get(i).getcostToSink();
		int route = Nodes.get(i).getrouteToSink();
		Neighbors = Nodes.get(i).GetNeighborList();
		newRow.setInt("id",i);
		newRow.setInt("route",route);
		newRow.setFloat("cost",cost);
		if(cost>maxCost) {
			maxCost=cost;
		}

	}
}
```

Στη συνέχεια δημιουργούμε έναν ακόμα πίνακα που περιέχει για κάθε κόστος πόσοι κόμβοι έχουν αυτό το κόστος, και ποιοι είναι αυτοί οι κόμβοι.

```
Table Costs = new Table();
String nodestring;
Costs.addColumn("cost");
Costs.addColumn("nodeCount");
Costs.addColumn("nodestring");
for(int j=1;j<=maxCost;j++) {
	int counter=0;
	nodestring="";
	for(int k=1;k<numberNodes;k++) {
		if(Nodes.get(k).getcostToSink() == j) {
			nodestring=nodestring+","+k;
			counter++;
		 }

	}
	if(nodestring.length()>0){
		nodestring=nodestring.substring(1);
	}
	TableRow newRow = Costs.addRow();
	newRow.setInt("cost", j);
	newRow.setInt("nodeCount", counter);
	newRow.setString("nodestring", nodestring);
}
```

Αφού συλλέξαμε τις πληροφορίες που θέλαμε και τις τοποθετήσαμε σε δομές δεδομένων που μας βολεύουν, σχεδιάζουμε τους ομόκεντρους κύκλους με κέντρο τον sink node. Στην περίμετρο αυτών των κύκλων θα τοποθετήσουμε στην συνέχεια τους κόμβους:

```
int step=0;
for(int j=0;j<maxCost;j++) {
	noFill();
	stroke(200);
	ellipse(saveXsize/2,(ySize-40)/2,ySize/maxCost+step,ySize/maxCost+step);
	if(flaag) {
		step+=(ySize)/maxCost;
	} else {
		step+=(ySize-60)/maxCost;
	}
}
```

Μόλις ζωγραφιστούν οι βοηθητικοί κύκλοι ξεκινάει το δύσκολο κομμάτι: η τοποθέτηση των κόμβων σε δέντρα γύρω από ένα κεντρικό σημείο. Το δέντρο δημιουργείται από μέσα προς τα έξω, επομένως χρειαζόμαστε μία επανάληψη που να τρέχει για κάθε κόστος. Μέσα στην επανάληψη αναζητούμε στους πίνακες που δημιουργήσαμε πριν τους κόμβους που έχουν κόστος i και ορίζεται ως αρχική γωνία η 2π.

```
for (int i=1;i<=maxCost;i++) {
	TableRow result = Costs.matchRow(str(i), "cost");
	int NodeCount=result.getInt("nodeCount");
	nodestring= result.getString("nodestring");
	if(nodestring.length()>0){
		String[] NodeIds = nodestring.split(",");
		float  StartAngle=2*PI;
```

Έπειτα ξεκινάει μία άλλη επανάληψη για κάθε κόμβο του συγκεκριμένου κόστους. Αν το κόστος είναι 1 υπολογίζεται η αρχική απόσταση από τον 0 και η αρχική γωνία αυξάνεται ανάλογα με το πόσοι κόμβοι θα τοποθετηθούν στον κύκλο. Βάση της απόστασης και της γωνίας υπολογίζονται οι συντεταγμένες του κόμβου και τοποθετείται στην οθόνη. Έπειτα καλείται η μέθοδος `CreateSaveTree` για αυτόν τον κόμβο με σκοπό να δημιουργηθεί το υποδέντρο που τον έχει ως ρίζα. Το πώς δημιουργείται το υποδέντρο είναι μία πολύπλοκη διαδικασία που θα εξηγήσουμε στην συνέχεια.

```
for(int c=0;c<NodeIds.length;c++) {
	int StartNode = Integer.parseInt(NodeIds[c]);
	if(Nodes.get(StartNode).isConnected) {
		if(i==1) {
			float StartDistance = (((ySize-60)/maxCost)/2*i);
			StartAngle+= 2*PI/NodeCount;
			int x=(int)(saveXsize/2 + StartDistance * cos(StartAngle));
			int y=(int)((ySize-40)/2 + StartDistance * sin(StartAngle));
			Nodes.get(StartNode).setAngle(StartAngle);
			Nodes.get(StartNode).setSaveX(x);
			Nodes.get(StartNode).setSaveY(y);
			fill(Nodes.get(StartNode).saveColor);
			ellipse(x,y,20,20);
			fill(0);
```

```
                text(StartNode,x,y+5);
                noFill();
            }
            NodeTable                    =                    CreateSaveTree(StartNode,
      StartAngle,i,maxCost,Costs,NodeTable);
```

Εάν δεν είμαστε σε κόμβο κόστους 1, τότε παίρνουμε την γωνία του κόμβου μας και καλούμε την `CreateSaveTree` δίνοντας ως παράμετρο αυτή τη γωνία, ώστε το υποδέντρο να δημιουργηθεί στην μεριά του κόμβου αυτού.

```
      } else {
            StartAngle = Nodes.get(StartNode).getAngle();
            NodeTable = CreateSaveTree(StartNode,
StartAngle,i,maxCost,Costs,NodeTable);
      }
```

Σε αυτό το σημείο θα αναλύσουμε την `CreateSaveTree` για να γίνει κατανοητό πως δημιουργείται το υποδέντρο για κάθε κόμβο.

```
Table CreateSaveTree(int Node, float fatherangle,float fathercost, float maxCost,
Table Costs,Table NodeTable) {

noFill();
ArrayList<Integer> Neighbors = new ArrayList<Integer>();
ArrayList<Integer> ChildrenList = new ArrayList<Integer>();
Node thisChild;
Node itsParent;
int savetimer=0;
float distance=0;
int children=0;
int NodeCount=0;
float angle=fatherangle;
float centerangle=fatherangle;
float arc=0;
float rightend=0;
float leftend=0;
float anglestep=0;
int x=0;
int y=0;
Neighbors = Nodes.get(Node).GetNeighborList();
float childcost=fathercost+1;
String[] NodeIds=null;
```

Η μέθοδος δέχεται ως παραμέτρους τον κόμβο για τον οποίο θέλουμε να δημιουργήσουμε το υποδέντρο `Node`, την γωνία του κόμβου αυτού `fatherangle`, το κόστος του `fathercost`, το μέγιστο κόστος του δικτύου `maxCost` και τους πίνακες `Costs` και `NodeTable` που δημιουργήσαμε προηγουμένως.

Αρχικοποιούμε τα Arraylists `Neighbors` και `ChildrenList` στα οποία θα αποθηκευτούν οι γείτονες και τα παιδιά του υπό εξέταση κόμβου. Μετά αρχικοποιούμε τα αντικείμενα `thisChild` και `itsParent` που θα μας βοηθήσουν να έχουμε γρήγορη πρόσβαση στο κάθε παιδί και τον πατέρα του. Τέλος αρχικοποιούμε τις απαραίτητες μεταβλητές:

1. `Distance` : η απόσταση των παιδιών από το κέντρο του κύκλου (κόμβος 0).
2. `Children` : ο αριθμός των παιδιών του `Node`.

3. `NodeCount` : το σύνολο των κόμβων που έχουν κόστος ίσο με αυτό των παιδιών
4. `Angle` : η γωνία που θα έχει το κάθε παιδί.
5. `Centerangle` : η κεντρική γωνία, αρχικά ίση με του πατέρα, με κέντρο την οποίο θα σχηματιστεί ένα τόξο που στη περίμετρο του θα τοποθετηθούν τα παιδιά.
6. `Arc` : το τόξο σε rad που αναφέραμε στο 5.
7. `Rightend` : το δεξί πέρας του `arc` σε rad.
8. `Leftend` : το αριστερό πέρας του `arc` σε rad.
9. `Anglestep :` το βήμα κατά το οποίο θα αυξάνεται η `Angle.`
10. `x, y` : Προσωρινές μεταβλητές για τις συντεταγμένες του κάθε παιδιού.
11. `Childcost` : το κόστος των παιδιών, ίσο με το κόστος του πατέρα +1.
12. `NodeIds`: Ένας πίνακας με τα ids των κόμβων που βρίσκονται στον ίδιο κύκλο με τα παιδιά.

Αφού τελειώσουμε με την αρχικοποίηση μεταβλητών πρέπει τώρα να τις αναθέσουμε τιμές. Το πρώτο βήμα είναι να υπολογίσουμε την απόσταση. `Η flaag` αλλάζει την απόσταση σε περίπτωση που κάποιοι κόμβοι έχουν μεγαλύτερη ακτίνα.

```
if(childcost<=maxCost) { //check if we are at the end of the tree
	boolean flaag=false;
		for(int d=1;d<numberNodes;d++) {
			if(Nodes.get(d).getSaveRadius()>30) {
			  flaag=true;
			}
		    }
		    if(flaag) {
		      distance = ((((ySize)/maxCost)/2)*childcost);
		    } else {
		      distance = ((((ySize-60)/maxCost)/2)*childcost);
		    }
```

Στη συνέχεια βρίσκουμε το `NodeCount,` τα `NodeIds` και τα παιδιά του `Node.`

```
	TableRow result = Costs.matchRow("["+childcost+"]", "cost");
		    NodeCount=result.getInt("nodeCount"); //get how many nodes are on
this circle
	String nodestring= result.getString("nodestring");
	NodeIds = nodestring.split(","); //The ids of the nodes on the same circle as
the children

}
ChildrenList = Nodes.get(Node).getChildren();
children=ChildrenList.size();
```

Εάν υπάρχουν κόμβοι προς τοποθέτηση σε αυτόν τον κύκλο. Υπολογίζουμε το μέγεθος του τόξου ανάλογα με τον αριθμό των παιδιών και τον συνολικό αριθμό των κόμβων στον κύκλο. Βάση αυτού του τόξου υπολογίζουμε τα 2 άκρα του.

```
if(NodeCount>0) {
	centerangle=fatherangle;
	arc=2*PI*children+1/NodeCount;
	leftend=centerangle+arc/2;
	rightend=centerangle-arc/2;
	Nodes.get(Node).rightend=rightend;
```

```
Nodes.get(Node).leftend=leftend;
```

Οι τιμές `offset, offset2, offset3` έχουν υπολογιστεί πειραματικά και χρειάζονται για τον υπολογισμό του βήματος της γωνίας. Το βήμα αυτό είναι πολύ σημαντικό να υπολογιστεί σωστά ώστε οι κόμβοι που θα τοποθετηθούν να μην πέφτουν ο ένας πάνω στον άλλον. Το μέγεθος του βήματος εξαρτάται από τον αριθμό των παιδιών του κόμβου, το κόστος των παιδιών και τον συνολικό αριθμό κόμβων του δικτύου. Ο σκοπός είναι το βήμα να είναι μεγαλύτερο όταν υπάρχουν πολλά παιδιά, να μικραίνει όταν το κόστος είναι μικρότερο από 3 και μεγαλύτερο από 6, αλλά να μεγαλώνει για κόστος μεταξύ 3 και 5. Τέλος θέλουμε το βήμα να είναι μικρότερο όταν έχουμε λίγους κόμβους στο δίκτυο και μεγαλύτερο για περισσότερους.

```
float offset=4;
if(children > 5) {
	offset=1.3;
} else if (children <=5 && children >= 3) {
	offset=1.1;
} else {
	offset=1;
}
float offset2=0.6;
if(childcost <= 3) {
	offset2=0.4;
} else if (childcost <=5 && childcost > 3) {
	offset2=0.2;
} else {
	offset2=0.8;
}
float offset3=0;
if(numberNodes<=50) {
	offset3=0.5;
} else if(numberNodes>50 && numberNodes<=150) {
	offset3=0.8;
} else {
	offset3=1.1;
}

anglestep = offset3*arc*offset/(children*offset2*childcost);
```

Μόλις υπολογιστεί και το βήμα της γωνίας σειρά έχει η τοποθέτηση των παιδιών που είναι και το βασικό ζητούμενο. Ο τρόπος με τον οποίο θα τοποθετηθούν τα παιδιά εξαρτάται από τον αριθμό των παιδιών. Η πρώτη περίπτωση είναι όταν υπάρχει ένα παιδί και θα πρέπει να τοποθετηθεί στην ίδια ευθεία με τον πατέρα του. Συνεπώς οι γωνίες τους θα είναι ίσες. Με βάση την γωνία και την απόσταση υπολογίζονται οι συντεταγμένες και τοποθετείται το παιδί πάνω στον κύκλο. Αξίζει να σημειωθεί πως οι αλλαγές στην τοπολογία του δικτύου αλλά και ο μεγάλος αριθμός κόμβων ωθούσαν κάποιους κόμβους να επικαλύπτουν άλλους και να μην φαίνονται τα id τους. Για αυτόν τον λόγο δημιουργήσαμε την μέθοδο `checkCollisions` που ανιχνεύει αυτές τις επικαλύψεις και υπολογίζει μία νέα θέση ώστε οι κόμβοι να φαίνονται σωστά.

```
if(children == 1) {
	thisChild=Nodes.get(ChildrenList.get(0));
	if(thisChild.angleChanged && abs(thisChild.getAngle()-centerangle)<PI/2) {
```

```
            angle=thisChild.getAngle();
      } else {
             angle=centerangle;
      }

      x=(int)(saveXsize/2 + distance * cos(angle));
      y=(int)((ySize-40)/2 + distance * sin(angle));
      ArrayList<Float> returnres =
checkCollisions(thisChild,x,y,angle,distance,NodeIds,ChildrenList);
      thisChild.setAngle(returnres.get(2));
      thisChild.setSaveX((int)Math.ceil(returnres.get(0)));
      thisChild.setSaveY((int)Math.ceil(returnres.get(1)));
      fill(Nodes.get(ChildrenList.get(0)).saveColor);
      strokeWeight(thisChild.strokeWeight);
      ellipse(thisChild.getSaveX(),thisChild.getSaveY(),thisChild.getSaveRadius()*2,
thisChild.getSaveRadius()*2);
      strokeWeight(1);
      fill(0);
      text(thisChild.getId(),thisChild.getSaveX(),thisChild.getSaveY()+5);
  }
```

Η επόμενη περίπτωση είναι όταν υπάρχει ζυγός αριθμός παιδιών όπου χωρίζουμε την `ChildrenList` στη μέση. Τα μισά παιδιά θα τοποθετηθούν στο δεξί μισό του τόξου, και τα υπόλοιπα στο αριστερό μισό. Ένας μετρητής `counter` θα μας βοηθήσει να καταλάβουμε πότε τοποθετούμε το πρώτο παιδί, ώστε να υποδιπλασιάσουμε το `anglestep` και να το βάλουμε πιο κοντά στο κέντρο του τόξου. Για τα παιδιά μετά το πρώτο η `angle` αυξάνεται κατά `anglestep`. Αφού πραγματοποιηθούν οι έλεγχοι για συγκρούσεις τοποθετούμε τα υπόλοιπα παιδιά ένα προς ένα.

```
else if(children%2==0) {
      angle=centerangle;
      int counter=0;
      for(int i=0;i<ChildrenList.size()/2;i++) {
            thisChild = Nodes.get(ChildrenList.get(i));
            counter++;
            if(counter==1) {
                  angle+=anglestep/2;
            } else {
                 anglestep +=anglestep;
            }
            if(thisChild.angleChanged && abs(thisChild.getAngle()-angle)<PI/2) {
                  angle=thisChild.getAngle();
            }
            x=(int)(saveXsize/2 + distance * cos(angle));
            y=(int)((ySize-40)/2 + distance * sin(angle));
            ArrayList<Float> returnres =
checkCollisions(thisChild,x,y,angle,distance,NodeIds,ChildrenList);
            thisChild.setAngle(returnres.get(2));
            thisChild.setSaveX((int)Math.ceil(returnres.get(0)));
            thisChild.setSaveY((int)Math.ceil(returnres.get(1)));
            fill(thisChild.saveColor);
            strokeWeight(thisChild.strokeWeight);
            ellipse(thisChild.getSaveX(),thisChild.getSaveY(),thisChild.getSaveRadiu
      s()*2,thisChild.getSaveRadius()*2);
            strokeWeight(1);
            fill(0);
            text(thisChild.getId(),thisChild.getSaveX(),thisChild.getSaveY()+5);
```

Το δεύτερο μισό της λίστας των παιδιών τοποθετείται με τον ίδιο τρόπο, με μόνη διαφορά ότι αυτή τη φορά μειώνουμε τη γωνία κατά `anglestep` αντί να την αυξάνουμε.

```
	angle=centerangle;
	counter=0;
	for(int k=ChildrenList.size()/2;k<ChildrenList.size();k++) {
		thisChild=Nodes.get(ChildrenList.get(k));
		counter++;
		if(counter==1) {
			angle-=anglestep/2;
		} else {
			angle-=anglestep;
		}
		if(thisChild.angleChanged && abs(thisChild.getAngle()-angle)<PI/2) {
			angle=thisChild.getAngle();
		}
		x=(int)(saveXsize/2 + distance * cos(angle));
		y=(int)((ySize-40)/2 + distance * sin(angle));
		ArrayList<Float> returnres =
checkCollisions(thisChild,x,y,angle,distance,NodeIds,ChildrenList);
		thisChild.setAngle(returnres.get(2));
		thisChild.setSaveX((int)Math.ceil(returnres.get(0)));
		thisChild.setSaveY((int)Math.ceil(returnres.get(1)));
		fill( Nodes.get(ChildrenList.get(k)).saveColor);
		strokeWeight(thisChild.strokeWeight);
		ellipse(thisChild.getSaveX(),thisChild.getSaveY(),thisChild.getSaveRadiu
	s()*2,thisChild.getSaveRadius()*2);
		strokeWeight(1);
		fill(0);
		text(thisChild.getId(),thisChild.getSaveX(),thisChild.getSaveY()+5);
  }
```

Τέλος εάν υπάρχει περιττός αριθμός παιδιών τοποθετούμε το πρώτο παιδί στην ίδια γωνία με τον πατέρα του και τα υπόλοιπα (ζυγός πλέον αριθμός) τα χωρίζουμε στην μέση και τα τοποθετούμε όπως τα παραπάνω, ο κώδικας είναι παρόμοιος για αυτόν τον λόγο θα παραληφθεί.

Αισίως έχουμε φτάσει στο τέλος της `CreateSaveTree`, το Save View δεν τελειώνει όμως εδώ. Έως τώρα έχουμε τοποθετήσει τους κόμβους, αλλά δεν έχουμε ζωγραφίσει τις ακμές που τους ενώνουν. Για να το πετύχουμε αυτό χρειαζόμαστε μία επανάληψη που να διατρέχει των πίνακα των κόμβων. Για κάθε κόμβο βρίσκουμε την γωνία του, τον πατέρα του και την γωνία του πατέρα του.

```
for(int d=1;d<numberNodes;d++) {
	if(Nodes.get(d).isConnected) {
		int parent = Nodes.get(d).getrouteToSink();
			float parentAngle = Nodes.get(parent).getAngle();
			float thisAngle = Nodes.get(d).getAngle();
```

Μετά πρέπει να υπολογίσουμε την απόσταση του κόμβου και του πατέρα του από το κέντρο του κύκλου:

```
float parentDist;
float thisDist;
if(flaag) {
	parentDist =(((ySize)/maxCost)/2*Nodes.get(parent).getcostToSink());
	thisDist = (((ySize)/maxCost)/2*Nodes.get(d).getcostToSink());
} else {
	parentDist =(((ySize-60)/maxCost)/2*Nodes.get(parent).getcostToSink());
	thisDist = (((ySize-60)/maxCost)/2*Nodes.get(d).getcostToSink());
}
```

Αφού ολοκληρώσουμε και αυτό το βήμα πρέπει να υπολογίσουμε το σημείο από το οποίο ξεκινάει η γραμμή και το σημείο στο οποίο τελειώνει. Το πρώτο σημείο βρίσκεται στην περιφέρεια του κόμβου του πατέρα (κάθε κόμβος είναι ένας κύκλος) και το δεύτερο στην περιφέρεια του κόμβου υπό εξέταση. Αν χρησιμοποιούσαμε τις συντεταγμένες του κόμβου που υπολόγισε η `CreateSaveTree` οι γραμμές θα ξεκινούσαν από το κέντρο του κύκλου και θα φαίνονταν άσχημα.

```
int lineparentX;
int lineparentY;
if(parent==0) {
	lineparentX = (int)(saveXsize/2 + (25/2) * cos(thisAngle));
	lineparentY =(int)((ySize-40)/2 + (25/2) * sin(thisAngle));
} else {
	lineparentX = (int)(saveXsize/2 + (parentDist+20/2) * cos(parentAngle));
	lineparentY = (int)((ySize-40)/2 + (parentDist+20/2) * sin(parentAngle));
}
int linethisX = (int)(saveXsize/2 + (thisDist-20/2) * cos(thisAngle));
int linethisY = (int)((ySize-40)/2 + (thisDist-20/2) * sin(thisAngle));
```

Εφόσον γνωρίζουμε τις συντεταγμένες, ζωγραφίζουμε την γραμμή και ολοκληρώνεται ο κώδικας του Save View:

```
line(linethisX,linethisY,lineparentX,lineparentY);
```

Έλεγχος συγκρούσεων

Θα ήταν αρκετά ενδιαφέρον σε αυτό το σημείο να περιγράψουμε πως γίνεται ο έλεγχος συγκρούσεων των κόμβων με τη μέθοδο

```
ArrayList<Float> checkCollisions(Node thisChild, int x, int y, float angle, float
distance, String[] NodeIds,ArrayList<Integer> ChildrenList) {
```

Η μέθοδος δέχεται ως παραμέτρους έναν κόμβο `thisChild`, τις συντεταγμένες του `x, y`, τη γωνία του `angle`, την απόσταση από το κέντρο `distance`, έναν πίνακα με τα id των κόμβων που βρίσκονται στον ίδιο κύκλο με αυτόν `NodeIds` και την λίστα των παιδιών του πατέρα του `ChildrenList,` γνωστά και ως αδέρφια του. Αφού γίνει ο έλεγχος και διαπιστωθεί ότι δεν υπάρχει άλλος κόμβος σε σύγκρουση, η μέθοδος επιστρέφει ένα Arraylist με τις νέες συντεταγμένες και τη γωνία του κόμβου.

Η μέθοδος έχει 2 σκέλη, πρώτα να γίνει έλεγχος αν ο δεδομένος κόμβος συγκρούεται με άλλους κόμβους του ίδιου κόστους και έπειτα να διαπιστωθεί εάν ο κόμβος συγκρούεται με τα αδέρφια του. Για αρχή χρειαζόμαστε μία επανάληψη που θα τρέχει για κάθε κόμβο στο ίδιο κόστος με τον αυτόν που θέλουμε να ελέγξουμε:

```
for(int c=0; c<NodeIds.length; c++) {
	Node checkNode = Nodes.get(Integer.parseInt(NodeIds[c]));
	int checkX = checkNode.getSaveX();
	int checkY = checkNode.getSaveY();
```

Μέσα στην επανάληψη τσεκάρουμε εάν ο κόμβος μας συγκρούεται με κάθε έναν από τους κόμβους ίδιου κόστους. Αυτό το πετυχαίνουμε συγκρίνοντας αν η απόσταση των κέντρων τους είναι μεγαλύτερη από την ακτίνα του κύκλου τους, με άλλα λόγια συγκρίνουμε εάν τέμνονται οι κύκλοι τους:

```
if( (pow((checkX-x),2) + pow((checkY-y),2)) <
pow((thisChild.getSaveRadius()+checkNode.getSaveRadius()),2) && checkNode!=thisChild)
{
```

Εάν όντως τέμνονται οι κύκλοι τους ξεκινάμε να αυξάνουμε την γωνία του κόμβου κατά *π/95* και υπολογίζουμε νέες συντεταγμένες με βάση τη γωνία αυτή. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να μην υπάρχει σύγκρουση ή μέχρι να φτάσουμε σε κάποιο όριο. Όσο διαρκεί αυτή η διαδικασία στον κόμβο έχει δηλωθεί πως η γωνία του έχει αλλάξει (`angleChanged=true`), αυτή η μεταβλητή είναι απαραίτητη για να γνωρίζει ο αλγόριθμος του Save View να μην υπολογίσει αυτός τη γωνία, αλλά να χρησιμοποιήσει την γωνία που βρήκε η μέθοδος `checkCollisions`.

```
tries=0;
while( (pow((checkX-x),2) + pow((checkY-y),2)) <=
pow((thisChild.getSaveRadius()+checkNode.getSaveRadius()),2) && tries<100) {
	angle+=PI/95;
	x=(int)(saveXsize/2 + distance * cos(angle));
	y=(int)((ySize-40)/2 + distance * sin(angle));
	tries++;
	thisChild.angleChanged=true;
}
```

Στο δεύτερο σκέλος ελέγχουμε εάν ο κόμβος συγκρούεται με κάποιο από τα αδέρφια του. Η διαδικασία αυτή μοιάζει περιττή, επειδή τα αδέρφια του περιέχονται και στον έλεγχο του προηγούμενου σκέλους, αλλά είναι απαραίτητη στην πράξη. Αν την παραλείψουμε θα υπάρχουν συγκρούσεις μεταξύ των αδερφιών. Ο κώδικας είναι παρόμοιος με τον προηγούμενο:

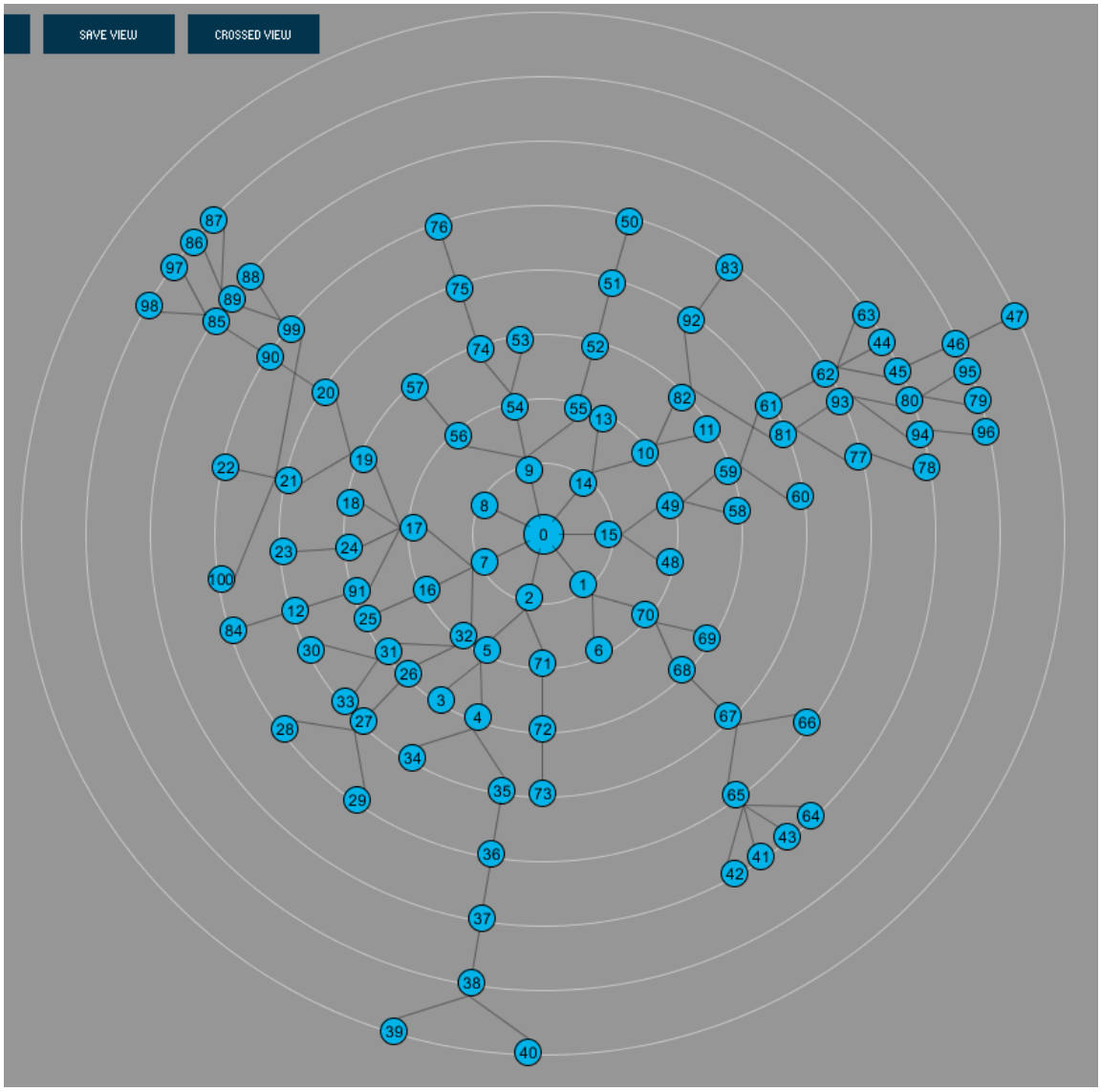
```
for(int k=0;k<ChildrenList.size();k++) {
	Node checkNode = Nodes.get(ChildrenList.get(k));
	int checkX = checkNode.getSaveX();
	int checkY = checkNode.getSaveY();
	if( (pow((checkX-x),2) + pow((checkY-y),2) <
pow((thisChild.getSaveRadius()+checkNode.getSaveRadius()),2)) &&
checkNode!=thisChild) {

		tries=0;
		while( (pow((checkX-x),2) + pow((checkY-y),2) <=
pow((thisChild.getSaveRadius()+checkNode.getSaveRadius()),2) ) && tries<50) {
		angle+=PI/55;
		x=(int)(saveXsize/2 + distance * cos(angle));
		y=(int)((ySize-40)/2 + distance * sin(angle));
		tries++;
```

```
            thisChild.angleChanged=true;
        }

}
```

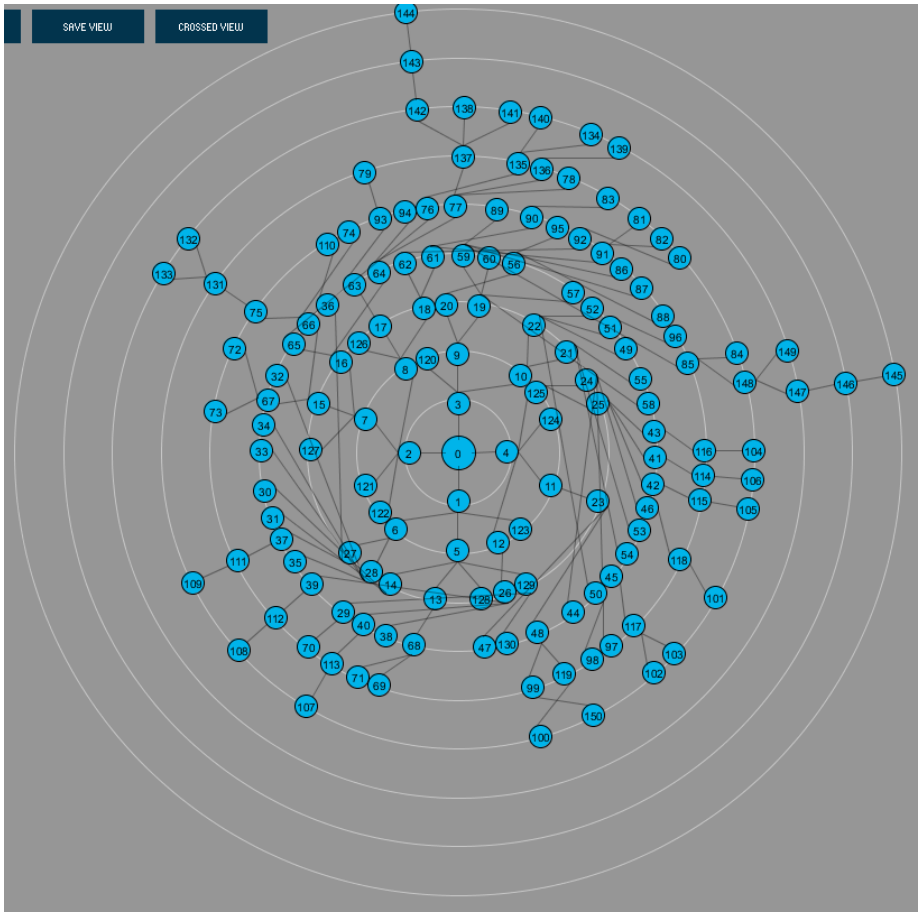
Τέλος επιστρέφουμε τις συντεταγμένες και τη γωνία που υπολογίστηκαν

```
    returnres.add(0,(float)x);
    returnres.add(1,(float)y);
    returnres.add(2,angle);
    return returnres;
```

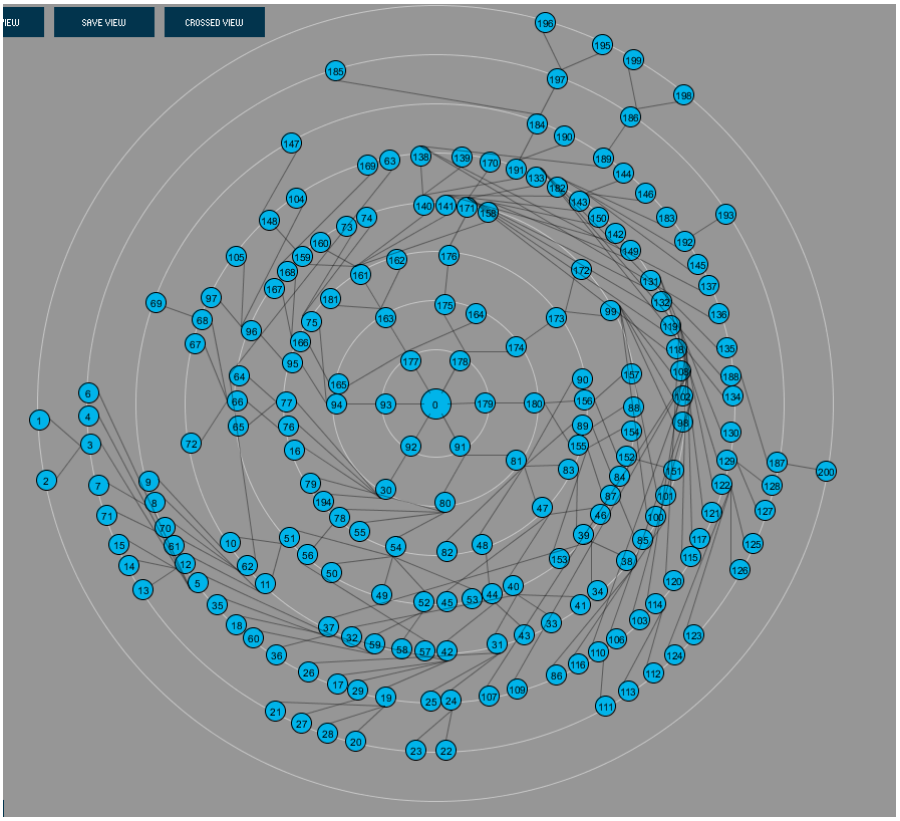
Το τελικό αποτέλεσμα φαίνεται στις εικόνες που ακολουθούν:



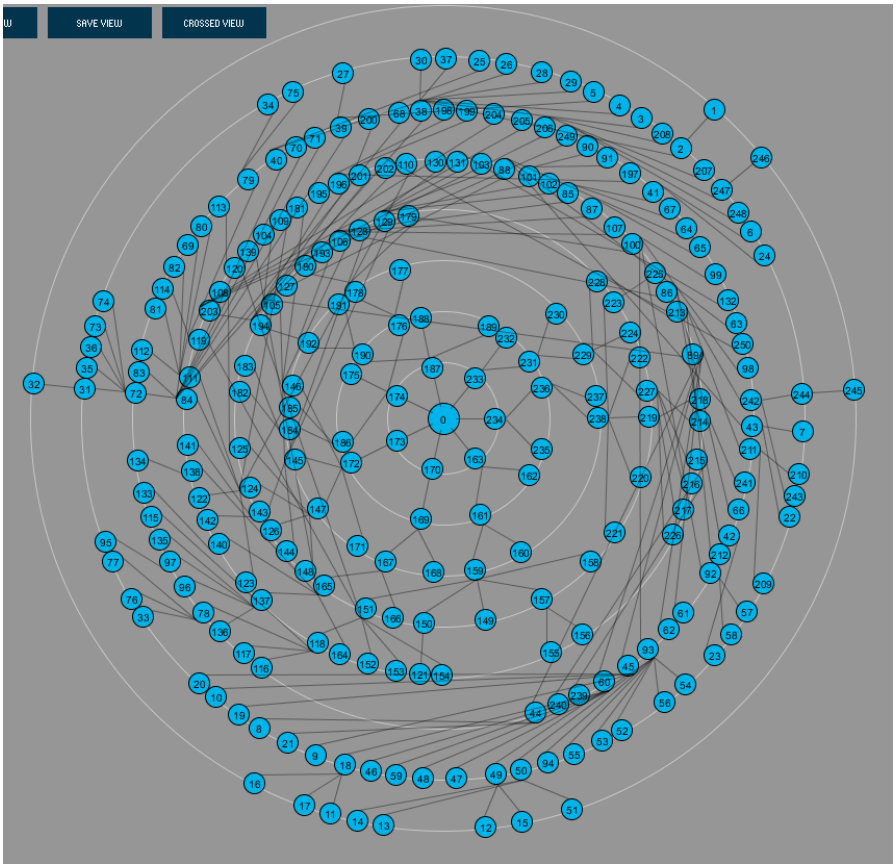
Εικόνα.21a: 100 κόμβοι



Εικόνα21b: 150 κόμβοι



Εικόνα21c: 200 κόμβοι



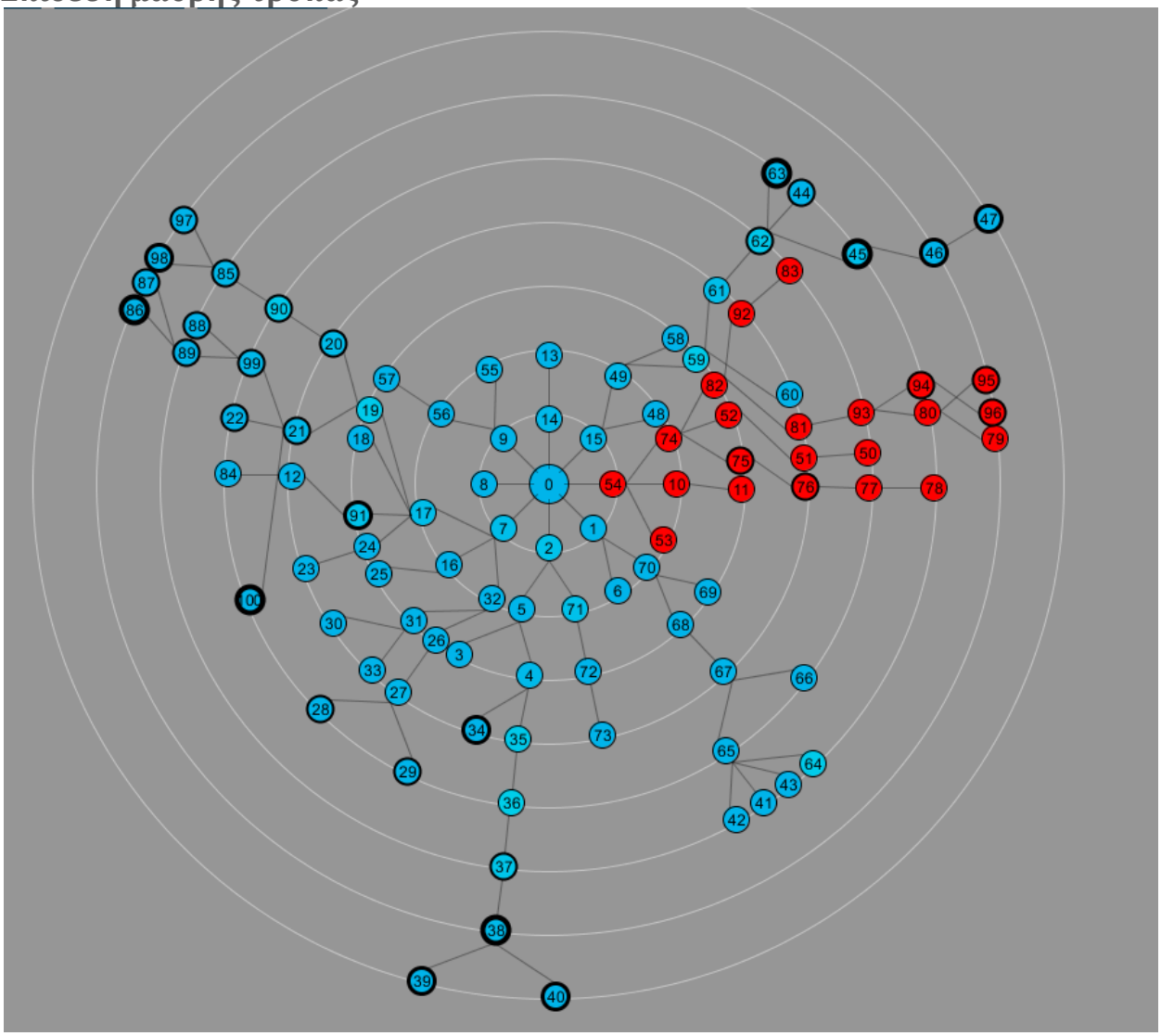
Εικόνα21d: 250 κόμβοι

Από τους 250 κόμβους και επάνω αντιμετωπίζουμε πρόβλημα αλληλοεπικαλύψεων επειδή δεν υπάρχει αρκετός χώρος για όλους. Η ανάλυση του παραθύρου της εφαρμογής είναι 1440x850. Ίσως σε μεγαλύτερη ανάλυση να μπορέσουμε να χωρέσουμε περισσότερους κόμβους.

Επιθέσεις

Στο κεφάλαιο 4 αναλύσαμε, περιγράψαμε και προτείναμε τρόπους αντιμετώπισης για τις σημαντικότερες επιθέσεις σε ένα ασύρματο δίκτυο αισθητήρων. Τις επιθέσεις αυτές ενσωματώσαμε στην εφαρμογή μας ώστε ο χρήστης να μπορεί να δει σε προσομοίωση πως η κάθε επίθεση επηρεάζει ένα ΑΔΑ. Θα ξεκινήσουμε τη εξηγούμε πως υλοποιήσαμε τις επιθέσεις με την μαύρη τρύπα.

Επίθεση μαύρης τρύπας



Εικόνα.22: Απεικόνιση MT στο Save View

Αρχικά έπρεπε να δημιουργήσουμε ένα GUI ώστε ο χρήστης να μπορεί να χειρίζεται τις επιθέσεις.

Εικόνα. 23: Black hole GUI

Για την black hole χρειαζόμαστε ένα πεδίο εισαγωγής κειμένου και 2 κουμπιά για την εκκίνηση και τον τερματισμό επίθεσης. Στο πεδίο εισαγωγής κειμένου ο χρήστης πληκτρολογεί το id του κόμβου που θα ενεργήσει ως μαύρη τρύπα. Μόλις πατηθεί το κουμπί “Start BH” εκτελείται ο controller του κουμπιού:

```
if(theEvent.isFrom("StartBH")) {
      int BlackHoleid=-1;
      String tempBlackHole="";
```

```
        if (cp5.get(Textfield.class,"BlackHole").getText().length()>0) {
              tempBlackHole = cp5.get(Textfield.class,"BlackHole").getText();
              BlackHoleid = Integer.parseInt(tempBlackHole);
              BHstoppedFlag=true;
              timeflag=true;
              times=millis();
    }

    if(!BHNodes.contains(Nodes.get(BlackHoleid))) {
       BHNodes.add(Nodes.get(BlackHoleid));
        dNow = new Date( );
        Messages.put(ft.format(dNow),"Node "+tempBlackHole+" is acting as a
blackhole");
      BHFlag=true;
    }
}
```

Αυτό που κάνει ο controller είναι να διαβάζει το id που έδωσε ο χρήστης και να βάζει τον αντίστοιχο κόμβο στο ArrayList `BHNodes`, εφόσον δεν υπάρχει ήδη. Επίσης θέτει τιμές στα αντίστοιχα flags ώστε να ξεκινήσει η επίθεση. Τέλος ενημερώνει τον χρήστη με μήνυμα για το ποιος κόμβος είναι ΜΤ. Μόλις το `BHFlag` γίνει `true` καλείται η μέθοδος `void BlackHoleInit()` που περιέχει την λογική της επίθεσης. Σε μία επανάληψη που τρέχει για κάθε κόμβο μέσα στο ArrayList `BHNodes`, γίνονται οι εξής αλλαγές:

1. Το κόστος του κόμβου που ενεργεί ως μαύρη τρύπα γίνεται 1. Με αυτόν τον τρόπο ο κόμβος καταφέρνει να δελεάσει τους γείτονες του για να δρομολογήσουν μέσω αυτού.
2. Το επόμενο hop του κόμβου γίνεται ο κόμβος 0.
3. Το χρώμα του αλλάζει σε κόκκινο για εύκολη αναγνώριση.

```
for(int i=0;i<BHNodes.size();i++){
      Node BHNode = BHNodes.get(i);
      BHNode.setcostToSink(1); //set cost to 1
      BHNode.setrouteToSink(0); //set the route to 0
      BHNode.saveColor=color(255,0,0);
```

Στην συνέχεια η μέθοδος βρίσκει τους κόμβους που δρομολογούν μέσω της μαύρης τρύπας και τους σημειώνει κατάλληλα:

```
      for(int k=1;k<numberNodes;k++) {
            ArrayList<Integer> HopList = new ArrayList<Integer>();
            HopList=Nodes.get(k).getFullRoute();
            if(HopList.contains(BHNode.getId())){
                  BHAffectedNodes.add(Nodes.get(k)); //keep track of the nodes
routing through the BH
                  Nodes.get(k).type="BHAff";
                  Nodes.get(k).saveColor=color(255,0,0);
                  Nodes.get(k).angleChanged=false;
            }
      }
```

Έχοντας επιτυχώς δημιουργήσει μία μαύρη τρύπα πρέπει να αρχίσουμε να διαχειριζόμαστε τα πακέτα που φτάνουν σε αυτή. Έτσι λοιπόν στην μέθοδο `ProccessQueue` προσθέτουμε έναν έλεγχο. Κάθε πακέτο που φτάνει στην μαύρη τρύπα

το καταγράφουμε, και στη συνέχεια το διαγράφουμε χωρίς να το προωθήσουμε. Επίσης οι μαύρες τρύπες δεν παράγουν πακέτα με δεδομένα, στέλνουν μόνο πακέτα δρομολόγησης.

```
if(BHNodes.contains(ThisPacket.getCurrent())) { //if this packet reached the
BlackHole
	PacketsBHed+=1; //log it
	ThisNode.queue.remove(ThisPacket); //delete it
	ThisPacket=null;
	continue; //go to the next packet
}
```

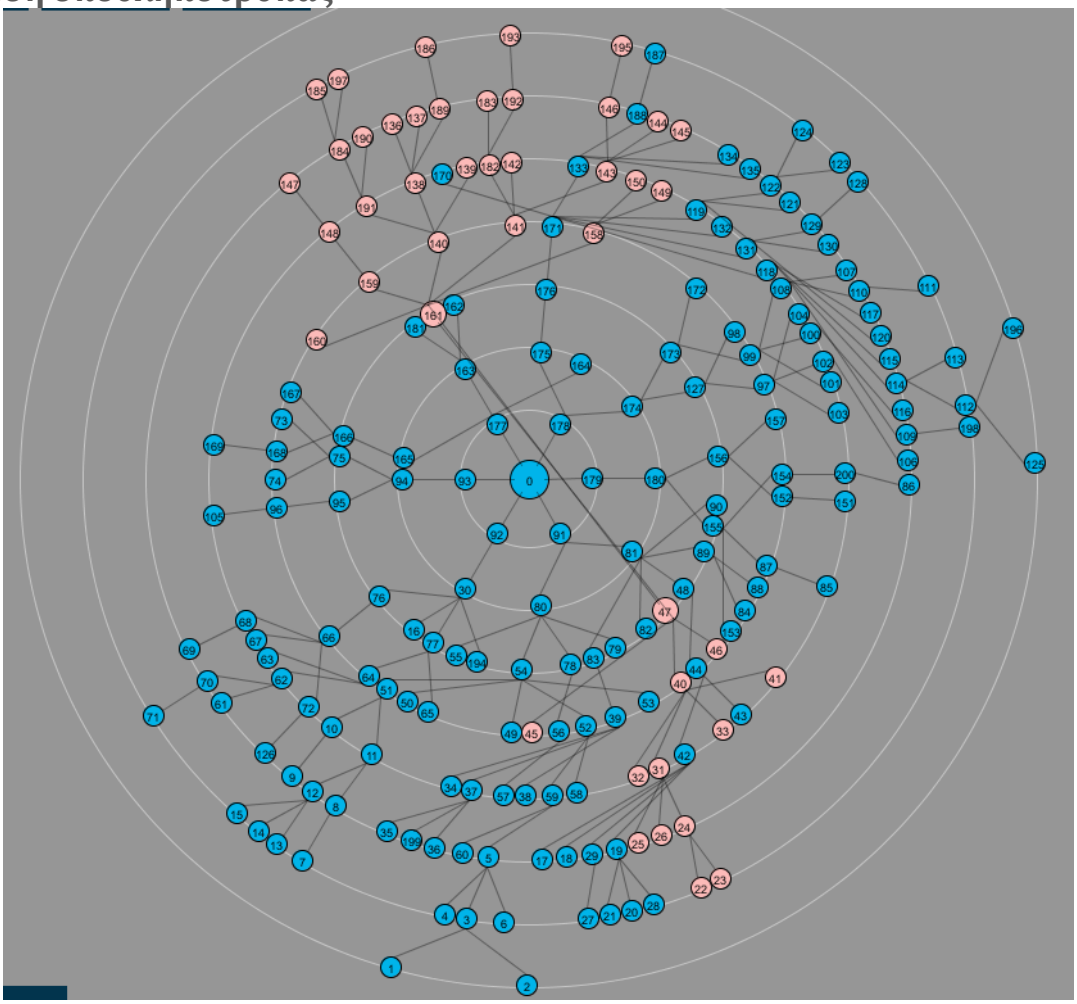
Τέλος όταν θέλουμε να σταματήσουμε την επίθεση πατάμε το κουμπί Stop BH, το οποίο καλεί τον controller που θα σταματήσει την επίθεση:

```
  public void StopBH() {
	BHFlag=false;
	BlackHoleStop();
	dNow = new Date( );
	Messages.put(ft.format(dNow),"All blackholes resolved.");

   }
```

Η μέθοδος `BlackHoleStop()` επαναφέρει τους κόμβους που ήταν μαύρες τρύπες ή επηρεάστηκαν από αυτές στην αρχική τους κατάσταση. Αυτό το πετυχαίνουμε απλά αλλάζοντας το χρώμα στο προκαθορισμένο για υγιείς κόμβους και θέτοντας το κόστος του κόμβου μαύρη τρύπα σε 9999, ο αλγόριθμος δρομολόγησης θα κάνει τα υπόλοιπα. Για τους κόμβους που επηρεάστηκαν από κάποια μαύρη τρύπα, αλλάζουμε επίσης το χρώμα και επαναφέρουμε το type τους σε Normal. Τέλος καθαρίζουμε τα Arraylists `BHNodes` και `BHAffectedNodes`.

```
void BlackHoleStop(){
	for(int i=0;i<BHNodes.size();i++){
		Node BHNode = BHNodes.get(i);
		BHNode.setcostToSink(9999);
		BHNode.saveColor=color(0,180,234);
	}
	BHNodes.clear();
	for(int k=0;k<BHAffectedNodes.size();k++){
		BHAffectedNodes.get(k).type="Normal";
		BHAffectedNodes.get(k).saveColor=color(0,180,234);
	}
	BHAffectedNodes.clear();
}
```

Επίθεση σκουληκότρυπας



Εικόνα.24: Απεικόνιση Wormhole στο Save View

Η επίθεση σκουληκότρυπας που δημιουργήσαμε χρησιμοποιεί ζεύγη εσωτερικών κόμβων του δικτύου. Δύο κόμβοι συνεργάζονται για να δημιουργήσουν μία εικονική ζεύξη μεταξύ τους και να ξεγελάσουν άλλους κόμβους στο να δρομολογήσουν μέσω αυτής. Όπως και στην Black hole, έτσι και στην Wormhole πρέπει να δημιουργήσουμε ένα GUI και έναν controller για αυτό.

Εικόνα.25: Wormhole GUI

Ο controller λαμβάνει τα ids των κόμβων που εισήγαγε ο χρήστης και τα τοποθετεί σε 2 ArrayLists αφού πρώτα βεβαιωθεί ότι δεν υπάρχουν ήδη. Στο `WHNodes1` μπαίνουν τα id από το πρώτο input, και στο `WHNodes2` από το δεύτερο. Οι σκουληκότρυπες δουλεύουν σε ζευγάρια, οπότε ένας κόμβος που βρίσκεται στην πρώτη λίστα δεν μπορεί να είναι στην δεύτερη και το αντίστροφο. Επίσης τα ζευγάρια καθορίζονται από τα indexes των ArrayLists. Στη συνέχεια ενεργοποιείται η `WHFlag` και καλείται η μέθοδος `WHStart()`. Τέλος ο χρήστης ενημερώνεται με σχετικό μήνυμα για την ύπαρξη σκουληκότρυπας.

```
  if(theEvent.isFrom("StartWH")) {
          int WH1=-1;
          int WH2=-1;
          String tempWH1="";
          String tempWH2="";
          if (cp5.get(Textfield.class,"WH1").getText().length()>0) {
              tempWH1 = cp5.get(Textfield.class,"WH1").getText();
              WH1 = Integer.parseInt(tempWH1);
            }
          if (cp5.get(Textfield.class,"WH2").getText().length()>0) {
            tempWH2 = cp5.get(Textfield.class,"WH2").getText();
            WH2 = Integer.parseInt(tempWH2);
          }
          if(!WHNodes1.contains(Nodes.get(WH1)) &&
!WHNodes2.contains(Nodes.get(WH1)) &&!WHNodes2.contains(Nodes.get(WH2)) &&
!WHNodes1.contains(Nodes.get(WH2))) {
            WHNodes1.add(Nodes.get(WH1));
            WHNodes2.add(Nodes.get(WH2));
              dNow = new Date( );
              Messages.put(ft.format(dNow),"Wormhole detected! ["+WH1+","+WH2+"]");
            WHFlag=true;
            WHStart();
          }
        }
```

Για να ξεκινήσει η επίθεση η `WHStart` πρέπει να εκτελέσει ορισμένες ενέργειες. Αρχικά πρέπει να βεβαιωθεί ότι τα 2 ArrayLists έχουν το ίδιο μέγεθος και περιέχουν τουλάχιστον έναν κόμβο το καθένα. Έπειτα ξεκινάει μία επανάληψη για όλους τους κόμβους που βρίσκονται στα 2 αυτά ArrayLists. Κάθε κόμβος του `WHNodes1` θέτει ως γείτονα του τον αντίστοιχο στο `WHNodes2` και το αντίστροφο, στη συνέχεια δηλώνουμε πως ο ένας κόμβος δρομολογεί μέσω του άλλου, με άλλα λόγια ο πρώτος είναι πατέρας του δεύτερου και ο δεύτερος πατέρας του πρώτου. Τέλος ορίζουμε το χρώμα και τον τύπο για κάθε έναν από τους κόμβους.

```
void WHStart(){
if((WHNodes1.size()>0 && WHNodes2.size()>0) && WHNodes1.size()==WHNodes2.size()) {
      for(int i=0;i<WHNodes1.size();i++){
            Node WH1 = WHNodes1.get(i);
            Node WH2 = WHNodes2.get(i);
            WH1.saveColor=color(252,186,184);
            WH2.saveColor=color(252,186,184);
            //Set the 2 nodes as neighbors
            ArrayList<Integer> NewNeighborList1 = WH1.GetNeighborList();
            if(!NewNeighborList1.contains(WH2.getId())){
                  NewNeighborList1.add(WH2.getId());
                  WH1.UpdateNeighborList(NewNeighborList1);
            }
            ArrayList<Integer> NewNeighborList2 = WH2.GetNeighborList();
            if(!NewNeighborList2.contains(WH1.getId())){
                  NewNeighborList2.add(WH1.getId());
                  WH2.UpdateNeighborList(NewNeighborList2);
            }
            //Make them route through each other
            WH1.setrouteToSink(WH2.getId());
            WH2.setrouteToSink(WH1.getId());
            WH1.setType("WH1");
            WH2.setType("WH2");
```

Κάθε φορά που γίνεται όμως δρομολόγηση οι κόμβοι ανιχνεύουν ξανά τους γείτονες τους. Οι κόμβοι όμως που σχηματίζουν την σκουληκότρυπα δεν είναι στην πραγματικότητα γείτονες. Για αυτόν τον λόγο θα πρέπει να τροποποιήσουμε την `DiscoverNeighbors()` ώστε να λαμβάνει υπόψη της τις wormholes:

```
if(WHNodes1.contains(Nodes.get(i))) {
	int index = WHNodes1.indexOf(Nodes.get(i));
	NeighborList.add(WHNodes2.get(index).getId());
}
if(WHNodes2.contains(Nodes.get(i))) {
	int index = WHNodes2.indexOf(Nodes.get(i));
	NeighborList.add(WHNodes1.get(index).getId());
}
```

Το επόμενο βήμα είναι να διαχειριστούμε τα πακέτα που φτάνουν στα άκρα της σκουληκότρυπας. Η διαχείριση τους περιλαμβάνει την καταγραφή των πακέτων και την διαγραφή τους στη συνέχεια:

```
if(WHNodes1.contains(ThisPacket.getCurrent()) ||
WHNodes2.contains(ThisPacket.getCurrent())) { //if this packet reached the either end
of the WormHole
	PacketsWHed+=1; //log it
	ThisNode.queue.remove(ThisPacket); //delete it
	ThisPacket=null;
	continue; //go to the next packet
}
```

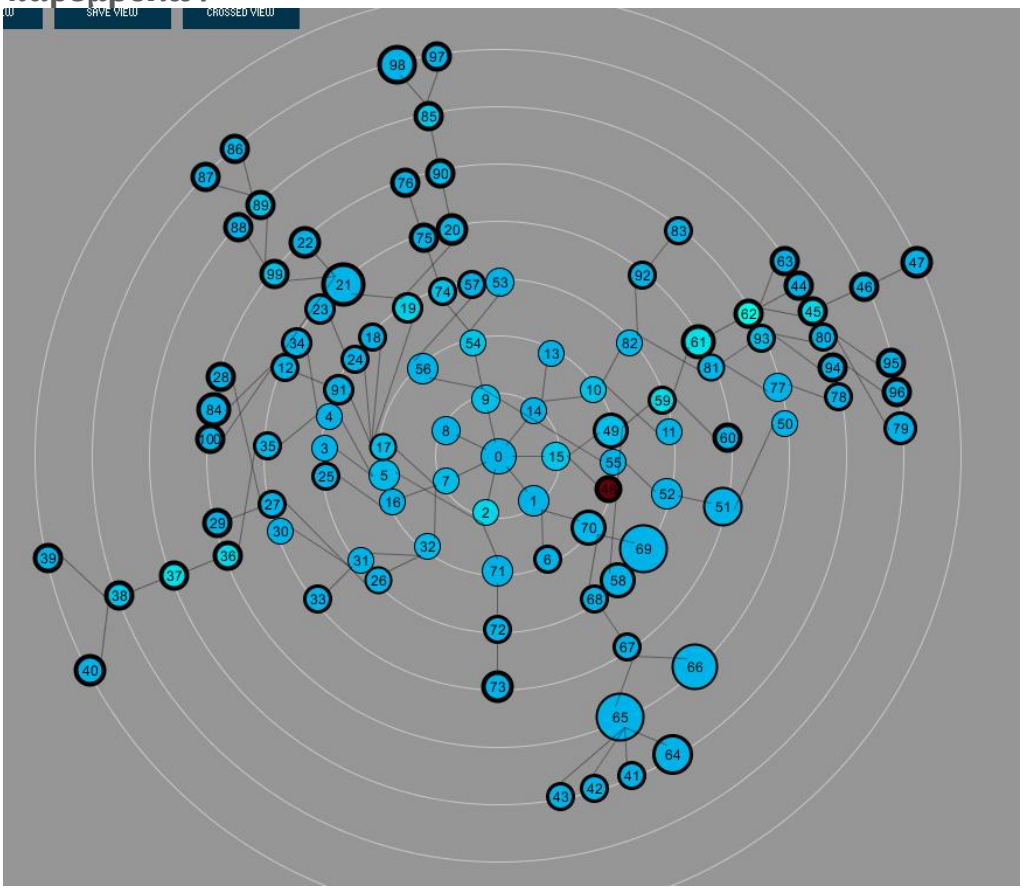
Τέλος το SaveView αρχικά δημιουργήθηκε χωρίς να ληφθεί υπόψη ότι κάποιοι κόμβοι θα είναι ταυτόχρονα και πατεράδες και παιδιά και η ύπαρξη σκουληκότρυπας προκαλούσε πρόβλημα στην οπτικοποιήση. Το πρόβλημα το ξεπεράσαμε αναγκάζοντας τους κόμβους που είναι άκρα σκουληκότρυπας να μένουν στην ίδια θέση που βρίσκονταν πριν ξεκινήσει η επίθεση:

```
if(WHNodes1.contains(thisChild) || WHNodes2.contains(thisChild)) {
	fill( thisChild.saveColor);
	strokeWeight(thisChild.strokeWeight);
	ellipse(thisChild.getSaveX(),thisChild.getSaveY(),20,20);
	strokeWeight(1);
	fill(0);
	text(thisChild.getId(),thisChild.getSaveX(),thisChild.getSaveY()+5);
	continue;
}
```

Για να τερματιστεί η επίθεση σκουληκότρυπας ο χρήστης αρκεί να πατήσει το κουμπί STOP WH, ο controller του οποίου θα καλέσει την `WormHoleStop()` και θα θέσει την τιμή της `WHFlag` σε **`false`**. Η `WormHoleStop()` επαναφέρει την λίστα γειτόνων των κόμβων που συμμετείχαν στην επίθεση στην κανονική της κατάσταση, θέτει ως επόμενο hop τον κόμβο 0, αδειάζει τα ArrayLists και επαναφέρει τα χρώματα στα φυσιολογικά. Ο αλγόριθμος δρομολόγησης θα τοποθετήσει τους κόμβους στις σωστές τους θέσεις.

```
void WormHoleStop() {
	for(int i=0;i<WHNodes1.size();i++){
		Node WH1 = WHNodes1.get(i);
		Node WH2 = WHNodes2.get(i);
		WH1.setType("Normal");
		WH2.setType("Normal");
		WH1.saveColor=color(0,180,200);
		WH2.saveColor=color(0,180,200);
		ArrayList<Integer> NewNeighborList1 = WH1.GetNeighborList();
		int index = NewNeighborList1.indexOf(WH2.getId());
		if(index!=-1){
			NewNeighborList1.remove(index);
			WH1.UpdateNeighborList(NewNeighborList1);
		}
		ArrayList<Integer> NewNeighborList2 = WH2.GetNeighborList();
		index = NewNeighborList2.indexOf(WH1.getId());
		if(index!=-1){
			NewNeighborList2.remove(index);
			WH2.UpdateNeighborList(NewNeighborList2);
		}
		WH1.setrouteToSink(0);
		WH2.setrouteToSink(0);

	}
	WHNodes1.clear();
	WHNodes2.clear();
}
```

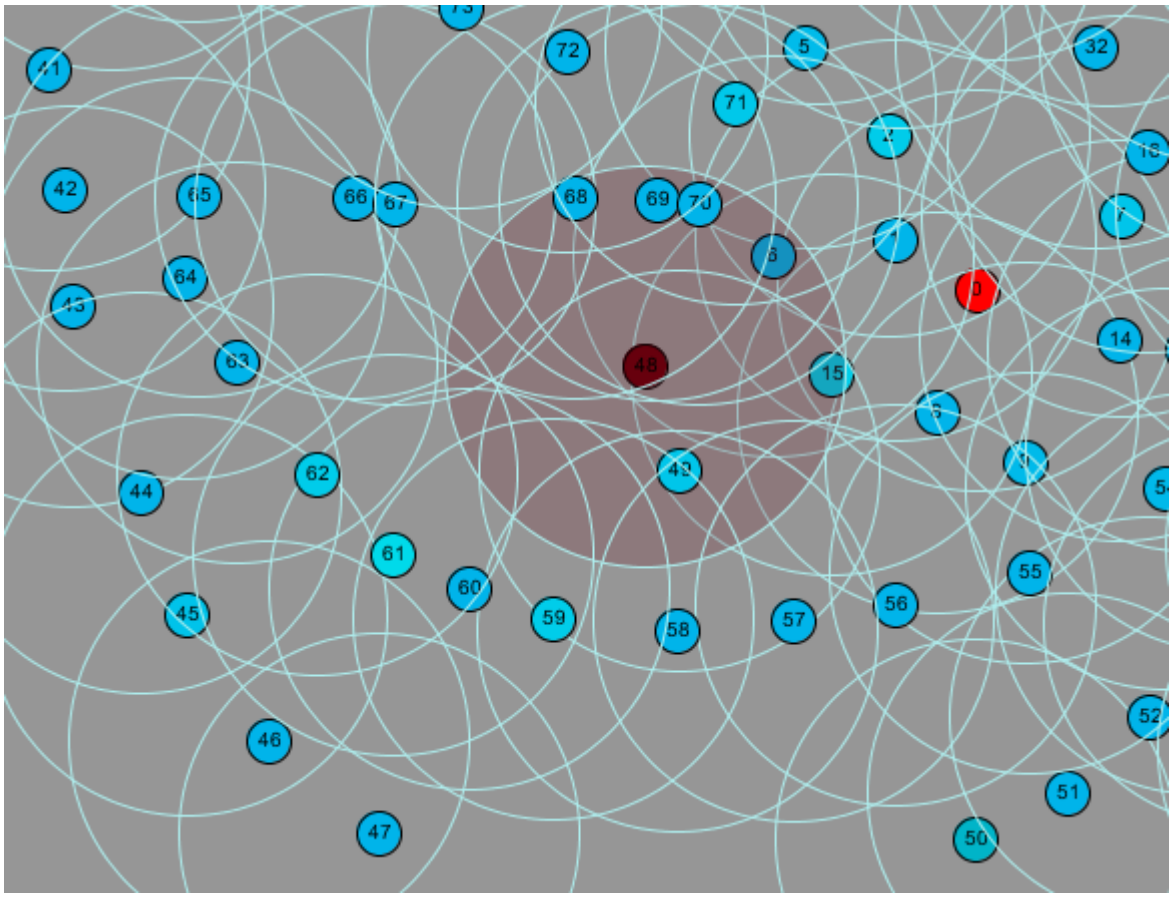
Επίθεση παρεμβολών



Εικόνα.26: Απεικόνιση Jammer στο Save View

Έχουμε υλοποιήσει έναν deceptive jammer στην εφαρμογή μας. Ο jammer είναι ένας επιλεγμένος κόμβος του δικτύου ο οποίος στέλνει συνεχώς πακέτα στους γείτονες του με σκοπό να γεμίσει την ουρά τους με άχρηστα πακέτα. Όταν η ουρά ενός κόμβου είναι γεμάτη αναγκάζονται να απορριφθούν τα κανονικά πακέτα που στέλνουν οι υπόλοιποι κόμβοι. Η επίδραση αυτής της επίθεσης γίνεται αντιληπτή στο Save View παρατηρώντας το πάχος της περιμέτρου των κόμβων, αλλά και την αύξηση της ακτίνας τους. Η παχιά περίμετρος δηλώνει αυξημένο latency στον κόμβο και η ακτίνα αυξάνεται όταν ο sink node έχει ώρα να λάβει πακέτα από αυτούς. Επομένως όσο μεγαλύτερη είναι η ακτίνα για τόσο περισσότερο χρόνο ο κόμβος θεωρείται ότι δεν αποκρίνεται.

Το Normal View μας δείχνη την εμβέλεια του Jammer και τους κόμβους που επηρεάζονται από αυτόν.



Εικόνα.27: Απεικόνιση Jammer στο Normal View

Όπως και σε όλες τις προηγούμενες επιθέσεις ξεκινήσαμε με την δημιουργία του GUI. Ένα πεδίο εισαγωγής κειμένου και δύο κουμπιά

Εικόνα.28: Jammer GUI

Ο controller τοποθετεί τον κόμβο που επιλέξαμε στην λίστα με τους `Jammers` και ενεργοποιεί την σημαία `JamFlag`. Τέλος ενημερώνει τον χρήστη.

```
  if(theEvent.isFrom("StartJam")) {
          if(millis()-start_time<10000){return;}
          resetTries();
          int JammerId=-1;
          String tempJammer="";
              if (cp5.get(Textfield.class,"Jammer").getText().length()>0) {
              tempJammer = cp5.get(Textfield.class,"Jammer").getText();
              JammerId = Integer.parseInt(tempJammer);

            }
            if(!Jammers.contains(Nodes.get(JammerId))) {
            Jammers.add(Nodes.get(JammerId));
              dNow = new Date( );
              Messages.put(ft.format(dNow),"Node "+tempJammer+" is jamming. I'm
jamming too!");
            JamFlag=true;
          }
        }
```

Η σημαία πυροδοτεί την μέθοδο `JammingInit()`. Η μέθοδος αυτή αλλάζει τον τύπο του κόμβου και το χρώμα του.

```
void JammingInit(){
      for(int i=0;i<Jammers.size();i++) {
            Node Jammer = Jammers.get(i);
            Jammer.setType("Jammer");
            Jammer.saveColor=color(103,0,13);
  }
```

Η πραγματική δουλειά που κάνει το jammer καταστροφικό για τους γειτονικούς του κόμβους γίνεται στην μέθοδο παραγωγής κίνησης. Σε αντίθεση με τους κανονικούς κόμβους, όσοι είναι jammers έχουν μικρότερο κύκλο μεταδόσεων (100ms) και δεν μεταδίδουν τυχαία σε κάθε κύκλο. Επίσης οι jammers παράγουν πολύ μεγάλο αριθμό πακέτων και τον παράγουν συνέχεια:

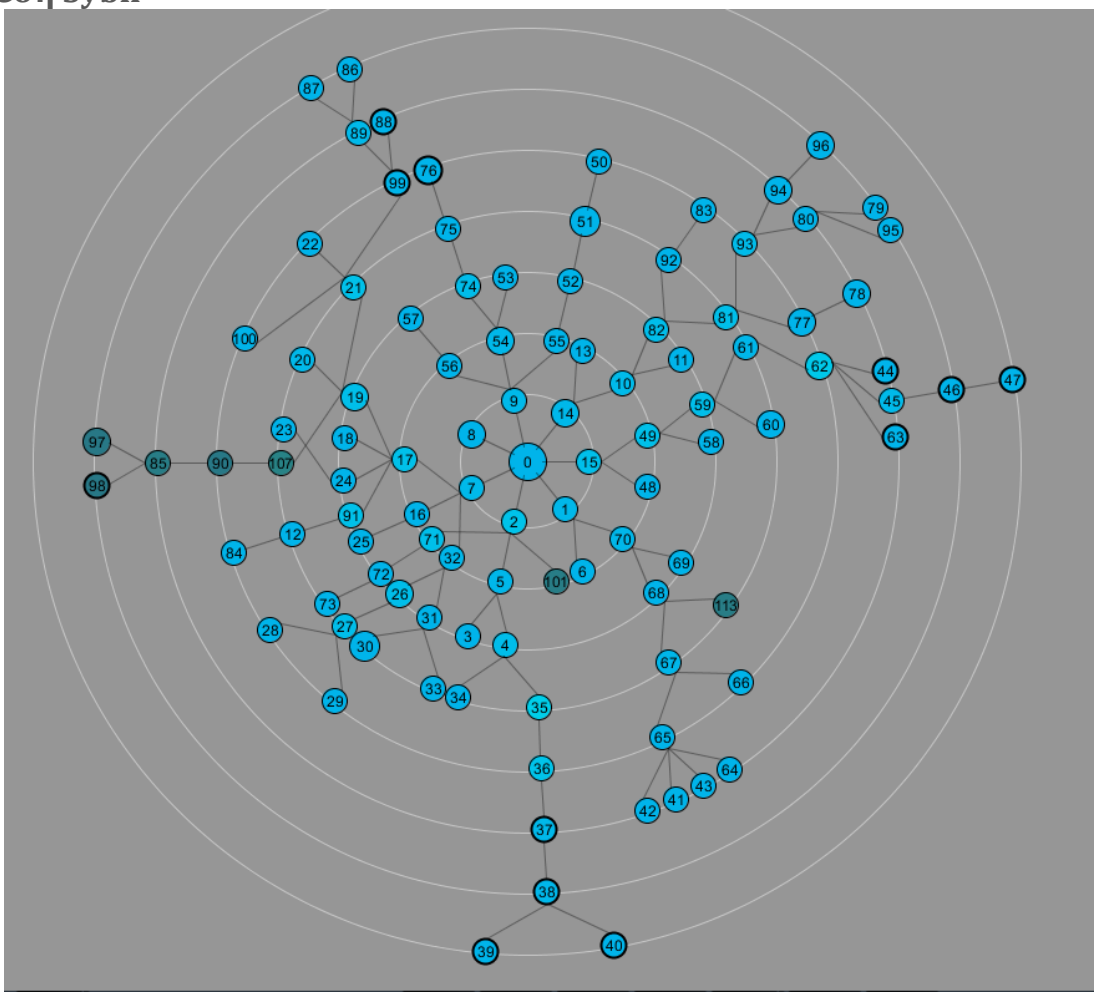
```
if(millis()-packet_timer1>=100) {
      if(CreatePacketsFlag) {
            for(int t=0;t<Jammers.size();t++) {
                Node Jammer = Jammers.get(t);
                int packetNumber = 2000;
                ArrayList<Integer> JammerNeighborList = new ArrayList<Integer>();
                JammerNeighborList = Jammer.GetNeighborList();
                for(int j=0;j<JammerNeighborList.size();j++){
                   Node Neighbor = Nodes.get(JammerNeighborList.get(j));
                   if(Neighbor.queue.size()<MaxNodeQueueSize){
                         for(int w=packetCounter;w<packetCounter+packetNumber;w++)
{
                               SimulationTimer=millis()-NowTimer;
                               Packet jPacket = new
Packet(w,Jammer,Neighbor,Neighbor,MTU,SimulationTimer,false,null);
                               Neighbor.queue.add(jPacket);
                               jPacket=null;
                         }
                   }
```

```
                packetCounter+=packetNumber;
                int curPackets = Jammer.GetpacketsSent();
                Jammer.SetpacketsSent(curPackets+packetNumber);
            }


        }
        }
        packet_timer1=millis();
    }
```

Για να τερματίσουμε την επίθεση Jammer καλούμε μέσω του αντίστοιχου controller την `JammingStop()` η οποία επαναφέρει τους κόμβους στην κανονική τους κατάσταση:

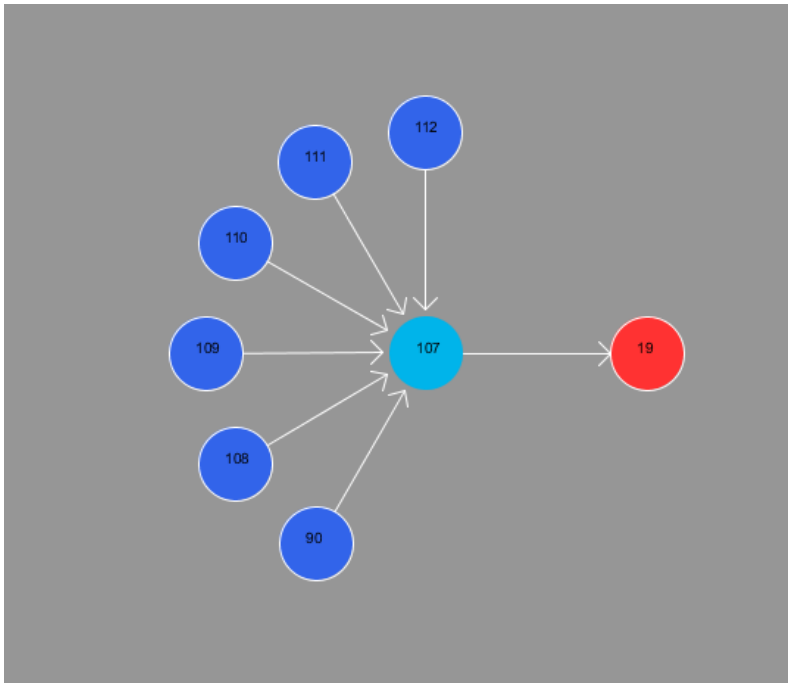
```
void JammingStop(){
      for(int i=0;i<Jammers.size();i++) {
            Node Jammer = Jammers.get(i);
            Jammer.setType("Normal");
            Jammers.remove(Jammer);
            Jammer.saveColor=color(0,180,200);
      }
}
```

Επίθεση Sybil



Εικόνα.29: Απεικόνιση Sybil στο Save View

Στην υλοποίηση της επίθεσης Sybil υπάρχει έμμεση επικοινωνία μεταξύ των κόμβων. Ο κακόβουλος κόμβος δημιουργεί έναν αριθμό από Sybil Nodes, ή Sybil Ghosts όπως τα ονομάσαμε, και αναφέρει πως έχει μονοπάτι προς αυτά. Οι υγιείς κόμβοι του δικτύου δεν επικοινωνούν απευθείας με τα Sybil Ghosts. Ο κακόβουλος κόμβος μπορεί να δημιουργήσει μία πλαστή ταυτότητα μέσα στο δίκτυο ή να κλέψει το id κάποιου υπάρχοντος κόμβου. Οι Sybil Ghosts φαίνεται να υπάρχουν ταυτόχρονα μέσα στο δίκτυο.



Εικόνα.30: Τα Sybil Ghosts στο Node View

Στην παραπάνω εικόνα ο 107 είναι ο κακόβουλος κόμβος και οι 108-112 οι Sybil Nodes που δημιούργησε. Ο κόμβος 90 είναι υγιής κόμβος που δρομολογεί μέσω του 107. Το GUI είναι στο ίδιο στυλ με των υπολοίπων επιθέσεων. Αν γράψουμε το id κάποιου υγιούς κόμβου αυτός θα μετατραπεί στον κόμβο που θα πραγματοποιήσει την επίθεση Sybil. Αν το id δεν υπάρχει μέσα στο δίκτυο θα δημιουργηθεί ένας νέος κόμβος σε τυχαία θέση. Στο πεδίο “#ADDITIONAL NODES” εισάγουμε τον αριθμό των Sybil Ghosts που θα δημιουργηθούν.

Εικόνα.31: Sybil GUI

Οι παραπάνω εργασίες πραγματοποιούνται στον controller μόλις πατηθεί το «START SYBIL”.

```
if(theEvent.isFrom("StartSybil")) {
  int SybilId=-1;
  String tempSybil="";
  String tempNumber="";
  if (cp5.get(Textfield.class,"SybilNode").getText().length()>0) {
      tempSybil = cp5.get(Textfield.class,"SybilNode").getText();
      SybilId = Integer.parseInt(tempSybil);

  }
  if (cp5.get(Textfield.class,"SybilGhosts").getText().length()>0) {
    tempNumber = cp5.get(Textfield.class,"SybilGhosts").getText();
    NumberSybils=Integer.parseInt(tempNumber);
  }
     Random rand = new Random();
     int maxx=1000;
     int maxy=390;
     int minx=220;
     int miny= 230;

     int randomX = rand.nextInt((maxx - minx) + 1) + minx;
     int randomY = rand.nextInt((maxy - miny) + 1) + miny;
     int SybilForMsg;
    if(SybilId>=numberNodes) {

      Node SybilNode = new Node(numberNodes,randomX,randomY,range,radius);
      SybilForMsg = numberNodes;
      numberNodes++;
      SybilNode.type="SybilNode";
      SybilNode.saveColor=color(40,120,134);
      Nodes.add(SybilNode);
      SybilNodes.add(SybilNode);
    } else {
     SybilNodes.add(Nodes.get(SybilId));
     SybilForMsg = SybilId;
     Nodes.get(SybilId).type="SybilNode";
     Nodes.get(SybilId).saveColor=color(40,120,134);
     randomX = Nodes.get(SybilId).getXpos();
     randomY = Nodes.get(SybilId).getYpos();

    }
```

Στη συνέχεια ο controller δημιουργεί τα Sybil Ghosts και τα θέτει να δρομολογούν μέσω του κόμβου που επιλέξαμε να πραγματοποιήσει την επίθεση. Στο τέλος ο χρήστης ενημερώνεται για την επίθεση με μήνυμα και καλείται η `SybilInit()`.

```
      String aliases="";
      for(int j=1;j<=NumberSybils;j++){
         Node SybilGhost = new Node(numberNodes,randomX,randomY,range,radius);
         aliases+=numberNodes+",";
         numberNodes++;
         SybilGhost.type="SybilGhost";
         SybilGhost.saveColor=color(40,120,134);
         SybilGhost.SybilParent = Nodes.get(SybilForMsg);
         SybilGhost.setrouteToSink(SybilForMsg);
         Nodes.add(SybilGhost);
         SybilGhosts.add(SybilGhost);

      }
      dNow = new Date( );
```

```
        Messages.put(ft.format(dNow),"Node "+SybilForMsg+" is acting as a Sybil node.
Possible aliases "+aliases);
        routing(false);
        SybilInit();
        SybilFlag=true;

    }
```

Η `SybilInit` καταγράφει τους κόμβους που δρομολογούν είτε μέσω του Sybil Node ή των Sybil Ghosts και τους αλλάζει χρώμα

```
void SybilInit(){
      for(int i=0;i<SybilNodes.size();i++){
            Node SybilNode = SybilNodes.get(i);
            SybilNode.saveColor=color(40,120,134);
            SybilNode.angleChanged=false;
            for(int k=1;k<numberNodes;k++) {
                  ArrayList<Integer> HopList = new ArrayList<Integer>();
                   HopList=Nodes.get(k).getFullRoute();
                   if(HopList.contains(SybilNode.getId())){
                         SybilAffectedNodes.add(Nodes.get(k));
                         Nodes.get(k).angleChanged=false;
                         Nodes.get(k).type="SybilAff";
                         Nodes.get(k).saveColor=color(40,120,134);
                   }
              }
      }
      for(int i=0;i<SybilGhosts.size();i++){
            Node SybilGhost = SybilGhosts.get(i);
            SybilGhost.saveColor=color(40,120,134);
            for(int k=1;k<numberNodes;k++) {
                  ArrayList<Integer> HopList = new ArrayList<Integer>();
                   HopList=Nodes.get(k).getFullRoute();
                   if(HopList.contains(SybilGhost.getId())){
                         SybilAffectedNodes.add(Nodes.get(k));
                         Nodes.get(k).angleChanged=false;
                         Nodes.get(k).saveColor=color(40,120,134);
                   }
             }
      }
}
```

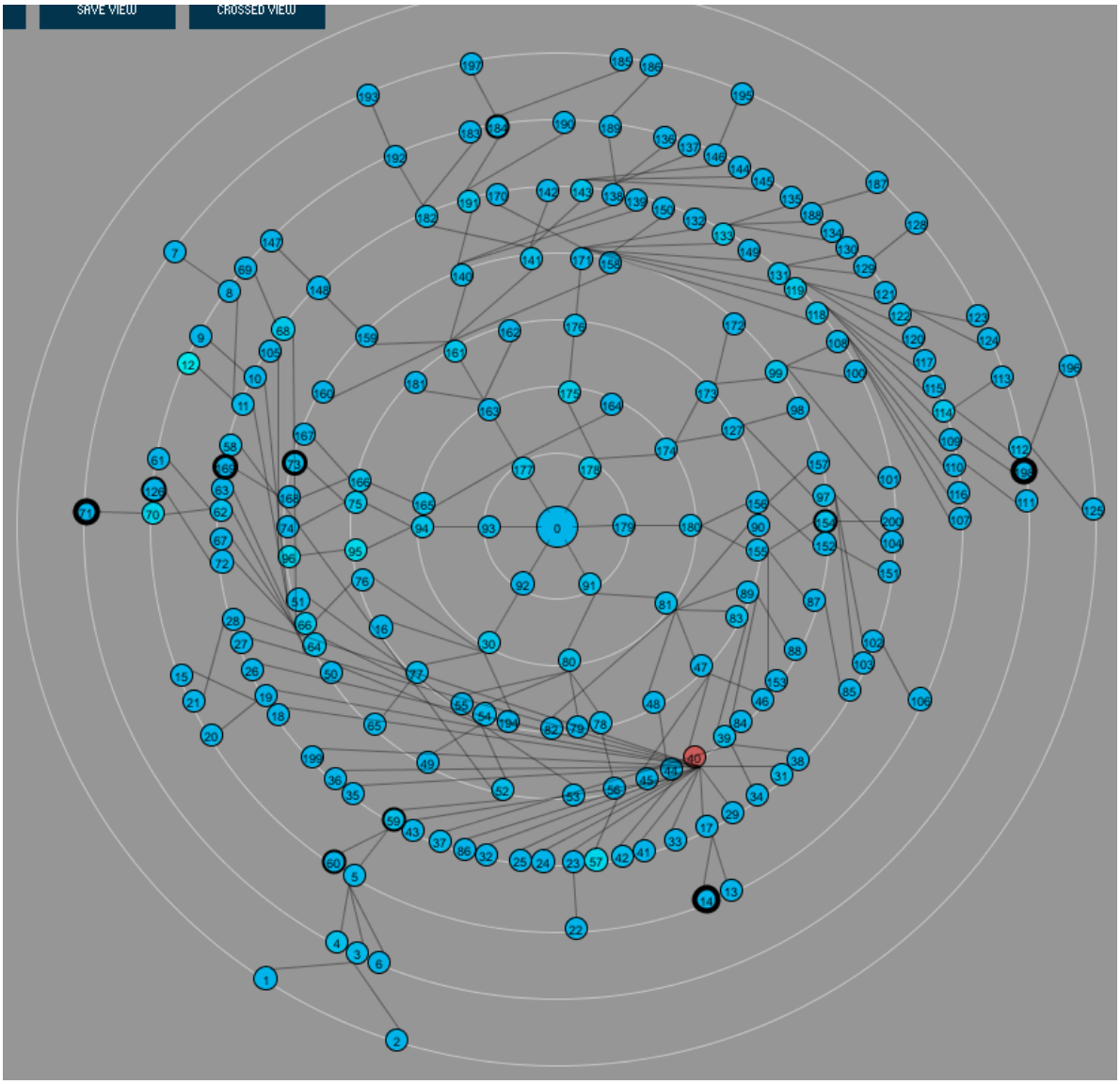
Για να μπορέσει να εκτελεστεί σωστά η επίθεση χρειάστηκε να γίνουν αλλαγές στον κώδικα βασικών μεθόδων της εφαρμογής. Η μέθοδος ανακάλυψης γειτόνων έπρεπε να τροποποιηθεί έτσι ώστε οι Sybil Ghosts να μην είναι ορατοί από τους υγιείς κόμβους. Ο αλγόριθμος δρομολόγησης τροποποιήθηκε επίσης ώστε να μην υπάρχουν index out of bounds exceptions όταν τερματίζουμε την επίθεση Sybil. Πιο συγκεκριμένα όταν αφαιρούσαμε τους Sybil Nodes και Sybil Ghosts από το δίκτυο κάποια πακέτα που ήταν στην φάση μετάδοσης προσπαθούσαν να δρομολογηθούν μέσα από ανύπαρκτους κόμβους. Η τελευταία αλλαγή έγινε στο Save View ώστε τα SybilGhosts να μην εμφανίζονται καθόλου.

Η επίθεση τερματίζεται με την `stopSybilFunc()`. Όσοι κόμβοι επηρεάστηκαν από την επίθεση επανέρχονται στην φυσιολογική κατάσταση. Οι κακόβουλοι κόμβοι αφαιρούνται από το δίκτυο όπως επίσης και τα Sybil Ghosts.

```
void stopSybilFunc(){
	for(int i=0; i<SybilAffectedNodes.size();i++) {
		Node SAffNode = SybilAffectedNodes.get(i);
		SAffNode.setcostToSink(100);
		SAffNode.GetNeighborList().clear();
		SAffNode.saveColor=color(0,180,234);
		SAffNode.type="Normal";
	}
	SybilAffectedNodes.clear();
	for(int i=0; i<SybilNodes.size();i++) {
		if(SybilNodes.get(i).getId()>=initialNodes) {
			Nodes.remove(SybilNodes.get(i));
			numberNodes--;
		} else {
			SybilNodes.get(i).type="Normal";
			SybilNodes.get(i).saveColor=color(0,180,234);

		}
		routing(false);
	}
	SybilNodes.clear();
	for(int i=0; i<SybilGhosts.size();i++) {
		Nodes.remove(SybilGhosts.get(i));
		numberNodes--;
		routing(false);
	}
	SybilGhosts.clear();
}
```

Επίθεση Hello

Η τελευταία επίθεση που μπορούμε να πραγματοποιήσουμε με την εφαρμογή είναι η επίθεση Hello. Ένας επιλεγμένος κόμβος που εκτελεί την επίθεση έχει αυξημένη εμβέλεια μετάδοσης και στέλνει συνεχώς πακέτα δρομολόγησης σε όσους κόμβους είναι στην ακτίνα του. Ο σκοπός είναι να ξεγελάσει τους κόμβους του δικτύου ώστε να νομίζουν πως είναι γείτονας τους. Το αποτέλεσμα θα είναι οι κόμβοι να προσπαθούν να στείλουν πακέτα σε αυτόν, αλλά τα πακέτα να μην φτάνουν ποτέ λόγω της μεγάλης απόστασης, έτσι οι κόμβοι δεν μπορούν να επικοινωνήσουν με τον Sink Node και ταυτόχρονα σπαταλούν την μπαταρία τους. Επίσης τα πακέτα δρομολόγησης στέλνονται συνεχώς από τον Hello Node, προκαλώντας ακόμα ταχύτερη μείωση της μπαταρίας των κόμβων που τα λαμβάνουν.



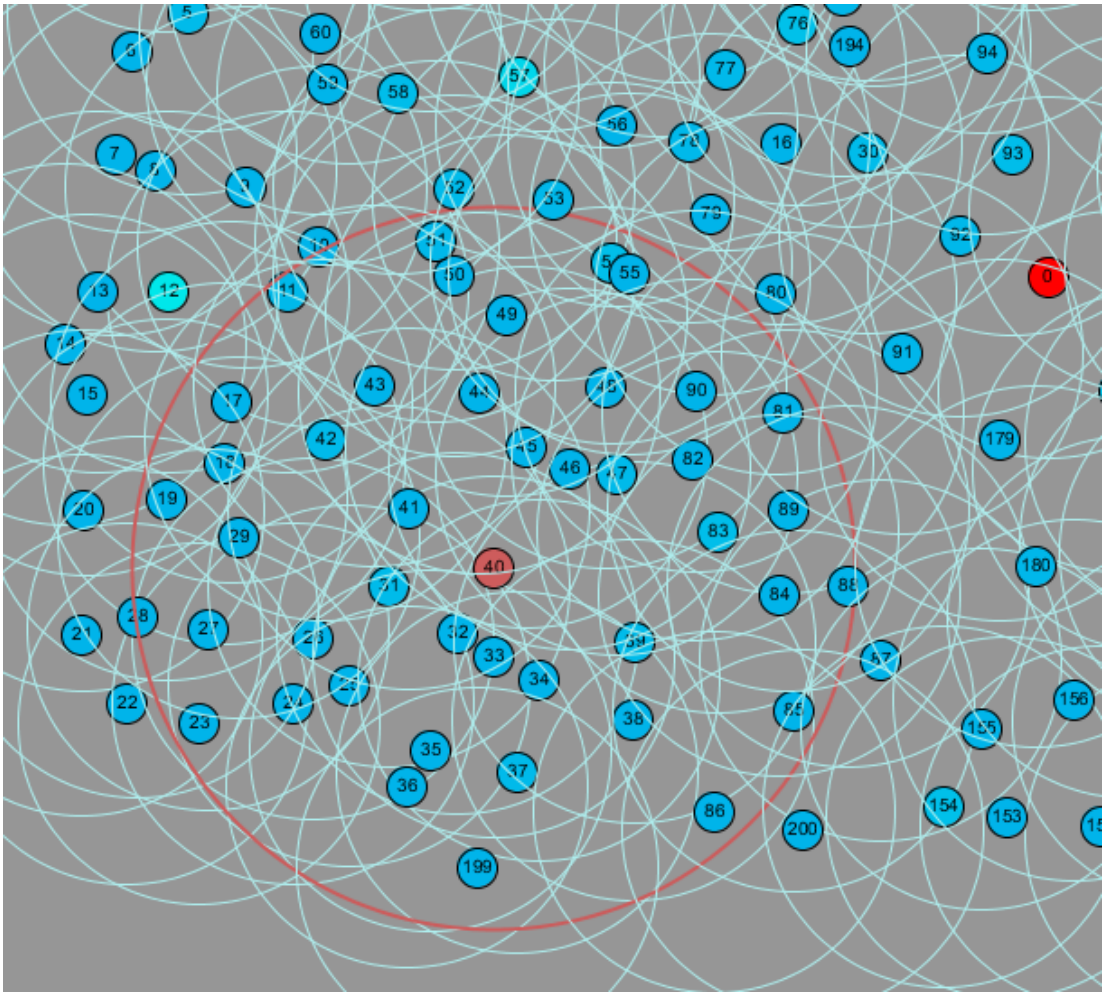
Εικόνα.32: Απεικόνιση επίθεσης Hello στο Save View

Για την προσομοίωση της μπαταρίας των κόμβων ορίσαμε μία float μεταβλητή με αρχική τιμή το 100.0, η τιμή αυτή θα μειώνεται κατά λίγο κάθε φορά που ο κόμβος στέλνει πακέτα. Όταν η μπαταρία εξαντληθεί ο κόμβος τίθεται εκτός λειτουργίας και οι κόμβοι που τον είχαν ως next hop θα πρέπει να αναζητήσουν μία διαφορετική διαδρομή. Εάν δεν μπορέσει να βρεθεί διαδρομή οι κόμβοι αποσυνδέονται από το δίκτυο. Για να υλοποιηθεί αυτό έγιναν οι εξής αλλαγές:

- **Αλγόριθμος δρομολόγησης:** Οι κόμβοι που δεν έχουν μπαταρία δεν στέλνουν πακέτα δρομολόγησης και μαρκάρονται ως μη συνδεδεμένοι. Οι υγιείς κόμβοι θα δεχτούν ως επόμενο hop μόνο κόμβους που έχουν μπαταρία και είτε έχουν path προς τον sink node είτε προς κάποιο άκρο σκουληκότρυπας. Η αλλαγή αυτή είναι απαραίτητη για μπορέσει να βρεθεί μονοπάτι όταν εξαντληθεί η μπαταρία κάποιου κόμβου. Σε αντίθετη περίπτωση υπήρχε κίνδυνος κόμβοι να προσπαθούν να δρομολογήσουν ο

ένας μέσω του άλλου νομίζοντας ο καθένας πως ο άλλος έχει μονοπάτι προς τον sink.

- **Παραγωγή κίνησης:** Κανένας κόμβος χωρίς μπαταρία δεν δημιουργεί ούτε δρομολογεί πακέτα. Πακέτα που προορίζονταν για κόμβο που ξαφνικά έμεινε χωρίς μπαταρία απορρίπτονται. Πακέτα που προορίζονται για Hello node και ο αποστολέας τους δεν έχει αρκετή ισχύ χάνονται στην άβυσσο και καταγράφονται.
- **Save View:** Όσοι κόμβοι δεν είναι συνδεδεμένοι στο δίκτυο δεν εμφανίζονται καθόλου. Οι κόμβοι με εξαντλημένη μπαταρία εμφανίζονται με διαφορετικό χρώμα.



Εικόνα.33: Απεικόνιση επίθεσης Hello στο Normal View

Το GUI είναι ίδιο με τα προηγούμενα, επιλογή κόμβου, κουμπιά εκκίνησης και τερματισμού.

Εικόνα.34: Hello GUI

Ο controller διαβάζει τα id που εισάγει ο χρήστης και βάζει τους αντίστοιχους κόμβους σε ένα ArrayList, επίσης ενεργοποιεί την σημαία `HelloFlag` και ενημερώνει τον χρήστη.

```
if(theEvent.isFrom("StartHello")) {
    int HelloId=-1;
    String tempHello="";
    if (cp5.get(Textfield.class,"HelloNode").getText().length()>0) {
        tempHello = cp5.get(Textfield.class,"HelloNode").getText();
        HelloId = Integer.parseInt(tempHello);
        if(!HelloNodes.contains( Nodes.get(HelloId)) && HelloId<numberNodes &&
HelloId!=0) {
             HelloNodes.add(Nodes.get(HelloId));
             HelloFlag=true;
             dNow = new Date( );
             Messages.put(ft.format(dNow),"Node "+HelloId+" is performing a Hello
attack");
        }
    }


   }
```

Όταν ενεργοποιηθεί η σημαία και υπάρχει κίνηση στο δίκτυο κάθε 50ms καλείται η `helloStart()`

```
if((millis() - helloTimer > 50) && HelloFlag && !InitFlag && TrafficFlag &&
(NormalViewFlag || SaveViewFlag || NodeViewFlag || CrossedViewFlag)) {
    helloStart();
    helloTimer=millis();
}
```

η οποία για κάθε Hello Node, βρίσκει τους γείτονες του και τους στέλνει συνεχώς πακέτα δρομολόγησης. Στην πραγματικότητα όταν ένας κόμβος λάβει πακέτο δρομολόγησης(HELLO) πρέπει να απαντήσει σε αυτό με πακέτο επιβεβαίωσης(ACK). Για λόγους απλότητας δεν βάλαμε τέτοιου είδους πακέτα αλλά μειώνουμε την μπαταρία του κόμβου προσομοιώνοντας την αποστολή τους που θα είχε το ίδιο αποτέλεσμα:

```
void helloStart() {
      for (int i=0;i<HelloNodes.size();i++) {
             Node HelloNode = HelloNodes.get(i);
             HelloNode.setRange(150);
             ArrayList<Integer>NeighboorList = new ArrayList<Integer>();
             NeighboorList = HelloNode.GetNeighborList();
             for(int j=0;j<NeighboorList.size();j++) {
                   if(NeighboorList.get(j)!=0) {
                         if(packetCounter==0){packetCounter=1;}
                         Packet rPacket = new Packet(packetCounter+1, HelloNode,
Nodes.get(NeighboorList.get(j)), Nodes.get(NeighboorList.get(j)), 20, 10, true,
null);
                         packetCounter++;
                         routingPacketCounter++;
                         Nodes.get(NeighboorList.get(j)).queue.add(rPacket);
                         Nodes.get(NeighboorList.get(j)).battery-=0.001;
```

Για να τερματιστεί η επίθεση διαγράφουμε τους κόμβους από το ArrayList των `HelloNodes`, επαναφέρουμε την ακτίνα και το χρώμα τους, και θέτουμε την `HelloFlag=false;`

```
if(theEvent.isFrom("StopHello")) {
	for(int i=0;i<HelloNodes.size();i++) {
		HelloNodes.get(i).range=range;
		HelloNodes.get(i).saveColor=color(0,180,234);
	}
	for(int k=0;k<Nodes.size();k++){
		Nodes.get(k).angleChanged=false;
	}
	HelloNodes.clear();
	HelloFlag=false;
	dNow = new Date( );
	Messages.put(ft.format(dNow),"All HELLO attacks have stopped");
}
```

Σε αυτό το σημείο έχουμε ολοκληρώσει την περιγραφή των επιθέσεων που μπορεί να προσομοιώσει η εφαρμογή μας. Στην επόμενη ενότητα θα αναφερθούμε στα προβλήματα που προέκυψαν και στην αντιμετώπιση τους.

Προβλήματα που αντιμετωπίστηκαν

Κατά την διάρκεια δημιουργίας της εφαρμογής βρεθήκαμε αντιμέτωποι με διάφορα προβλήματα σε όλα τα στάδια. Τα προβλήματα αυτά ήταν αναγκαίο να ξεπεραστούν για να μπορέσει να λειτουργήσει ορθά η εφαρμογή.

Το πρώτο και πιο σημαντικό πρόβλημα ήταν η διάταξη των κόμβων στο Save View. Έπρεπε να δημιουργήσουμε κατευθυνόμενα δέντρα, οι κόμβοι των οποίων πρέπει να βρίσκονται πάνω στην περίμετρο των κύκλων. Τα μόνα δεδομένα που έχουμε είναι το κόστος και το επόμενο hop των κόμβων. Το ζητούμενο είναι να υπολογιστούν συντεταγμένες x και y που θα καθορίζουν την θέση των κόμβων. Φυσικά οι συντεταγμένες αυτές δεν μπορούσαν να είναι τυχαίες πάνω στην περίμετρο των κύκλων αλλά κάθε κόμβος έπρεπε να είναι στην μεριά του πατέρα του και έναν κύκλο πιο έξω. Με αναζήτηση στο διαδίκτυο μάθαμε πως για να υπολογίσουμε συντεταγμένες στην περίμετρο ενός κύκλου ο τύπος είναι:

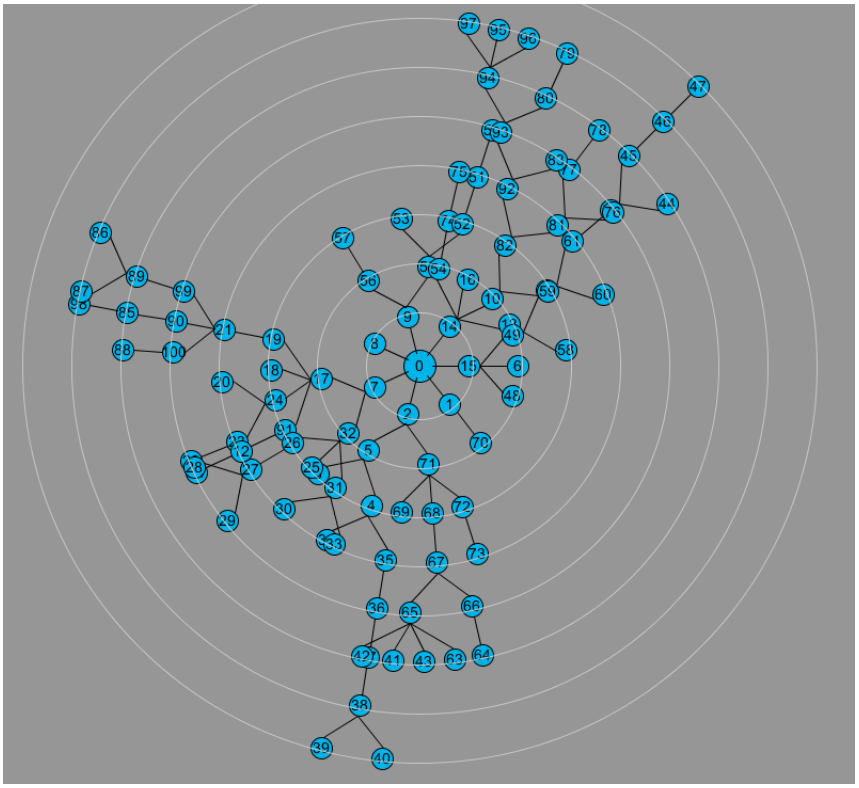
```
int x=(int)(A * cos(Angle));
int y=(int)(A * sin(Angle));
```

Όπου `A` η απόσταση από το κέντρο του κύκλου και `Angle` μία γωνία. Ο υπολογισμός της απόστασης ήταν εύκολος γιατί γνωρίζουμε που είναι το κέντρο του κύκλου και γνωρίζουμε πόσο μακριά από αυτό θα είναι ο κύκλος για κάθε κόστος. Το δύσκολο είναι ο υπολογισμός της γωνίας. Γνωρίζοντας πως ο κύκλος είναι 2π έπρεπε να χωρίσουμε την περιφέρεια του σε τόξα ανάλογα με τον αριθμό των κόμβων που έπρεπε να τοποθετηθεί. Εάν έχουμε 4 κόμβους με κόστος 1 θα τους τοποθετήσουμε στις θέσεις 2π, π/2, π. 3π/2. Εφόσον γνωρίζουμε την θέση των αρχικών κόμβων μετά πρέπει να υπολογίσουμε τις θέσεις των παιδιών του καθενός. Όπως και πριν χωρίζουμε τον κύκλο σε 4 τόξα (όσα και ο αριθμός των πατεράδων), αλλά το μέγεθος του κάθε τόξου

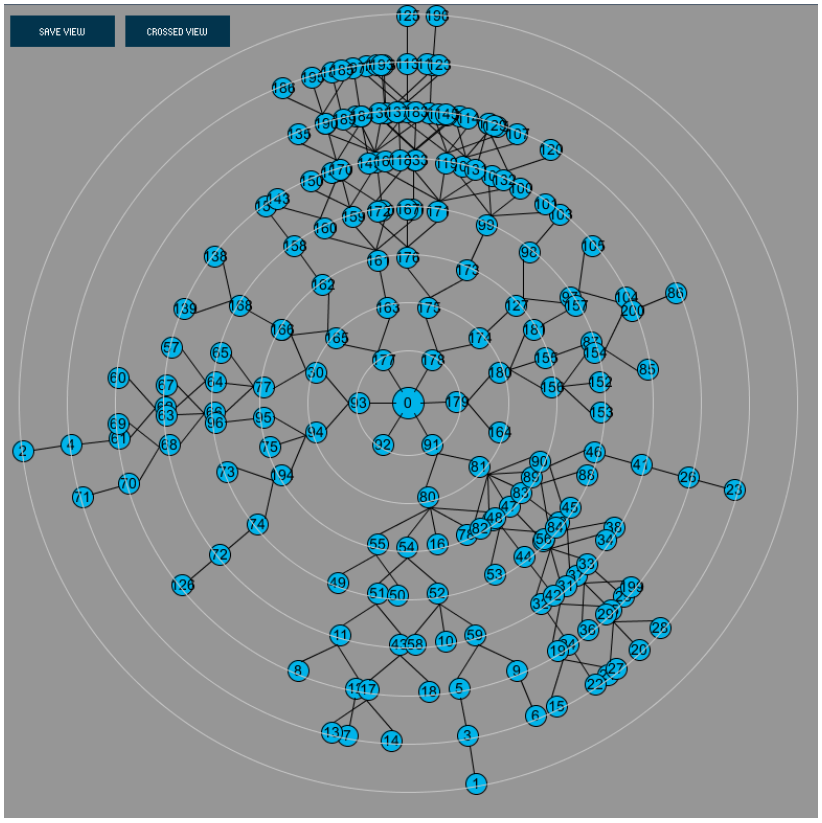
εξαρτάται από τον αριθμό των παιδιών. Το πρώτο παιδί θα έχει την γωνία του πατέρα του και τα υπόλοιπα θα τοποθετηθούν δεξιά και αριστερά βάση ενός βήματος που περιγράψαμε στην ενότητα του Save View.

Η λογική και τα μαθηματικά στον παραπάνω συλλογισμό είναι σωστά, αλλά στην πράξη αποδείχθηκε πως δεν δουλεύουν. Οι κόμβοι είναι τοποθετημένοι στο δίκτυο με τυχαίο τρόπο και τα δέντρα που σχηματίζουν είναι άνισα. Κάποιοι κόμβοι μπορεί να έχουν 10 παιδιά και κάποια κανένα. Το αποτέλεσμα ήταν αρκετοί κόμβοι να συμπίπτουν ο ένας με τον άλλον και αν αυξανόταν ο αριθμός των κόμβων επικρατούσε ένα χάος(Εικόνες 35a/b). Δοκιμάσαμε πολλά διαφορετικά βήματα αύξησης της γωνίας πχ μεγαλύτερο σε μικρό κόστος και σταδιακή αύξηση όσο αυξάνεται το κόστος, το αντίθετο από αυτό, αύξηση ανάλογα με τον αριθμό των παιδιών, μείωση ανάλογα με τον αριθμό των παιδιών κλπ. Καθένα από αυτά προκαλούσε καλύτερο αποτέλεσμα σε ορισμένα δίκτυα και χειρότερα σε άλλα, δεν μπορούσαμε όμως να βρούμε κάτι καθολικό.

Η μόνη λύση ήταν να τοποθετήσουμε τους κόμβους όσο καλύτερα μπορούμε και στη συνέχεια να κάνουμε έλεγχο για συγκρούσεις. Έτσι λοιπόν δημιουργήθηκε η `checkCollisions`. Μέχρι να φτάσει η μέθοδος στην τελική της μορφή πέρασε αρκετός καιρός και ακόμα περισσότερες δοκιμές, όμως καταφέραμε να μηδενίσουμε τις συγκρούσεις σε δίκτυα μέχρι 250 κόμβων. Ο περιορισμός αυτός είναι ένας φυσικός περιορισμός, δεν επαρκεί η περιφέρεια του κύκλου για να χωρέσουν όλοι οι κόμβοι για το οποίο υπάρχουν τρεις λύσεις: α)Να μεγαλώσουμε την περιφέρεια του κύκλου το οποίο προϋποθέτει να αυξήσουμε την ανάλυση της οθόνης, β)Να ελαττώσουμε τον αριθμό των κόμβων πάνω στον κύκλο, αυτό μπορούμε να το πετύχουμε μετακινώντας τους κόμβους σε άλλο μέρος του δικτύου και γ)Να μειώσουμε την ακτίνα του κύκλου το κόμβου, κάτι τέτοιο όμως θα κάνει δύσκολη την ανάγνωση του id, η γραμματοσειρά του οποίου πρέπει επίσης να μικρύνει για να χωρέσει στον κύκλο.



Εικόνα.35a: 100 κόμβοι



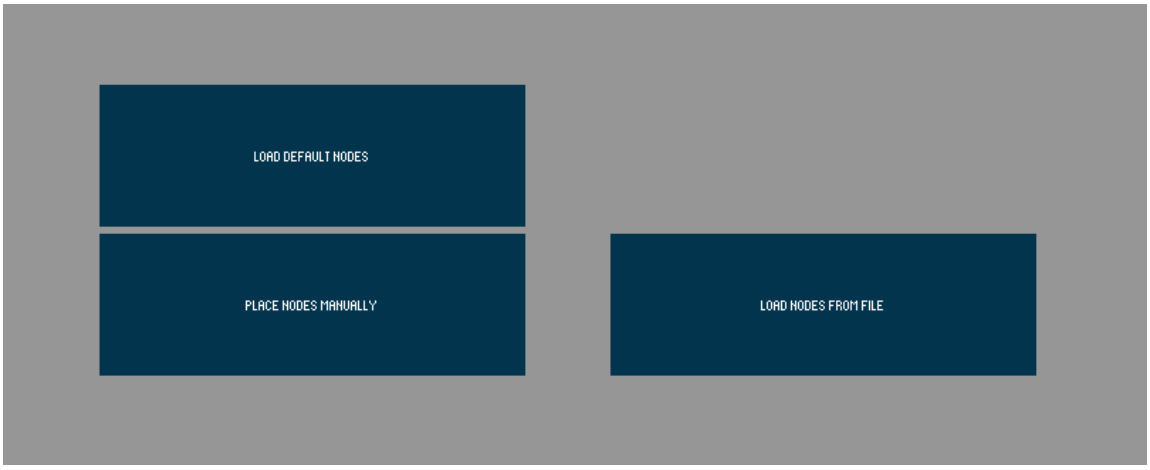
Εικόνα.35b: 200 κόμβοι

Ένα μικρό πρόβλημα που αντιμετωπίστηκε επίσης είχε να κάνει με τα κουμπιά του GUI. Όπως ανακαλύψαμε με αναζήτηση στο Internet ο controller των κουμπιών της βιβλιοθήκης cp5 ενεργοποιούταν μόνος του μόλις ξεκινούσε το πρόγραμμα και εκτελούσε τη λειτουργία που του είχε ανατεθεί. Μπορεί κάποιος να φανταστεί τι γινόταν όταν πατιόταν το Exit button κάθε φορά που εκτελούσαμε την εφαρμογή… Ένα workaround σε αυτό το bug είναι η δημιουργία ενός μετρητή που ξεκινάει με την εκτέλεση του προγράμματος και κρατάει τα milliseconds που πέρασαν. Σε κάθε κουμπί έπειτα βάζουμε έλεγχο να μην πατιέται αν έχουν περάσει λιγότερα από 1000ms από την έναρξη του προγράμματος.

Όταν έπρεπε να προγραμματίσουμε την επίθεση Sybil βρεθήκαμε μπροστά σε ένα ακόμα εμπόδιο. Η επίθεση αυτή απαιτεί την εισαγωγή νέων κόμβων στο δίκτυο. Όταν όμως ξεκινήσαμε την ανάπτυξη της εφαρμογής χρησιμοποιήσαμε στατικό πίνακα αποθήκευσης των κόμβων στον οποίο δεν μπορούσαμε να αλλάξουμε το μέγεθος για να βάλουμε τους νέους κόμβους. Έπρεπε λοιπόν ο πίνακας να αλλάξει σε ArrayList και μαζί με αυτόν να γίνουν αλλαγές σε όλα τα σημεία όπου διαβάζουμε ή γράφουμε στον πίνακα.

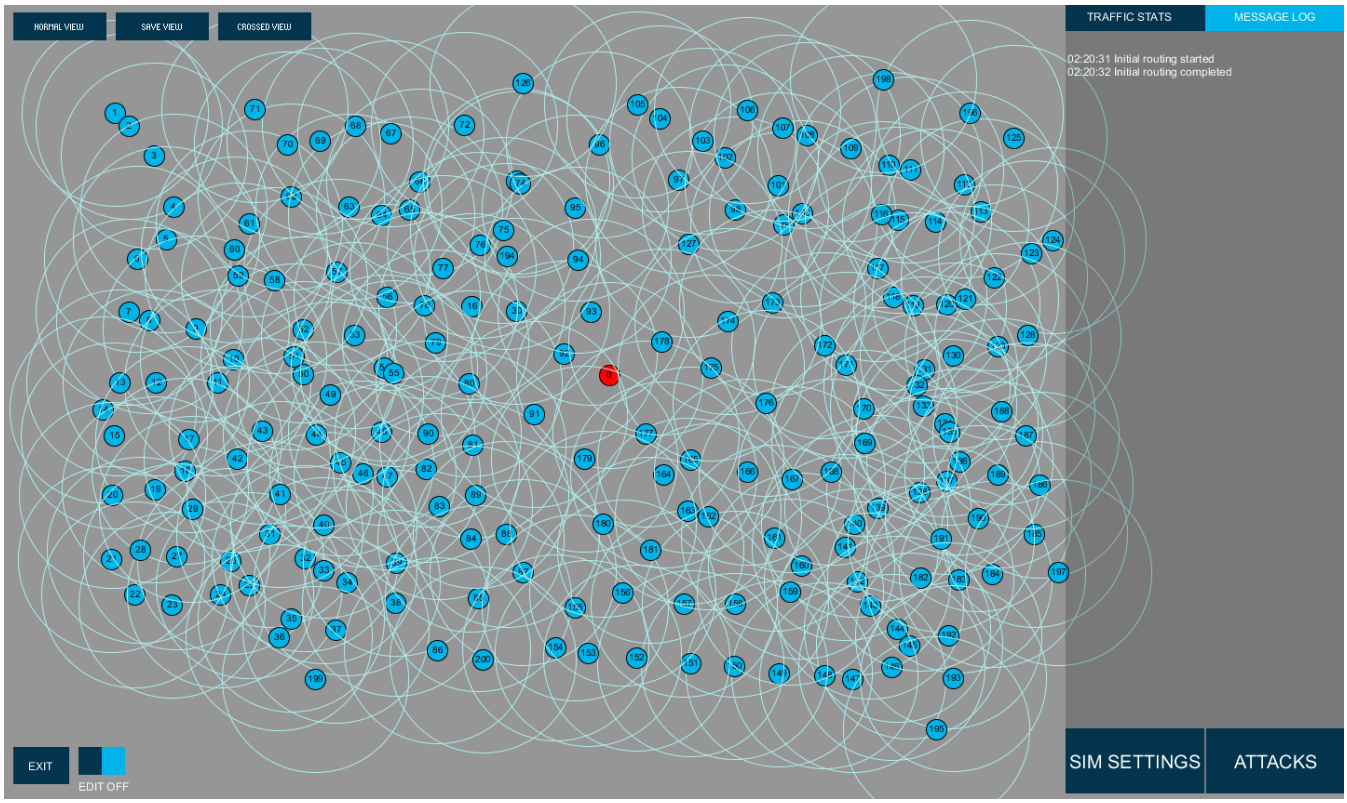
Η επίθεση Sybil δημιούργησε εκτός από το παραπάνω και ακόμη ένα πρόβλημα. Πιο συγκεκριμένα το πρόβλημα παρουσιαζόταν τη στιγμή που τερματίζαμε την επίθεση. Ο τερματισμός της επίθεσης περιλάμβανε την αφαίρεση κόμβων από το δίκτυο. Αυτό είχε ως συνέπεια την δημιουργία java exceptions στον αλγόριθμο δρομολόγησης και κίνησης πακέτων που τερμάτιζαν την εκτέλεση της εφαρμογής. Το πρόβλημα αυτό ήταν αδύνατο να ξεπεραστεί με κάποιον τρόπο οπότε αναγκαστήκαμε να περικλύσουμε τα κομμάτια κώδικα που «χτυπούσαν» σε try-catch blocks. Έτσι το exception που παρουσιαζόταν μία φορά δεν σταματούσε την εφαρμογή και στον επόμενο κύκλο δρομολόγησης όλα ερχόταν στη κανονική κατάσταση.

Εκτέλεση της εφαρμογής



Εικόνα.36: Οθόνη υποδοχής

Μόλις εκτελέσουμε την εφαρμογή μας παρουσιάζεται η αρχική οθόνη με τρεις επιλογές. Να φορτώσουμε τους προκαθορισμένους κόμβους, να δημιουργήσουμε δικούς μας κόμβους ή να φορτώσουμε κόμβους που δημιουργήσαμε στο παρελθόν και αποθηκεύσαμε σε αρχείο csv. Για το παράδειγμα αυτό θα χρησιμοποιήσουμε ένα δίκτυο με 200 κόμβους.



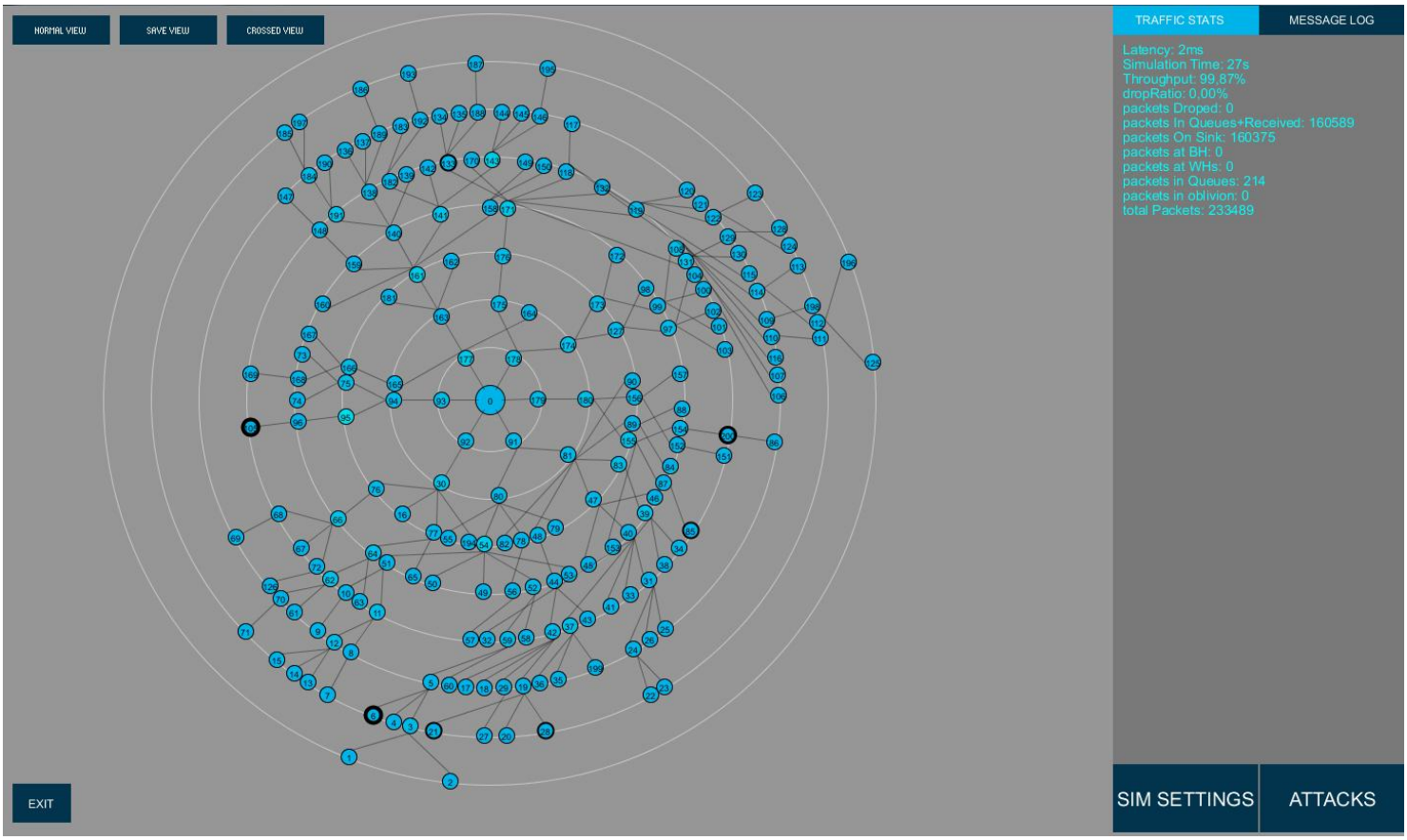
Εικόνα.37:ΑΔΑ με 200 κόμβους. Πανοραμική άποψη (Normal View)

Το δίκτυο αυτή τη στιγμή βρίσκεται σε αδράνεια, δεν υπάρχει αποστολή δεδομένων. Οι κόμβοι ανταλλάσσουν μόνο πακέτα δρομολόγησης (hello packets) κάθε 500ms. Για να ξεκινήσουμε την προσομοίωση κίνησης πρέπει να θέσουμε τις αντίστοιχες παραμέτρους. Για το παράδειγμα ο χρόνος της προσομοίωσης θα είναι τα 10 λεπτά με

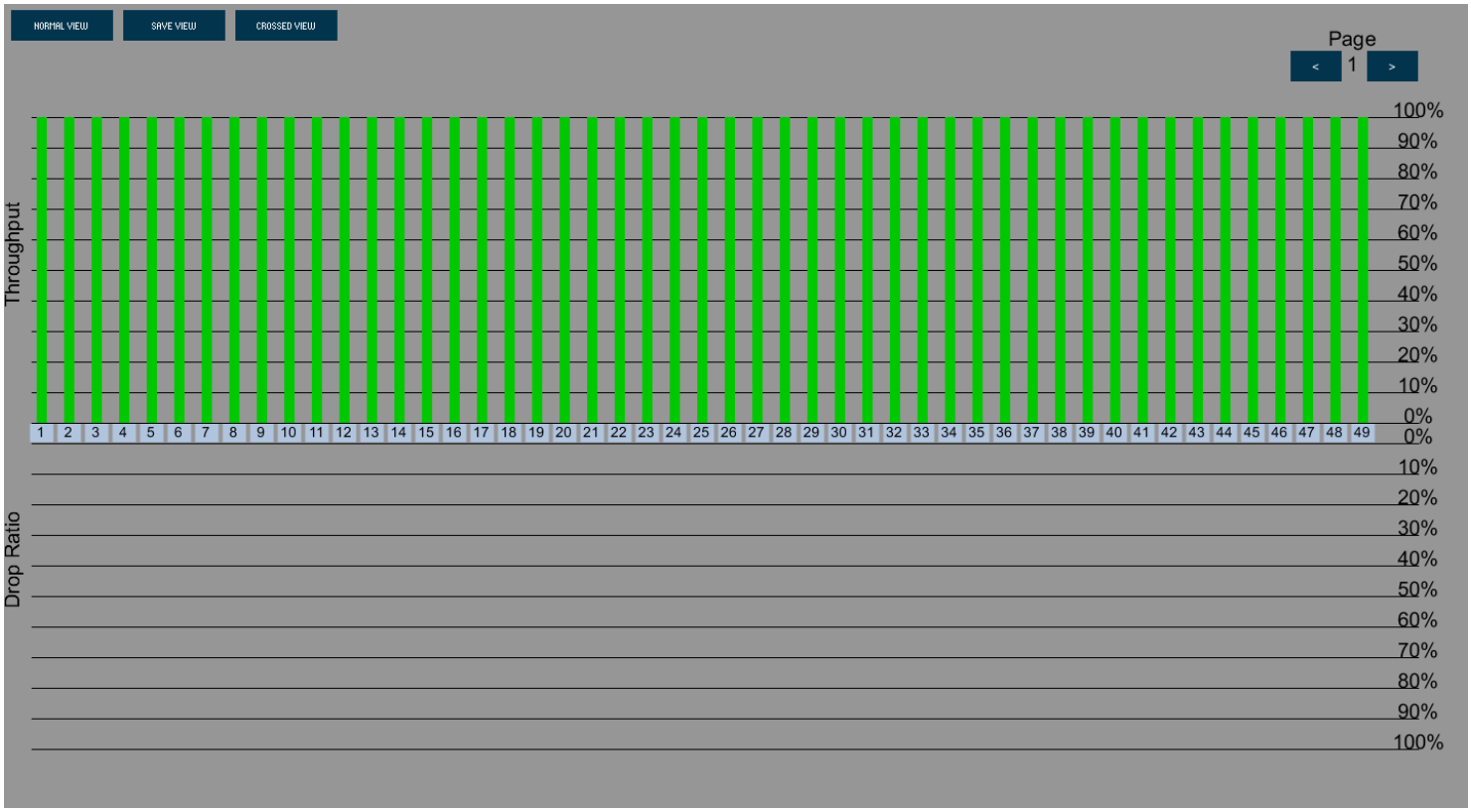
ταχύτητα δικτύου 150Mbps. Οι κόμβοι θα έχουν 5% πιθανότητα σε κάθε κύκλο να μεταδώσουν από 100 μέχρι 300 πακέτα των 200bytes. Η περίοδος του κύκλου μετάδοσης είναι 500ms. Θα εκτελέσουμε μία προσομοίωση για κάθε επίθεση.

Επίθεση μαύρης τρύπας

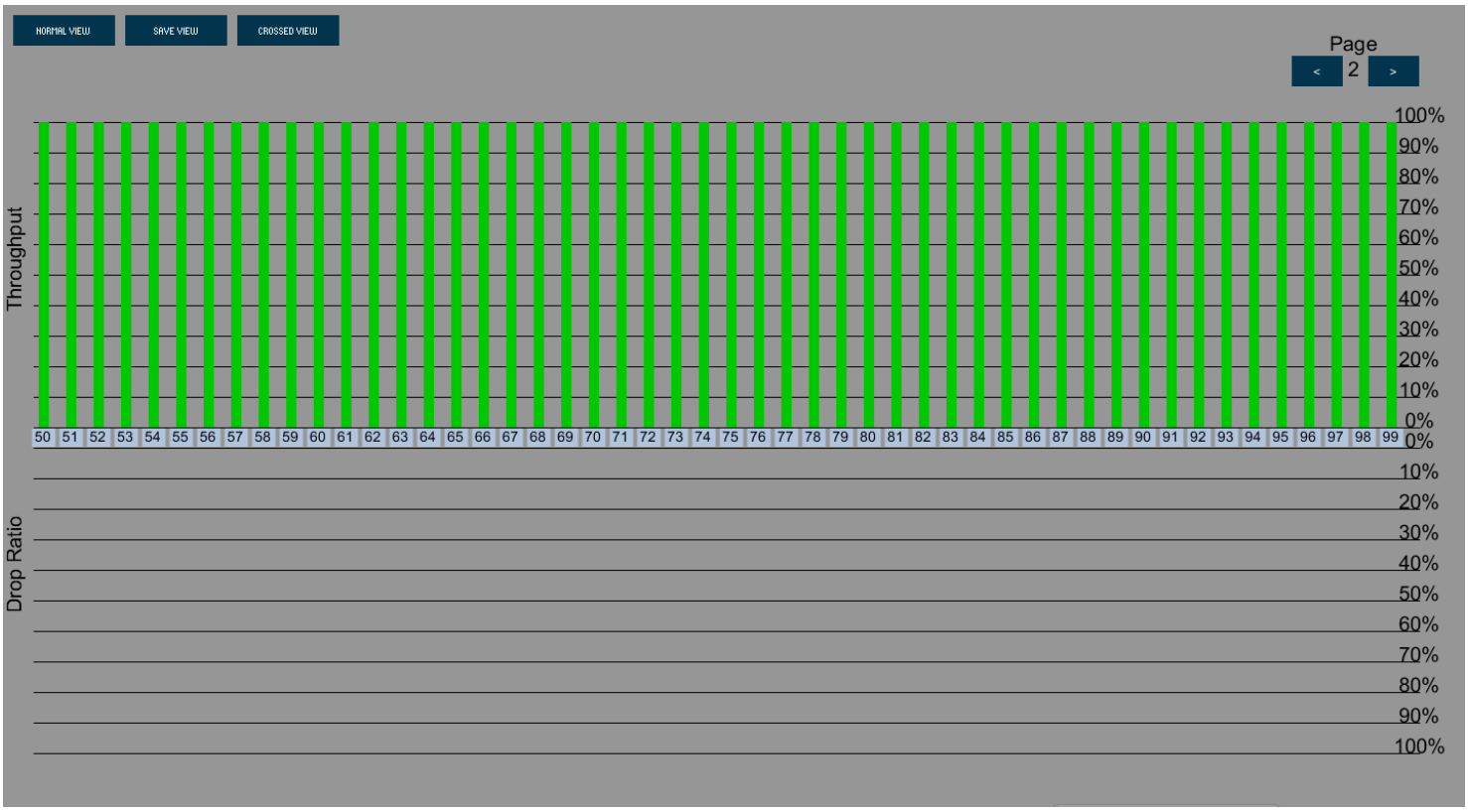
Η πρώτη επίθεση που θα προσομοιώσουμε είναι η MT. Στις παρακάτω εικόνες φαίνεται το δίκτυο στην κατάσταση του πριν την επίθεση.



Εικόνα.38a: Save View πριν τη MT. Throughput: σχεδόν 100%, χαμηλό latency

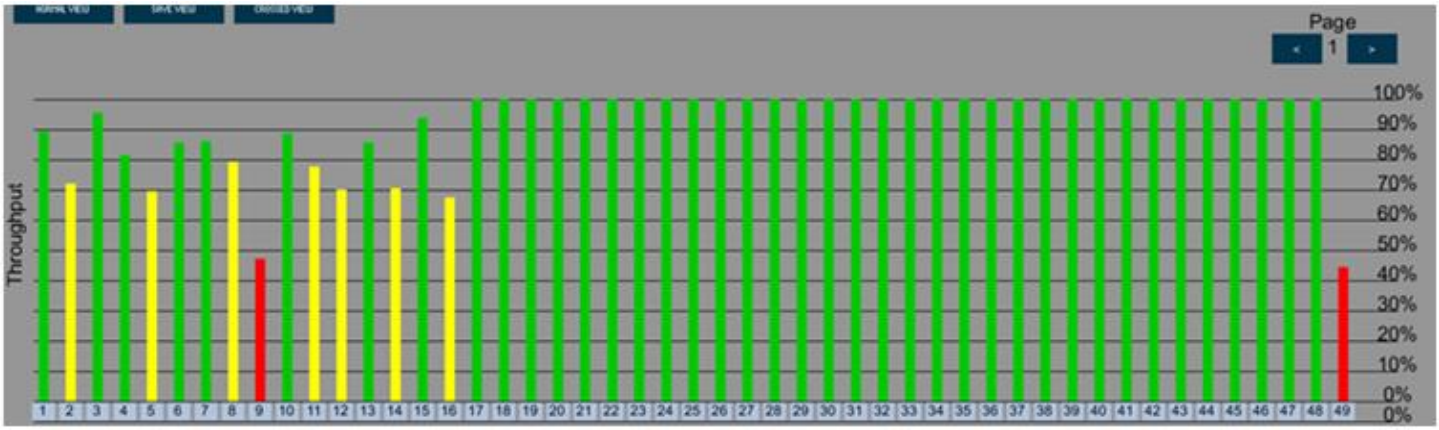


Εικόνα.38b: Crossed View πριν τη MT. Όλοι οι κόμβοι έχουν σχεδόν 100% throughput και μηδενικό drop ratio

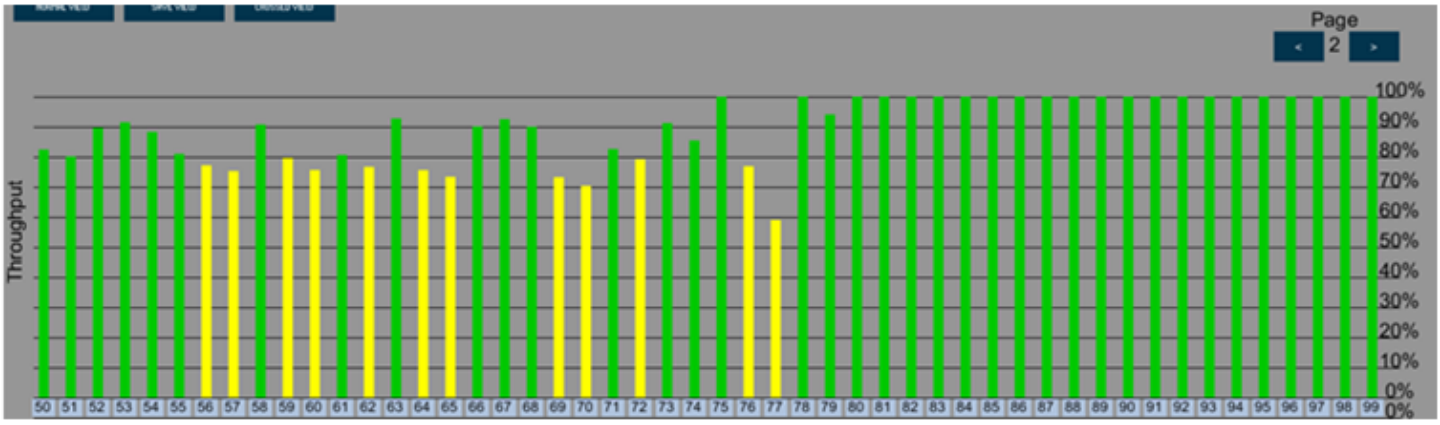


Εικόνα.38c: Crossed View πριν τη MT. Όλοι οι κόμβοι έχουν σχεδόν 100% throughput και μηδενικό drop ratio.

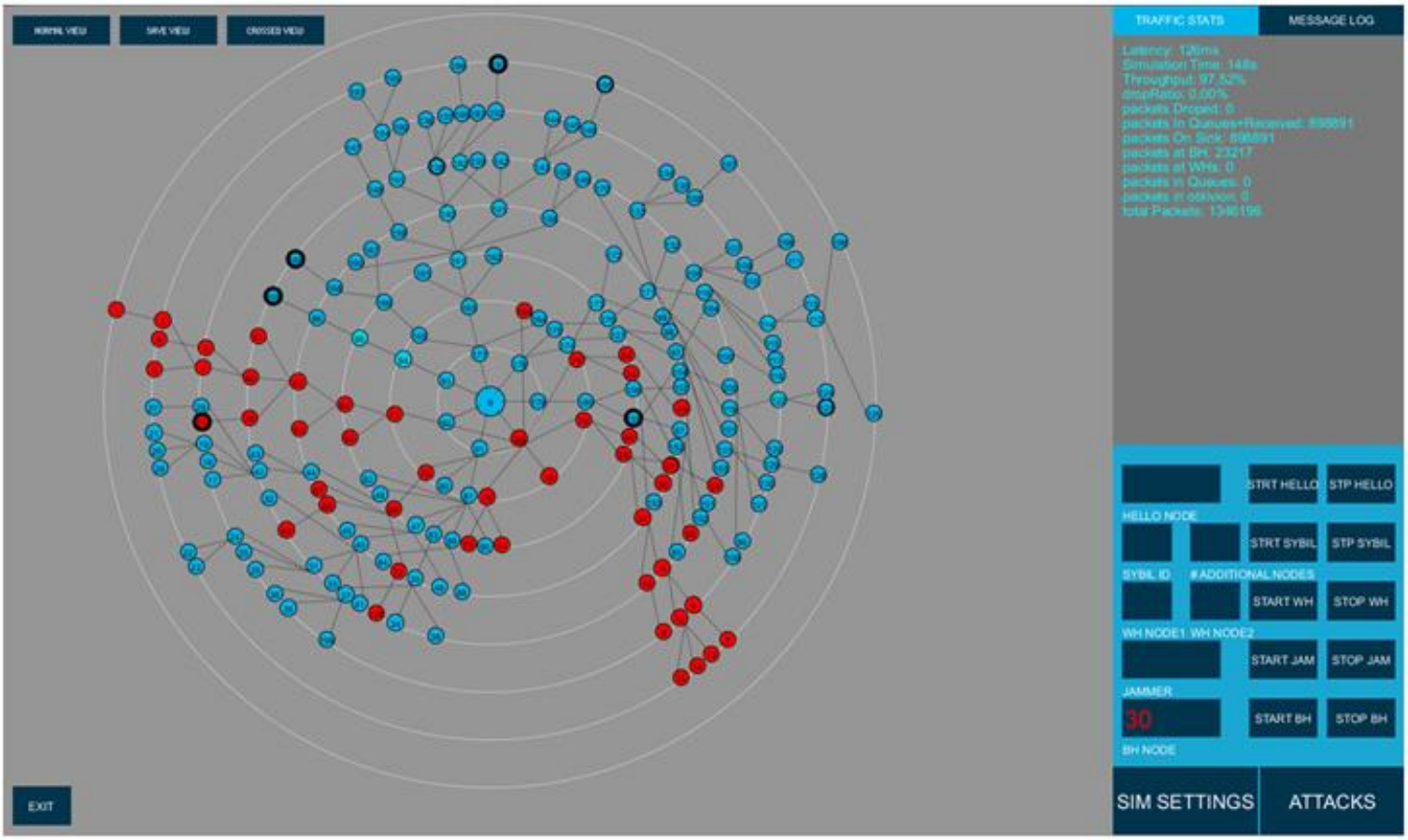
Μόλις επιλέξαμε τον κόμβο 30 να είναι αυτός η μαύρη τρύπα η διάταξη του δικτύου άλλαξε αισθητά. Το κόστος του 30 άλλαξε σε 1, από 2 που ήταν προηγουμένως, και από τους 17 κόμβους που είχε στο υποδέντρο του απέκτησε 49 κόμβους. Οι 49 αυτοί κόμβοι που έχουν σημειωθεί με κόκκινο χρώμα έχουν επηρεαστεί και στέλνουν τα δεδομένα τους στην μαύρη τρύπα. Παρατηρούμε στα στατιστικά ότι το throughput άρχισε να μειώνεται και πως ήδη 23217 πακέτα έχουν χαθεί στην μαύρη τρύπα. Παρακάτω μπορούμε να δούμε αναλυτικά την μείωση του throughput κάθε κόμβου στο Crossed View. Το drop ratio δεν αυξάνεται καθώς η μαύρες τρύπες δεν προκαλούν απόρριψη πακέτων, οι κόμβοι νομίζουν πως τα πακέτα τους φτάνουν κανονικά στον προορισμό τους.



Εικόνα.39a: Crossed View στην εκκίνηση της μαύρης τρύπας

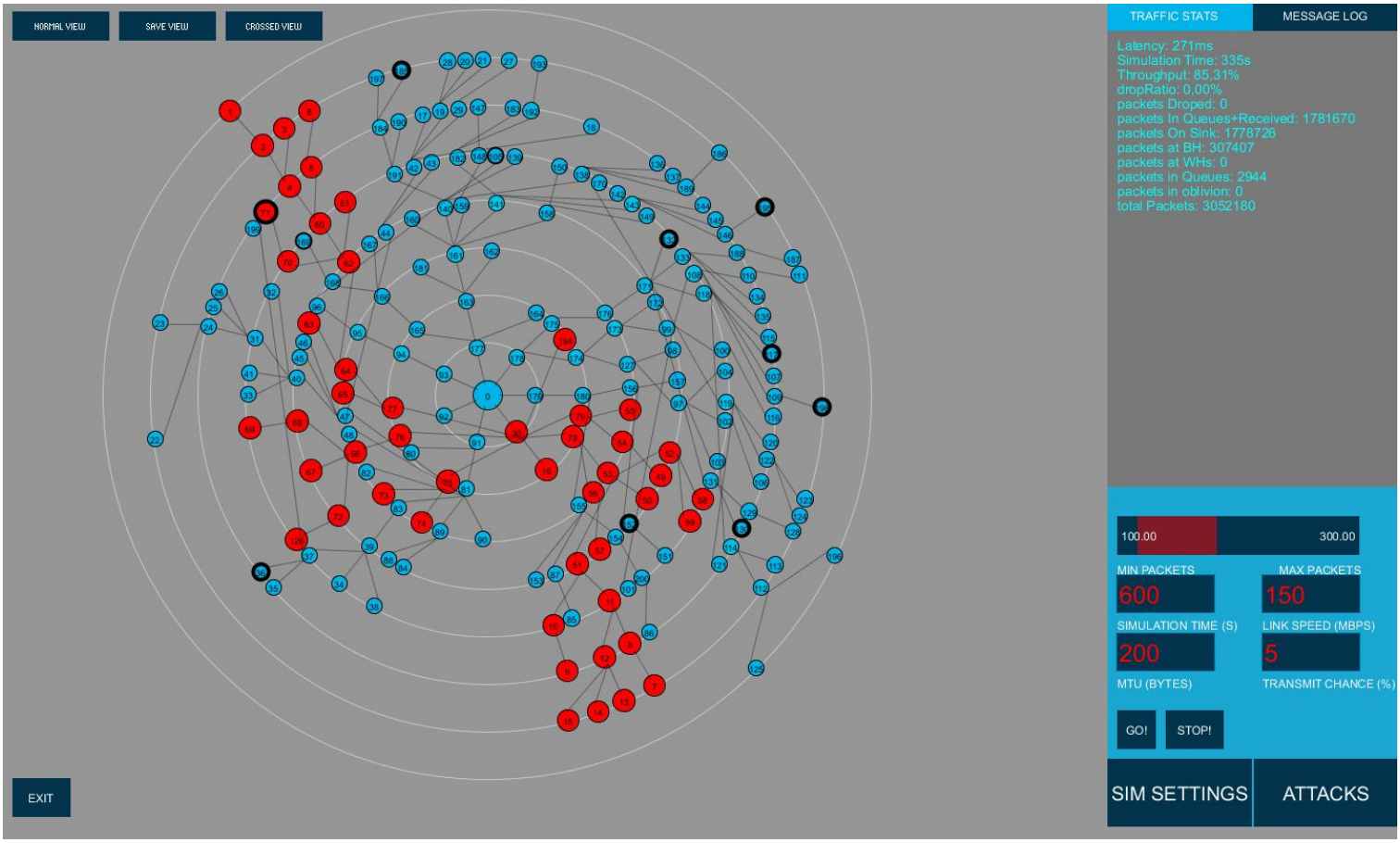


Εικόνα.39b: Crossed View στην εκκίνηση της μαύρης τρύπας



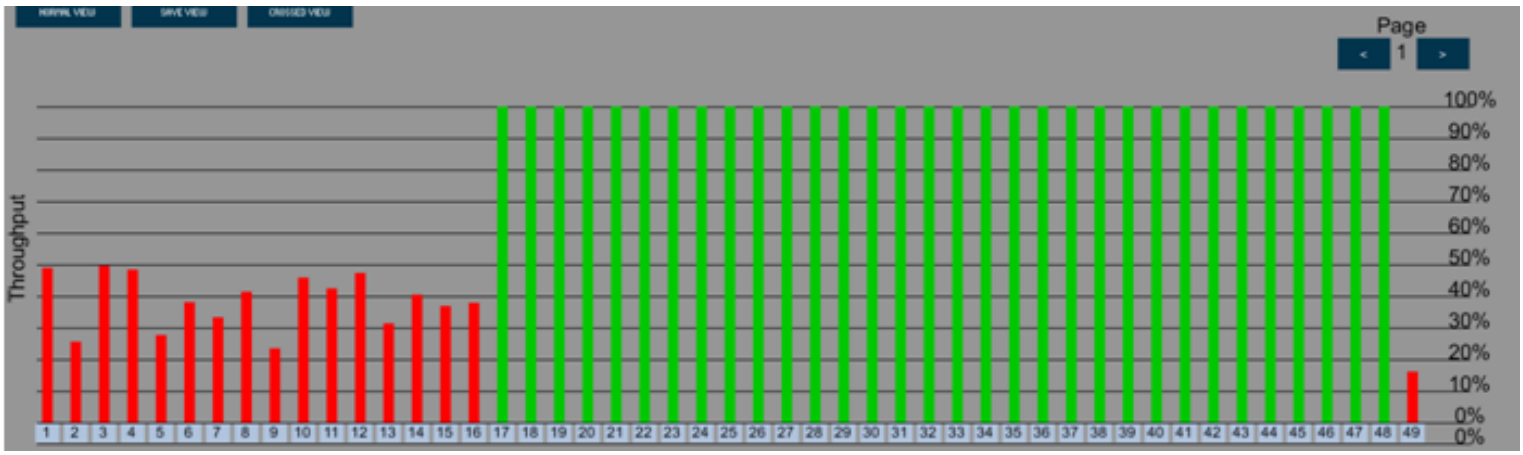
Εικόνα.39c: Save View στην εκκίνση της μαύρης τρύπας.

Παρακάτω παρατηρούμε την κατάσταση του δικτύου 3 λεπτά περίπου μετά την εμφάνιση της μαύρης τρύπας. Οι κόκκινοι κόμβοι έχουν αρχίσει να φουσκώνουν αισθητά, υποδηλώνοντας πως ο sink node έχει να λάβει πακέτα από αυτούς για αρκετή ώρα. Επίσης το throughput έχει μειωθεί στο 85% και 307.407 πακέτα από τα συνολικά 3.052.180 έχουν χαθεί στην μαύρη τρύπα.

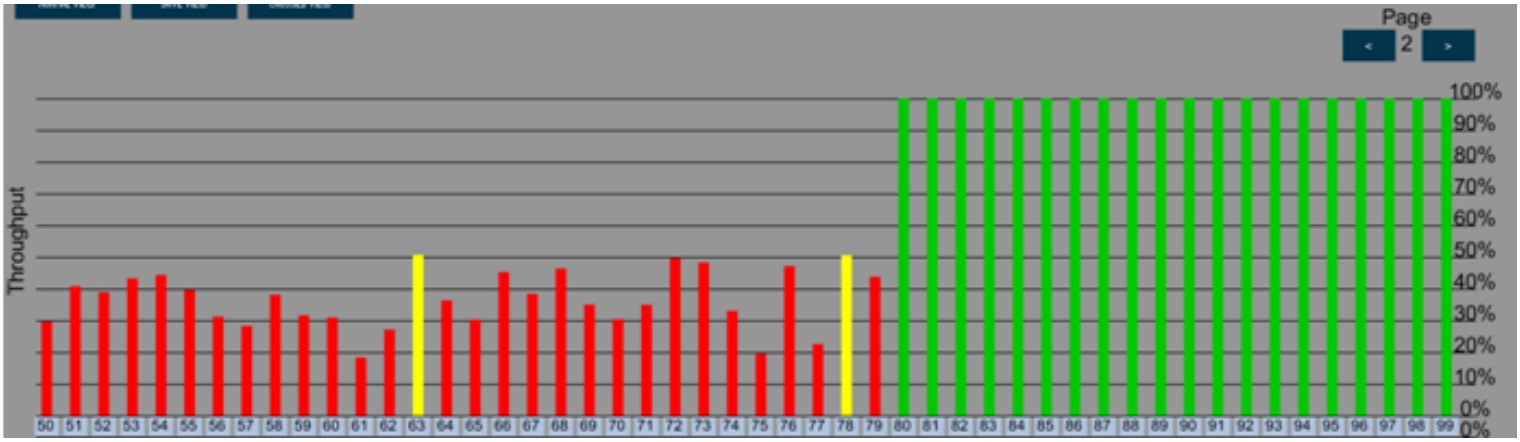


Εικόνα.40a: Save view, 3 λεπτά μετά την εμφάνιση μαύρης τρύπας

Στο Crossed View βλέπουμε αναλυτικά την πτώση του throughput. Οι κόμβοι με πράσινο throughput βρίσκονται σε άλλη περιοχή του δικτύου που δεν επιρρεάζεται από την Black hole.

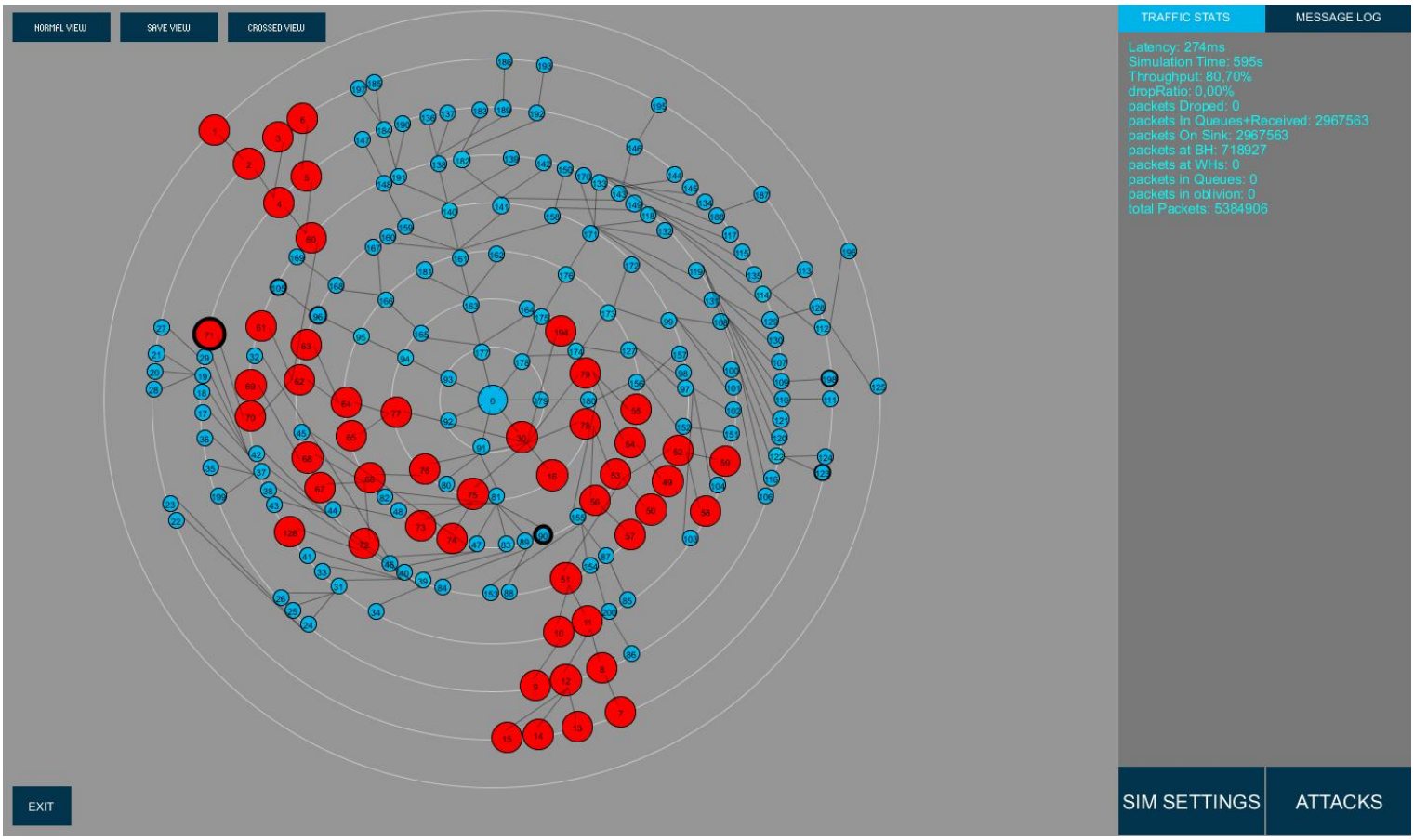


Εικόνα. 40b: Crossed View 3 λεπτά μετά την εμφάνιση μαύρης τρύπας



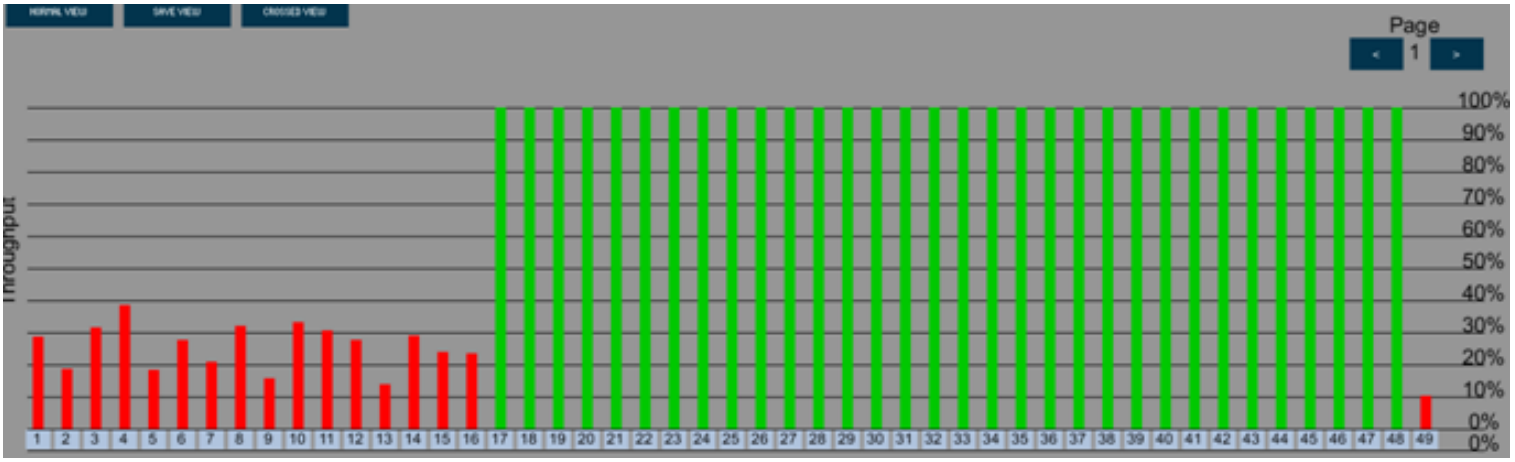
Εικόνα. 40c: Crossed View 3 λεπτά μετά την εμφάνιση μαύρης τρύπας

Λίγο πριν τελειώσει η προσομοίωση παρατηρούμε ότι ο κόμβος 30 ενεργώντας ως μαύρη τρύπα κατάφερε να μειώσει το συνολικό throughput του δικτύου κοντά στο 80% και να απορροφήσει 718.927 πακέτα από τα συνολικά 5.384.906, δηλαδή το 13,3% των πακέτων του δικτύου χάθηκαν στην μαύρη τρύπα.

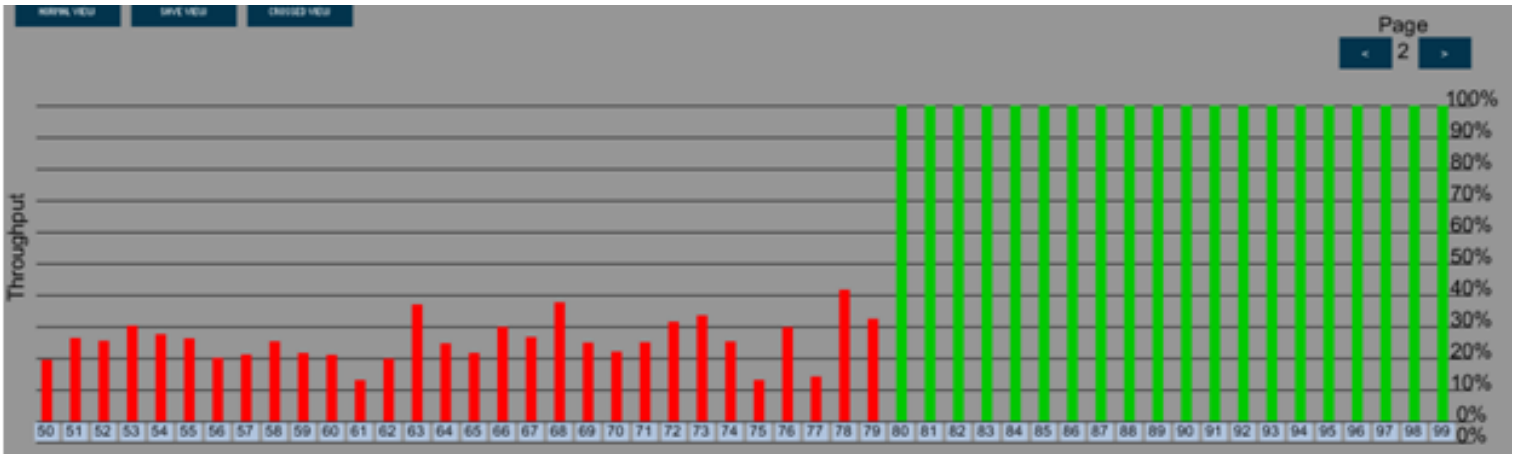


Εικόνα.41a: Save View στο τέλος της προσομοίωσης της MT.

Στο Crossed View βλέπουμε αναλυτικά την πτώση του throughput.

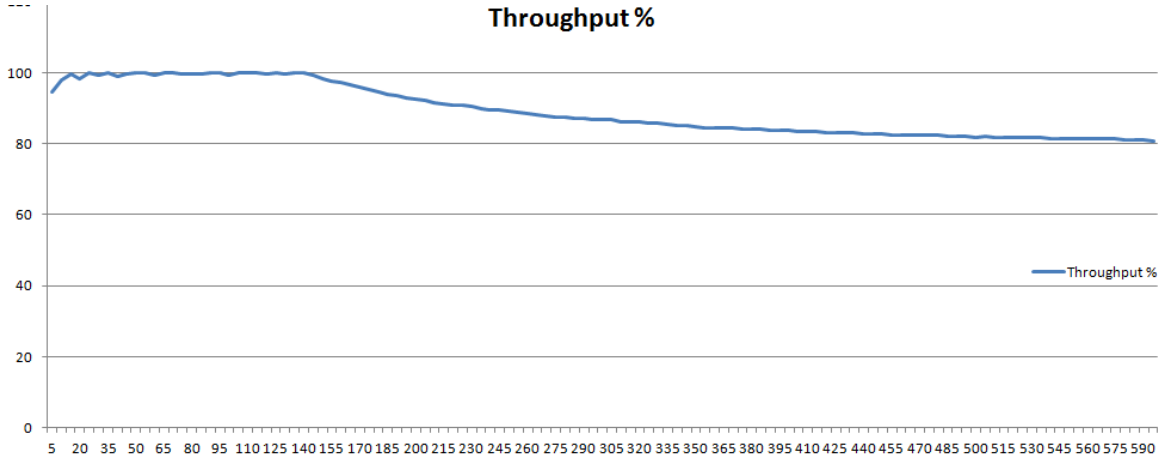


Εικόνα.41b: Crossed View στο τέλος της προσομοίωσης



Εικόνα 41c: Crossed View στο τέλος της προσομοίωσης

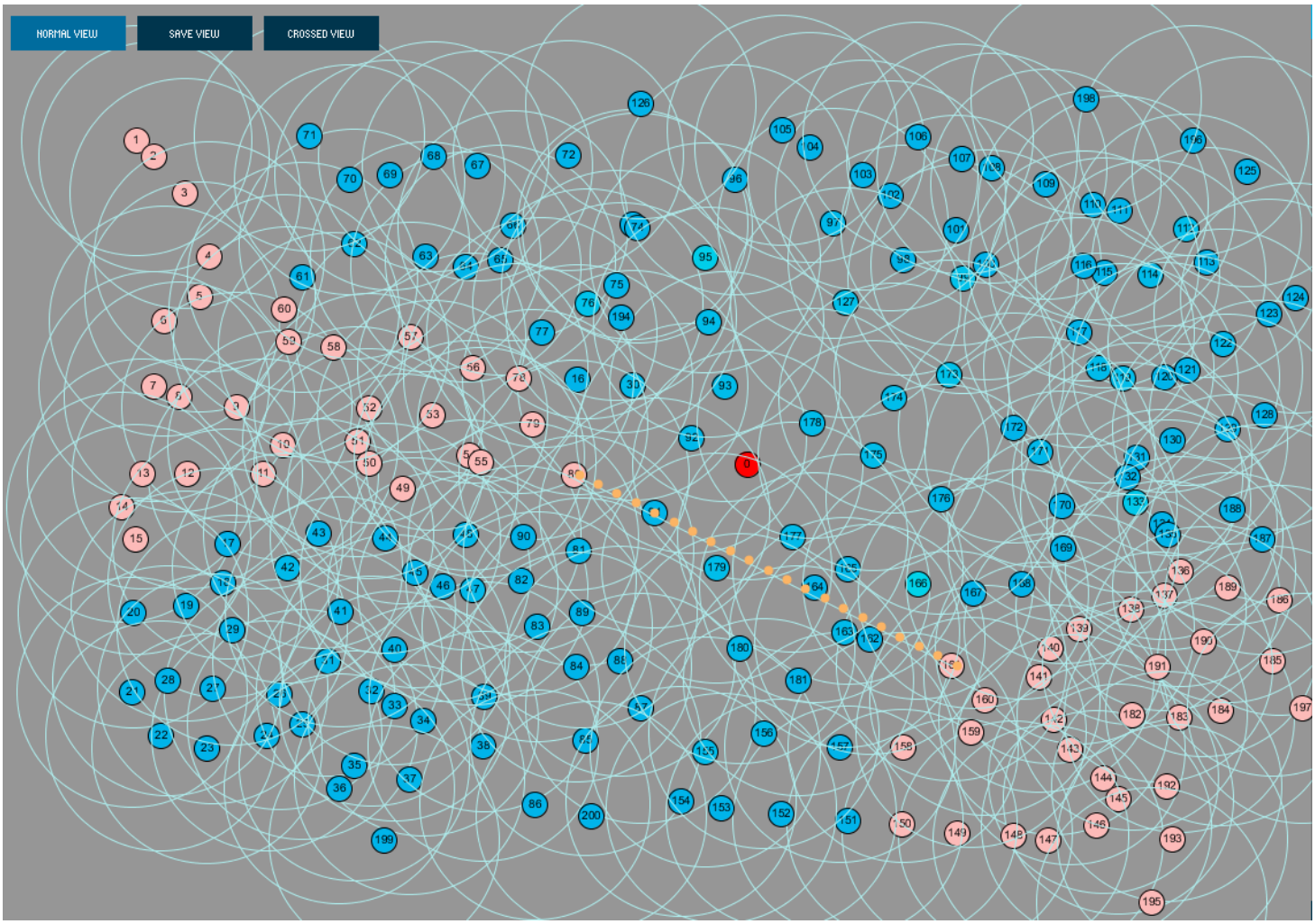
Τέλος στο παρακάτω γράφημα παρατηρούμε την μείωση του throughput κατά τη διάρκεια της 10λεπτης προσομοίωσης.



Εικόνα.42: Μείωση ρυθμαπόδοσης κατά τη διάρκεια επίθεσης MT

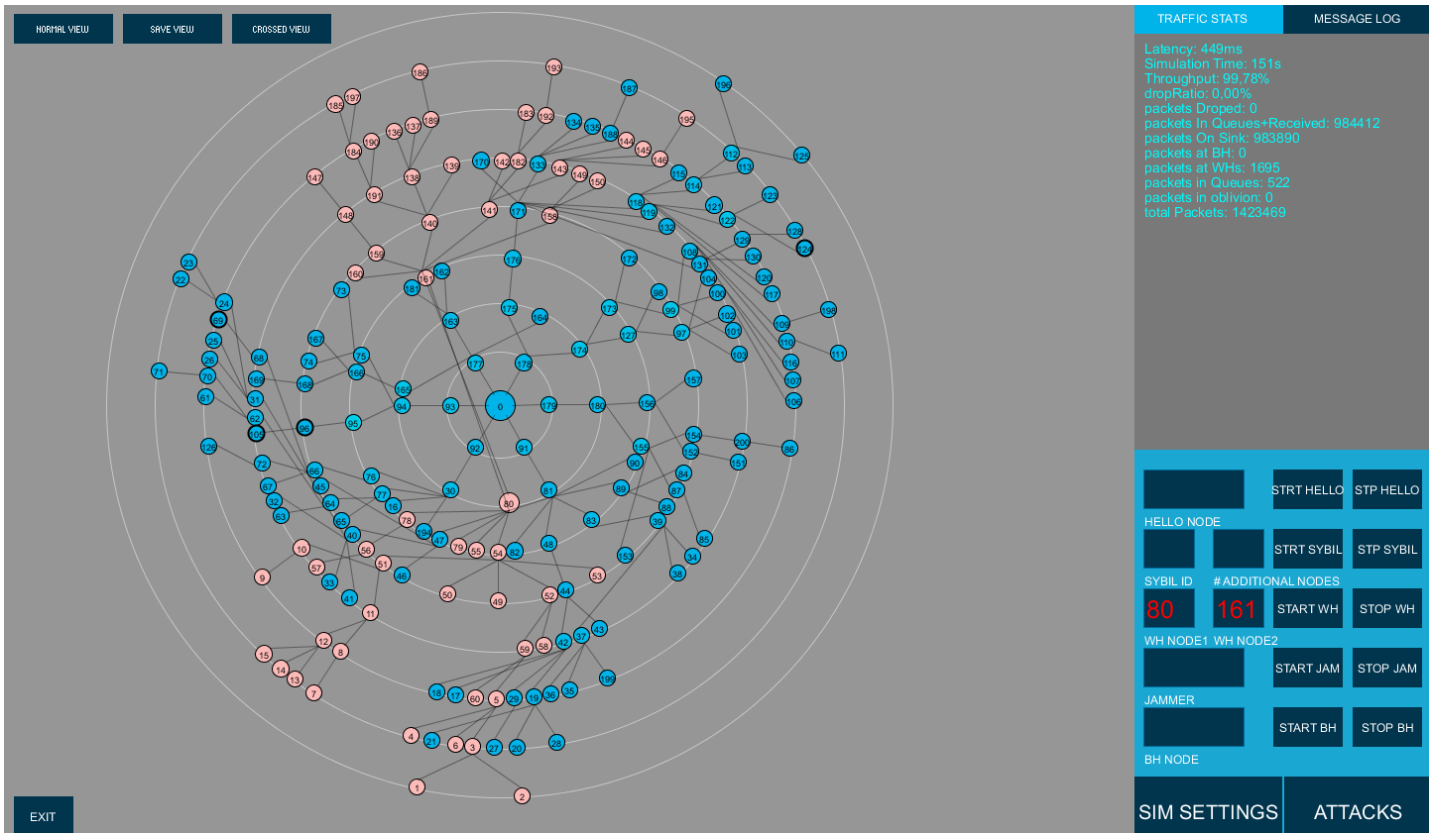
Επίθεση σκουληκότρυπας

Η επόμενη επίθεση που θα πραγματοποιήσουμε θα είναι η σκουληκότρυπα (wormhole). Οι παράμετροι τις προσομοίωσης είναι οι ίδιες με πριν. Επιλέγουμε στρατηγικά ως άκρα της σκουληκότρυπας τους κόμβους 80 και 161. Οι κόμβοι αυτοί βρίσκονται σε τελείως αντίθετα μέρη του δικτύου και έχουν αρκετούς κόμβους να δρομολογούν δια μέσω τους. Με την επίθεση αυτή οι χρωματισμένοι κόμβοι θα δρομολογούν τα πακέτα τους μέσω της σκουληκότρυπας και αυτά δεν θα φτάνουν ποτέ στον προορισμό τους.

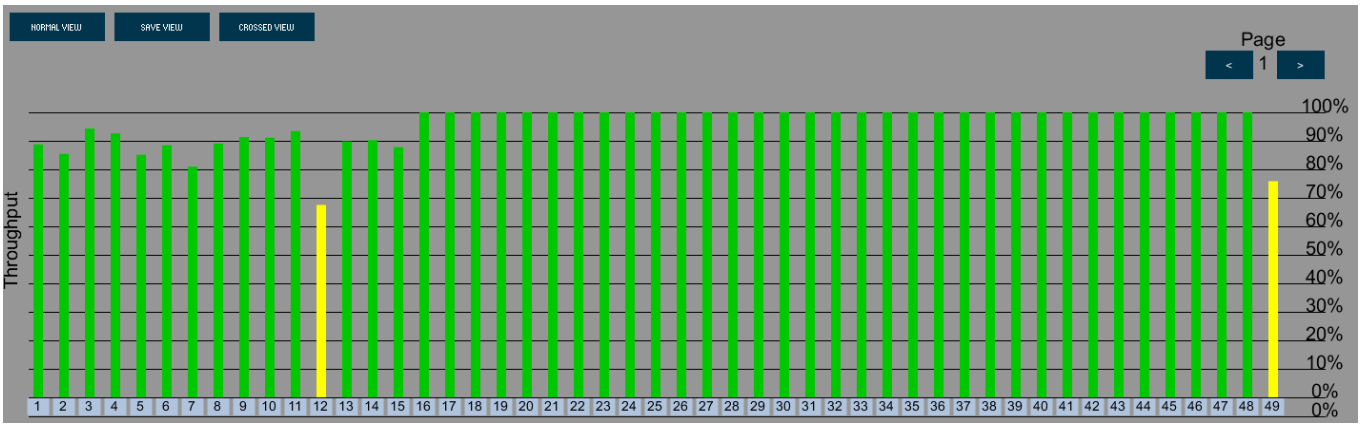


Εικόνα. 43a: Normal View επίθεση σκουληκότρυπας

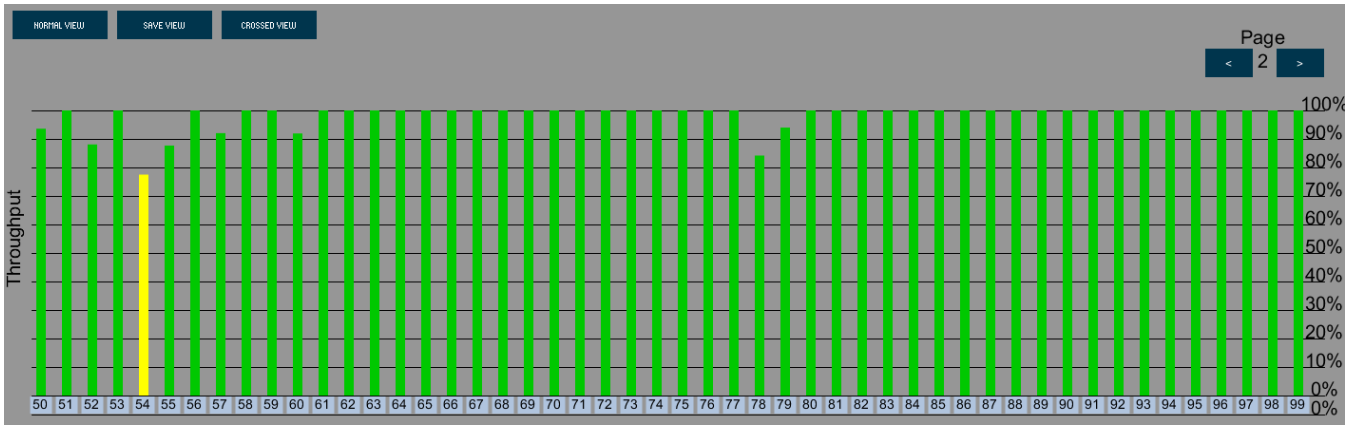
Στην επόμενη εικόνα φαίνεται η διάταξη του δικτύου κατά την εκκίνηση της επίθεσης. Οι χρωματισμένοι κόμβοι είναι αυτοί που έχουν επηρεαστεί. Όπως είπαμε στα προηγούμενα κεφάλαια η επίθεση σκουληκότρυπας είναι ένα είδος άρνησης υπηρεσίας, επομένως αναμένουμε να δούμε πτώση στο throughput των κόμβων. Η σκουληκότρυπα και η μαύρη τρύπα έχουν σχεδόν τις ίδιες επιπτώσεις στο δικτύο με την διαφορά ότι η πρώτη είναι πιο ισχυρή γιατί μπορεί να επιρρεάσει μεγαλύτερο ποσοστό των κόμβων. Θα μπορούσαμε εύκολα να αυξήσουμε ακόμα περισσότερο την επίδραση αν θέταμε το κόστος των WH Nodes σε 1. Με αυτόν τον τρόπο θα δελάζαμε περισσότερους κόμβους.



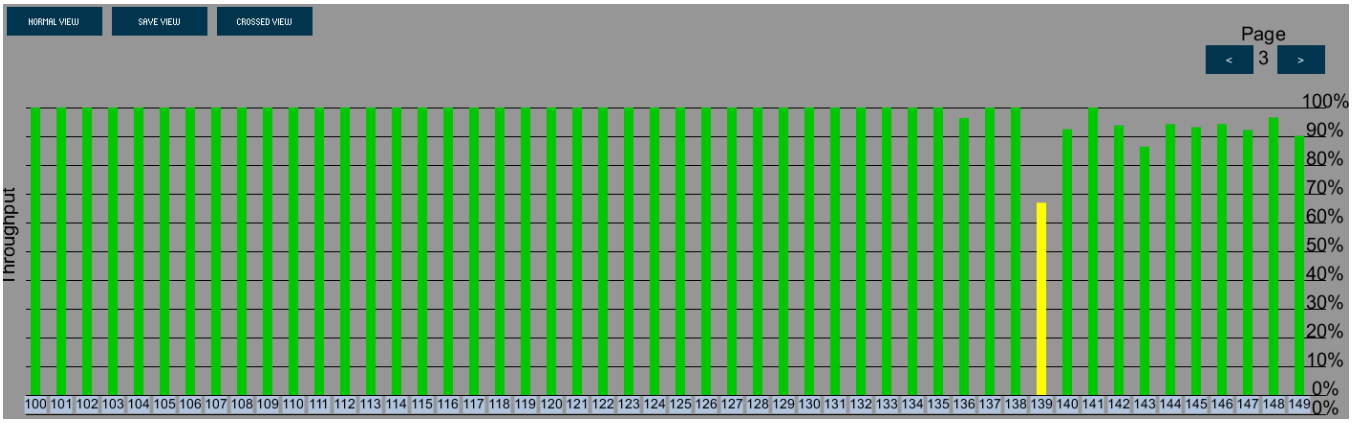
Εικόνα.43b: Απεικόνιση της επίθεσης σκουληκότρυπας στο Save View. Αρχή επίθεσης



Εικόνα 43c. Crossed View: Αναλυτική πτώση throughput. Αρχή επίθεσης

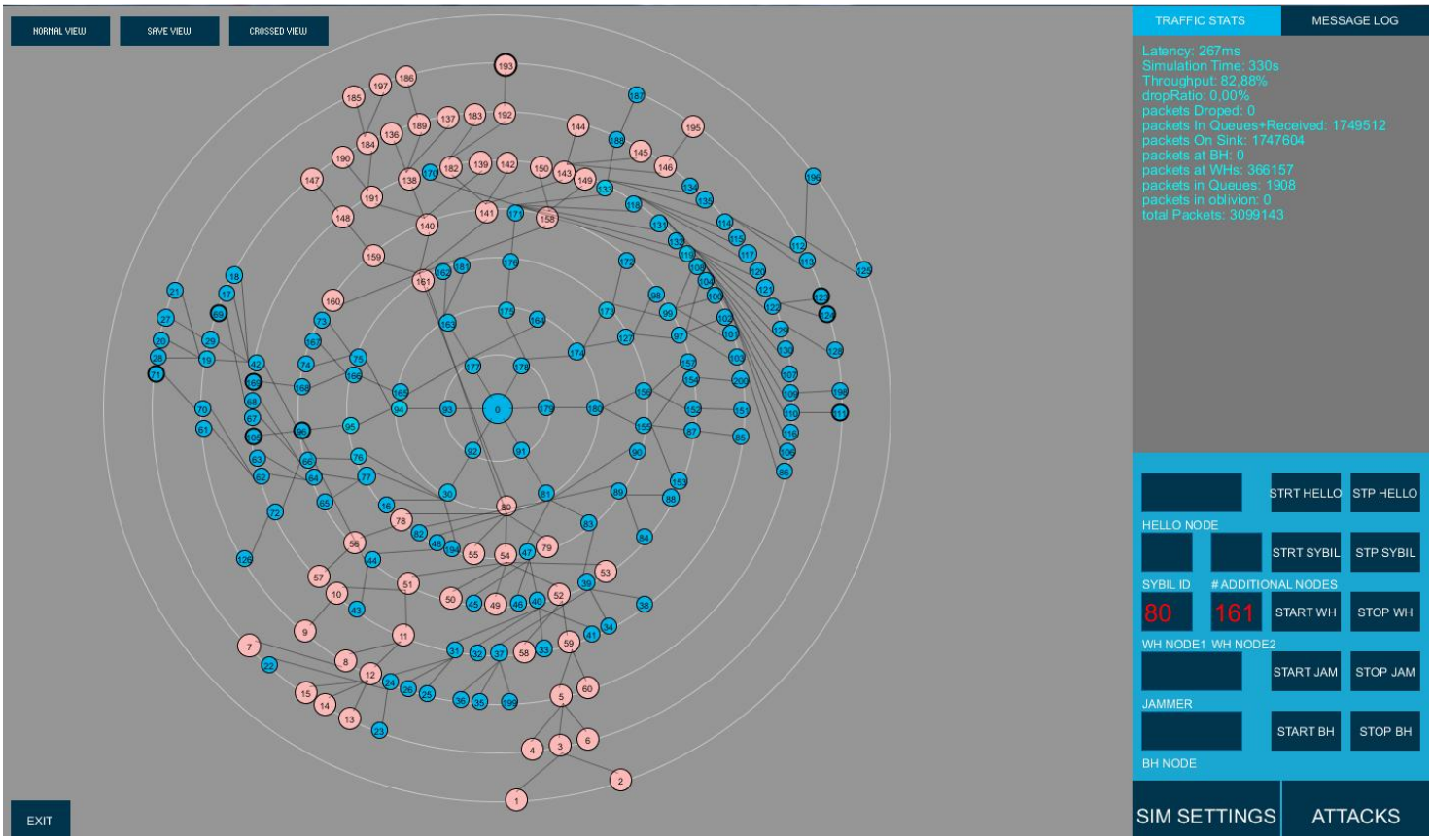


Εικόνα.43d: Crossed View: Αναλυτική πτώση throughput. Αρχή επίθεσης

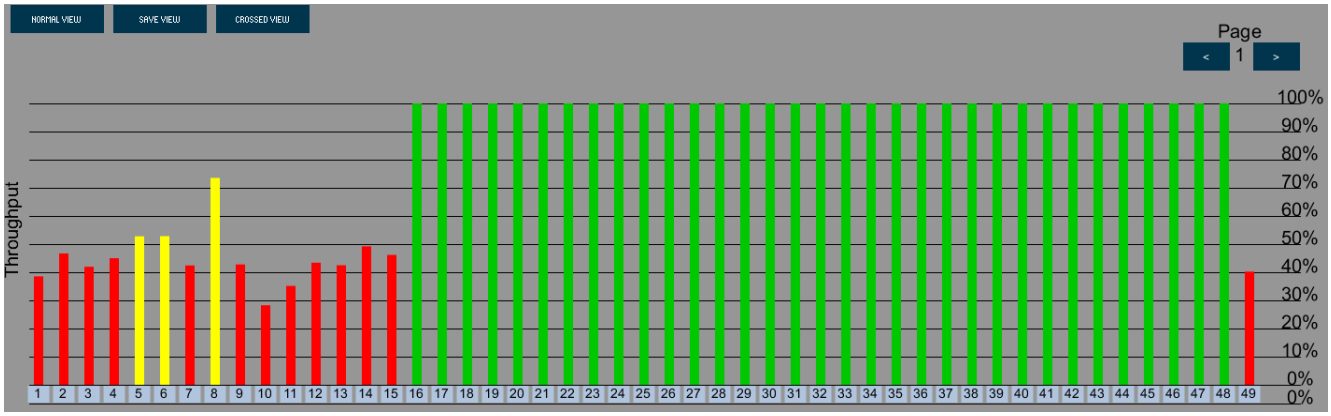


Εικόνα.43e: Crossed View: Αναλυτική πτώση throughput. Αρχή επίθεσης

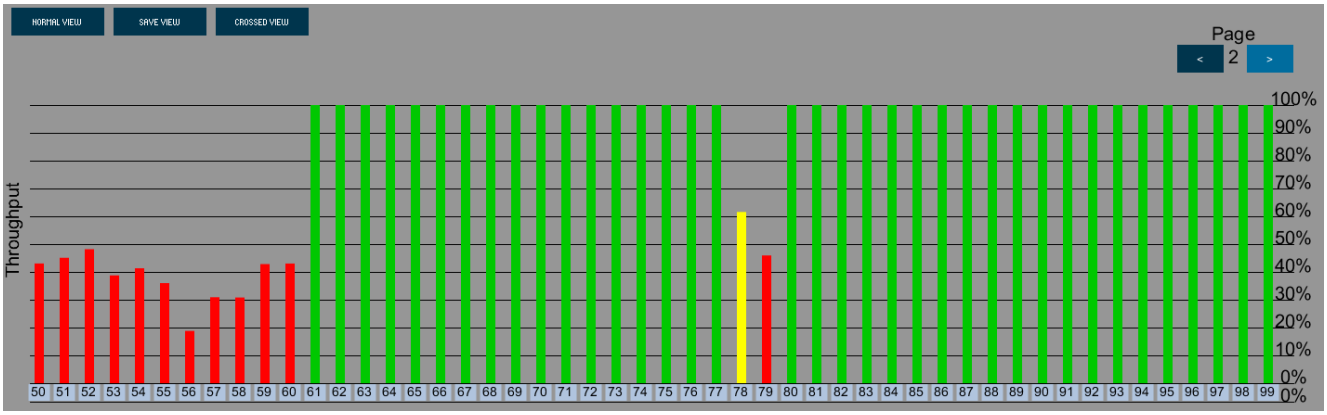
Στη συνέχεια παρατηρούμε την εικόνα του δικτύου 3 λεπτά από την εκκίνηση της επίθεσης. Το throughput έχει μειωθεί στο 82.33% και 366.157 από τα συνολικά 3.099.143 πακέτα ταξιδεύουν στην σκουληκότρυπα. Αξίζει να σημειωθεί ότι το latency δεν αυξάνεται γιατί υπολογίζεται από τον κόμβο 0 όταν καταφθάνουν σε αυτόν πακέτα συγκρίνοντας την ώρα άφιξης με την ώρα αποστολής. Λόγω της επίθεσης ο 0 δεν λαμβάνει πακέτα από τους επηρεασμένους κόμβους και συνεπώς δεν μπορεί να υπολογίσει το latency. Για αυτόν τον λόγο παρατηρούμε και αύξηση της ακτίνας των κόμβων.



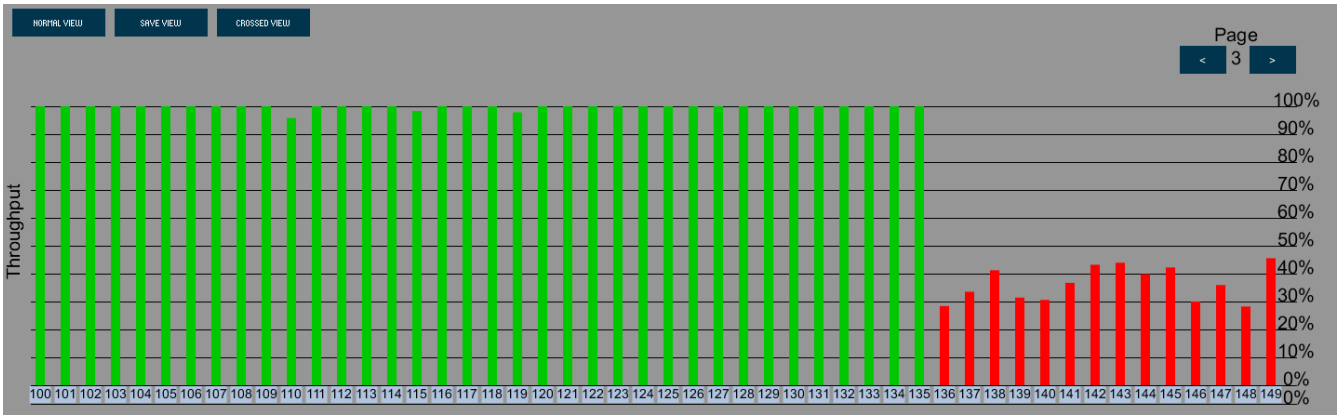
Εικόνα.44a: Save View, 3 λεπτά μετά την εμφάνιση ΣΤ



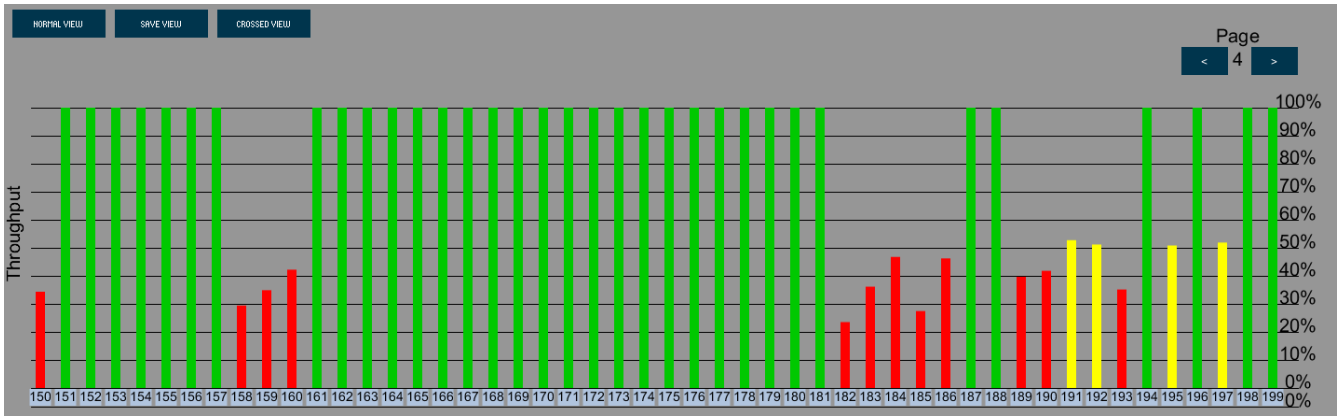
Εικόνα.44b : Crossed View, πτώση throughput 3 λεπτά από την εμφάνιση σκουληκότρυπας



Εικόνα.44c : Crossed View, πτώση throughput 3 λεπτά από την εμφάνιση σκουληκότρυπας

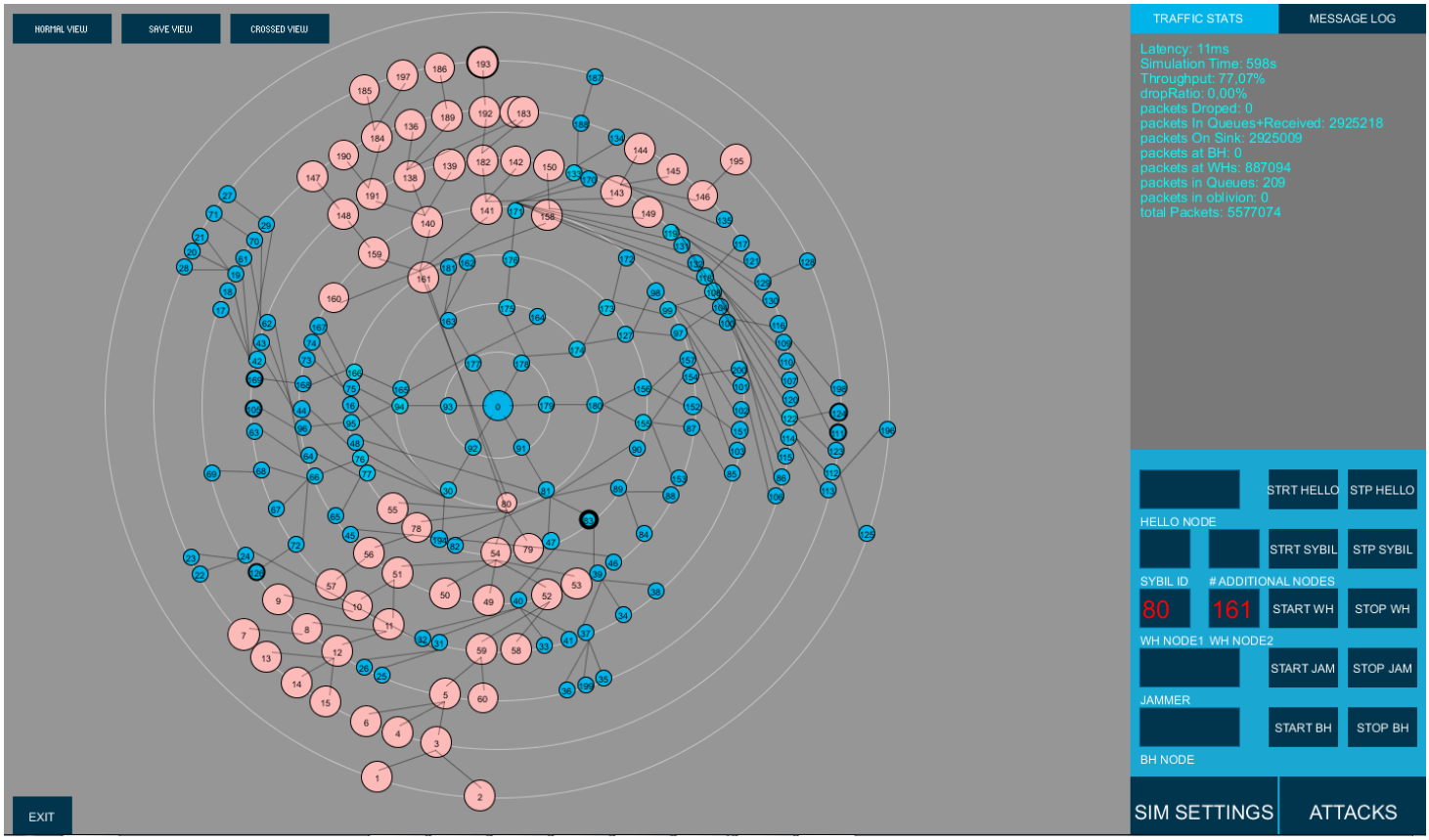


Εικόνα.44d: Crossed View, πτώση throughput 3 λεπτά από την εμφάνιση σκουληκότρυπας

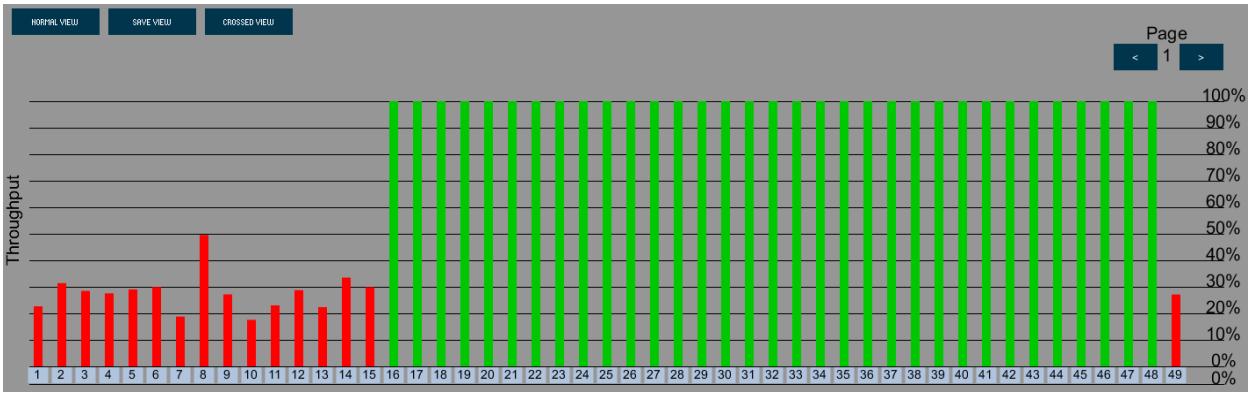


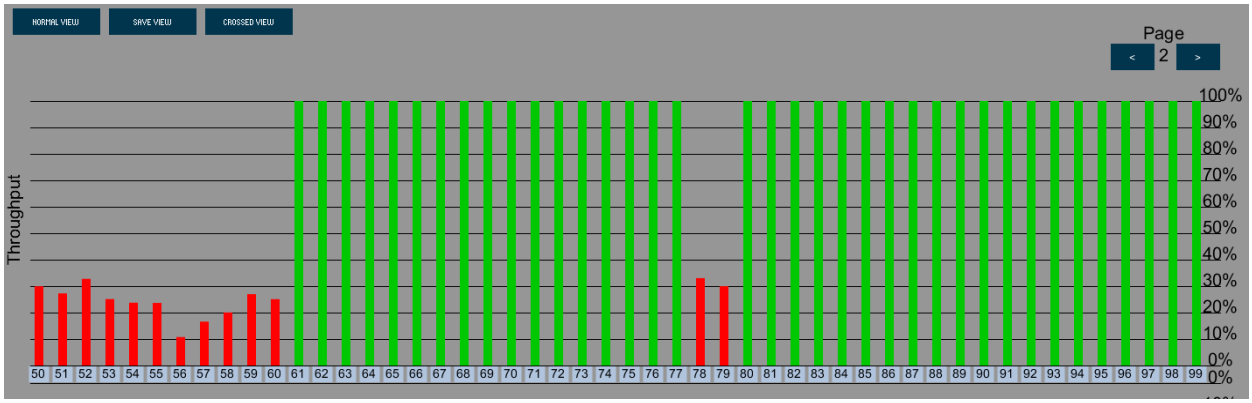
Εικόνα.44e: Crossed View, πτώση throughput 3 λεπτά από την εμφάνιση σκουληκότρυπας

Στο τέλος της προσομοίωσης η ΣΤ που δημιουργήθηκε από τους κόμβους 80 και 161 κατάφερε να μειώσει το συνολικό throughput του δικτύου στο 77% και να απομονώσει 887.094 από τα 5.577.074 πακέτα του δικτύου, κάτι λιγότερο από το 16% της συνολικής κίνησης.



Εικόνα.45a: Save View στο τέλος της προσομοίωσης

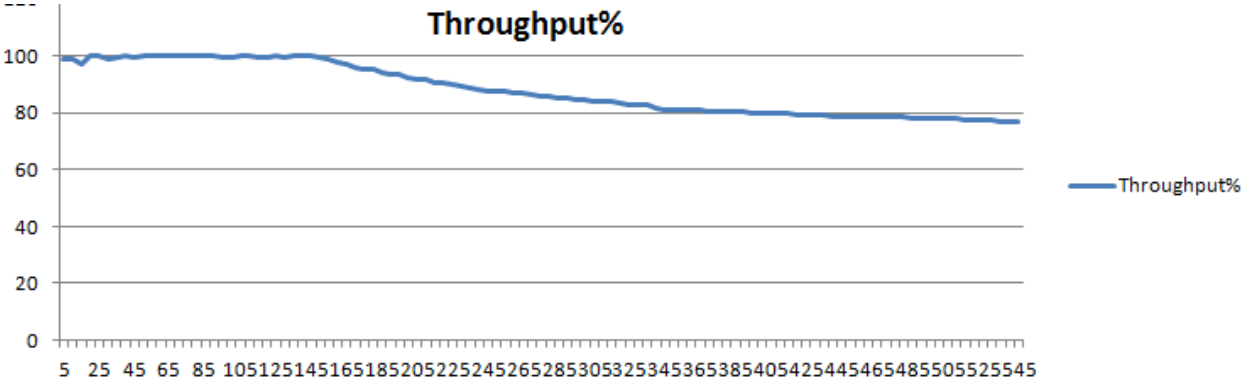




Εικόνα.45b,c: Crossed View στο τέλος της προσομοίωσης

Εικόνα.45d,e: Crossed View στο τέλος της προσομοίωσης.

Τέλος στο γράφημα παρουσιάζεται η πτώση του throughput κατά την διάρκεια της προσομοίωσης.

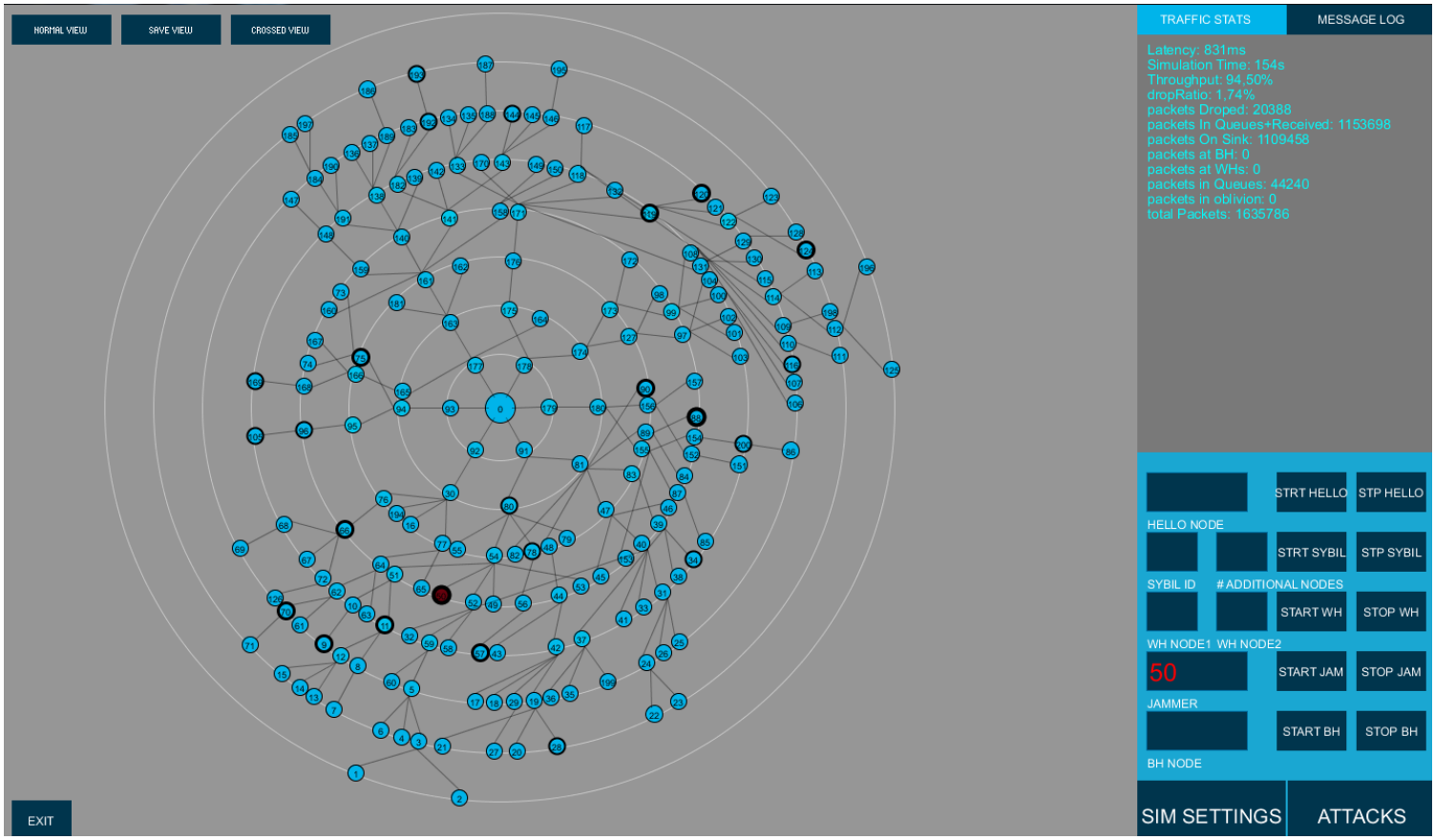


Εικόνα.45f: Πτώση throughput κατά τη διάρκεια της επίθεσης ΣΤ

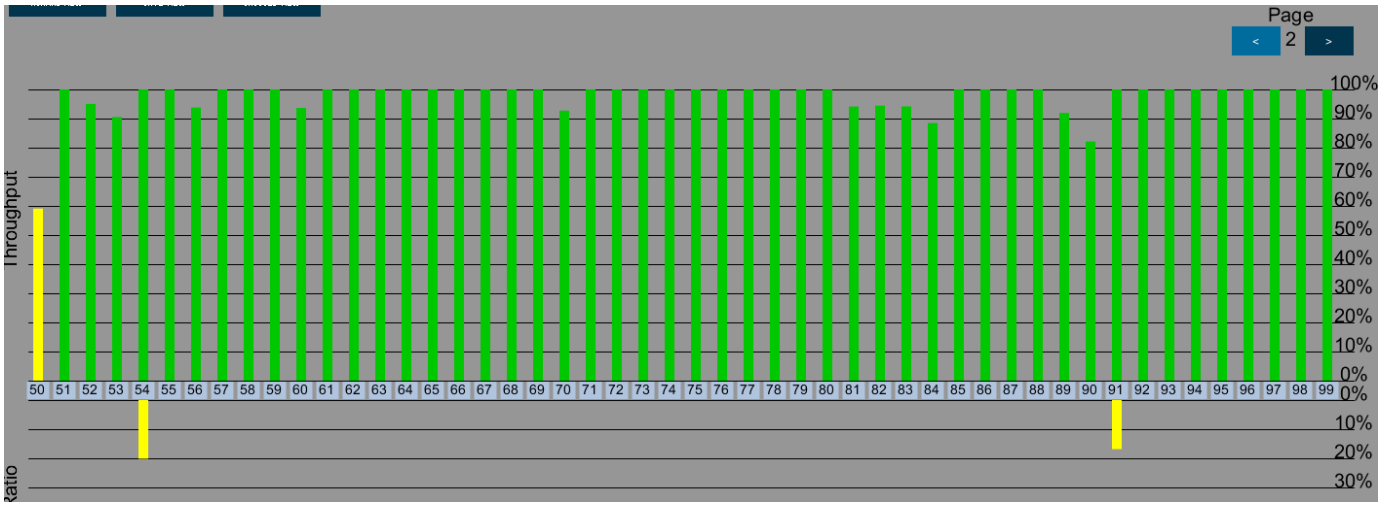
Επίθεση παρεμβολών

Η επόμενη επίθεση που θα παρουσιάσουμε είναι η επίθεση παρεμβολών. Η επίθεση αυτή είναι μία καταστροφική επίθεση ακόμα και αν πραγματοποιείται από έναν μόνο κόμβο. Ο κόμβος αυτός πλημμυρίζει τους γείτονες του με άχρηστα πακέτα αναγκάζοντας τους από την μία να μην μπορούν να μεταδώσουν τα δικά τους πακέτα και από την άλλη να απορρίπτουν πακέτα που δέχονται από άλλους κόμβους. Με την επίθεση αυτή αναμένουμε να δούμε μεγάλη αύξηση στο latency του δικτύου και στο drop ratio, όπως επίσης και μείωση του throughput. Οι παράμετροι τις προσομοίωσης παραμένουν ίδιες και ο jammer μας θα είναι ο κόμβος 50. Παρατηρούμε ότι μόλις 4

δευτερόλεπτα μετά την εμφάνιση του jammer το drop ratio έγινε 1,74% και το latency 831ms

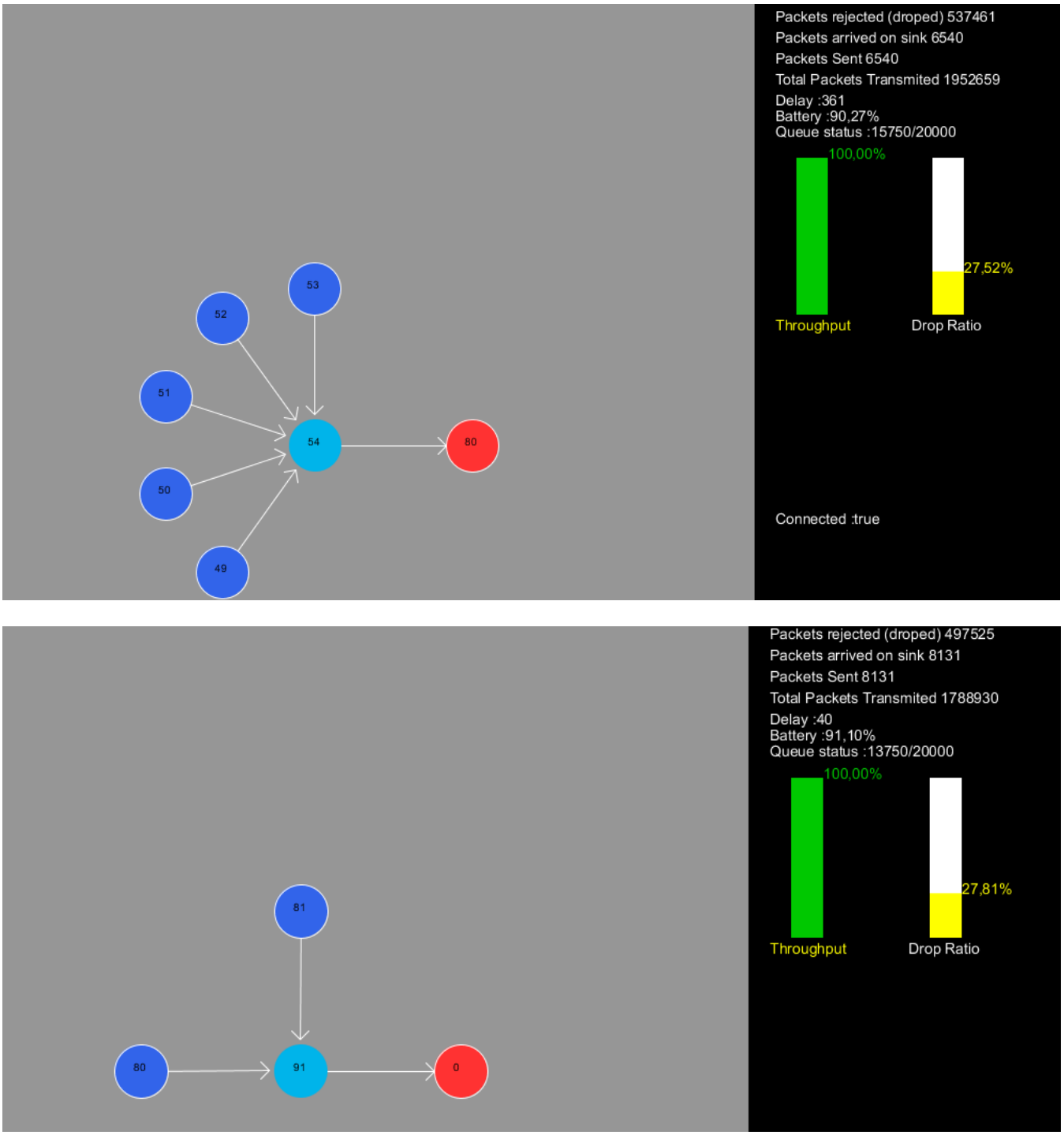


Εικόνα.46a: Save View παρουσία jammer. Αρχή της επίθεσης



Εικόνα.46b: Crossed View παρουσία jammer. Αρχή της επίθεσης

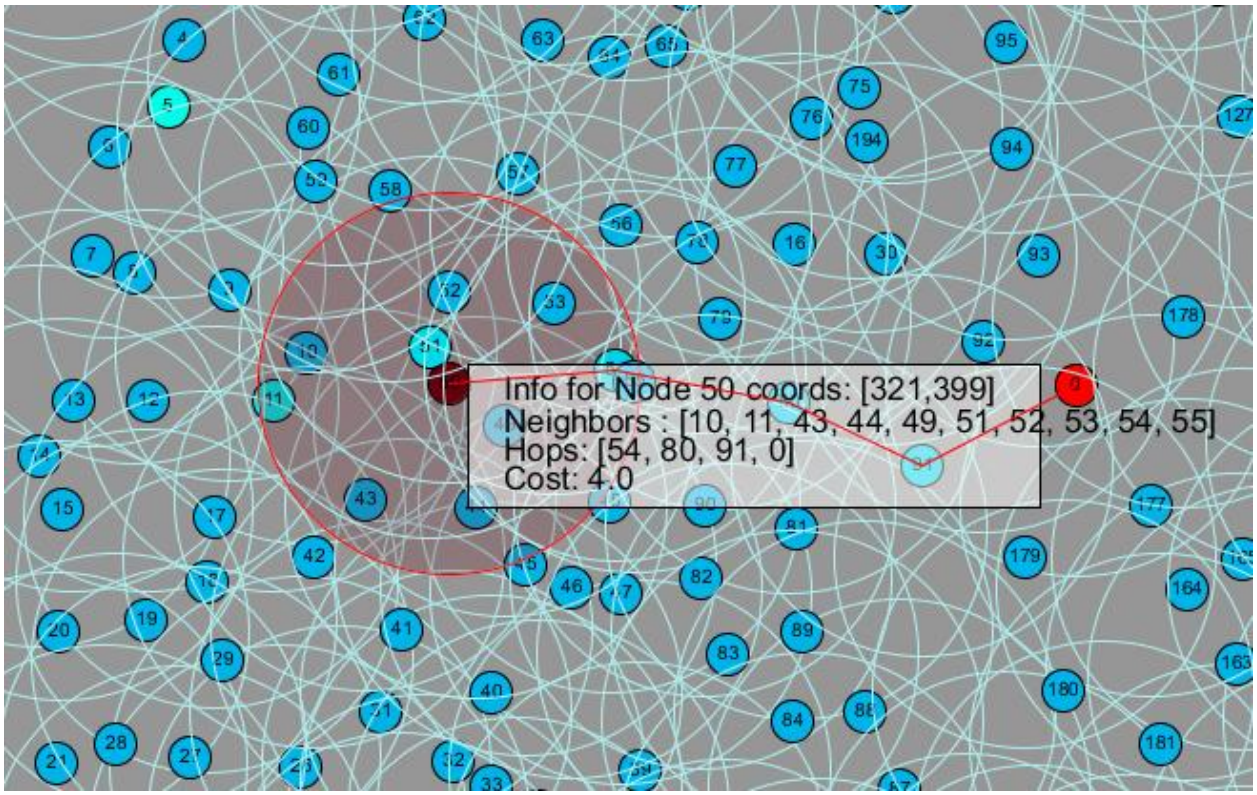
Μπορούμε επίσης να δούμε τι συμβαίνει στους κόμβους 54 και 91 οι οποίοι είναι gateway κόμβοι και αναγκάζονται να απορρίψουν πακέτα. Σε αυτό το σημείο αξίζει να σημειωθεί ότι τα πακέτα που απορρίπτονται δεν είναι μόνο κανονικά πακέτα, αλλά και ψεύτικα πακέτα που στέλνει ο jammer.



Εικόνα. 46c,d: Node View των κόμβων 54 και 91

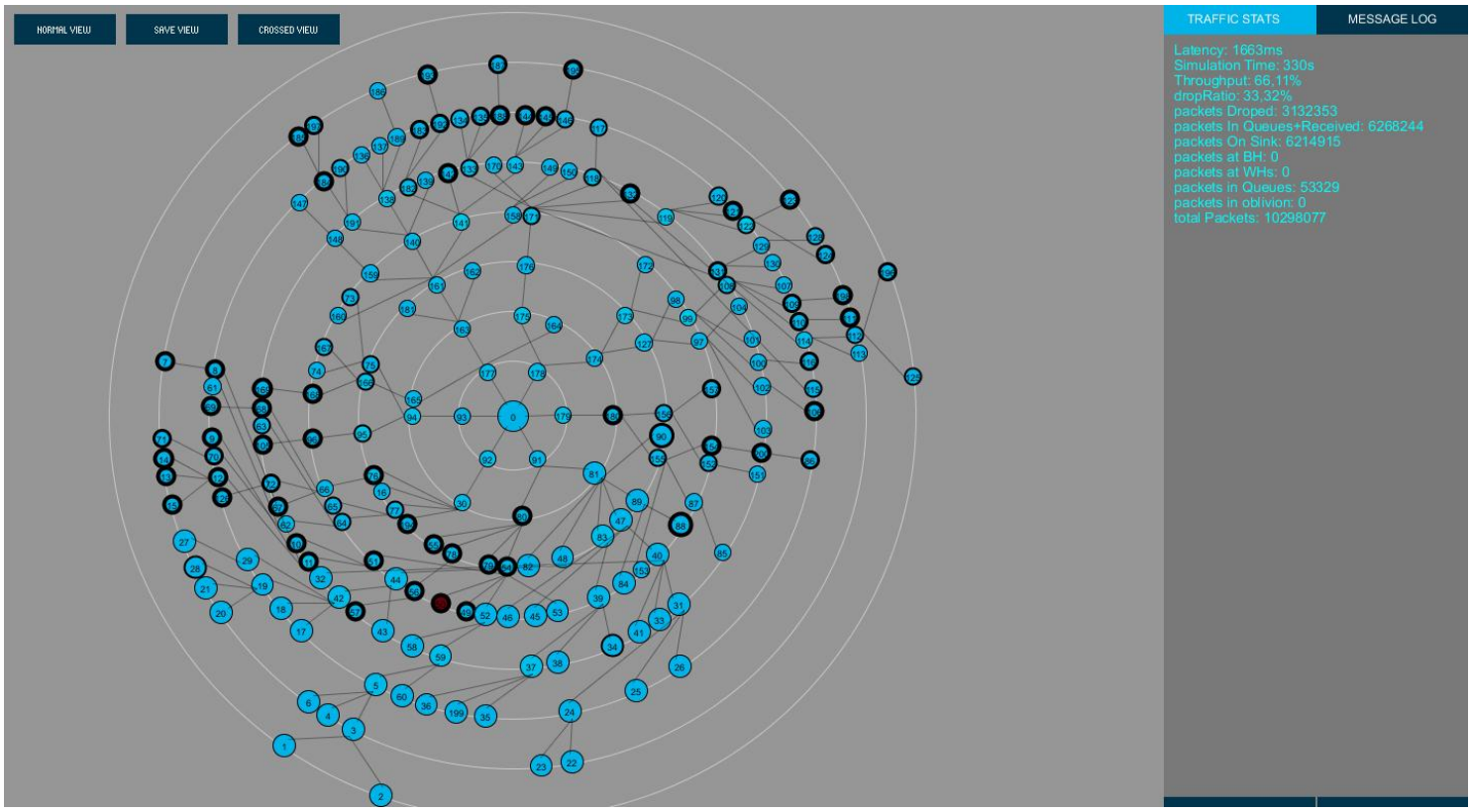
Μέσω του Node View φαίνονται ποιοι κόμβοι βρίσκονται στην εμβέλεια του jammer. Παρατηρούμε πως ο κόμβος 91 περιέχεται στο route του 50. Ταυτόχρονα ακόμη 69 κόμβοι δρομολογούν μέσω του 91, οπότε ένας σωστά τοποθετημένος jammer μπορεί

να εμποδίσει τις επικοινωνίες ενός μεγάλου μέρος του δικτύου, ακόμα και αν όλοι οι επηρεαζόμενοι κόμβοι δεν βρίσκονται εντός της εμβέλειας του. Τα ψεύτικα πακέτα που στέλνει θα μεταδοθούν από κόμβο σε κόμβο και θα πλημμυρίσουν το δίκτυο αν δεν υπάρχει κατάλληλος μηχανισμός ελέγχου.

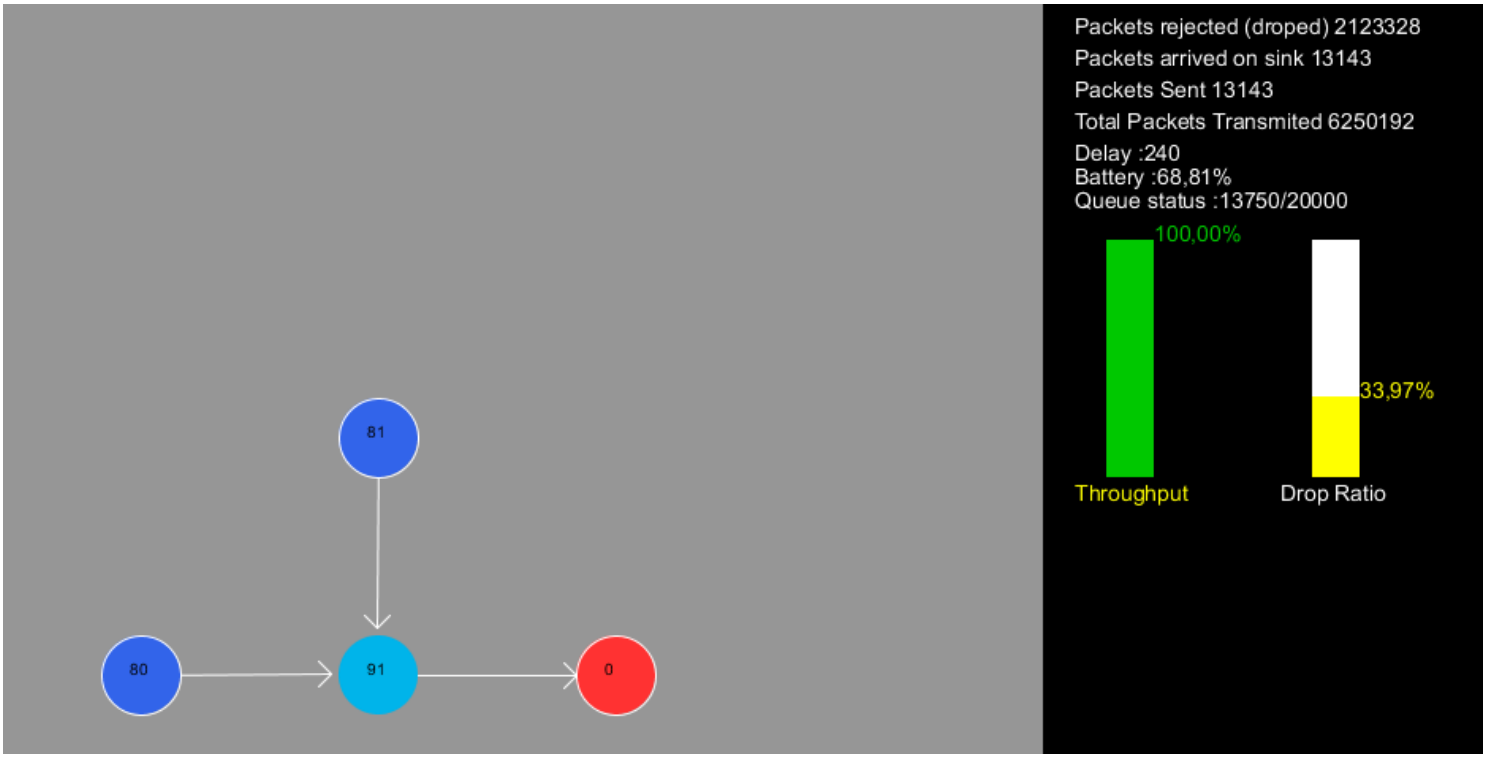


Εικόνα.46e: Normal View με jammer τον κόμβο 50

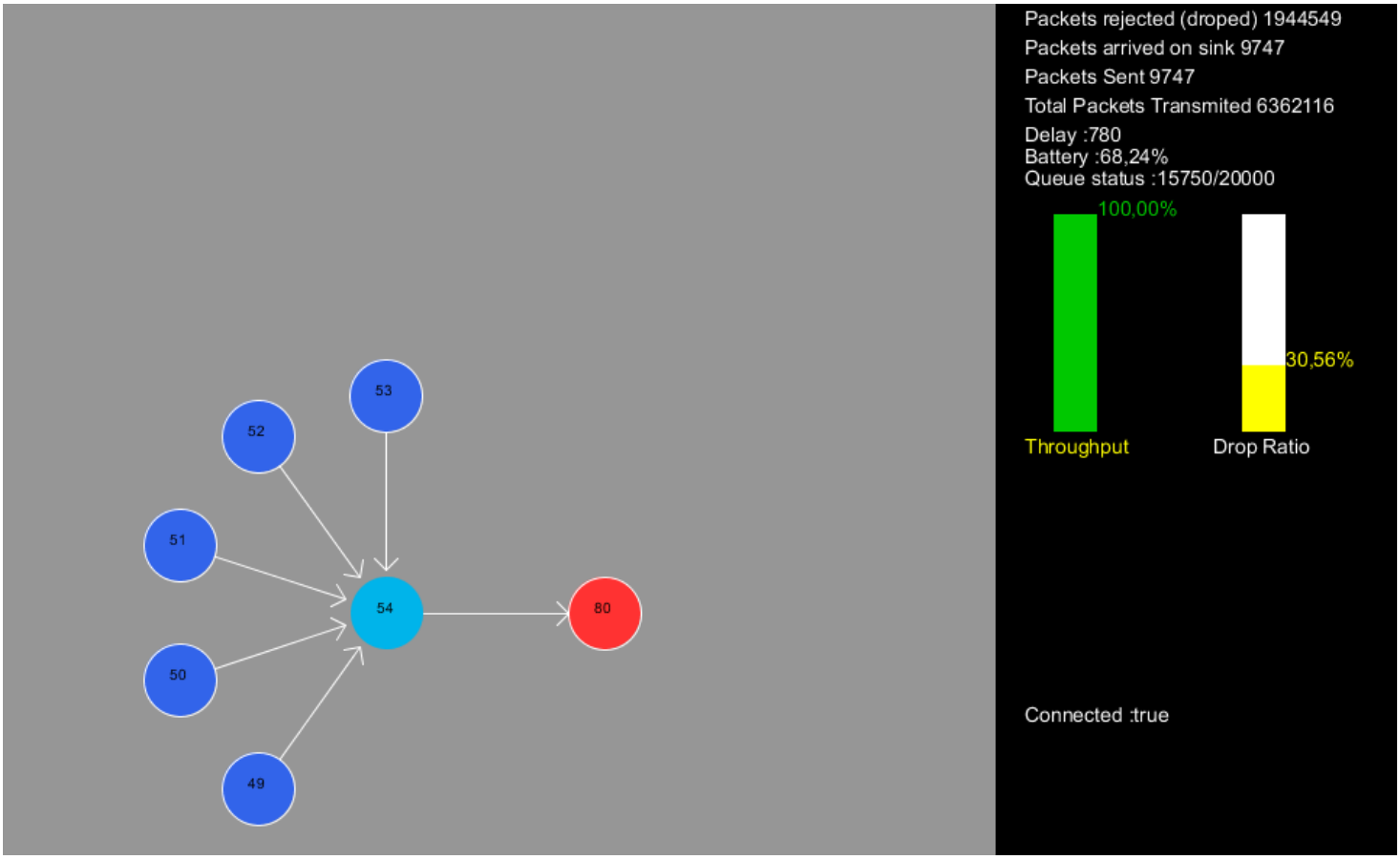
Στη συνέχεια απεικονίζεται το δίκτυο στο Save View, 3 λεπτά μετά την εμφάνιση του jammer. Το drop ratio και το latency έχουν αυξηθεί αισθητά στο 33% και 1663ms αντίστοιχα. Επίσης υπάρχει αρκετά μεγάλη μείωση του throughput γιατί κόμβοι που βρίσκονται πίσω από τον jammer αδυνατούν να επικοινωνήσουν με τον 0 λόγω της μεγάλης συμφόρησης. Τέλος σε αρκετούς κόμβους η επικοινωνία έχει διακοπεί τελείως.



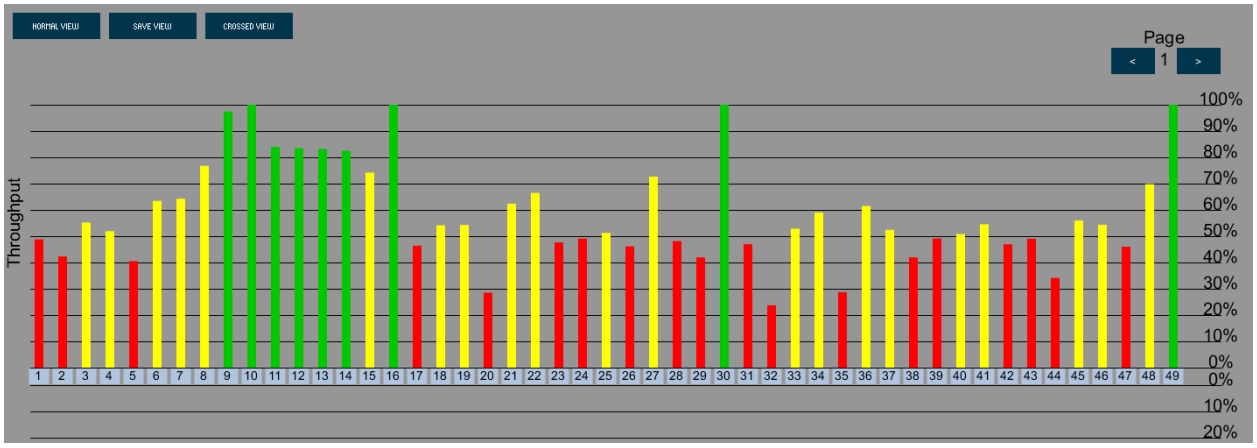
Εικόνα.47a: Save View 3 λεπτά μετά την εμφάνιση του jammer



Εικόνα.47b: Node 91 View 3 λεπτά μετά την εμφάνιση του jammer

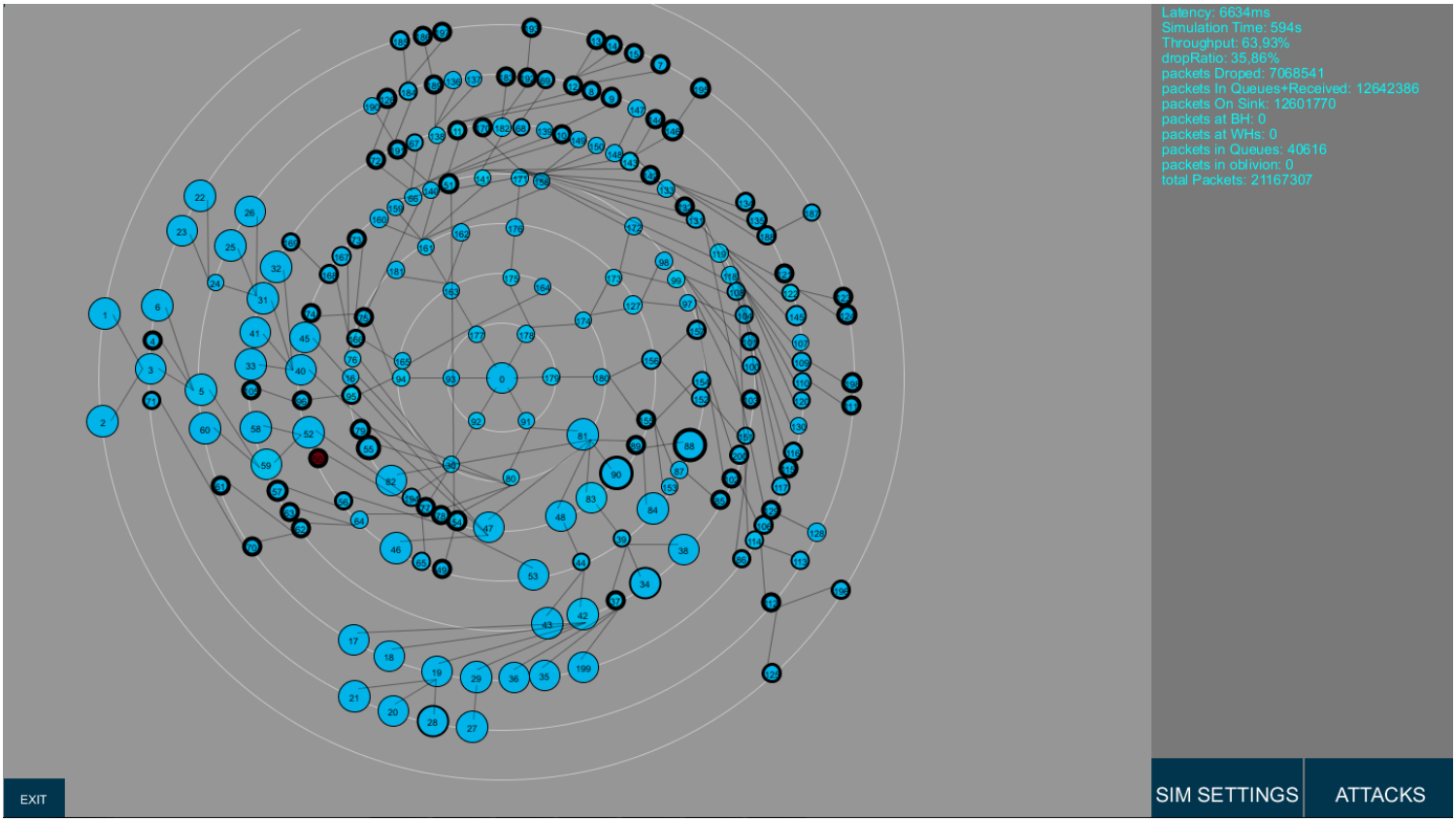


Εικόνα.47c: Node 54 View 3 λεπτά μετά την εμφάνιση του jammer



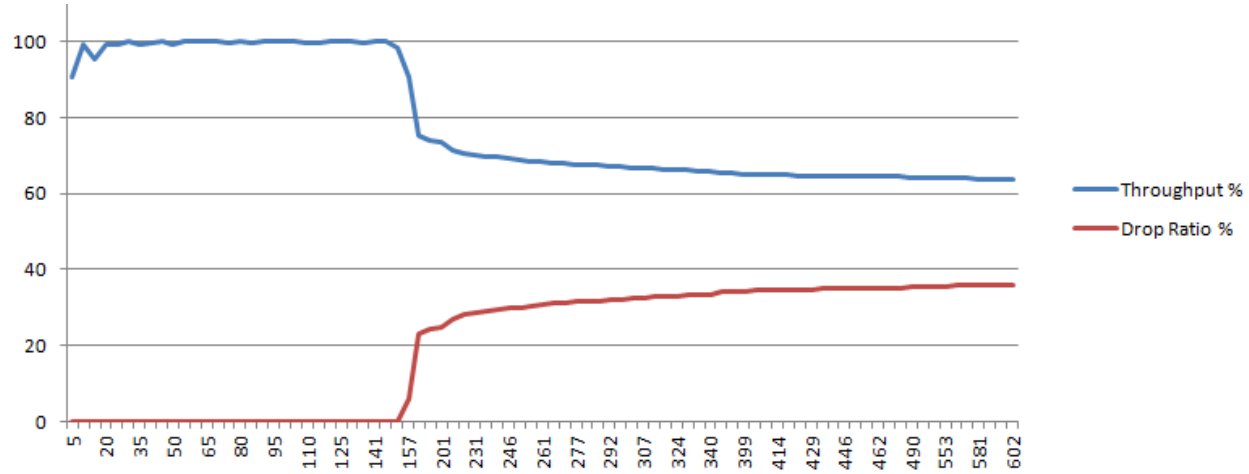
Εικόνα.47d: Crossed View 3 λεπτά μετά την εμφάνιση του jammer

Εικόνα.47e: Crossed View 3 λεπτά μετά την εμφάνιση του jammer

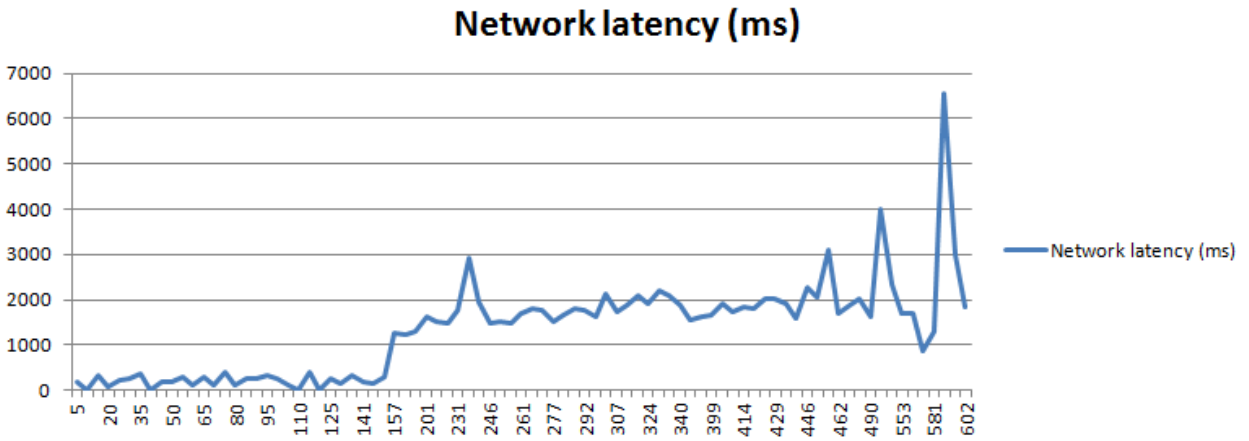


Εικόνα.48: Save View: Διάταξη του δικτύου στο τέλος της προσομοίωσης

Στην εικόνα 48 παρουσιάζεται η τελική διάταξη του δικτύου μόλις τελείωσει η προσομοίωση της επίθεσης jamming. Βλέπουμε στα στατιστικά ότι το latency εκτοξεύθηκε στα 6634ms και ότι ένα πολύ σημαντικό μέρος (~36%) του συνόλου των πακέτων απορρίφθηκε, αυτό μεταφράζεται σε 7.068.541 από τα 21.167.307 πακέτα. Υπενθυμίζουμε ότι ο αριθμός των πακέτων είναι μεγάλος γιατί προσμετρούνται και τα ψεύτικα πακέτα του jammer. Τέλος το throughput μειώθηκε στο 63.93% και διακόπηκε τελείως η επικοινωνία από 44 κόμβους. Στα παρακάτω γραφήματα φαίνονται αναλυτικά τα στατιστικά που μας ενδιαφέρουν:



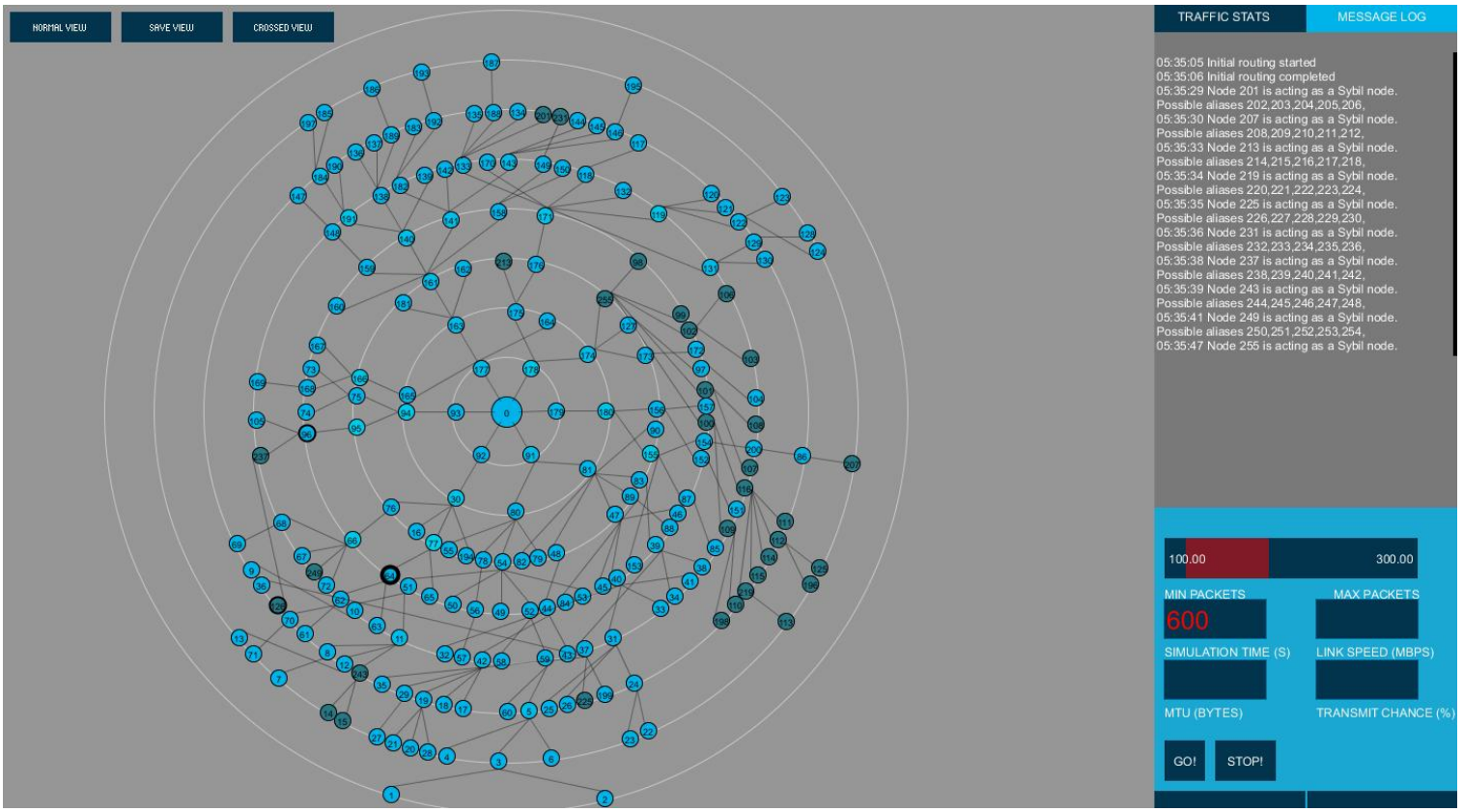
Εικόνα.49a,: Γράφημα ρυθμαπόδοσης και ποσοστού απόρριψης πακέτων κατά τη διάρκεια επίθεσης παρεμβολών



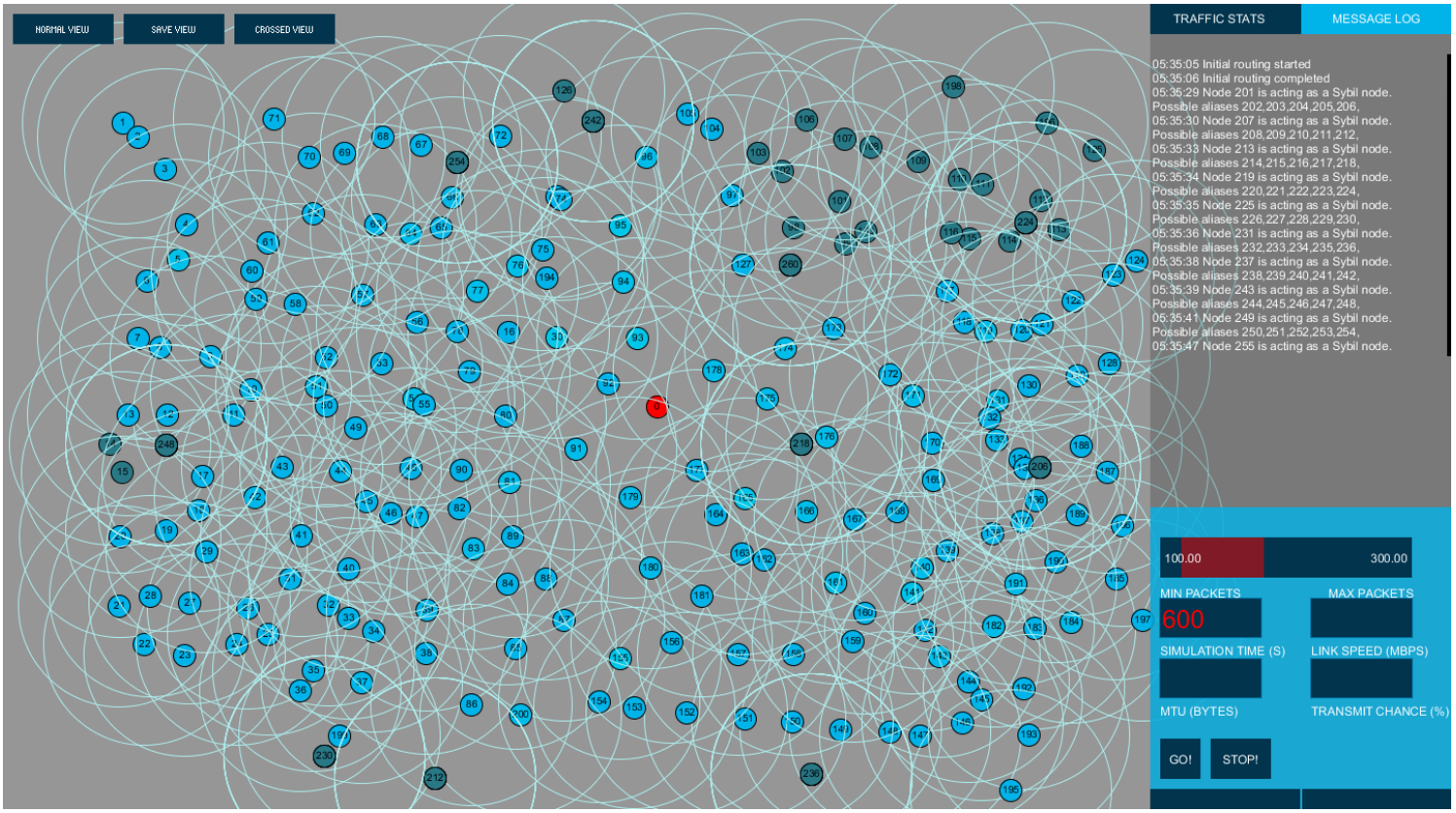
Εικόνα.49,b: Γράφημα μέσης καθηστέρισης του δικτύου κατά τη διάρκεια επίθεσης παρεμβολών

Επίθεση Sybil

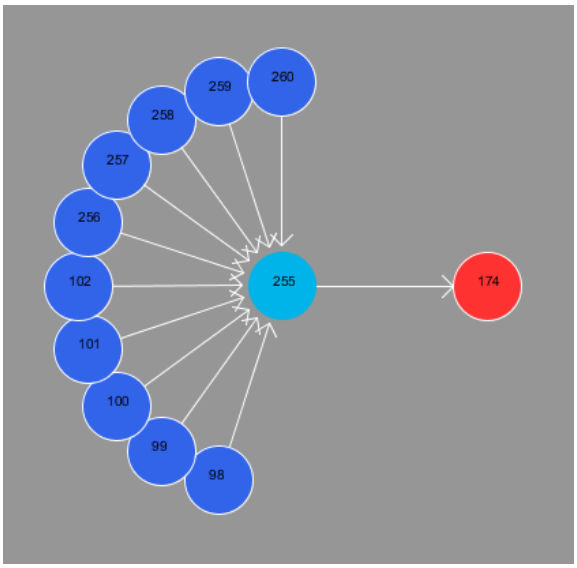
Η επίθεση Sybil είναι η προτελευταία επίθεση που θα εξετάσουμε. Κατά την διάρκεια της επίθεσης προστίθενται επιπλέον κόμβοι στο δίκτυο. Αυτοί οι κόμβοι ισχυρίζονται πως έχουν γείτονες κάποιους άλλους κόμβους που όμως δεν είναι ορατοί από τους υγιείς κόμβους. Εκτός από καινούριους κόμβους η επίθεση είναι δυνατόν να εκτελεστεί και από προσβεβλημένους υπάρχοντες κόμβους του δικτύου. Η επίθεση αυτής της μορφής δεν επηρεάζει τα στατιστικά όπως οι προηγούμενες αλλά υπάρχει η δυνατότητα οι Sybil Nodes να τροποποιήσουν πακέτα που λαμβάνουν ή να εισάγουν πακέτα με ψεύτικα δεδομένα στο δίκτυο. Στις επόμενες εικόνες φαίνεται η επίθεση Sybil όπως εμφανίζεται στα Save και Normal View.

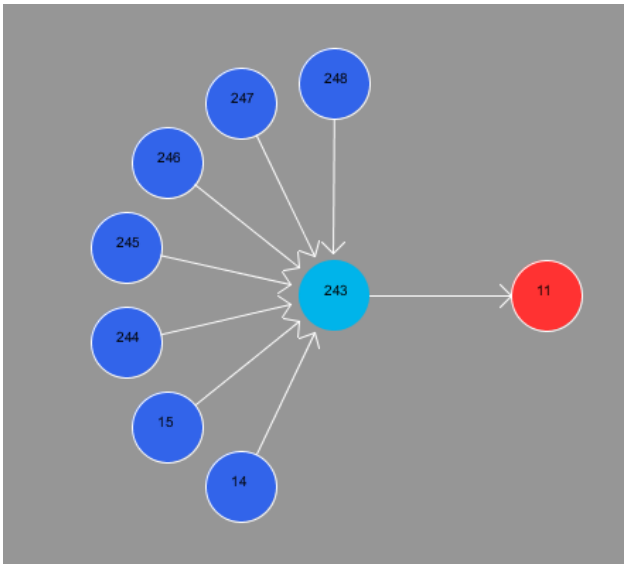


Εικόνα.50a: Save View απεικόνιση της επίθεσης Sybil.

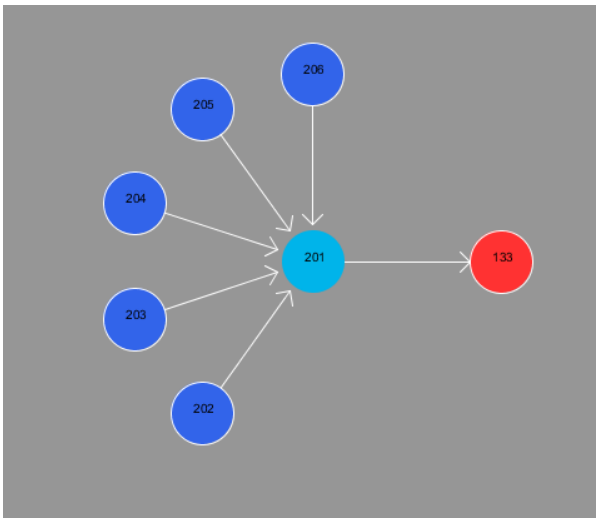


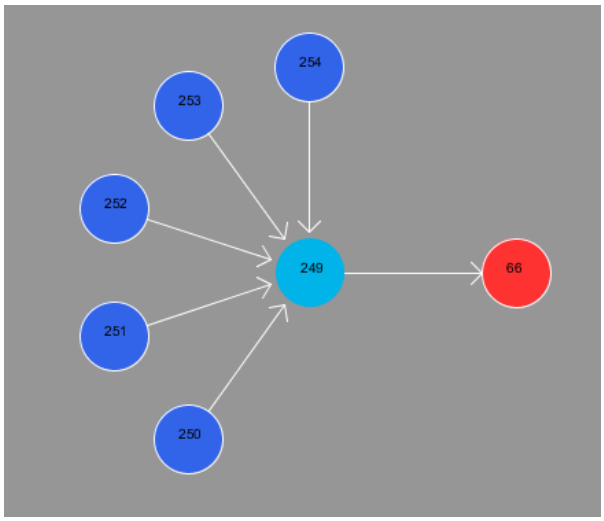
Εικόνα50b: Normal View φυσικές θέσεις των Sybil nodes στο δίκτυο. Τα Sybil Ghosts βρίσκονται στις ίδιες συντεταγμένες με τους κόμβους που τα δημιούργησαν



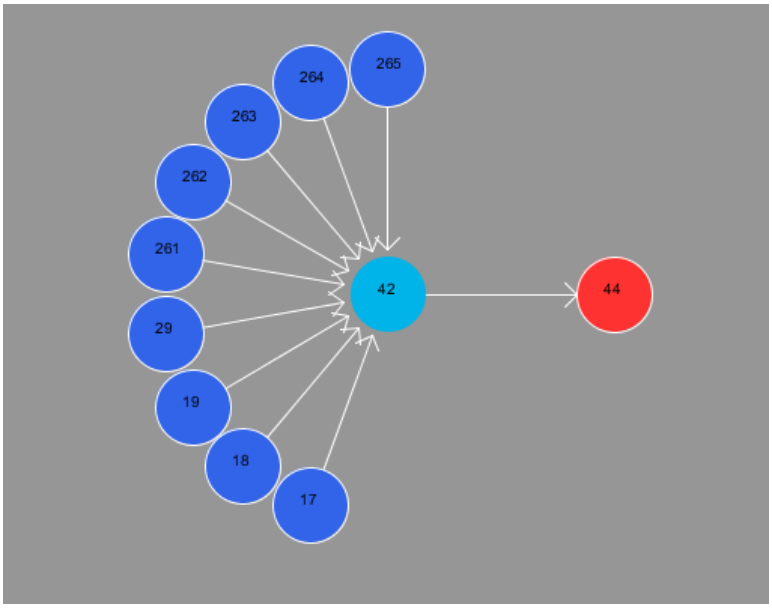


Εικόνα.51a,b: Sybil Nodes με τα Sybil Ghosts τους, αλλά και κανονικούς κόμβους ως παιδιά.





Εικόνα. 51c,d: Sybil Nodes με τα Sybil Ghosts τους χωρίς άλλους κόμβους ως παιδιά..

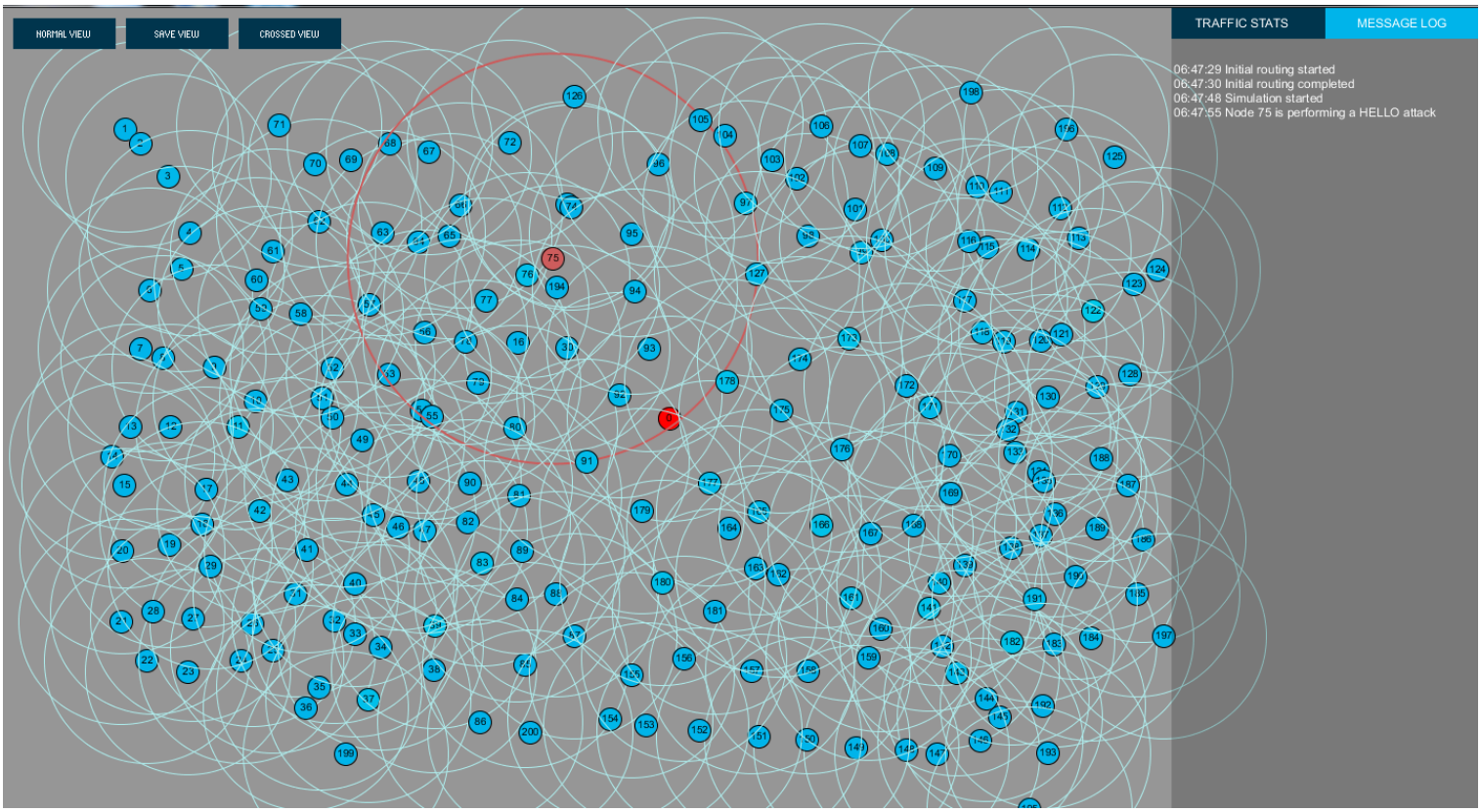


Εικόνα 51e: Πρώην κανονικός κόμβος που παρουσιάζει Sybil Ghosts. Έχει επίσης ως παιδιά του 4 ακόμα κόμβους.

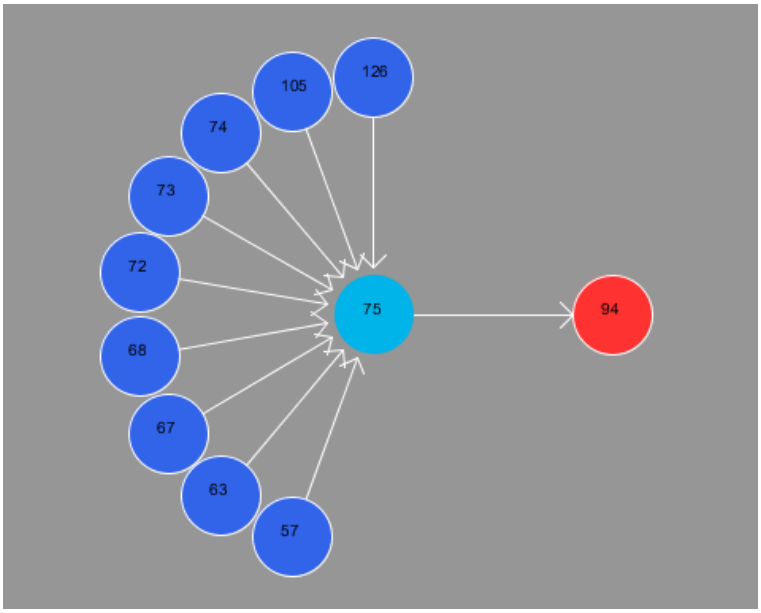
Επίθεση Hello

Για το τέλος αφήσαμε την επίθεση hello. Θεωρούμε πως ο επιτιθέμενος έχει στην κατοχή του μία συσκευή με μεγάλη εμβέλεια μετάδοσης και μεταδίδει συνεχώς hello πακέτα στους κόμβους που βρίσκονται εντός αυτής. Η επίδραση της επίθεσης είναι διπλή. Αρχικά οι κόμβοι ξοδεύουν άσκοπα την μπαταρία τους λαμβάνοντας τα hello packets και έπειτα οι μακρινοί κόμβοι που ακούνε τα hello και προσπαθούν να δρομολογήσουν μέσω του κακόβουλου κόμβου στέλνουν τα πακέτα τους στο κενό.

Για την προσομοίωση επιλέξαμε τον κόμβο 75 να εκτελέσει την επίθεση, οι παράμετροι είναι ίδιες με πριν.

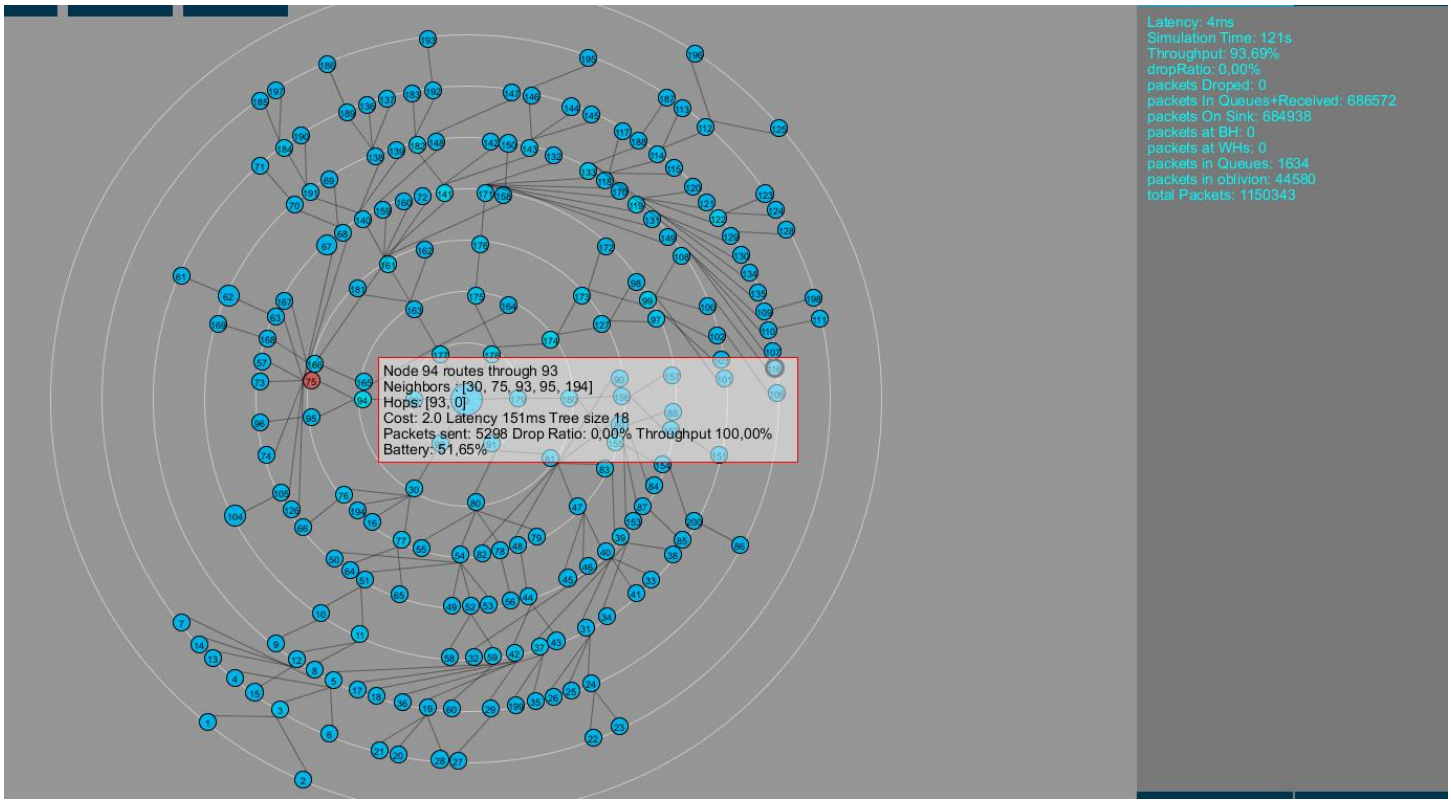


Εικόνα.52a: Normal View, ο κόμβος 75 εκτελεί επίθεση Hello και έχει μεγαλύτερη εμβέλεια από τους υπόλοιπους

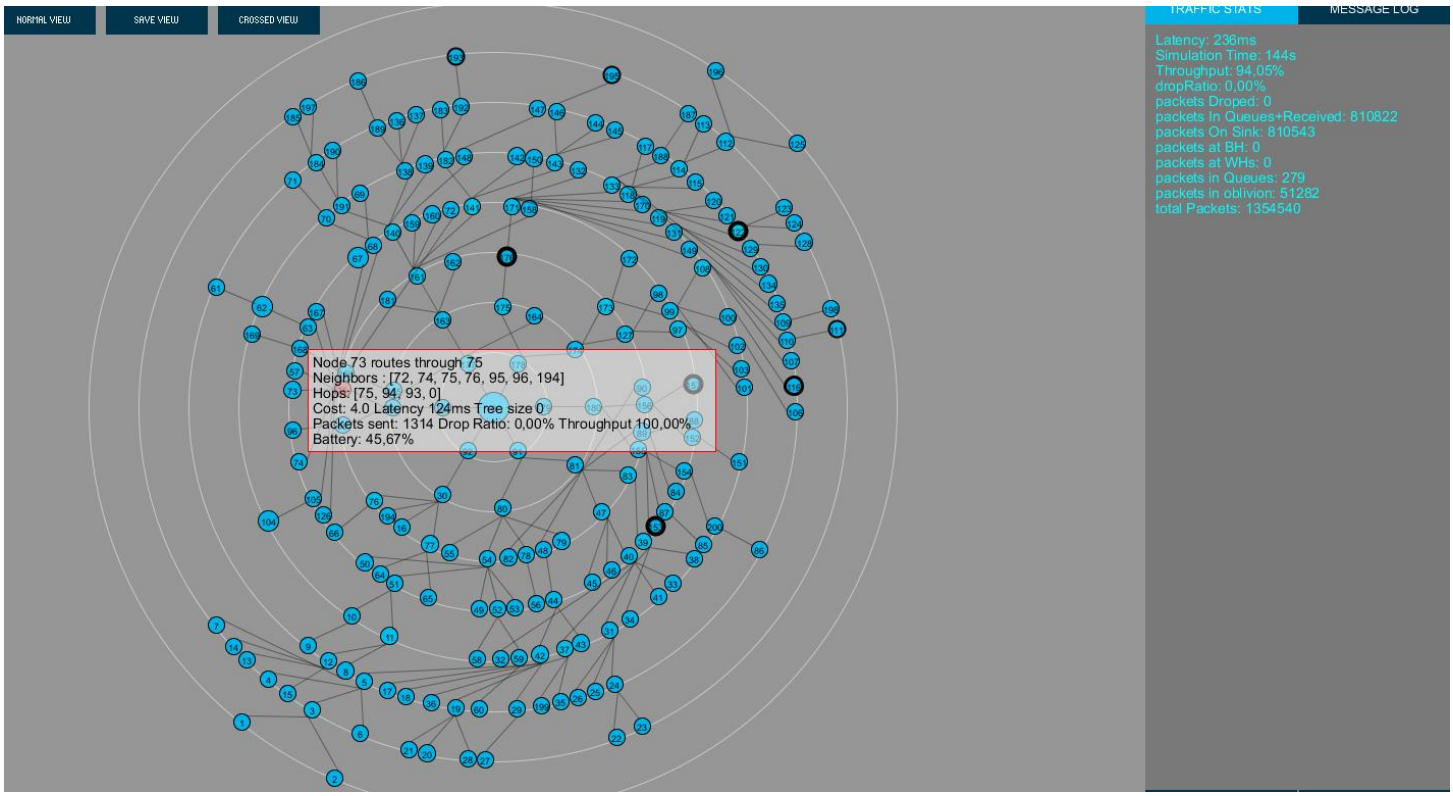


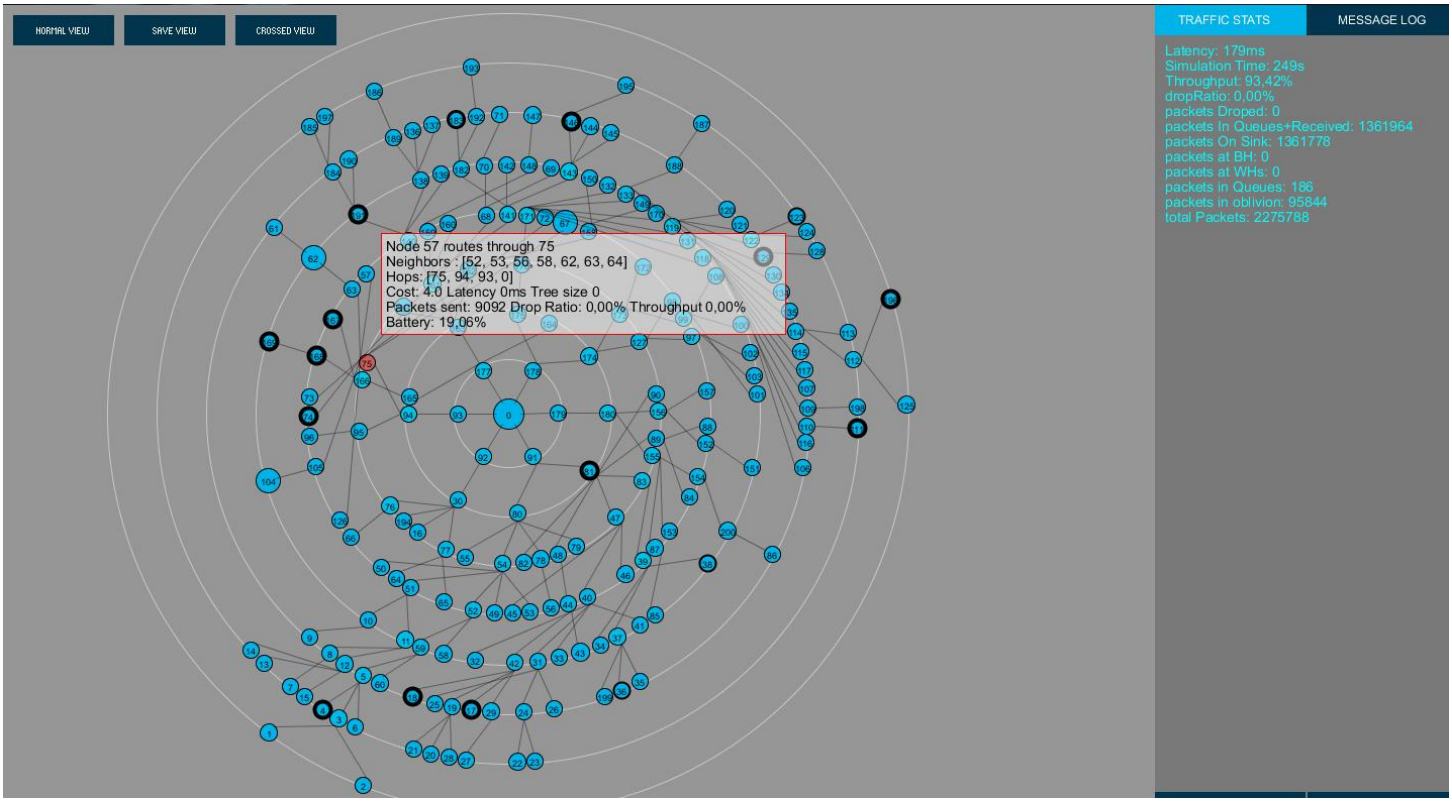
Εικόνα.52b: Node View, τα παιδιά του 75. Υπό κανονικές συνθήκες τα παιδιά του θα ήταν μόνο οι 73 και 74

Ακολουθούν εικόνες που δείχνουν την μείωση της μπαταρίας των κόμβων που επηρεάζονται από την επίθεση. Για χάρη της προσομοίωσης έχουμε αυξήσει τον ρυθμό ελάττωσης της μπαταρίας, στην πραγματικότητα είναι πολύ πιο αργός. Μπορούμε επίσης να δούμε στα στατιστικά τον αριθμό των πακέτων που χάθηκαν προσπαθώντας να φτάσουνε στον Hello Node. Είναι 44580 στα 121s και 51282 στα 144s και 95844 στα 249s.



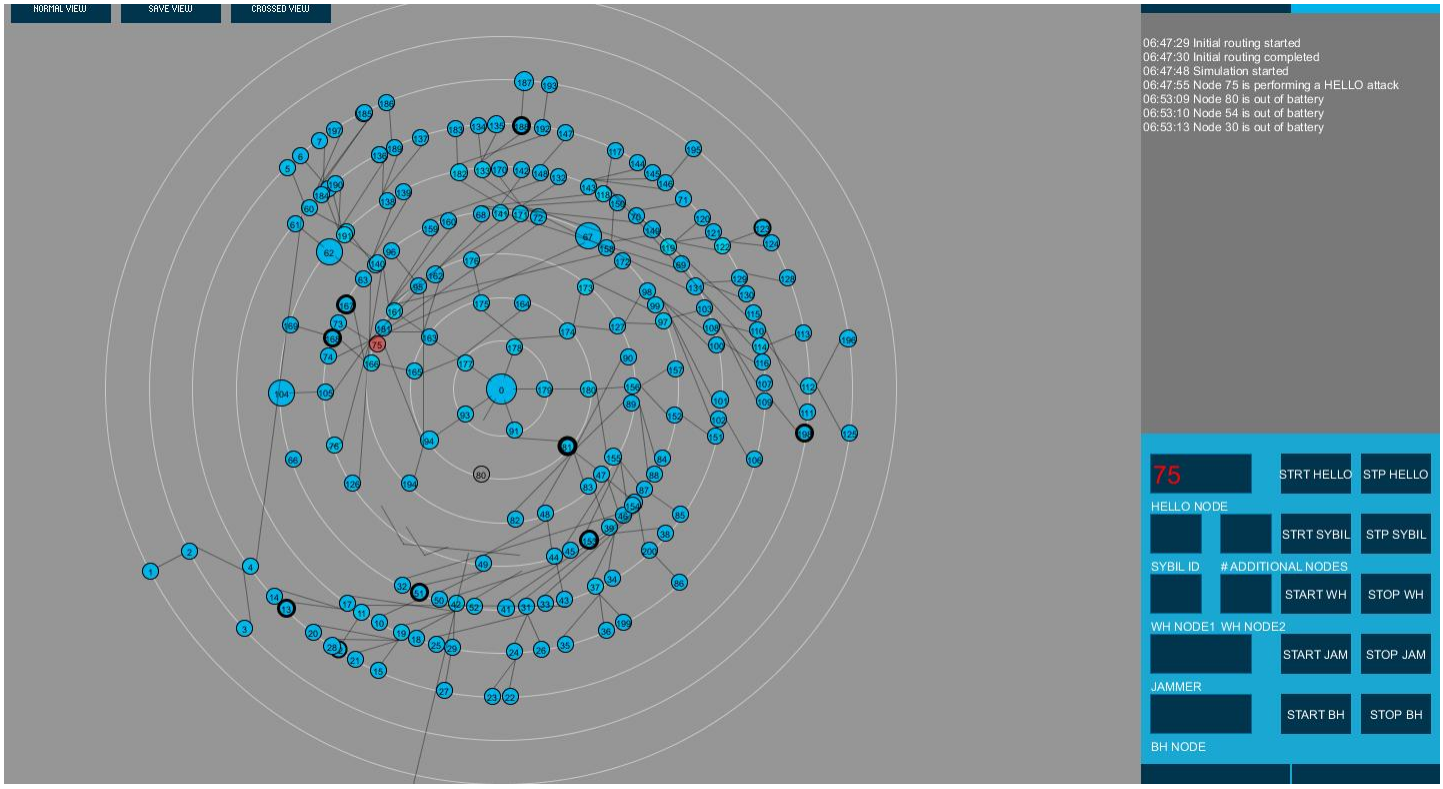
Εικόνα.52b,c,d: Save View ελάττωση μπαταρίας κόμβων υπό την επήρεια hello attack.





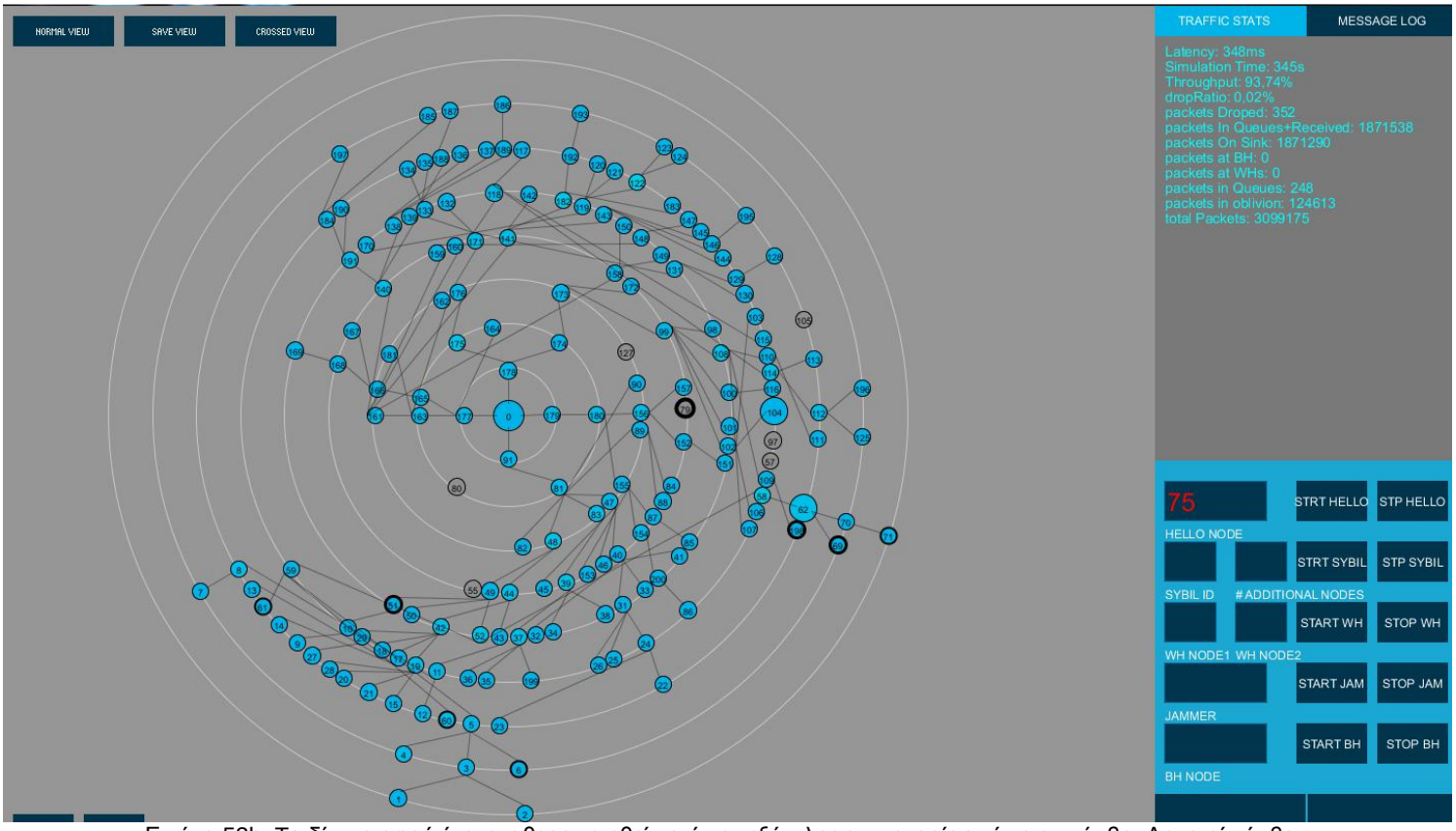
Εικόνα.52d: Save View ελάττωση μπαταρίας κόμβων υπό την επήρεια hello attack.

Στη συνέχεια φαίνεται η εικόνα του δικτύου την στιγμή που τελειώνει η μπαταρία των κόμβων 80,84 και 30. Σε αυτή την περίπτωση οι κόμβοι που δρομολογούσαν μέσω αυτών αναζητούν νέα μονοπάτια.



Εικόνα.53a: Η στιγμή που εξαντλείται η μπαταρία των κόμβων. Το δίκτυο βρίσκεται σε σύγχυση καθώς οι κόμβοι αναζητούν νέα μονοπάτια

Ακολουθεί η κατάσταση του δικτύου μόλις ο hello node πέτυχε τον σκοπό του και εξάντλησε την μπαταρία των κόμβων. Είναι φανερό πως ένα σημαντικό μέρος των κόμβων του δικτύου έχει αποσυνδεθεί. Συνολικά 124.613 από τα 3.099.175 πακέτα (4%) χάθηκαν προσπαθώντας μάταια να φτάσουν στον hello node, σε αυτό οφείλεται. και η μικρή μείωση του throughput (93,74%). Η επίθεση διήρκησε γύρω στα 5,5 λεπτά. Παρατηρούμε επίσης ότι 352 (0,02%) πακέτα απορρίφθηκαν. Αυτά ήταν πακέτα τα οποία προορίζονταν για κάποιον κόμβο του οποίο τέλειωσε η μπαταρία πριν προλάβουν να φτάσουν



Εικόνα.53b: Το δίκτυο αφού έχει σταθεροποιηθεί μετά την εξάντληση μπαταρίας κάποιον κόμβο. Αρκετοί κόμβοι έχουν αποσυνδεθεί.

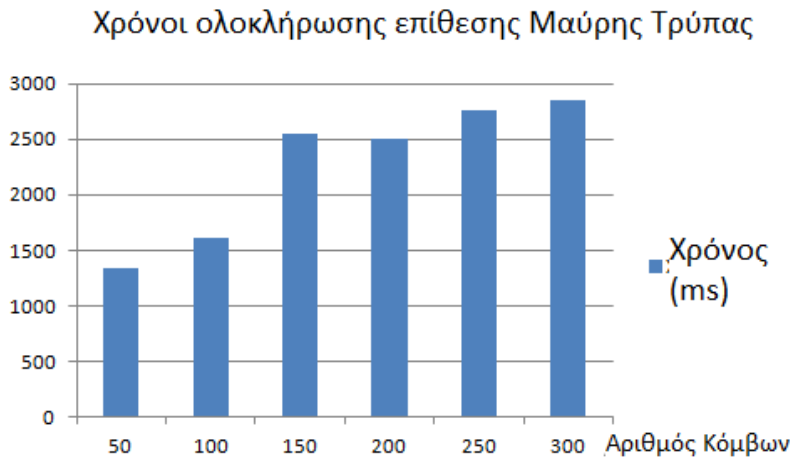
Σε αυτό το σημείο έχει ολοκληρωθεί η παρουσίαση της εφαρμογής. Αναλύσαμε σχεδόν σειρά προς σειρά τον κώδικα της εφαρμογής ώστε να μπορέσει ο αναγνώστης να καταλάβει το σκεπτικό του συγγραφέα και τις δυσκολίες που αντιμετώπισε. Στην αρχή έγινε παρουσίαση των βασικών κλάσεων και μεθόδων που είναι απαραίτητες για την λειτουργία της εφαρμογής, αλλά και κάποιον βοηθητικών. Έπειτα ο αναγνώστης ενημερώθηκε για τον τρόπο με τον οποίο δημιουργήθηκε η κάθε απεικόνιση. Στη συνέχεια εξηγήσαμε πως δημιουργήσαμε την κάθε επίθεση και τέλος τις εκτελέσαμε και παρουσιάσαμε σχολαστικά τις επιπτώσεις τους στο δίκτυο.

Αποτίμηση της Εφαρμογής

Ο σκοπός αυτού του κεφαλαίου είναι να κάνουμε μία αξιολόγηση της εφαρμογής μας. Πιο συγκεκριμένα θα μετρήσουμε τον χρόνο που χρειάζεται για να ολοκληρωθούν οι επιθέσεις. Πρώτα θα πάρουμε μετρήσεις για την επίθεση μαύρης τρύπας. Το ζητούμενο σε αυτή την επίθεση είναι να μάθουμε πόσος χρόνος χρειάζεται από την έναρξη της επίθεσης μέχρι η μαύρη τρύπα να απορροφήσει τον μέγιστο αριθμό κόμβων που μπορεί. Για να το πετύχουμε αυτό ξεκινάμε έναν μετρητή χρόνου σε ms την στιγμή που θα πατηθεί το κουμπί «START BH». Ο μετρητής αυξάνεται όσο αλλάζει το μέγεθος του υποδέντρου του κόμβου-μάυρη τρύπα. Όταν σταθεροποιηθεί το μέγεθος του υποδέντρου σημαίνει ότι η μαύρη τρύπα έχει απορροφήσει τον μέγιστο αριθμό κόμβων και σταματάει ο μετρητής. Ο χρόνος αυτός εξαρτάται από το πόσο συχνά τρέχει ο αλγόριθμος δρομολόγησης (στην περίπτωση μας κάθε 500ms) και τον αριθμό των κόμβων του δικτύου. Η μέθοδος που μετράει το μέγεθος του υποδέντρου κάθε κόμβου ανήκει στην κλάση `Node` και είναι αναδρομική:

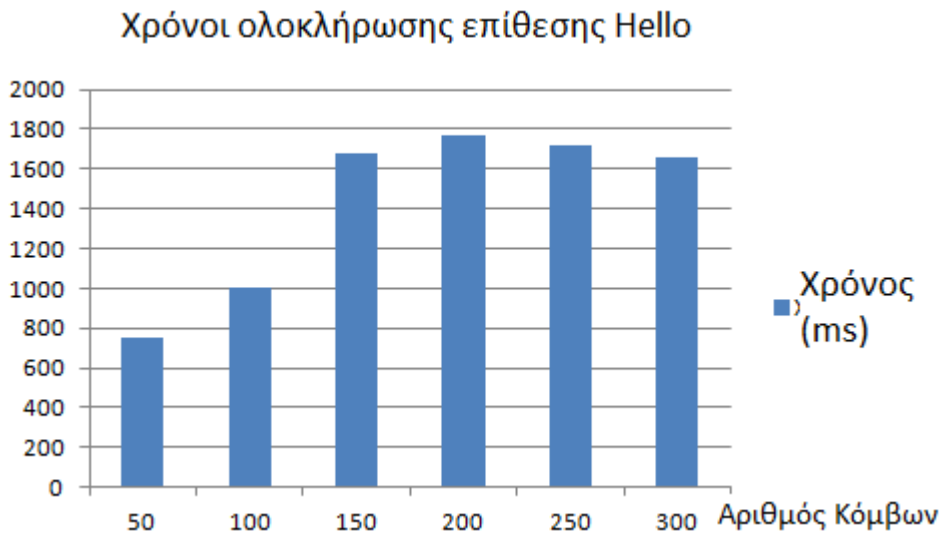
```
public int treesize(int size) {
          ArrayList<Integer> children = new ArrayList<Integer>();
          children=this.getChildren();
          int newsize = children.size()+size;
          int counter=0;
          while (children.size() >0 && counter<children.size()) {
             newsize = Nodes.get(children.get(counter)).treesize(newsize);
             counter++;
          }
          return newsize;
}
```

Εκτελέσαμε την επίθεση σε δίκτυα διαφόρων μεγεθών και σε κάθε δίκτυο πήραμε αρκετές μετρήσεις με διαφορετικό κόμβο-μαύρη τρύπα κάθε φορά. Υπολογίσαμε τον μέσο όρο των μετρήσεων αυτών και τα αποτελέσματα φαίνονται στο παρακάτω γράφημα:



Εικόνα 54: Χρόνοι ολοκλήρωσης επίθεσης Μαύρης Τρύπας

Στην συνέχεια θα μετρήσουμε τον χρόνο που απαιτείται για να ολοκληρωθεί η επίθεση Hello. Η διαδικασία με τον μετρητή είναι ίδια με την επίθεση μαύρης τρύπας και όλες οι υπόλοιπες παράμετροι, καθώς και τα δίκτυα αισθητήρων που χρησιμοποιήθηκαν είναι επίσης τα ίδια. Κατά την διάρκεια των μετρήσεων παρατηρήσαμε ότι ο χρόνος ολοκλήρωσης εξαρτάται κατά μεγάλο βαθμό από την θέση που έχει ο κόμβος στο δίκτυο. Εάν βρίσκεται σε «πυκνοκατοικημένη» περιοχή ο χρόνος αυξάνεται γιατί υπάρχουν περισσότεροι κόμβοι που θα δελεαστούν. Εάν βρίσκεται σε περιοχή με λίγους κόμβους κοντά του τότε η επίθεση ολοκληρώνεται πολύ γρήγορα. Επιλέγοντας κόμβο που είναι στα όρια του δικτύου και δεν δελεάζει κανέναν κόμβο η επίθεση ολοκληρώνεται σε περίπου 60ms. Φροντίσαμε όμως να μην συμπεριλάβουμε ακραίες τιμές στις μετρήσεις μας οι οποίες φαίνονται παρακάτω.



Εικόνα 55: Χρόνοι ολοκλήρωσης επίθεσης Hello.

Στις υπόλοιπες επιθέσεις δεν έχουμε τη δυνατότητα να πάρουμε αντίστοιχες μετρήσεις γιατί δεν μεταβάλλεται το μέγεθος του υποδέντρου των κόμβων. Η επίθεση εκτελείται άμεσα (σε μερικές δεκάδες ms) και ο χρόνος αυτός εξαρτάται από την ταχύτητα του επεξεργαστή.

Η εφαρμογή σε κατάσταση αδράνειας με την οθόνη να δείχνει το Save View με 200 κόμβους καταναλώνει περίπου το 25% του επεξεργαστή και 110 με 120MB της RAM. Κατά τη διάρκεια προσομοίωσης με τις προκαθορισμένες ρυθμίσεις η χρήση της CPU άγγιξε το 27% και χρησιμοποιήθηκαν μέχρι και 150MB RAM. Εκτελώντας την πιο υπολογιστικά ακριβή επίθεση Jamming η χρήση της CPU άγγιξε το 35% και η κατανάλωση της RAM ξεπέρασε τα 500MB μέσα στα πρώτα 5 λεπτά. Η ραγδαία αυτή αύξηση της χρήσης μνήμης οφείλεται στον τεράστιο αριθμό πακέτων που παράγει ο jammer. Η εκτέλεση των υπόλοιπων επιθέσεων δεν παρουσίασε σημαντική αύξηση στην κατανάλωση πόρων του υπολογιστή.

Μελλοντική δουλειά - Επίλογος

Η ανάγκες για ασφάλεια στα ασύρματα δίκτυα αισθητήρων αυξάνονται συνεχώς καθώς ανακαλύπτονται συνεχώς νέες παραλλαγές επιθέσεων για να αποφεύγεται η ανίχνευση τους. Ο τομέας της ασφάλειας είναι ένας αγώνας ανάμεσα στους ανθρώπους που θέλουν να καταστρέψουν και σε αυτούς που θέλουν να προστατέψουν τα δίκτυα. Εμείς μπήκαμε στον αγώνα και δημιουργήσαμε μία εφαρμογή που είναι ικανή να παρουσιάσει οπτικά στον χρήστη τις επιθέσεις που συμβαίνουν στο ασύρματο δίκτυο αισθητήρων του. Οι κόμβοι που συμμετέχουν ή επηρεάζονται από κάθε επίθεση επισημαίνονται με διαφορετικά χρώματα για την εύκολη αναγνώριση. Το Save View που έχουμε δημιουργήσει είναι από τους πιο εύχρηστους τρόπους για την απεικόνιση επιθέσεων σε ένα δίκτυο, ο χρήστης μπορεί να καταλάβει άμεσα όταν κάτι κακό συμβαίνει στο δίκτυο. Επίσης το Crossed View είναι ένα ακόμα εργαλείο που εμφανίζει συγκεντρωτικά το throughput και το drop ratio του κάθε κόμβου και βοηθάει στην αποτελεσματική παρακολούθηση του δικτύου.

Στο μέλλον θα θέλαμε να υλοποιήσουμε τη δυνατότητα εισαγωγής στην εφαρμογή δεδομένων κίνησης και συμβάντων από πραγματικά ασύρματα δίκτυα αισθητήρων και να αναπτύξουμε τεχνικές για την αυτόματη αναγνώριση και οπτικοποίηση των επιθέσεων. Όταν υλοποιηθεί αυτό, η εφαρμογή θα μπορεί να εκτελείται στον σταθμό βάσης σαν ένα κεντρικοποιημένο σύστημα προστασίας του δικτύου. Στόχος επίσης είναι να δημιουργηθεί ένας client της εφαρμογής και να εγκατασταθεί στο cloud ώστε οι χρήστες να μπορούν να έχουν απομακρυσμένη πρόσβαση στο δίκτυο τους από όπου και αν βρίσκονται.

Όσον αφορά το κομμάτι της προσομοίωσης καθώς εξελίσσεται η εφαρμογή θα προστεθούν επιπλέον παράμετροι και επιλογές για την κίνηση των πακέτων στο δίκτυο, καθώς και διαφορετικές τοπολογίες. Επίσης ο χρήστης θα μπορεί να επιλέξει διαφορετικούς αλγόριθμους δρομολόγησης και πρωτόκολλα επικοινωνίας. Τέλος οι επιθέσεις θα είναι πλήρως παραμετροποιήσιμες μέσω ενός εύχρηστου μενού.

Λίστα ακρώνυμων

Ακρωνύμιο	Επεξήγηση
ΑΔΑ	Ασύρματο Δίκτυο Αισθητήρων – Wireless Sensor Network – WSN
ΓΑΠΑ	Γραφική Απεικόνιση Προβλημάτων Ασφαλείας - Security Visualization
ΙΗΗΜ	Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (Institute of Electrical and Electronics Engineers – IEEE)
ΑΤΔ	Ασύρματο τοπικό δίκτυο WLAN
ΠΑΔ	Προσωπικό ασύρματο δίκτυο - Wireless Personal Area network
ΥΕΖ	Υπερ Ευρείας Ζώνης – Ultra WideBand
ΔΣ	Δίκτυο Σώματος – Body Area Network
ΧΡ-ΠΑΔ	Χαμηλού Ρυθμού-Προσωπικό Ασύρματο Δίκτυο – Low Rate-PAN
ΕΔΚΩ	Έξυπνα Δίκτυα Κοινής Ωφέλειας – Smart Utility Networks
ΕΧ	Εγγυημένες Χρονοθυρίδες – Guaranteed timeslots - GTSs
ΑΕ	Ανίχνευση Ενέργειας - Energy detection
ΕΠΖ	Ένδειξη Ποιότητας Ζεύξης - Link quality indication
ΠΛΣ	Πλήρως Λειτουργική Συσκευή – Fully Functional Device – FFD
ΜΛΣ	Μερικώς Λειτουργική Συσκευή – Reduced Functionality Device - RFD
ΠΡ	Πομποδέκτης Ραδιοσυχνοτήτων – Radio Transmiter
ΜΔΠ	Μονάδες Δεδομένων Πρωτοκόλλου - protocol data units - PDUs
ΕΕΚ	Εκτίμηση Ελεύθερου Καναλιού – Clear channel assessment
ΟΔΥΜ	Οντότητα Διαχείρισης Υποεπιπέδου MAC - MAC sublayer management entity (MLME)
ΣΠΥ	Σημείο Πρόσβασης Υπηρεσίας service access point (SAP)
ΑΑΕΦ	Άμεσης Ακολουθίας Εκτεταμένου Φάσματος - direct sequence spread spectrum -DSSS
ΑΜΠΠ-ΑΣ	Αίσθηση του Μεταφορέα Πολλαπλής Πρόσβασης με Αποφυγή των Συγκρούσεων - Carrier Sense Multiple Access with Collision Avoidance - CSMA-CA
ΠτΥ	Ποιότητα της Υπηρεσίας – Quality of Service - QoS
ΑτΥ	Άρνηση της Υπηρεσίας – Denial of Service - DoS
ΜΤ	Μαύρη Τρύπα – Black hole - BH
ΣΤ	Σκουληκότρυπα – Wormhole – WH
ΚΚ	Κεντρικός Κόμβος – Sink Node
ΠΕ	Παράγοντας εμπιστοσύνης – Trust Factor
ΠΚ	Πολυδιάστατη Κλιμάκωση multi-dimensional scaling
ΧΜΑ	Χρονικό Μοντέλο Ανάπτυξης – Temporal Expansion Model - TEM
ΑΡΣ	chips Αναγνώρισης ΡαδιοΣυχνότητας RFID chips

Λίστα Εικόνων

38b,c	Crossed View πριν τη ΜΤ. Όλοι οι κόμβοι έχουν σχεδόν 100% throughput και μηδενικό drop ratio.
39a,b	Crossed View στην εκκίνηση της μαύρης τρύπας
39c	Save View στην εκκίνση της μαύρης τρύπας
40a	Save view, 3 λεπτά μετά την εμφάνιση μαύρης τρύπας
40b,c	Crossed View 3 λεπτά μετά την εμφάνιση μαύρης τρύπας
41a	Save View στο τέλος της προσομοίωσης της ΜΤ
41b,c	Crossed View στο τέλος της προσομοίωσης της ΜΤ
42	Πτώση throughput κατά τη διάρκεια της επίθεσης ΜΤ
43a	Normal View επίθεση σκουληκότρυπας
43b	Απεικόνιση της επίθεσης σκουληκότρυπας στο Save View. Αρχή επίθεσης
43c,d,e	Crossed View: Αναλυτική πτώση throughput. Αρχή επίθεσης ΣΤ
44a	Save View, 3 λεπτά μετά την εμφάνιση ΣΤ
44b,c,d,e	Crossed View, πτώση throughput 3 λεπτά από την εμφάνιση σκουληκότρυπας
45a	Save View στο τέλος της προσομοίωσης
45b,c,d,e	Crossed View στο τέλος της προσομοίωσης.
45f	Πτώση throughput κατά τη διάρκεια της επίθεσης ΣΤ
46a	Save View παρουσία jammer. Αρχή της επίθεσης
46b	Crossed View παρουσία jammer. Αρχή της επίθεσης
46c,d	Node View των κόμβων 54 και 91 που έχουν επηρεαστεί από τον jammer
46e	Normal View με jammer τον κόμβο 50
47a	Save View 3 λεπτά μετά την εμφάνιση του jammer
47b	Node 91 View 3 λεπτά μετά την εμφάνιση του jammer
47c	Node 54 View 3 λεπτά μετά την εμφάνιση του jammer
47e,d	Crossed View 3 λεπτά μετά την εμφάνιση του jammer
48	Save View: Διάταξη του δικτύου στο τέλος της προσομοίωσης του jammer
49a,b	Γραφικές παραστάσεις του a)throughtput & drop ratio b)latency του δικτύου κατά τη διάρκεια της επίθεσης jamming.
50a	Save View απεικόνιση της επίθεσης Sybil.
50b	Normal View φυσικές θέσεις των Sybil nodes στο δίκτυο. Τα Sybil Ghosts βρίσκονται στις ίδιες συντεταγμένες με τους κόμβους που τα δημιούργησαν
51a,b	Sybil Nodes με τα Sybil Ghosts τους, αλλά και κανονικούς κόμβους ως παιδιά
51c,d	Sybil Nodes με τα Sybil Ghosts τους χωρίς άλλους κόμβους ως παιδιά
51e	Πρώην κανονικός κόμβος που παρουσιάζει Sybil Ghosts. Έχει επίσης ως παιδιά του 4 ακόμα κόμβους.
52a	Normal View, ο κόμβος 75 εκτελεί επίθεση Hello και έχει μεγαλύτερη εμβέλεια από τους υπόλοιπους
52b	Node View, τα παιδιά του 75. Υπό κανονικές συνθήκες τα παιδιά του θα ήταν μόνο οι 73 και 74
52b,c,d	Save View ελάττωση μπαταρίας κόμβων υπό την επήρεια hello attack
53a	Η στιγμή που εξαντλείται η μπαταρία των κόμβων. Το δίκτυο βρίσκεται σε σύγχυση καθώς οι κόμβοι αναζητούν νέα μονοπάτια

53b	Το δίκτυο αφού έχει σταθεροποιηθεί μετά την εξάντληση μπαταρίας κάποιον κόμβο. Αρκετοί κόμβοι έχουν αποσυνδεθεί.
54	Χρόνοι ολοκλήρωσης επίθεσης Μαύρης Τρύπας
55	Χρόνοι ολοκλήρωσης επίθεσης Hello.

Λίστα πινάκων

Αριθμός Πίνακα	Επεξήγηση
1	Επιλογές ανάθεση συχνότητας του προτύπου IEEE 802.15.4

Βιβλιογραφία

[1] "Processing", Available at: processing.org

[2] E. Karapistoli, P. Sarigiannidis, and A. A. Economides, "Visual-Assisted Wormhole Attack Detection for Wireless Sensor Networks, in 10th SecureComm, Beijing, Sept. 2014, pp.1-8

[3] Virmani, Dr, Manas Hemrajani, and Shringarica Chandel (2014, Jan.). "Exponential Trust Based Mechanism to Detect Black Hole attack in Wireless Sensor Network." pp.1 Available: http://arxiv.org/abs/1401.2541.

[4] Y. Wu, Y. Cho, G.Qu, "Insider Threats against Trust Mechanism with Watchdog and Defending Approaches in Wireless Sensor Networks", IEEE CS Security and Privacy Workshops, April 2012, pp.134-141

[5] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00), August 2000, pp.255-265

[6] A. Forootaniniaand, M. B. Ghaznavi-Ghoushchi, "An Improved Watchdog Technique Based On Power Aware Hierarchical Design for IDS in Wireless Sensor Networks", International Journal of Network Security & Its Applications (IJNSA), 4(4), July 2012.

[7] S. Banerjee. "Detection/Removal of Cooperative Black and Gray Hole in Mobile Ad-Hoc Network", in WCECS 2008, San Francisco, USA, October 22-24, 2008

[8] J. Yin, S. K. Madria, "A Hierarchical Secure Routing Protocol against Blackhole Attacks in Sensor Networks", IEEE International Conference on Sensor Networks,2006

[9] M. Khabbazian, H. Mercier, and V. Bhargava, "Severity analysis and countermeasure for the wormhole attack in wireless ad hoc networks," Wireless Communications, IEEE Transactions on, 8(2), 2009, pp. 736–745.

[10] X. Ban, R. Sarkar, and J. Gao, "Local connectivity tests to identify wormholes in wireless networks," in Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing, ser. MobiHoc, 2011, pp. 1–11.

[11]B. Awerbuch, R. Curtmola, D. Holmer, H. Rubens, and C. Nita-Rotaru, "On the survivability of routing protocols in ad hoc wireless networks," in Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on, Sept 2005, pp. 327–338.

[12]. B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "Odsbr: An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks," ACM Trans. Inf. Syst. Secur., vol. 10, no. 4, Jan. 2008, pp. 6:1–6:35

[13] X. Wang and J. Wong, "An end-to-end detection of wormhole attack in wireless ad-hoc networks," in Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, vol. 1, 2007, pp. 39–48.

[14] W. Wang and B. Bhargava, "Visualization of wormholes in sensor networks," in ACM workshop on Wireless Security. ACM Press, 2004, pp. 51–60.

[15] W. Wang and A. Lu, "Interactive wormhole detection in large scale wireless networks," in Visual Analytics Science And Technology, IEEE Symposium On, 2006,pp. 99–106.

[16] Singh, Shio Kumar, M. P. Singh, and D. K. Singh. (2011), "A survey on network security and attack defense mechanism for wireless sensor networks." In International Journal of Computer Trends and Technology 1(2), pp:9-17. Available: http://www.ijcttjournal.org/Volume1/issue-2/ijcttjournal-v1i2p2.pdf

[17] Jun Zheng and Abbas Jamalipour, "Wireless Sensor Networks: A Networking Perspective", a book published by A John & Sons, Inc, and IEEEE, 2009.

[18] E. Yoneki and J. Bacon, "A survey of Wireless Sensor Network technologies: research trends and middleware's role", Technical Report. Available: http://www.cl.cam.ac.uk/TechReports, ISSN 1476-2986, 2005

[19] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security - a survey", Security in Distributed, Grid, Mobile, and Pervasive Computing, Auerbach Publications, CRC Press, 2007.

[20] L.L. Fernandes, (2007, June) "Introduction to Wireless Sensor Networks Report", University of Trento. Available: http://dit.unitn.it/~fernand/downloads/iwsn.pdf

[21] Zia, T., & Zomaya, A, "A security framework for wireless sensor networks", in Proceedings of the IEEE Sensors Applications Symposium, Feb. 2006, pp. 49-53.

[22] P. Mohanty, S. A. Pangrahi, N. Sarma, and S. S. Satapathy, "Security Issues in Wireless Sensor Network Data Gathering Protocols: A Survey". Journal of Theoretical and Applied Information Technology, 2010, pp. 14-27.

[23] Wenyuan Xu, Wade Trappe, Yanyong Zhang and Timothy Wood. "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks", in Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing. ACM, May 2005

[24] Hamid, Azzoune, Md Mamun-Or-Rashid, and Choong Seon Hong. "Defense against lap-top class attacker in wireless sensor network." Presented at Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference. Vol. 1. IEEE, 2006.

[25] Singh, V. P., Jain, S., & Singhai, J. (2010). Hello flood attack and its countermeasures in wireless sensor networks. International Journal of Computer Science, 7(3), pp. 2-3. Available: http://www.ijcsi.org/papers/IJCSI-Vol-7-Issue-3-No--11.pdf#page=37

[26] Giruka, V. C., Singhal, M., Royalty, J., & Varanasi, S. (2008). Security in wireless sensor networks. Wireless communications and mobile computing, 8(1),. Available: http://www.cs.arizona.edu/classes/cs625/fall10/Survey_security_sensors.pdf

[27] Dr. Moh. Osama K., "Hello flood counter measure for wireless sensor network", in International Journal of Computer Science and Security, 2 (3), 2007

[28] Chris Karlof, David Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", In First IEEE International Workshop on Sensor Network Protocols and Applications, May 2003, pp 113–127

[29] Cao, Z., Zhou, X., Xu, M., Chen, Z., Hu, J., & Tang, L., "Enhancing base station security against DoS attacks in wireless sensor networks", Presented at Wireless Communications, Networking and Mobile Computing, 2006. WiCOM 2006. International Conference on, IEEE, Sept. 2006, pp. 1-4

[30] Pires, W. R., de Paula Figueiredo, T. H., Wong, H. C., & Loureiro, A., "Malicious node detection in wireless sensor networks.", Presented at Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International IEEE, April 2004, pp. 24.

[31] Haghighi, M. S., & Mohamedpour, K., "Securing wireless sensor networks against broadcast attacks.", Presented at Telecommunications, 2008. IST 2008. International Symposium on, IEEE, Aug. 2008, pp. 49-54

[32] Newsome, J., Shi, E., Song, D., & Perrig, A, "The sybil attack in sensor networks: analysis & defenses.", Presented at Proceedings of the 3rd international symposium on Information processing in sensor networks. ACM, April 2004, pp. 259-268

[33] J. R. Douceur (2002, Mar.). "The Sybil attack", Presented at First International Workshop on Peer-to-Peer Systems (IPTPS '02). Available: http://freehaven.net/anonbib/cache/sybil.pdf

[34] Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., & Shenker, S, "GHT: a geographic hash table for data-centric storage", in Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, ACM September 2002 pp. 78-87

[35] Raffael Marty, "Visualization" in" Applied Security Visualization". Upper Saddle River: Addison-Wesley 2009, pp.2-7

[36] Colin Ware in "Information Visualization: Perception for Design", San Francisco, CA, 2012

[37] Jeffery Undercoffer, Sasikanth Avancha, Anupam Joshi, and John Pinkston, "Wireless Sensor Networks", an edited book, Kluwer Publications, 2004

[38] Sénéchal, N., Hong, S., & Eades, P,"Display of sensor networks: a feasibility study", in Daintree Networks, July 2006

[39] Turon, M., & Suh, J. (2005, May). Mote-view: A sensor network monitoring and management tool. Presented at The Second IEEE Workshop on Embedded Networked Sensors pp. 11-18.

[40] Buschmann, C., Pfisterer, D., Fischer, S., Fekete, S. P., & Kröller, A. Spyglass: a wireless sensor network visualizer. ACM Sigbed Review, 2(1), pp. 1-6, 2005

[41] Y. Yang, L. Huang, Q. Zhou, Y. Xu, and X. Li), "Snamp: A multisniffer and multi-view visualization platform for wireless sensor networks," in 1st IEEE Conference on Industrial Electronics and Applications (ICIEA '06), (Singapore),May 24-26, 2006, pp. 1523–1526.

[42] Shi, L., Liao, Q., He, Y., Li, R., Striegel, A., & Su, Z. "SAVE: Sensor anomaly visualization engine. Presented at Visual Analytics Science and Technology (VAST)", in 2011 IEEE Conference on, IEEE, October 2011, pp. 201-210

[43] "ControlP5 graphics library" Available: http://www.sojamo.de/libraries/controlP5

[44] Low-Rate Wireless Personal Area Networks, IEEE Standard 802.15.4, 2003

[45] Lou Frenzel, "What's The Difference Between IEEE 802.15.4 And ZigBee Wireless?" 2013, Mar. Unpublished. Available: http://electronicdesign.com/what-s-difference-between/what-s-difference-between-ieee-802154-and-zigbee-wireless