Ατομική Διπλωματική Εργασία

ΑΝΙΧΝΕΥΣΗ ΕΠΙΘΕΣΕΩΝ RPL ΣΕ ΔΙΚΤΥΑ IOT

Μαρία Ευθυμίου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανίχνευση επιθέσεων RPL σε δίκτυα IoT

Μαρία Ευθυμίου

Επιβλέπων Καθηγητής
Δρ. Βάσος Βασιλείου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2019

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέπων καθηγητή μου Δρ. Βάσο Βασιλείου, ο οποίος μου έδωσε την ευκαιρία, στα πλαίσια της διπλωματικής εργασίας, να ασχοληθώ ένα από τα πιο ενδιαφέρων, για εμένα, θέματα στον κλάδο της πληροφορικής, αλλά και για την καθοδήγησή του κατά την υλοποίηση της εργασίας.

Ακόμη, θα ήθελα να ευχαριστήσω την Δρ. Χριστιάνα Ιωάννου για τις γνώσεις και την υποστήριξη που μου παρείχε κατά τη διάρκεια διεκπεραίωσης της διπλωματικής εργασίας.

Περίληψη

Ο στόχος της παρούσας εργασίας είναι η ανίχνευση επιθέσεων RPL (Routing Protocol for Low-Power and Lossy Networks - Πρωτόκολλο δρομολόγησης για δίκτυα χαμηλής κατανάλωσης και με απώλειες) σε δίκτυα IoT (Internet of Things). Για την επίτευξη του στόχου αυτού είναι απαραίτητο να παρακολουθηθεί η συμπεριφορά των δικτύων όταν εισαχθεί σε αυτά κάποιος ιός.

Με τη βοήθεια του προγράμματος contiki/cooja δημιουργήθηκαν εξομοιώσεις του δικτύου, μέσω των οποίων αναλύθηκαν τρεις διαφορετικές τοπολογίες δικτύων. Για κάθε τοπολογία εισάχθηκαν στην προσομοίωση τέσσερεις ιοί σε κάθε σενάριο προσομοίωσης. Για κάθε ένα από τους πιο πάνω συνδυασμούς τοπολογίας-ιού δημιουργήθηκαν δύο διαφορετικά είδη εξομοιώσεων, ένα όπου ο ιός αρχίζει την κακόβουλη συμπεριφορά από την αρχή και ένα όπου η κακόβουλη συμπεριφορά αρχίζει μετά από ένα προκαθορισμένο χρονικό διάστημα.

Έπειτα, μέσω των αρχείων καταγραφής της κάθε εξομοίωσης, εξήχθησαν δεδομένα για κάθε κόμβο του δικτύου. Μέσα από την ανάλυση των δεδομένων αυτών μπορούμε να εξάγουμε συμπεράσματα, τα οποία αφορούν την συμπεριφορά του δικτύου σε σχέση με τους κακόβουλους κόμβους, τα οποία μπορούν να μας βοηθήσουν στην ανίχνευσή τους.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή στο RPL

Το RPL πρωτόκολλο δημιουργήθηκε συγκεκριμένα για δίκτυα LLN (Low-power and Lossy Networks). Δρομολογεί πακέτα χρησιμοποιώντας ένα κατευθυνόμενο άκυκλο γράφο ως αναπαράσταση του δικτύου, όπου κάθε κόμβος στέλνει πακέτα στον κόμβο-πατέρα με τελικό προορισμό των πακέτων τον κόμβο sink. Κάθε κόμβος γνωρίζει τους ενεργούς γείτονές του και επιλέγει ως κόμβο-πατέρα του τον κόμβο που βρίσκεται πιο κοντά στον κόμβο sink. Αυτό έχει ως αποτέλεσμα, σε συγκεκριμένες περιπτώσεις, ένας κόμβος να έχει διαφορετικό κόμβο-πατέρα σε διαφορετική χρονική στιγμή.

Όπως προδιαγράφεται στο RFC 6550[2] , το πρωτόκολλο RPL πρέπει να πληροί τις πιο κάτω προδιαγραφές:

- Διαχωρίζει την επεξεργασία και προώθηση πακέτων από την διαδικασία βελτιστοποίησης της δρομολόγησης.
- Χρησιμοποιεί αμφίδρομους συνδέσμους μεταξύ των κόμβων του δικτύου, αντιμετωπίζοντας το δίκτυο ως ένα μη-κατευθυνόμενο γράφο. Χρειάζεται τους συνδέσμους αυτούς κατά την επιλογή του κόμβου-πατέρα.
- Χρησιμοποιεί σύστημα "RPL Packet Information" για να μεταφέρει πληροφορίες ελέγχου μέσα στα πακέτα.
- Διαδίδει πληροφορίες μέσα στο δίκτυο έτσι ώστε να επιτρέπει στους κόμβους να δρουν αυτόματα.
- Δημιουργεί τοπολογίες από δρομολογητές με δικό τους πρόθεμα το οποίο διαφημίζεται ως on-link.
- Έχει την δυνατότητα να δημιουργήσει υποδίκτυο μέσα στο οποίο να γίνεται δρομολόγηση με τη χρήση ενός κοινού προθέματος.
- Το RPL, αποτελεί IP πρωτόκολλο, και ως εκ τούτου δεν χρειάζεται συγκεκριμένα link-layer για να λειτουργήσει.

Ακόμη το RPL έχει την δυνατότητα να έχει ενεργά περισσότερα από ένα instances στο δίκτυο, με διαφορετικές παραμέτρους (configuration) για κάθε ένα από τα instances του δικτύου. Με αυτόν τον τρόπο κάθε instance μπορεί να είναι επικεντρωμένο σε διαφορετικές λειτουργίες του δικτύου.

Επιπλέον, το πρωτόκολλο RPL υποστηρίζει τρεις βασικές ροές κυκλοφορίας: multipoint-to-point, point-to-multipoint και point-to-point. Η ροή multipoint-to-point είναι η κυρίαρχη κυκλοφοριακή ροή σε πολλές εφαρμογές LLN δικτύων. Το RPL υποστηρίζει το πρωτόκολλο multipoint-to-point επιτρέποντας σε πακέτα να φτάσουν στους προορισμούς τους μέσω των ριζών DODAG[1]. Η ροή point-to-multipoint είναι ένα πρότυπο κυκλοφορίας που απαιτείται από πολλές εφαρμογές LLN. Το RPL υποστηρίζει την κυκλοφορία point-to-multipoint χρησιμοποιώντας ένα μηχανισμό διαφήμισης προορισμού που προμηθεύει δρομολόγια προς προορισμούς κατευθυνόμενη προς τα κάτω και μακριά από τις ρίζες. Τα RPL DODAG παρέχουν μια βασική δομή για κίνηση point-to-point. Για ένα δίκτυο RPL που υποστηρίζει την κυκλοφορία point-to-point, πρέπει να υπάρχει μια ρίζα που να μπορεί να δρομολογήσει τα πακέτα σε έναν προορισμό. Οι κόμβοι στο δίκτυο μπορεί να έχουν επίσης πίνακες δρομολόγησης προς προορισμούς.[2]

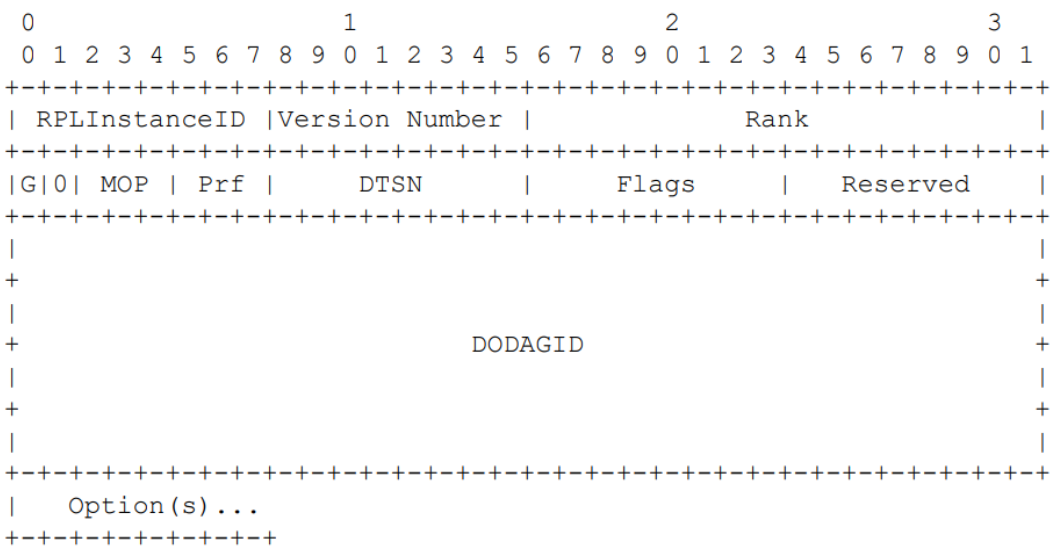
Κατά τη λειτουργία του, το RPL, στέλνει τέσσερεις κύριες κατηγορίες μηνυμάτων ελέγχου:

1.1.1. DIO
1.1.2. DAO
1.1.3. DAO-ACK
1.1.4. DIS

1.1.1 DIO

Τα DIO (DODAG Information Object) μηνύματα περιέχουν πληροφορίες που βοηθούν ένα κόμβο να ανακαλύψει στοιχεία σε σχέση με ένα ενεργό δίκτυο RPL. Μέσω των μηνυμάτων αυτών, που λαμβάνει, ένας κόμβος μπορεί να ανακαλύψει ένα instance του RPL και τις παραμέτρους του instance, τους ενεργούς γειτονικούς του κόμβους και, επομένως, να επιλέξει κόμβο-πατέρα, όπως φαίνεται και στην κεφαλίδα του πακέτου (Σχήμα 1.1).

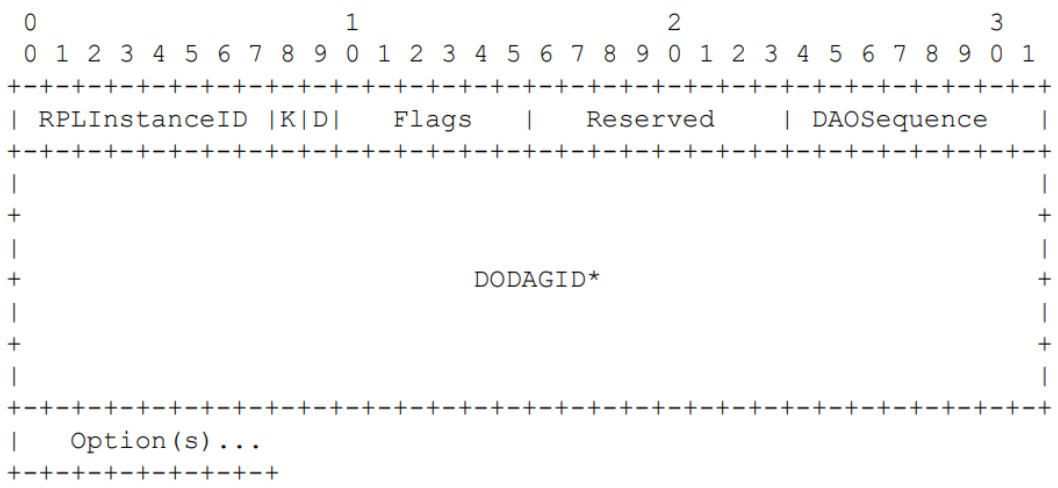
[1] Destination-Oriented DAG: Κατευθυνόμενος Άκυκλος Γράφος (DAG) με ρίζα μόνο ένα προορισμό[2].



Σχήμα 1.1 Κεφαλίδα Πακέτου μηνύματος DIO [3]

1.1.2 DAO

Τα μηνύματα DAO (Destination Advertisement Object) χρησιμοποιούνται για να μεταδώσουν, οι κόμβοι, πληροφορία σχετική με τον προορισμό τους πιο πάνω στην ιεραρχία, και σε σχέση με την επιλογή τους για τον κόμβο-πατέρα τους. Όπως φαίνεται και στην κεφαλίδα του πακέτου (Σχήμα 1.2), ένα πακέτο μηνύματος DAO περιέχει πληροφορίες σχετικά με το instance του RPL στο οποίο βρίσκεται, καθώς και σημαφόρο (flag K στο Σχήμα 1.2) μέσω του οποίου μπορεί να ζητήσει επιβεβαίωση παραλαβής του πακέτου από τον παραλήπτη. Αναλόγως με την λειτουργία του δικτύου, storing ή non-storing mode, το πακέτο γίνεται unicast από τον κόμβο-παιδί προς τον κόμβο-πατέρα ή γίνεται unicast στη ρίζα του DODAG αντίστοιχα.

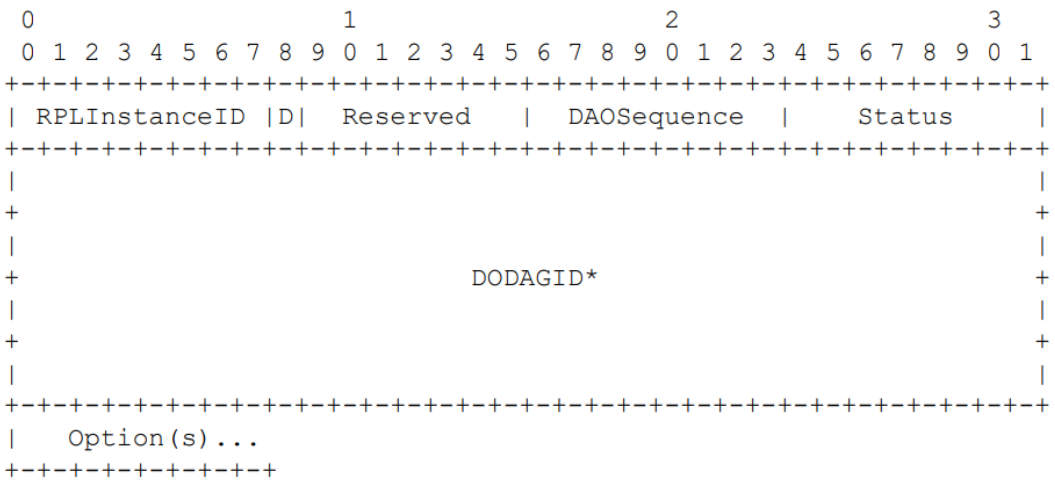


Σχήμα 1.2 Κεφαλίδα Πακέτου μηνύματος DAO [3]

1.1.3 DAO-ACK

Έπειτα από συγκεκριμένο αίτημα του κόμβου που στέλνει DAO, όπως αναφέραμε παραπάνω, ο κόμβος-παραλήπτης του αιτήματος στέλνει μήνυμα επιβεβαίωσης παραλαβής, το DAO-ACK. Όπως φαίνεται στην κεφαλίδα του πακέτου (Σχήμα 1.3), το πακέτο περιέχει πληροφορίες σχετικά με το RPL instance όπου βρίσκεται, καθώς και με το στάτους του κόμβου-παραλήπτη. Αναλόγως της τιμής που αποθηκεύεται στο πεδίο στάτους της κεφαλίδας ο κόμβος παραλήπτης στέλνει τα ακόλουθα μηνύματα:

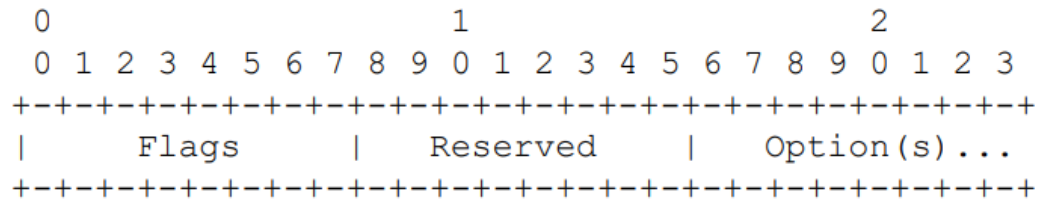
- Αποδέχεται το αίτημα DAO που έχει παραλάβει και είναι πρόθυμος να ενεργεί ως πατέρας του κόμβου από τον οποίο έλαβε το DAO.
- Είναι πρόθυμος να ενεργεί ως πατέρας του κόμβου από τον οποίο έλαβε το DAO, αλλά του προτείνει να επιλέξει άλλο κόμβο-πατέρα.
- Απορρίπτει το αίτημα DAO που έχει λάβει, δεν είναι πρόθυμος να ενεργεί ως κόμβος πατέρας.



Σχήμα 1.3 Κεφαλίδα Πακέτου μηνύματος DAO-ACK [3]

1.1.4 DIS

Τα μηνύματα DIS (DODAG Information Solicitation) στέλνονται για να ζητήσουν από κάποιο κόμβο μήνυμα DIO. Τα μηνύματα DIS χρησιμοποιούνται για να ανιχνεύσει, ένας κόμβος, ενεργά DODAG και κόμβους μέσα στην εμβέλειά του. Όπως βλέπουμε στην κεφαλίδα του πακέτου (Σχήμα 1.4), το μήνυμα δεν μεταφέρει οποιαδήποτε άλλη πληροφορία προς τον παραλήπτη του μηνύματος.



Σχήμα 1.4 Κεφαλίδα Πακέτου μηνύματος DIS [3]

Για τους σκοπούς της εργασίας, ως επί το πλείστον θα επικεντρωθούμε στην συχνότητα και τον αριθμό αποστολής μηνυμάτων DIO.

1.2 Οι Ιοί

Για να μελετήσουμε την συμπεριφορά του δικτύου σε σχέση με τους κακόβουλους κόμβους, πρέπει πρώτα να εισάγουμε ιούς στο δίκτυο. Θα εισαχθούν τέσσερεις ιοί:

- Black-hole:[1] Διαφημίζει μικρότερη απόσταση προς τον κόμβο-sink (ETX path to root), με σκοπό να προσελκύσει περισσότερα πακέτα. Ο ιός μπορεί να επιλέξει να ενεργήσει με ένα από τους ακόλουθους τρόπους:
    - να μην προωθήσει τα πακέτα που λαμβάνει.
    - να τροποποιήσει το περιεχόμενό τους.
    - να παρακολουθεί το περιεχόμενό τους.
- Flooding:[1] Στέλνει περιττά μηνύματα DIO, με σκοπό να δημιουργήσει συμφόρηση στο δίκτυο.
- Selective Forwarding:[1] Επιλέγει να μην προωθήσει συγκεκριμένα πακέτα.
- Sinkhole:[1] Μιμείται τον κόμβο-sink, διαφημίζοντας τη μικρότερη απόσταση προς το sink. Όταν λάβει πακέτα δεν τα προωθεί.

Για τους σκοπούς της εργασίας σε κάθε σενάριο υπάρχει μόνο ένας κακόβουλος κόμβος. Σε κάθε σενάριο προσομοίωσης, όπου το σενάριο έχει κακόβουλο κόμβο κατά τη διάρκεια της προσομοίωσης, ο κακόβουλος κόμβος είναι μολυσμένος μόνο με ένα ιό. Οι ιοί έχουν την δυνατότητα να επιλέξουν πότε θα ξεκινήσουν την κακόβουλη συμπεριφορά τους, με αποτέλεσμα να μπορούμε να εξετάσουμε σενάρια, κατά τα οποία επηρεάζουν το δίκτυο μετά από ένα συγκεκριμένο χρονικό διάστημα.

1.3 Μεθοδολογία

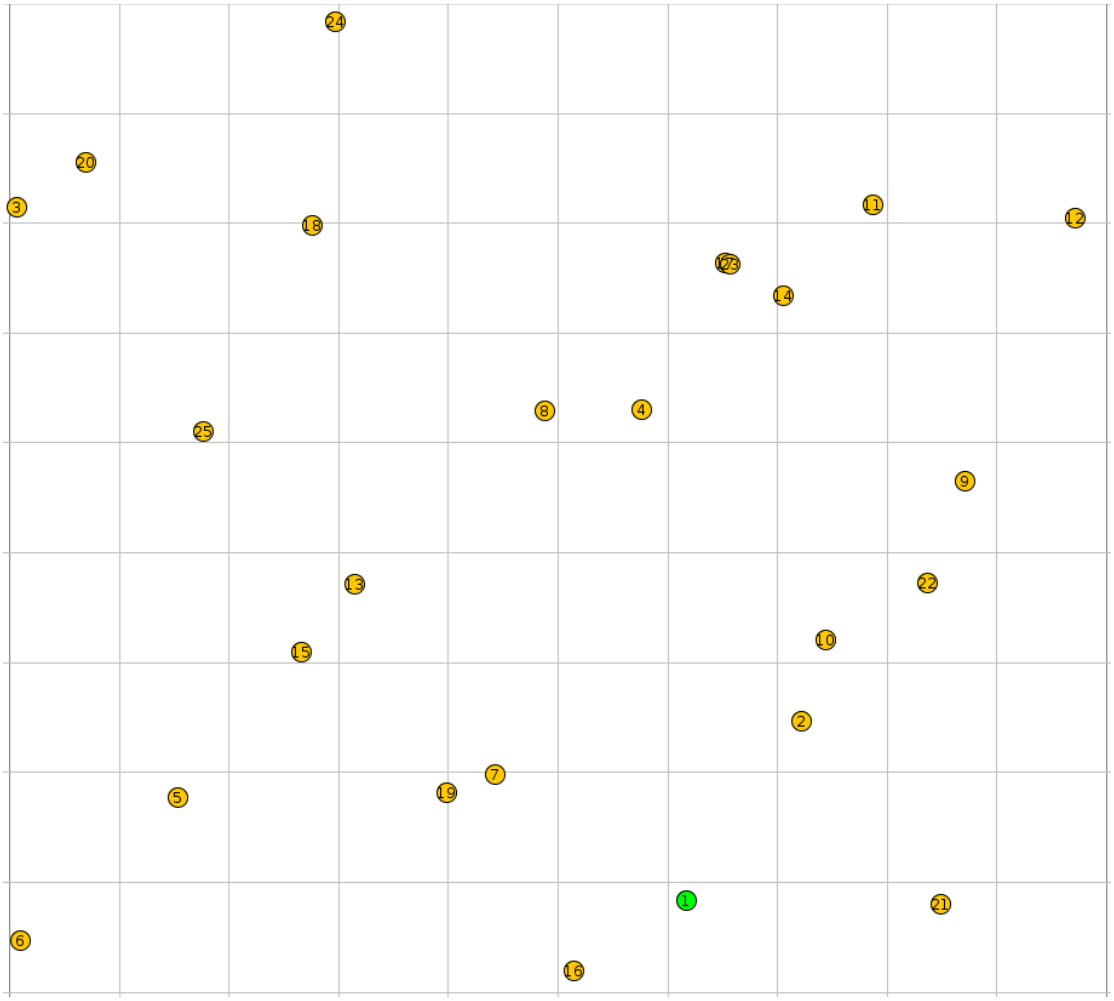
Χρησιμοποιώντας το πρόγραμμα contiki v3.0 και τον προσομοιωτή cooja τρέξαμε σενάρια με είκοσι-πέντε κόμβους και για περίοδο προσομοίωσης δεκαπέντε λεπτών. Συνολικά έτρεξαν πεντακόσιες εβδομήντα εννέα προσομοιώσεις. Για κάθε προσομοίωση συλλέχθηκαν πληροφορίες από κάθε κόμβο οι οποίες αποθηκεύτηκαν σε ένα αρχείο μορφής log file. Μέσω προγραμμάτων script, που είναι γραμμένα στη γλώσσα python, εξήξαμε τις πληροφορίες που χρειάζονται για την παρακολούθηση της συμπεριφοράς των ιών στο δίκτυο. Με τη χρήση των πληροφοριών που εξήξαμε, δημιουργήσαμε γραφικές παραστάσεις, τις οποίες θα αναλύσουμε.

Κεφάλαιο 2

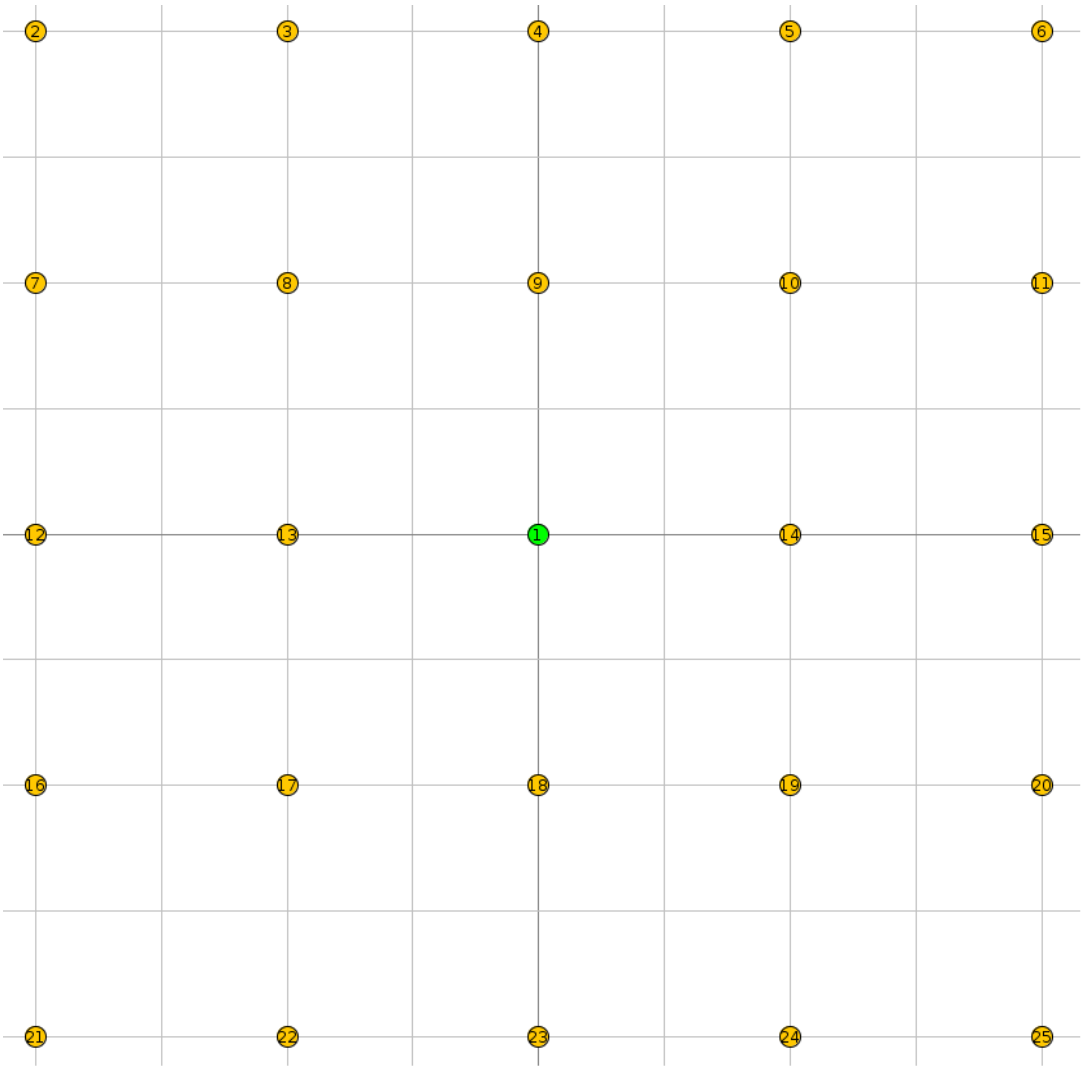
Δημιουργία και Προσομοίωση Σεναρίων

2.1 Τοπολογίες

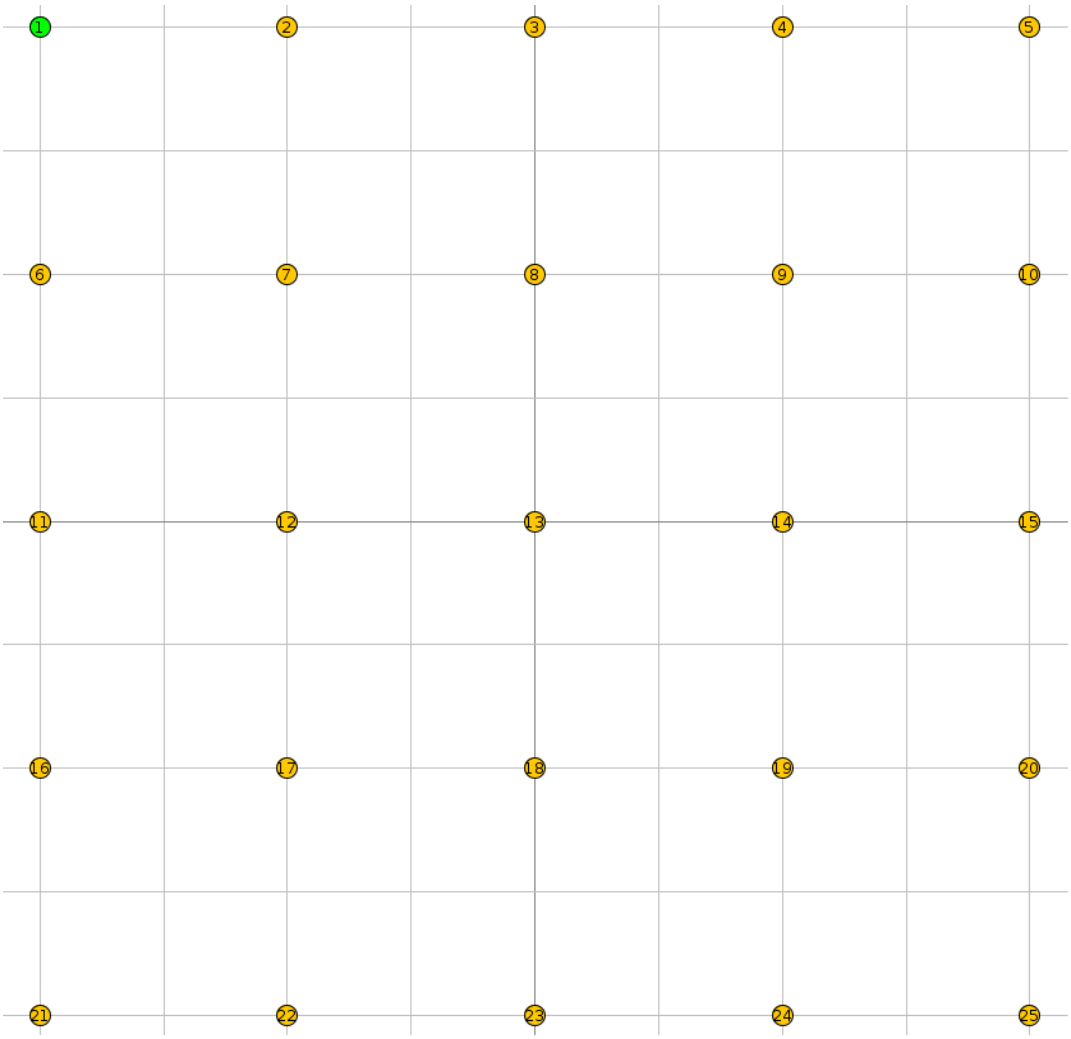
Τα σενάρια που προσομοιώθηκαν ανήκουν σε μία από τρεις τοπολογίες. Οι τοπολογίες είναι ως ακολούθως οι: Random, Sink on Top, Sink in Middle και θα αναφέρονται ως Random, Top και Middle αντίστοιχα. Για κάθε τοπολογία οι κόμβοι είναι τοποθετημένοι όπως φαίνονται στα Σχήμα 2.1, Σχήμα 2.2 και Σχήμα 2.3 πιο κάτω.



Σχήμα 2.1 Τοπολογία Random



Σχήμα 2.2 Τοπολογία Sink in Middle



Σχήμα 2.3 Τοπολογία Sink on Top

Σε όλες τις τοπολογίες και για όλα τα σενάρια ο κόμβος sink βρίσκεται πάντα στον κόμβο με ταυτότητα 1. Όπως βλέπουμε και στα πιο πάνω σχήματα, κάθε τοπολογία αποτελείται από είκοσι-πέντε κόμβους.

2.2 Σενάρια

Για τις προσομοιώσεις δημιουργήθηκαν τρεις κατηγορίες σεναρίων για κάθε τοπολογία. Στην πρώτη κατηγορία (Benign) κανένας κόμβος δεν έχει μολυνθεί από ιό. Στην δεύτερη οι μολυσμένοι κόμβοι έχουν κακόβουλη συμπεριφορά από την αρχή της προσομοίωσης και στην τρίτη έχουν κακόβουλη συμπεριφορά μετά από το χρονικό διάστημα που χρειάζεται για να ενωθούν όλοι οι κόμβοι στο δίκτυο. Στις δύο τελευταίες κατηγορίες,

για κάθε συνδυασμό τοπολογίας-ιού, δημιουργήθηκαν είκοσι-τέσσερα σενάρια. Σε κάθε ένα από αυτά τα σενάρια ένας διαφορετικός κόμβος, εκτός από τον sink, είναι μολυσμένος (κόμβος 2, κόμβος 3, κόμβος 4 κ.ο.κ.). Συνολικά, δημιουργήθηκαν πεντακόσια εβδομήντα εννέα σενάρια προσομοίωσης.

Όλοι οι κόμβοι, όλων των σεναρίων, χρησιμοποιούν το RPL σε συνδυασμό με το IP πρωτόκολλο IPv6. Επιπλέον, για να μπορέσουμε να μελετήσουμε καλύτερα την συμπεριφορά του δικτύου, αλλάξαμε την περίοδο αποστολής των πακέτων από ένα πακέτο το δευτερόλεπτο, σε ένα πακέτο κάθε δύο δευτερόλεπτα.

2.3 Μετά την Προσομοίωση

Μετά την προσομοίωσή του, κάθε σενάριο παρήγαγε ένα αρχείο logfile, το οποίο περιέχει όλες τις πληροφορίες που μπορούμε να πάρουμε για κάθε σενάριο. Το κάθε αρχείο περιέχει μεγάλο όγκο περιττής πληροφορίας, για το σκοπό μας, και μεγάλο ποσοστό raw data. Για τον λόγο αυτό δημιουργήσαμε μικρά προγράμματα (scripts) για να εξάξουμε την σημαντική, για το σκοπό μας, πληροφορία.

Κεφάλαιο 3

Επεξεργασία των Logfile

Έχουμε δημιουργήσει επτά script[2] για να παράγουν την σημαντική πληροφορία που χρειαζόμαστε από τα logfile. Τα logfile αποτελούνται από ένα σύνολο μηνυμάτων που περιέχουν πληροφορία για τον κάθε κόμβο. Όπως βλέπουμε στο Σχήμα 3.1 κάθε καταγραφή (κόκκινο) στο αρχείο αποτελείται από την χρονική στιγμή που καταγράφηκε το μήνυμα (πράσινο), τον αριθμό ταυτοποίησης του κόμβου (μπλε) και το περιεχόμενό του (πορτοκαλί). Ανάλογα με την πληροφορία που θέλει να εξάγει το κάθε script, διαβάζει διαφορετικό σημείο στην κάθε καταγραφή. Όλα τα script είναι γραμμένα στην προγραμματιστική γλώσσα python, η οποία έχει επιλεχθεί λόγω των μεγάλων δυνατοτήτων της σε εξόρυξη δεδομένων.

```
00:00.307   ID:24   Rime started with address 0.18.116.24.0.24.24.24
00:00.315   ID:24   MAC 00:12:74:18:00:18:18:18 Contiki 3.x started. Node id is set to 24.
00:00.324   ID:24   nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:00.329   ID:24   ctimer_set 0x1cc4 128
00:00.330   ID:24   RPL: Reset MRHOF
00:00.338   ID:24   Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7418:0018:1818
00:00.340   ID:24   Starting 'UDP client process'
00:00.343   ID:24   UDP client process started
00:00.344   ID:8    Rime started with address 0.18.116.8.0.8.8.8
00:00.348   ID:24   Client IPv6 addresses: aaaa::212:7418:18:1818
00:00.351   ID:24   fe80::212:7418:18:1818
00:00.352   ID:8    MAC 00:12:74:08:00:08:08:08 Contiki 3.x started. Node id is set to 8.
00:00.356   ID:24   Created a connection with the server :: local/remote port 8765/5678
00:00.361   ID:8    nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:00.366   ID:8    ctimer_set 0x1cc4 128
00:00.367   ID:8    RPL: Reset MRHOF
00:00.375   ID:8    Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7408:0008:0808
00:00.377   ID:8    Starting 'UDP client process'
00:00.379   ID:8    UDP client process started
00:00.384   ID:8    Client IPv6 addresses: aaaa::212:7408:8:808
00:00.387   ID:8    fe80::212:7408:8:808
```

Σχήμα 3.1 Παράδειγμα από ένα Logfile

3.1 totals.py

Το script αυτό διαβάζει το logfile και για κάθε logfile υπολογίζει και παράγει τις συνολικές τιμές για τα πιο κάτω:

- Συνολικό αριθμό πακέτων που έλαβε ο κόμβος
- Συνολικό αριθμό πακέτων που προώθησε ο κόμβος
- Συνολικό αριθμό πακέτων που έστειλε ο κόμβος (δεν συμπεριλαμβάνονται τα πακέτα που προώθησε).
- Συνολικό αριθμό πακέτων που απέρριψε ο κόμβος
- Συνολικό αριθμό μηνυμάτων DIO που έστειλε ο κόμβος
- Το τελευταίο μήκος για το μονοπάτι προς sink που διαφήμισε ο κόμβος

[2] Περισσότερες πληροφορίες σχετικά με την υλοποίηση των script βρίσκονται στο Παράρτημα Α.

- Το throughput της προσομοίωσης
- Την συνολική απώλεια πακέτων της προσομοίωσης

Για να υπολογίσει τα πιο πάνω, το script χρησιμοποιεί καταγραφές όπου οι κόμβοι ανακοινώνουν ότι έχουν λάβει, προωθήσει, στείλει ή απορρίψει ένα πακέτο ή έχουν στείλει ένα μήνυμα DIO. Έπειτα αφαιρεί των αριθμό των πακέτων που προωθήθηκαν από τον αριθμό των πακέτων που στάλθηκαν, έτσι ώστε ο αριθμός αυτός να αντιπροσωπεύει μόνο τα πακέτα που δημιουργήθηκαν από τον κόμβο. Εν τέλει χρησιμοποιεί τα τελικά αποτελέσματα για να υπολογίσει το throughput και την συνολική απώλεια πακέτων και εμφανίζει τα αποτελέσματα σε μορφή όπως φαίνεται στο Σχήμα 3.2.

```
ID: 17  Received: 709, Forwarded: 709, Sent: 446, Dropped: 56.
        DIO packets: 38, ETX path to root: 5.42.
ID: 18  Received: 323, Forwarded: 323, Sent: 448, Dropped: 53.
        DIO packets: 15, ETX path to root: 6.20.
ID: 19  Received: 84, Forwarded: 84, Sent: 448, Dropped: 53.
        DIO packets: 21, ETX path to root: 8.39.
ID: 20  Received: 126, Forwarded: 126, Sent: 446, Dropped: 59.
        DIO packets: 21, ETX path to root: 10.23.
ID: 21  Received: 1038, Forwarded: 1038, Sent: 444, Dropped: 46.
        DIO packets: 7, ETX path to root: 5.91.
ID: 22  Received: 885, Forwarded: 885, Sent: 451, Dropped: 49.
        DIO packets: 22, ETX path to root: 5.92.
ID: 23  Received: 916, Forwarded: 916, Sent: 451, Dropped: 50.
        DIO packets: 15, ETX path to root: 7.73.
ID: 24  Received: 449, Forwarded: 449, Sent: 445, Dropped: 48.
        DIO packets: 8, ETX path to root: 8.48.
ID: 25  Received: 0, Forwarded: 0, Sent: 448, Dropped: 37.
        DIO packets: 20, ETX path to root: 11.60.
**********************************************************************
        Throughput: 5.89 packets per second.
        Packet loss: 5486 packets.
**********************************************************************
```

Σχήμα 3.2 Παράδειγμα αποτελέσματος totals.py

3.2 interval_totals.py

Το script αυτό διαβάζει το logfile και για κάθε logfile υπολογίζει και παράγει τις συνολικές τιμές για τα πιο κάτω για κάθε χρονικό διάστημα[3]:

- Συνολικό αριθμό πακέτων που έλαβε ο κόμβος
- Συνολικό αριθμό πακέτων που προώθησε ο κόμβος
- Συνολικό αριθμό πακέτων που έστειλε ο κόμβος (δεν συμπεριλαμβάνονται τα πακέτα που προώθησε).
- Συνολικό αριθμό πακέτων που απέρριψε ο κόμβος
- Συνολικό αριθμό μηνυμάτων DIO που έστειλε ο κόμβος
- Το τελευταίο μήκος για το μονοπάτι προς sink που διαφήμισε ο κόμβος
- Το throughput της προσομοίωσης
- Την συνολική απώλεια πακέτων της προσομοίωσης

Για να υπολογίσει τα πιο πάνω, το script χρησιμοποιεί καταγραφές, οι οποίες βρίσκονται μέσα σε συγκεκριμένο χρονικό διάστημα. Για κάθε χρονικό διάστημα διαβάζει τις καταγραφές όπου οι κόμβοι ανακοινώνουν ότι έχουν λάβει, προωθήσει, στείλει ή απορρίψει ένα πακέτο ή έχουν στείλει ένα μήνυμα DIO. Έπειτα αφαιρεί των αριθμό των πακέτων που προωθήθηκαν από τον αριθμό των πακέτων που στάλθηκαν και χρησιμοποιεί τα τελικά αποτελέσματα για να υπολογίσει το throughput και την συνολική απώλεια πακέτων εμφανίζοντάς τα για κάθε διάστημα (Σχήμα 3.3). Εν τέλει υπολογίζει και τυπώνει τις μέγιστες και ελάχιστες τιμές για την απώλεια πακέτων και το throughput της προσομοίωσης (Σχήμα 3.4), καθώς και το διάστημα στο οποίο έχουν παρουσιασθεί.

[3] Στην περίπτωσή μας τα χρονικά διαστήματα διαρκούν ένα λεπτό.

```
************************************************************
Running for logs of upto 15 minutes in intervals of 1 minutes.
************************************************************
+----------------------------------------------------------+
+---------                Inetrval 1                ---------+
+----------------------------------------------------------+
ID: 1   Received: 393, Forwarded: 0, Sent: 0, Dropped: 459.
        DIO packets: 4, ETX path to root: 0.0.
ID: 2   Received: 98, Forwarded: 98, Sent: 29, Dropped: 23.
        DIO packets: 3, ETX path to root: 1.0.
ID: 3   Received: 0, Forwarded: 0, Sent: 29, Dropped: 4.
        DIO packets: 3, ETX path to root: 7.80.
ID: 4   Received: 0, Forwarded: 0, Sent: 29, Dropped: 12.
        DIO packets: 4, ETX path to root: 4.7.
ID: 5   Received: 23, Forwarded: 23, Sent: 29, Dropped: 12.
        DIO packets: 4, ETX path to root: 2.14.
ID: 6   Received: 0, Forwarded: 0, Sent: 28, Dropped: 5.
        DIO packets: 3, ETX path to root: 4.4.
```

Σχήμα 3.3 Παράδειγμα αποτελέσματος interval_totals.py

```
************************************************************
    Minimum throughput: 4.00 packets per second. (interval 0)
    Maximum throughput: 6.42 packets per second. (interval 4)
    Minimum packet loss: 337 packets. (interval 4)
    Maximum packet loss: 439 packets. (interval 0)
************************************************************
```

Σχήμα 3.4 Παράδειγμα αποτελέσματος interval_totals.py

3.3 end_to_end_delay.py

Το script αυτό διαβάζει το logfile και υπολογίζει και παράγει τον συνολικό χρόνο, σε χιλιοστά του δευτερολέπτου (ms), που πέρασε από τη στιγμή που το πακέτο στάλθηκε από τον κόμβο προς το sink, μέχρι τη χρονική στιγμή που παραλήφθηκε από το sink. Οι χρόνοι αυτοί εμφανίζονται σε ζευγάρια με τις χρονικές στιγμές που τα πακέτα έχουν σταλεί και κατηγοριοποιούνται με βάσει τον κόμβο από τον οποίο έχουν αποσταλεί τα πακέτα αρχικά. Το αποτέλεσμα αυτό εμφανίζεται σε μορφή αρχείου csv (Σχήμα 3.5).

	A	B	C
1	node 2		
2	00:13.564	1991	
3	00:18.162	5823	
4	00:19.471	5526	
5	00:21.744	6891	
6	00:24.229	5230	
7	00:25.394	7232	
8	00:39.314	876	
9	00:31.916	9608	

Σχήμα 3.5 Παράδειγμα αποτελέσματος end_to_end_delay.py

3.4 forwarding_delay.py

Το script αυτό διαβάζει το logfile και υπολογίζει και παράγει τον μέσο όρο του χρόνου που χρειάστηκε ο κάθε κόμβος για να προωθήσει τα πακέτα που έλαβε προς τον sink, για κάθε χρονικό διάστημα. Ο χρόνος υπολογίζεται σε χιλιοστά του δευτερολέπτου. Ακόμη υπολογίζει και εμφανίζει τον μέγιστο και ελάχιστο χρόνο καθυστέρησης του κάθε κόμβου για κάθε διάστημα (Σχήμα 3.6).

```
+-------------------------------------------------
+---------                  Inetrval 1
+-------------------------------------------------
ID:1    This is the sink.
ID:2    Average Delay: 7.67 (min: 7ms, max: 8ms).
ID:3    Average Delay: 7.35 (min: 7ms, max: 8ms).
ID:4    Average Delay: 7.32 (min: 7ms, max: 8ms).
ID:5    No packets were Received.
ID:6    Average Delay: 7.98 (min: 7ms, max: 10ms).
ID:7    Average Delay: 7.58 (min: 7ms, max: 8ms).
```

Σχήμα 3.6 Παράδειγμα αποτελέσματος forwarding_delay.py

3.5 setuptime.py

Το script αυτό διαβάζει το logfile και υπολογίζει το χρόνο που χρειάστηκε για να ενωθούν όλοι οι κόμβοι στον γράφο[4] που σχημάτισε το RPL για το δίκτυο. Ο χρόνος αυτός εκτυπώνεται σε δευτερόλεπτα με ακρίβεια δύο δεκαδικών ψηφίων (Σχήμα 3.7). Για να υπολογίσει το χρόνο αυτό παρακολουθεί τις επικοινωνίες μεταξύ κόμβων ψάχνοντας για μοτίβο επικοινωνίας DAO και DAO-ACK, η οποία γίνεται μέσω των διευθύνσεων IPv6 link-local[5] των κόμβων.

```
1 Tree set up time is 30.00s.
```

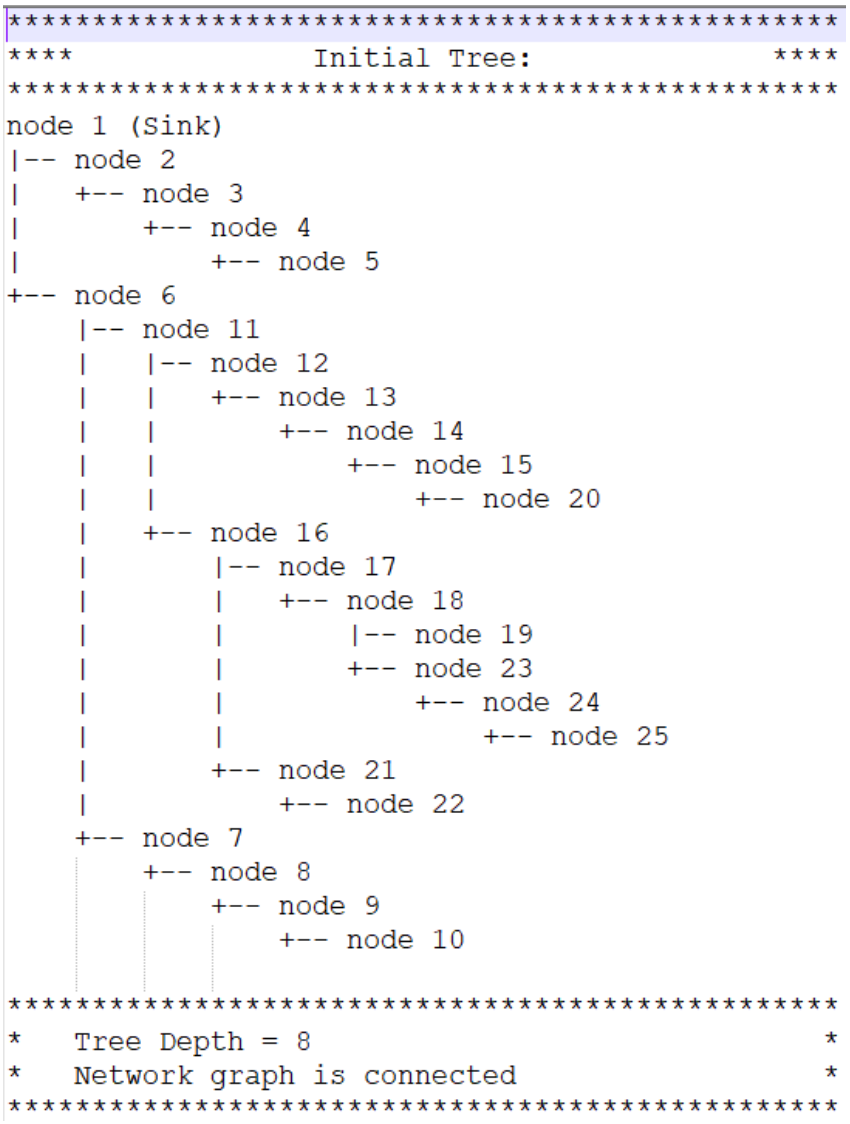
Σχήμα 3.7 Παράδειγμα αποτελέσματος setuptime.py

3.6 init_tree.py

Το script αυτό διαβάζει το logfile και υπολογίζει το πρώτο δέντρο που σχηματίζεται κατά την προσομοίωση. Παρακολουθεί την επικοινωνία μεταξύ των κόμβων μέσω των διευθύνσεων IPv6 link-local των κόμβων. Όταν εμφανιστεί το συγκεκριμένο μοτίβο επικοινωνίας DAO και DAO-ACK, το script προσθέτει τη σχέση στο δέντρο. Το δέντρο εκτυπώνεται σε δισδιάστατη γραφική αναπαράσταση με χρήση χαρακτήρων ASCII, όπως φαίνεται στο Σχήμα 3.8. Έπειτα υπολογίζει και εμφανίζει το βάθος του δέντρου και εάν το δέντρο που δημιουργείται συνδέει όλους τους κόμβους του δικτύου.

[4] Για τους σκοπούς της εργασίας, και λόγω των ιδιοτήτων του γράφου ως κατευθυνόμενος και άκυκλος, χειριζόμαστε τον γράφο ως δέντρο, όπου τα φύλλα αποτελούνται από τους κόμβους-παιδιά και στη ρίζα έχουμε τον κόμβο sink.

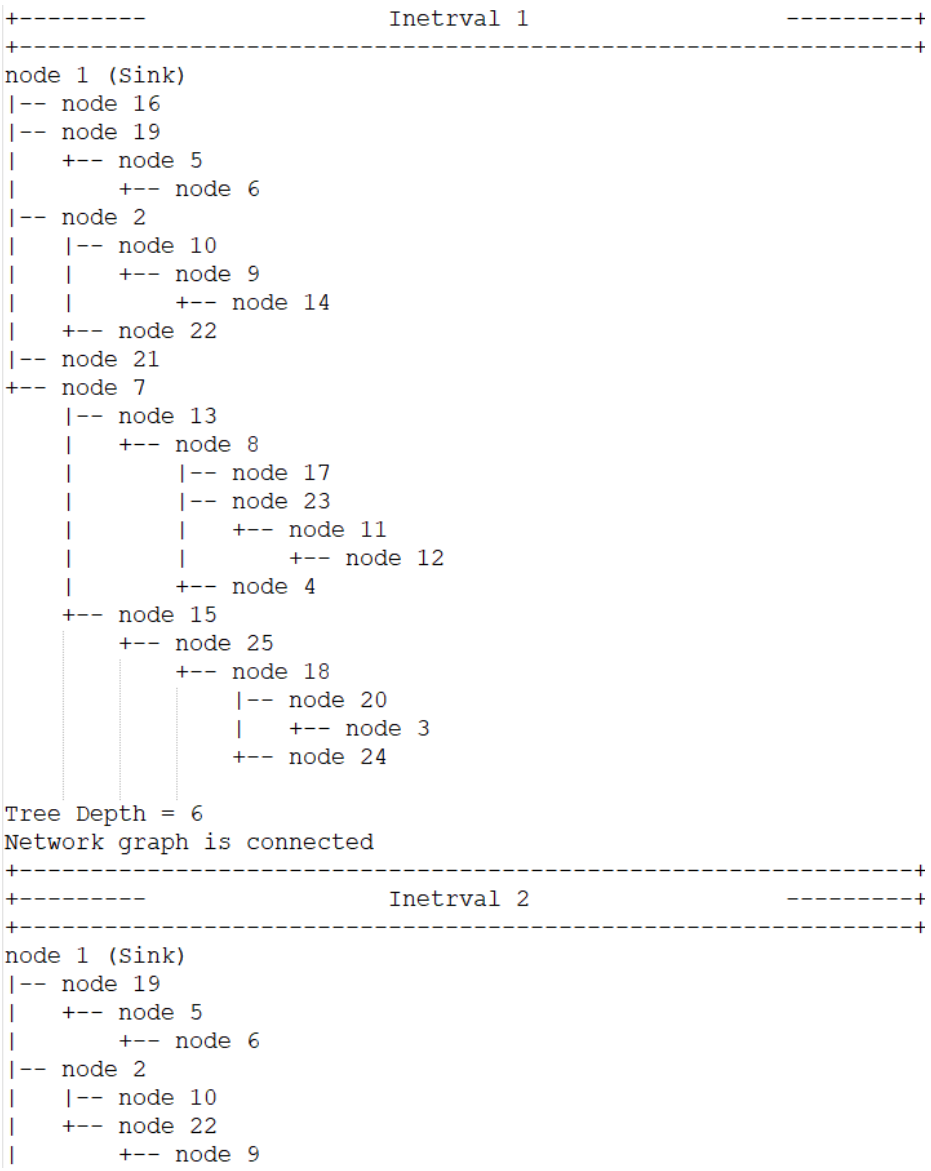
[5] Στο πρωτόκολλο IPv6 εκτός από τη διεύθυνση global unicast, μπορούμε να έχουμε κι άλλες διευθύνσεις όπως την link-local και την unique local.



Σχήμα 3.8 Παράδειγμα αποτελέσματος init_tree.py

3.7 trees.py

Το script αυτό διαβάζει το logfile και υπολογίζει το πρώτο δέντρο που σχηματίζεται σε κάθε χρονικό διάστημα. Όπως και το init_tree.py πιο πάνω, παρακολουθεί την επικοινωνία μεταξύ των κόμβων μέσω των διευθύνσεων IPv6 link-local των κόμβων ψάχνοντας το συγκεκριμένο μοτίβο επικοινωνίας DAO και DAO-ACK, μέχρι να σχηματίσει το δέντρο. Έπειτα εκτυπώνει το δέντρο με την ίδια δισδιάστατη γραφική αναπαράσταση ASCII (Σχήμα 3.9). Ακολούθως εμφανίζει το βάθος του δέντρου και εάν το δέντρο που δημιουργείται συνδέει όλους τους κόμβους του δικτύου.



Σχήμα 3.9 Παράδειγμα αποτελέσματος trees.py

Κεφάλαιο 4

Επεξεργασία Αποτελεσμάτων

4.1 Προεπεξεργασία Αποτελεσμάτων

Τα αποτελέσματα που εξήχθησαν από τα script, που αναφέρονται στο Κεφάλαιο 3, χρειάζονται περαιτέρω επεξεργασία πριν μπορέσουμε να τα αναλύσουμε σε σχέση με την κακόβουλη συμπεριφορά των ιών. Στόχος αυτής της περαιτέρω επεξεργασίας αποτελεί τη σχεδίαση γραφικών παραστάσεων. Για να σχεδιαστούν οι γραφικές χρειάζεται να αλλάξουμε την μορφή των πληροφοριών έτσι ώστε να μπορέσουν να διοχετευθούν ως είσοδος σε πρόγραμμα δημιουργίας γραφικών παραστάσεων. Οι γραφικές θα σχεδιαστούν μέσω προγραμμάτων που χρησιμοποιούν την βιβλιοθήκη matplotlib.pyplot της python. Επομένως, χρειάζονται script που θα μεταφράσουν τα αποτελέσματα σε μορφή που μπορεί να χρησιμοποιήσει η matplotlib.pyplot. Για την επίτευξη αυτού υλοποιήσαμε και χρησιμοποιήσαμε δύο script, στα οποία θα αναφερθούμε ακολούθως.

4.1.1 csv_convert.py

Το script αυτό διαβάζει το csv αρχείο που δημιουργεί το *end_to_end_delay.py*, μετατρέπει τη χρονική στιγμή σε άθροισμα χιλιοστών του δευτερολέπτου και δημιουργεί ένα καινούριο csv αρχείο (Σχήμα 4.1). Ακόμη μεταφέρει την κατηγορία στον y-άξονα, έτσι ώστε να επιτευχθεί η διαφοροποίηση της κάθε κατηγορίας με δικό της χρώμα.

	A	B	C
1		node 2	
2	13564	1991	
3	18162	5823	
4	19471	5526	
5	21744	6891	
6	24229	5230	
7	25394	7232	
8	39314	876	
9	31916	9608	

Σχήμα 4.1 Παράδειγμα αποτελέσματος csv_convert.py

4.1.2 file_convert.py

Το script αυτό διαβάζει το csv αρχείο που δημιουργεί το *interval_totals.py* και δημιουργεί δύο νέα αρχεία . Στο πρώτο (Σχήμα 4.2) αρχείο αποθηκεύει τις τιμές για την απώλεια πακέτων και το throughput κάθε διαστήματος. Στο δεύτερο αρχείο (Σχήμα 4.3) αποθηκεύει δεδομένα για τα πακέτα που παραλαμβάνονται, προωθούνται, στέλνονται, απορρίπτονται και τα μηνύματα DIO για κάθε διάστημα και τα ταξινομεί με βάση τον κόμβο στον οποίο αναφέρονται.

```
6.8, 8.5, 9.38, 9.35, 8.2, 8.53, 8.88, 9.13, 8.3, 9.75, 9.35, 9.58, 8.82, 9.02, 8.78,
271, 213, 177, 182, 263, 250, 214, 201, 246, 164, 183, 175, 211, 194, 209,
```

Σχήμα 4.2 Παράδειγμα αποτελέσματος file_convert.py

```
ID: 1   Received: 541, Forwarded: 0, Sent: 5, Dropped: 546. DIO packets: 0, no ETX path to root.
ID: 1   Received: 527, Forwarded: 0, Sent: 6, Dropped: 544. DIO packets: 0, no ETX path to root.
ID: 2   Received: 20, Forwarded: 13, Sent: 8, Dropped: 23. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 30, Forwarded: 13, Sent: 11, Dropped: 35. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 15, Forwarded: 7, Sent: 36, Dropped: 17. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 19, Forwarded: 5, Sent: 39, Dropped: 28. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 25, Forwarded: 15, Sent: 41, Dropped: 25. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 45, Forwarded: 20, Sent: 40, Dropped: 39. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 45, Forwarded: 19, Sent: 39, Dropped: 43. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 32, Forwarded: 9, Sent: 39, Dropped: 38. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 32, Forwarded: 16, Sent: 42, Dropped: 31. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 40, Forwarded: 13, Sent: 41, Dropped: 46. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 19, Forwarded: 10, Sent: 41, Dropped: 25. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 32, Forwarded: 14, Sent: 37, Dropped: 27. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 14, Forwarded: 8, Sent: 35, Dropped: 21. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 40, Forwarded: 12, Sent: 39, Dropped: 39. DIO packets: 0, ETX path to root: 0.50.
ID: 2   Received: 39, Forwarded: 15, Sent: 41, Dropped: 36. DIO packets: 0, ETX path to root: 0.50.
ID: 3   Received: 4, Forwarded: 4, Sent: 35, Dropped: 19. DIO packets: 8, ETX path to root: 4.17.
ID: 3   Received: 6, Forwarded: 6, Sent: 33, Dropped: 12. DIO packets: 7, ETX path to root: 1.57.
ID: 3   Received: 7, Forwarded: 7, Sent: 31, Dropped: 7. DIO packets: 6, ETX path to root: 1.62.
```

Σχήμα 4.3 Παράδειγμα αποτελέσματος file_convert.py

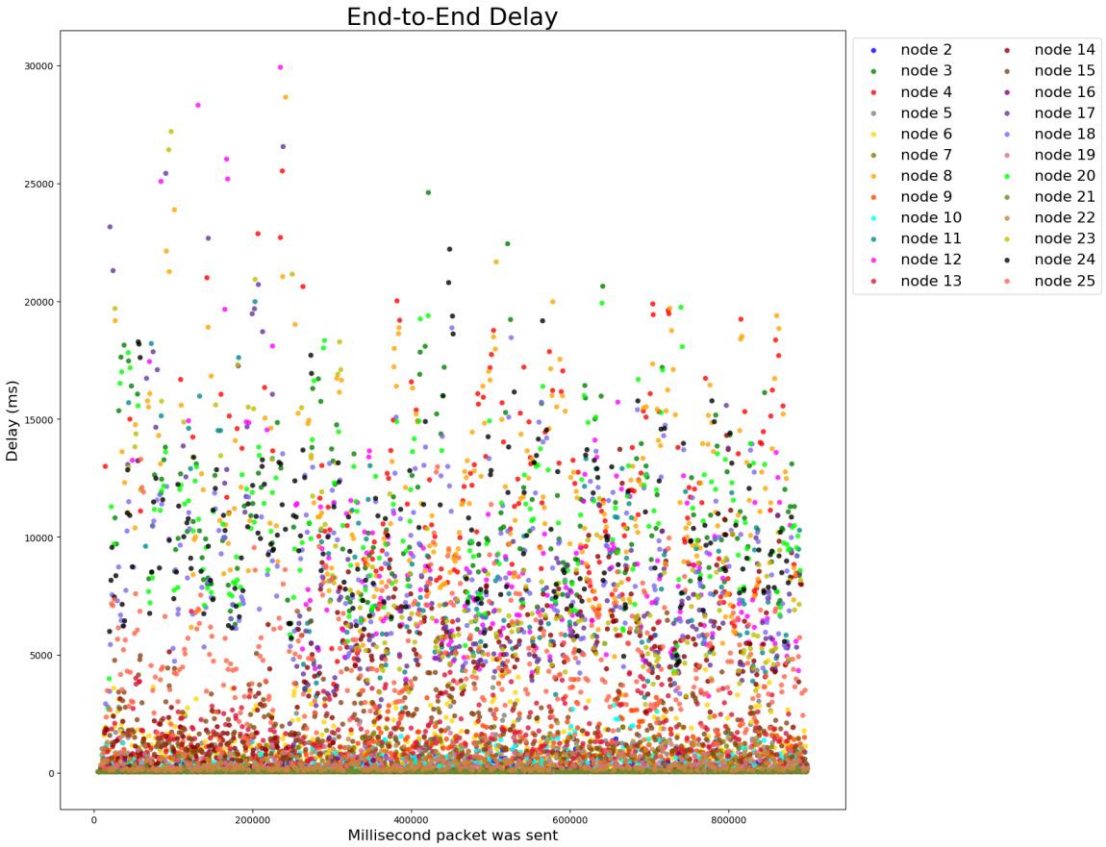
4.2 Δημιουργία Γραφικών Παραστάσεων

Μετά το μετασχηματισμό των αποτελεσμάτων, τα δεδομένα είναι σε μορφή που τους επιτρέπει να σχεδιαστούν σε γραφικές παραστάσεις. Για τη σχεδίαση των γραφικών χρησιμοποιούνται τα ακόλουθα script[6] στην γλώσσα python.

[6] Περισσότερες πληροφορίες σχετικά με την υλοποίηση των script βρίσκονται στο Παράρτημα Β.

4.2.1 scatterplot.py

Το script διαβάζει το csv που δημιουργήθηκε πιο πάνω και δημιουργεί μία γραφική παράσταση τύπου διασποράς. Για κάθε κόμβο η καθυστέρηση εμφανίζεται με δικό του χρώμα (Σχήμα 4.4). Συνολικά έχουν δημιουργηθεί πεντακόσιες εβδομήντα εννέα γραφικές διασποράς.



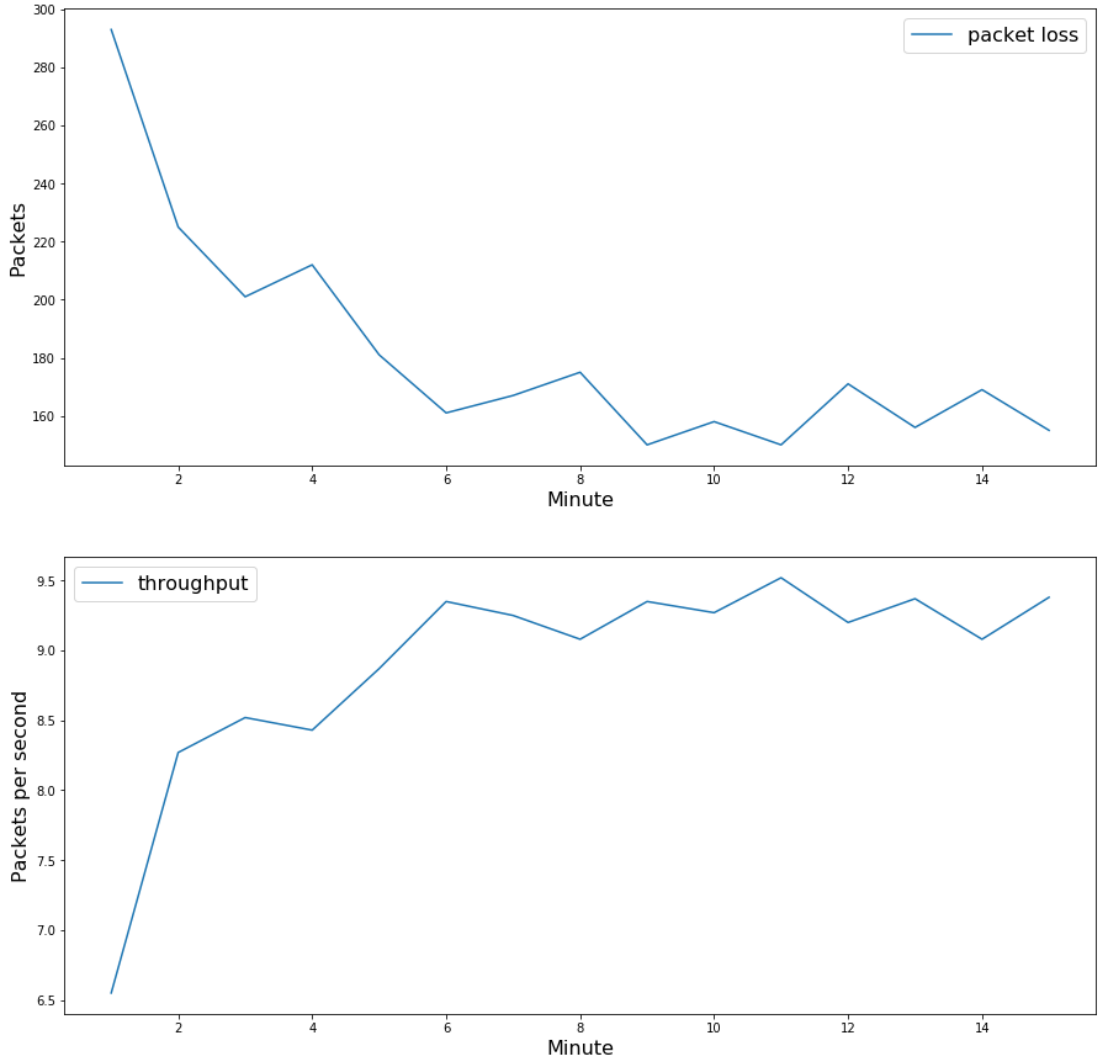
Σχήμα 4.4 Γραφική παράσταση Διασποράς

4.2.2 Για τα σενάρια benign

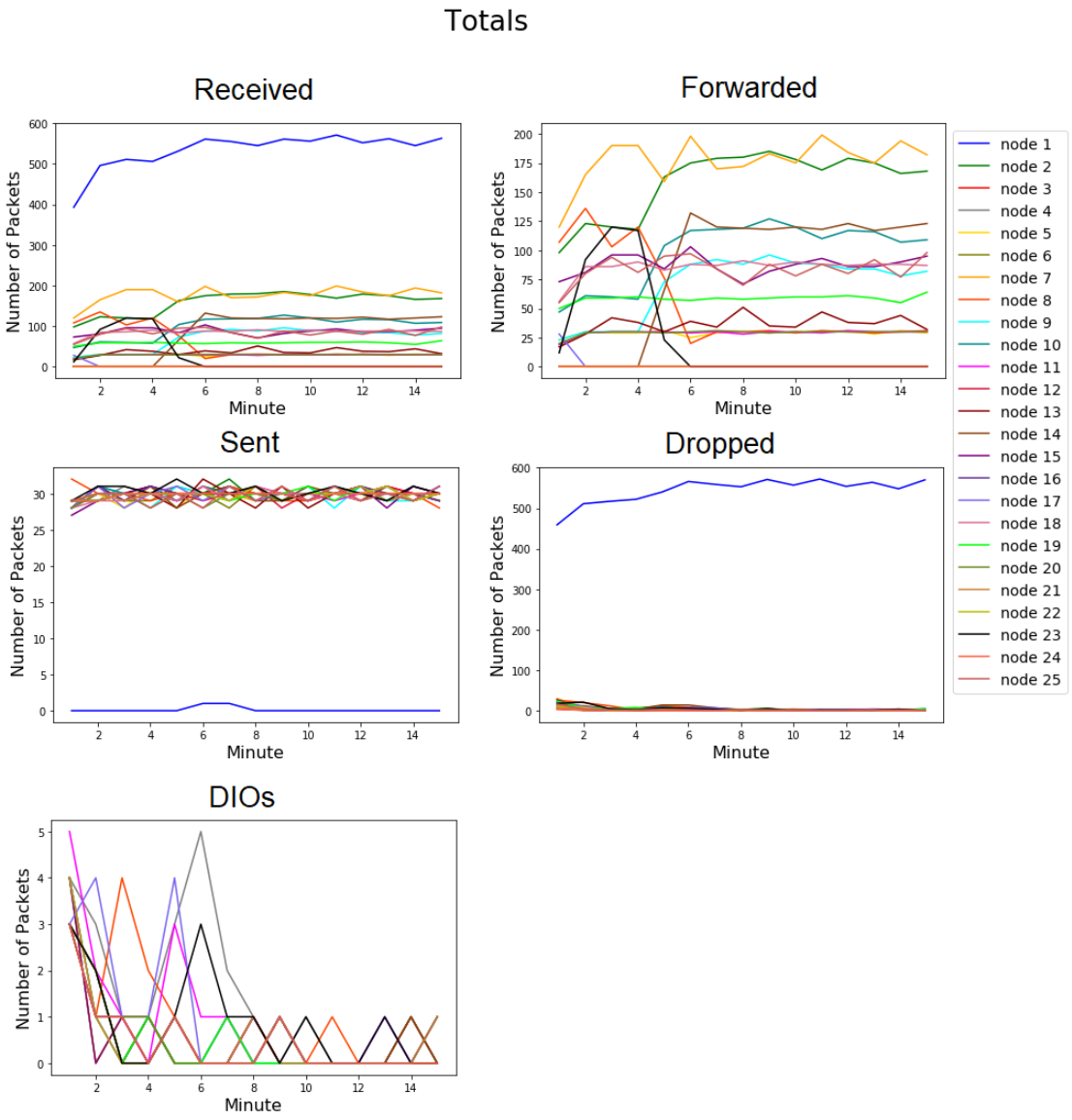
Τα ακόλουθα script είναι σχεδιασμένα μόνο για τα τρία σενάρια κατηγορίας Benign, καθώς δεν χρειάζεται να δημιουργηθεί ξεχωριστή παράσταση για τον κακόβουλο κόμβο. Συνολικά έχουν δημιουργηθεί τρεις γραφικές για κάθε είδος γραφικής που δημιουργεί το κάθε script.

4.2.2.1 totalsplot.py

Το script διαβάζει το αρχείο που έχει δημιουργηθεί από το *interval_totals.py* και υπολογίζει και δημιουργεί γραφικές παραστάσεις (Σχήμα 4.5) για την απώλεια πακέτων και για το throughput. Έπειτα δημιουργεί γραφικές παραστάσεις (Σχήμα 4.6) για τα σύνολα των πακέτων που έχουν παραληφθεί, προωθηθεί, αποσταλεί, απορριφθεί και που μεταφέρουν μηνύματα DIO.



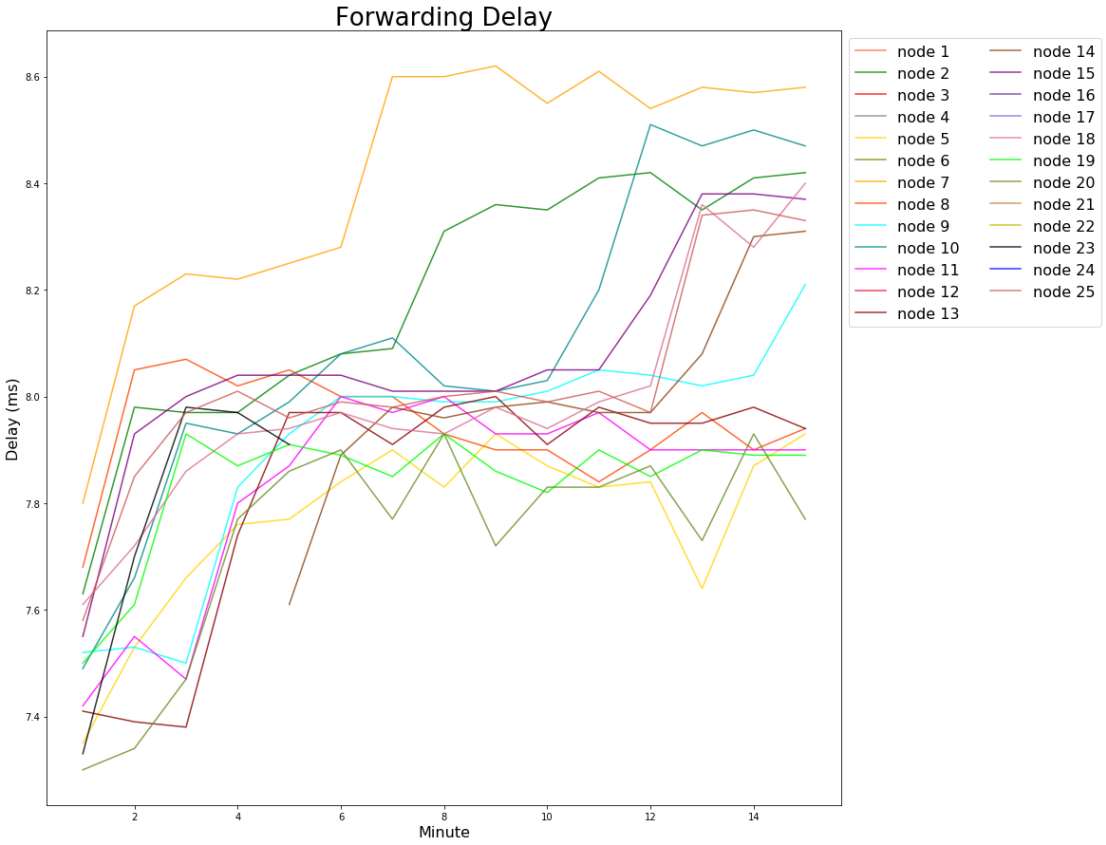
Σχήμα 4.5 Γραφική παράσταση throughput και packet loss



Σχήμα 4.6 Γραφική παράσταση συνόλων των πακέτων

4.2.2.2 forward_delay_plot.py

Το script διαβάζει το αρχείο που έχει δημιουργηθεί από το *forwarding_delay.py* και υπολογίζει και δημιουργεί μία γραφική παράσταση για την καθυστέρηση προώθησης των πακέτων για κάθε κόμβο (Σχήμα 4.7).



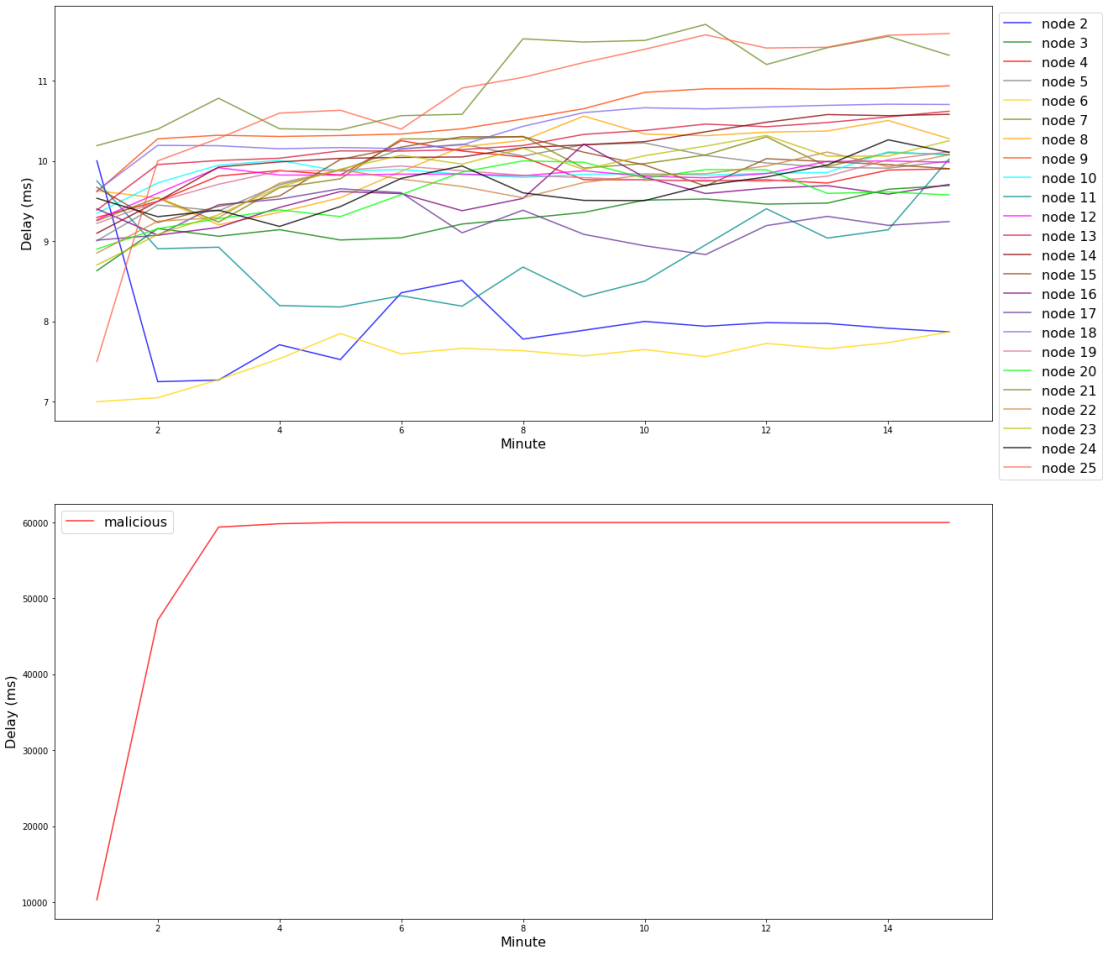
Σχήμα 4.7 Γραφική παράσταση forwarding delay

4.2.3 Για τα σενάρια malicious

Τα ακόλουθα script είναι σχεδιασμένα μόνο για τα σενάρια malicious. Συνολικά έχουν δημιουργηθεί είκοσι-τέσσερεις γραφικές για κάθε είδος γραφικής που δημιουργεί το κάθε script.

4.2.3.1 average_forwarding.py

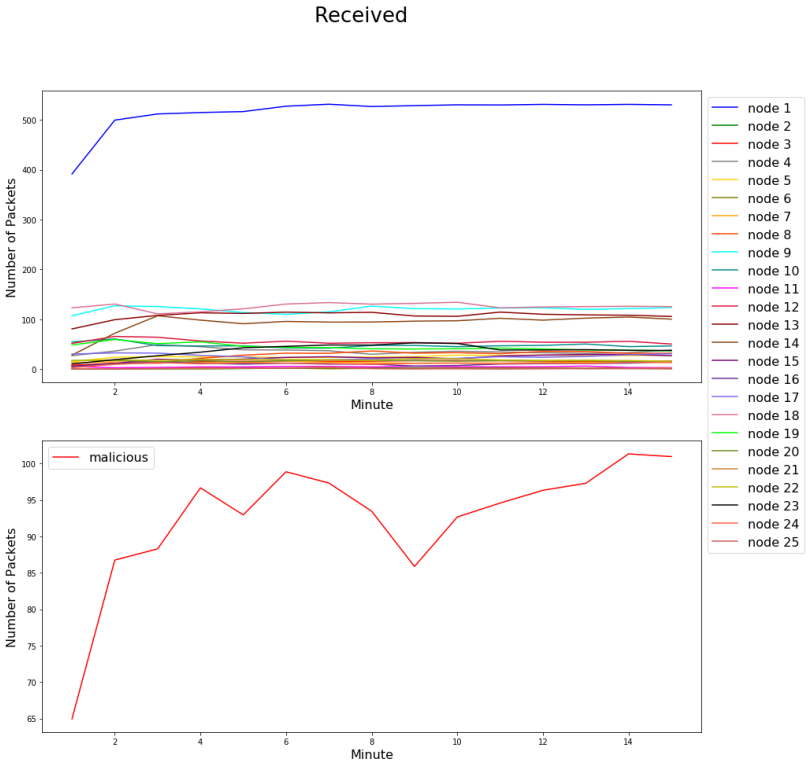
Το script διαβάζει το αρχείο που έχει δημιουργηθεί στο *forwarding_delay.py* για κάθε ένα από τα είκοσι-τέσσερα σενάρια για τον συγκεκριμένο συνδυασμό τοπολογίας-ιού και υπολογίζει τον μέσο όρο του forwarding delay για κάθε κόμβο και για τους μολυσμένους ιούς ξεχωριστά. Αφού υπολογίσει τους μέσους όρους σχεδιάζει μία γραφική τους παράσταση για τους υγιείς κόμβους (benign) και μία για τον κακόβουλο (Σχήμα 4.8).



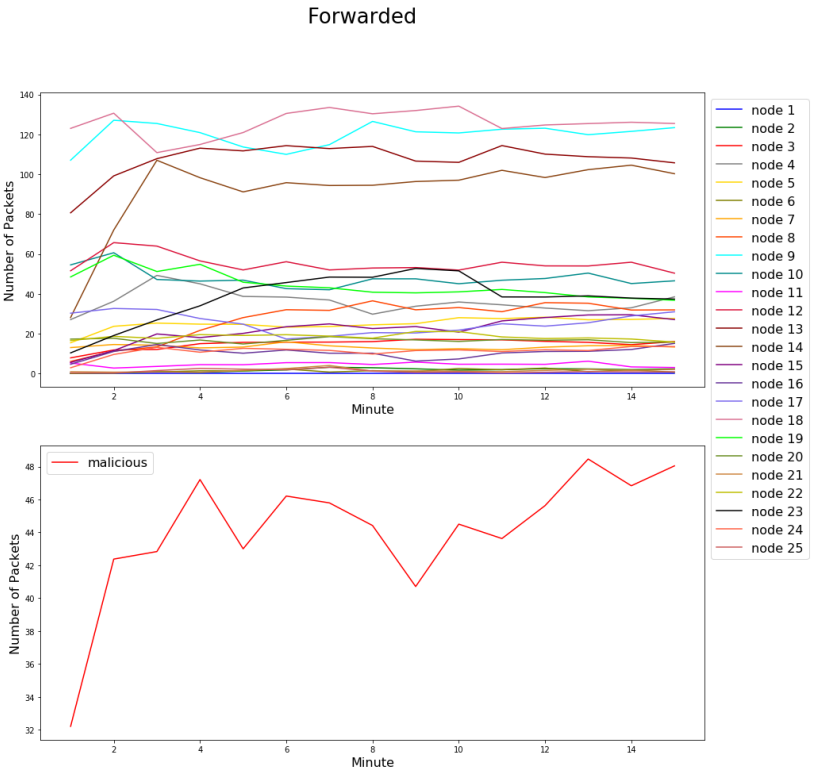
Σχήμα 4.8 Γραφική παράσταση forwarding delay με κακόβουλο κόμβο

4.2.3.2 average_totals.py

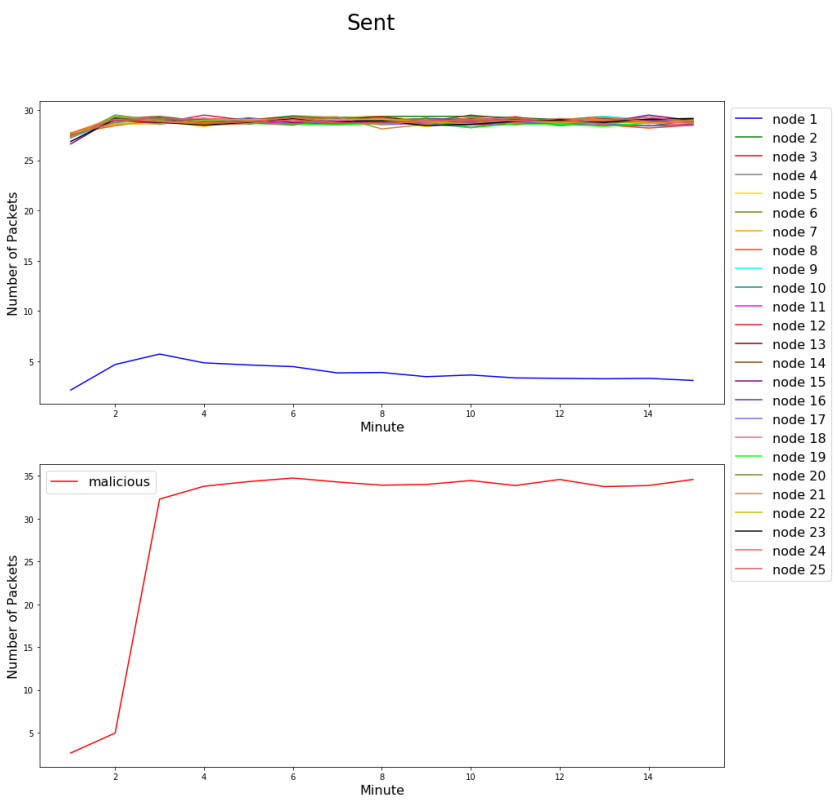
Το script διαβάζει το αρχείο που έχει δημιουργηθεί πιο πάνω στο κεφάλαιο για κάθε ένα από τα είκοσι-τέσσερα σενάρια για τον ίδιο συνδυασμό τοπολογίας-ιού και υπολογίζει τον μέσο όρο των πακέτων που έχουν παραληφθεί, προωθηθεί, αποσταλεί, απορριφθεί και που μεταφέρουν μηνύματα DIO, για κάθε κόμβο και για τους μολυσμένους ιούς ξεχωριστά. Αφού υπολογίσει τους μέσους όρους σχεδιάζει δύο γραφικές παραστάσεις για κάθε σύνολο, μία για τους υγιείς κόμβους και μία για τους κακόβουλους (Σχήμα 4.9 μέχρι Σχήμα 4.13). Έπειτα υπολογίζει τον μέσο όρο του throughput και της απώλειας πακέτων και σχεδιάζει την γραφική τους παράσταση (Σχήμα 4.14).



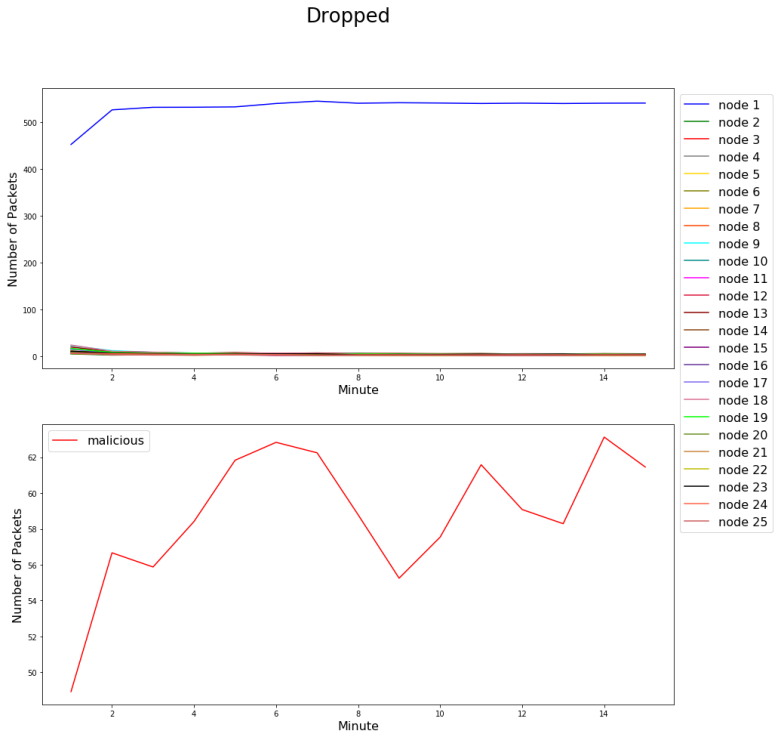
Σχήμα 4.9 Γραφική πακέτων που έχουν ληφθεί



Σχήμα 4.10 Γραφική πακέτων που έχουν προωθηθεί

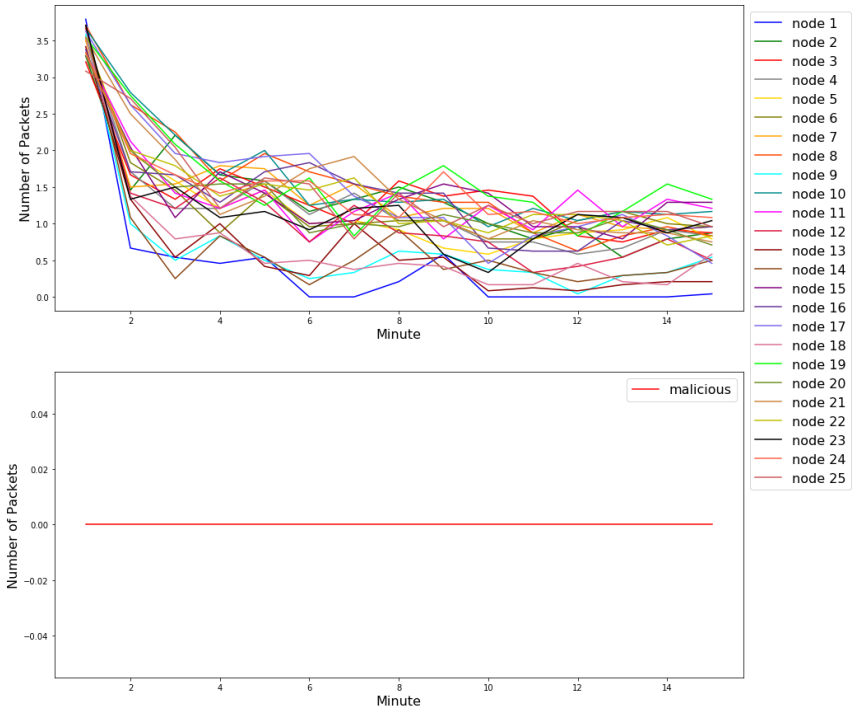


Σχήμα 4.11 Γραφική πακέτων που έχουν σταλεί

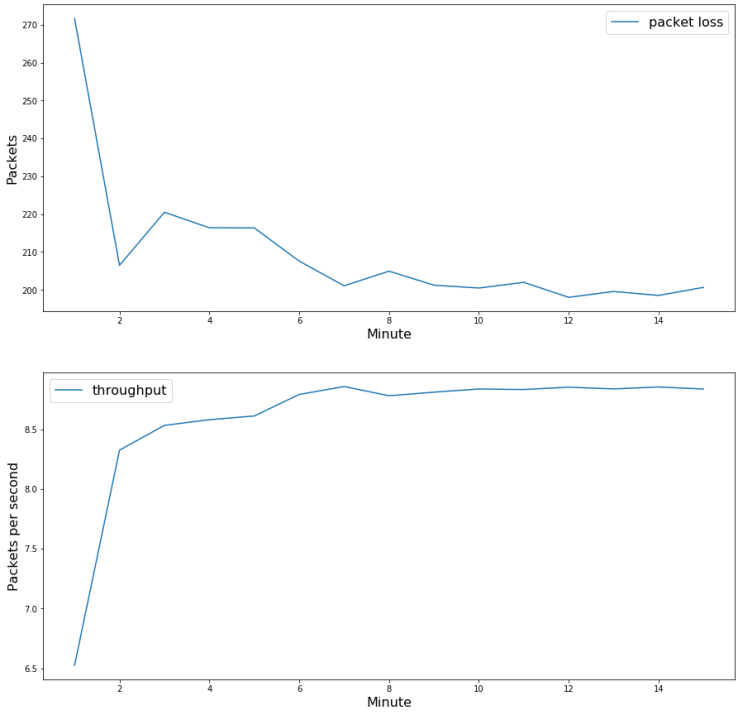


Σχήμα 4.12 Γραφική πακέτων που έχουν απορριφθεί

DIOs



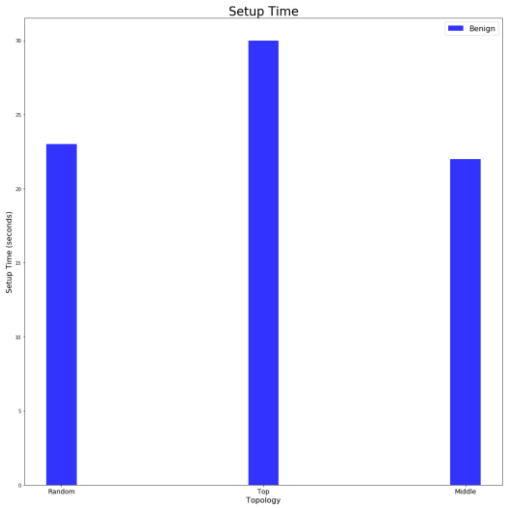
Σχήμα 4.13 Γραφική πακέτων μηνυμάτων DIO



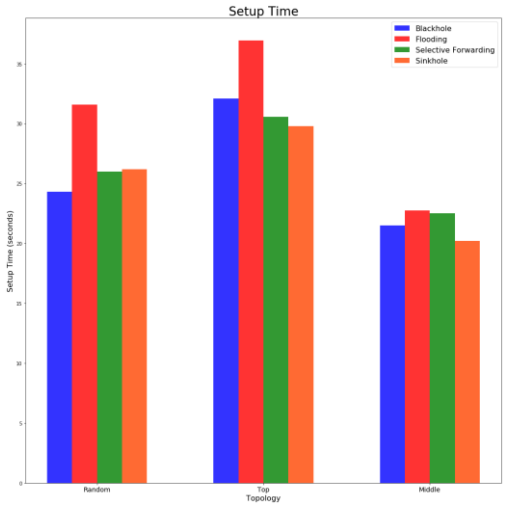
Σχήμα 4.14 Γραφική απώλειας πακέτων και throughput

4.2.4 av_setuptime.py

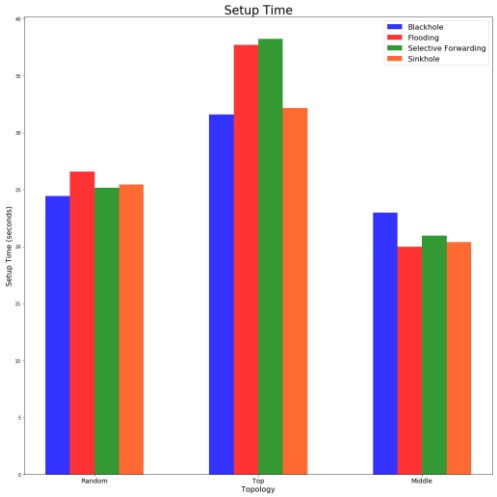
Το script χρησιμοποιήθηκε για να υπολογιστεί ο μέσος όρος του χρόνου setup time για τα είκοσι-τέσσερα σενάρια που έχουν τον ίδιο συνδυασμό τοπολογίας-ιού. Έπειτα δημιουργήθηκε η γραφική παράσταση για κάθε ομάδα σεναρίων (benign, οι ιοί είναι κακόβουλοι από την αρχή, οι ιοί είναι κακόβουλοι μετά από ένα χρονικό διάστημα). Έτσι δημιουργούνται τέσσερεις γραφικές (Σχήμα 4.15 μέχρι Σχήμα 4.18).



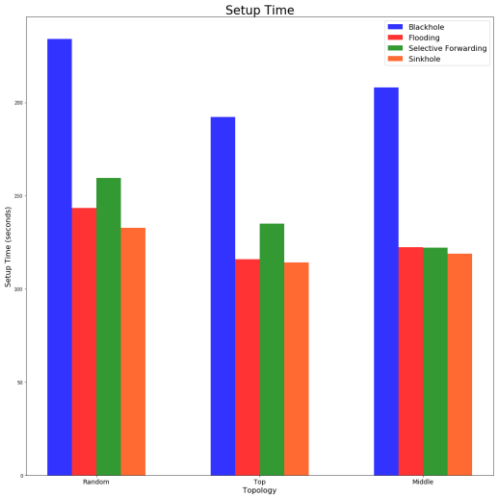
Σχήμα 4.15 Γραφική των Benign



Σχήμα 4.16 Γραφική των Κακόβουλων από την αρχή



Σχήμα 4.17 Γραφική των Κακόβουλων μετά από χρονικό διάστημα και πριν αρχίσει η κακόβουλη συμπεριφορά



Σχήμα 4.18 Γραφική των Κακόβουλων μετά από χρονικό διάστημα και μετά την αρχή της κακόβουλης συμπεριφορά

Κεφάλαιο 5

Ανάλυση Αποτελεσμάτων και Γραφικών Παραστάσεων

Σε αυτό το κεφάλαιο θα αναλύσουμε δεδομένα που αποκτούμε μέσα από 5784 αρχεία και 946 γραφικές παραστάσεις. Λόγω του μεγάλου όγκου τόσο των αρχείων όσο και του πλήθους τους δεν εμφανίζονται όλα τα αρχεία και οι γραφικές σε αυτό το κεφάλαιο, αλλά μόνο ένα μικρό δείγμα. Το κεφάλαιο είναι χωρισμένο σε υποκεφάλαια για κάθε μεγάλη κατηγορία σεναρίων, όπου και θα γίνει η ανάλυση για κάθε μια.

5.1 Benign

Τα σενάρια που ανήκουν σε αυτή την κατηγορία είναι ανεπηρέαστα από ιούς και για αυτό αποτελούν τον κανόνα με τον οποίο θα συγκρίνουμε τα σενάρια με ιούς.

5.1.1 Τοπολογία Random

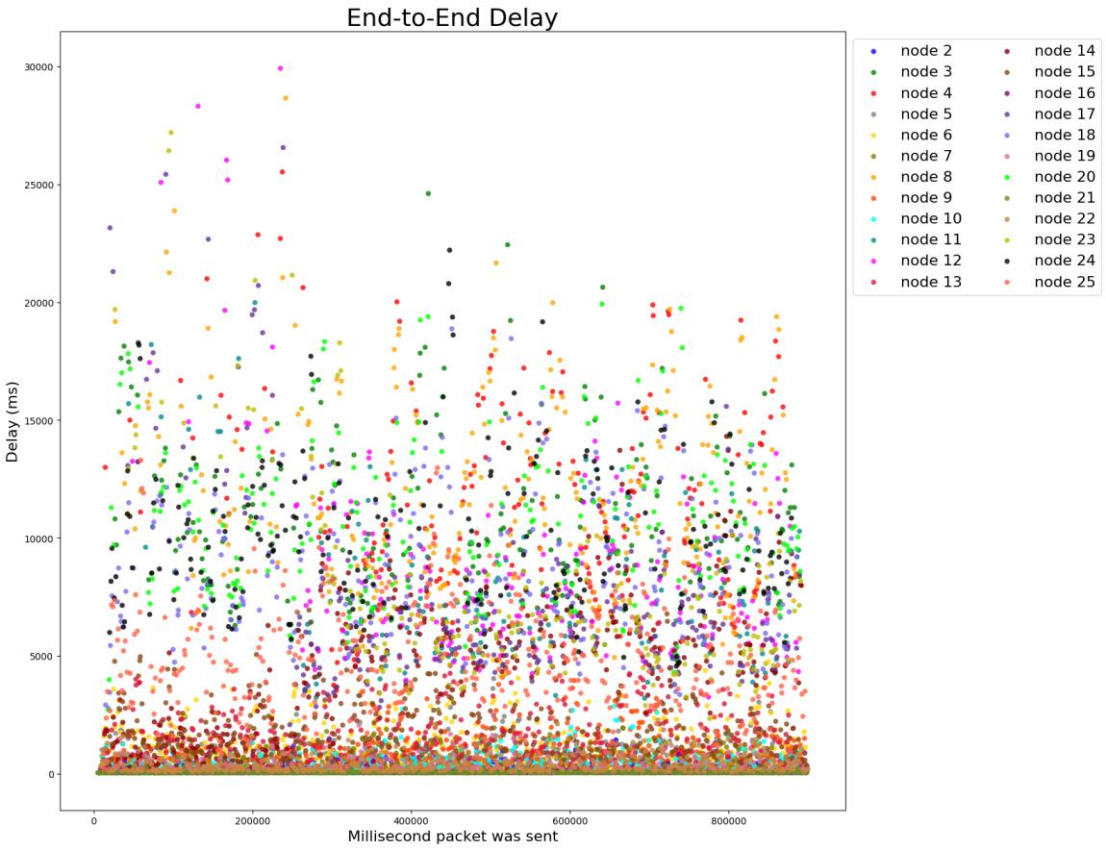
Σύμφωνα με τον Πίνακα 5.1, ο αριθμός των πακέτων που λαμβάνονται και που προωθούνται είναι ο ίδιος στα 694 πακέτα κατά μέσο όρο. Κάθε κόμβος στέλνει περίπου 447 πακέτα κατά μέσο όρο και απορρίπτει 48. Ακόμη στέλνει περίπου 10 DIO μηνύματα μέσα σε 15 λεπτά. Κατά την εξομοίωση έχει throughput 8.9 πακέτα το δευτερόλεπτο και απώλεια 3 πακέτα το δευτερόλεπτο.

	Πακέτα			
	Ελάχιστα	Μέγιστα	Μέσος Όρος	Sink
Received	0	2656	694	8009
Forwarded	0	2656	694	0
Sent	442	454	447.13	2
Dropped	9	101	47.88	8163
DIOs	7	22	9.29	7

Πίνακας 5.1 Σύνολα Random

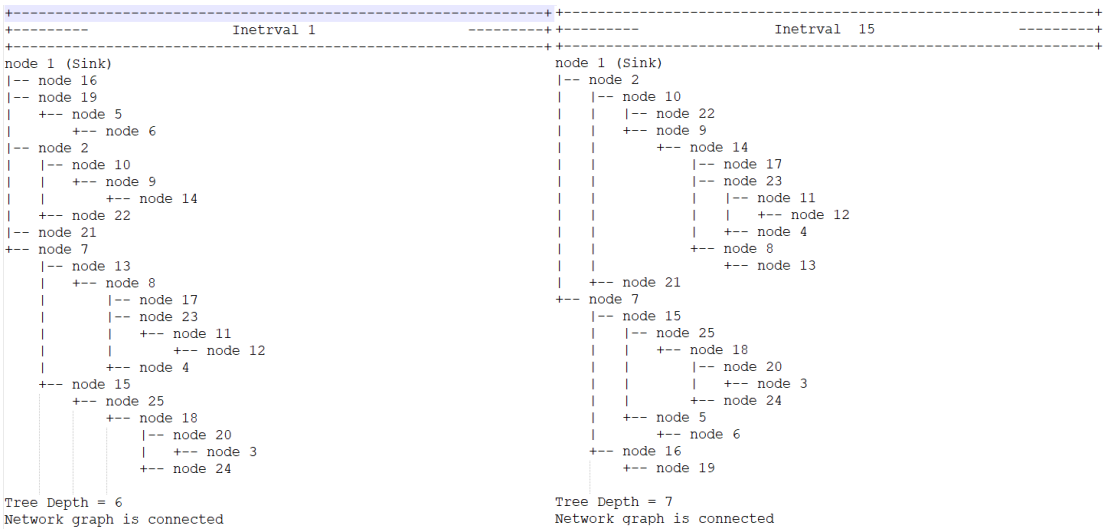
Σύμφωνα με τη γραφική διασποράς (Σχήμα 5.1), τα πλείστα πακέτα χρειάζονται λιγότερο από 5s για να φτάσουν στο sink, ενώ τα υπόλοιπα κατά το μέσο όρο χρειάζονται 10s. Ενώ, μεγαλύτερη καθυστέρηση που καταγράφηκε είναι 30s. Σύμφωνα με το αρχείο για

το forwarding delay, καθώς και την γραφική του, οι τιμές της καθυστέρησης κυμαίνονται μεταξύ των 9ms στο μέγιστο και των 7ms στο ελάχιστο.



Σχήμα 5.1 Γραφική Διασποράς για τοπολογία Random

Επιπλέον, όπως βλέπουμε στο Σχήμα 5.2, το πρώτο RPL δέντρο που δημιουργείται έχει βάθος 6, ενώ το τελευταίο έχει βάθος 7. Αυτό συμβαίνει, διότι οι κόμβοι δεν διαφημίζουν σταθερή απόσταση από τον sink, παρόλο που δεν μετακινούνται. Η διακύμανση στην απόσταση που διαφημίζεται προκαλεί αλλαγή επιλογής του κόμβου-πατέρα σε διαφορετικές χρονικές στιγμές, και έτσι μεταβάλλει το σχήμα του δέντρου.



Σχήμα 5.2 Πρώτο και τελευταίο δέντρο τοπολογίας Random

5.1.2 Τοπολογία Middle

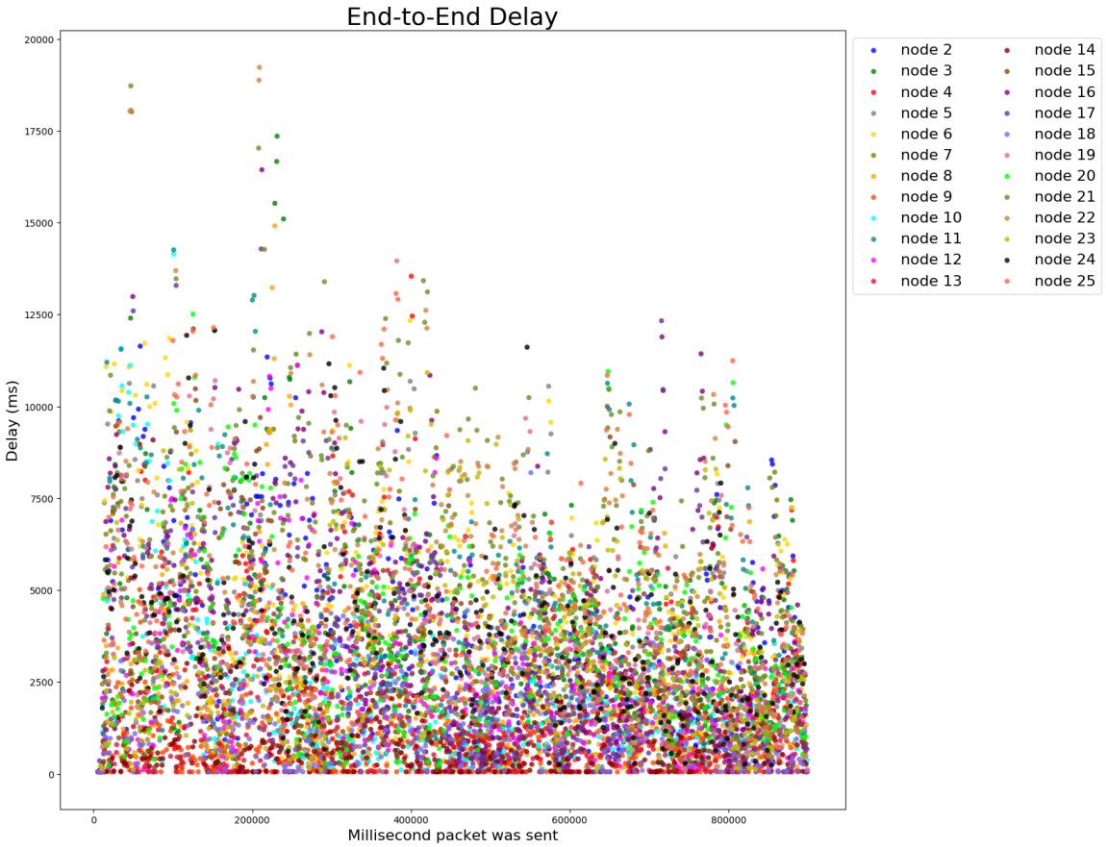
Σύμφωνα με τον Πίνακα 5.2 ο αριθμός των πακέτων που λαμβάνονται και που προωθούνται είναι ο ίδιος στα 628 πακέτα κατά μέσο όρο. Κάθε κόμβος στέλνει περίπου 447 πακέτα κατά μέσο όρο και απορρίπτει 49. Ακόμη στέλνει περίπου 12 DIO μηνύματα μέσα σε 15 λεπτά. Κατά την εξομοίωση έχει throughput 9.79 πακέτα το δευτερόλεπτο και απώλεια 2.2 πακέτα το δευτερόλεπτο.

	Πακέτα			
	Ελάχιστα	Μέγιστα	Μέσος Όρος	Sink
Received	0	2272	628.42	8810
Forwarded	0	2272	628.38	0
Sent	442	455	447.46	44
Dropped	27	86	48.92	8996
DIOs	7	25	12.38	7

Πίνακας 5.2 Σύνολα Middle

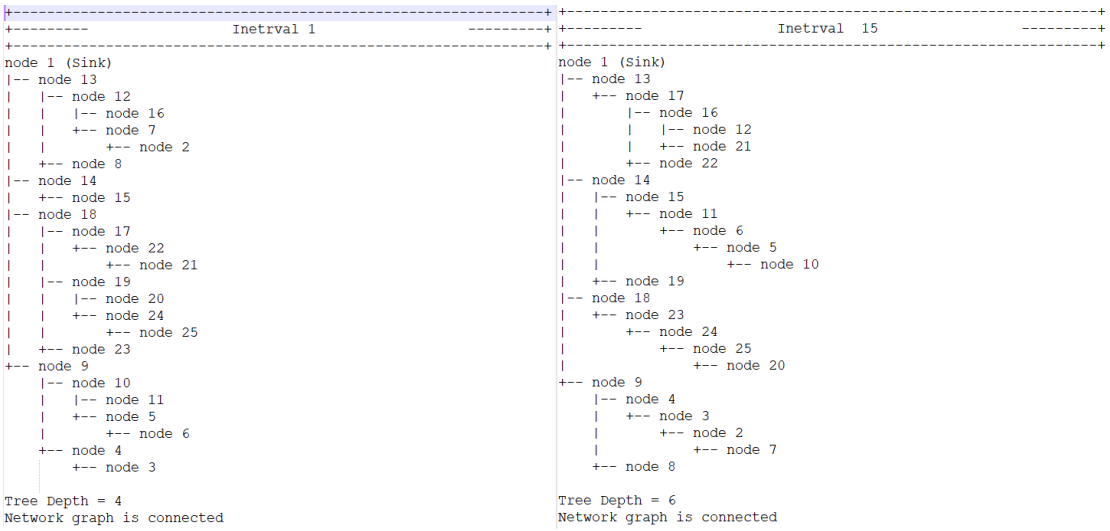
Σύμφωνα με τη γραφική διασποράς (Σχήμα 5.3) τα πλείστα πακέτα χρειάζονται κατά το μέσο όρο 2.5s με 5s για να φτάσουν στο sink. Ενώ, μεγαλύτερη καθυστέρηση που

καταγράφηκε είναι 19s. Σύμφωνα με το αρχείο για το forwarding delay, καθώς και την γραφική του, οι τιμές της καθυστέρησης κυμαίνονται μεταξύ των 9ms στο μέγιστο και των 7ms στο ελάχιστο.



Σχήμα 5.3 Γραφική Διασποράς για τοπολογία Middle

Όπως βλέπουμε στο Σχήμα 5.4, το πρώτο RPL δέντρο που δημιουργείται έχει βάθος 4, ενώ το τελευταίο έχει βάθος 6. Αυτό συμβαίνει, διότι οι κόμβοι δεν διαφημίζουν σταθερή απόσταση από τον sink, παρόλο που δεν μετακινούνται. Η διακύμανση στην απόσταση που διαφημίζεται προκαλεί αλλαγή επιλογής του κόμβου-πατέρα σε διαφορετικές χρονικές στιγμές, και μεταβάλλει το σχήμα του δέντρου.



Σχήμα 5.4 Πρώτο και τελευταίο δέντρο τοπολογίας Middle

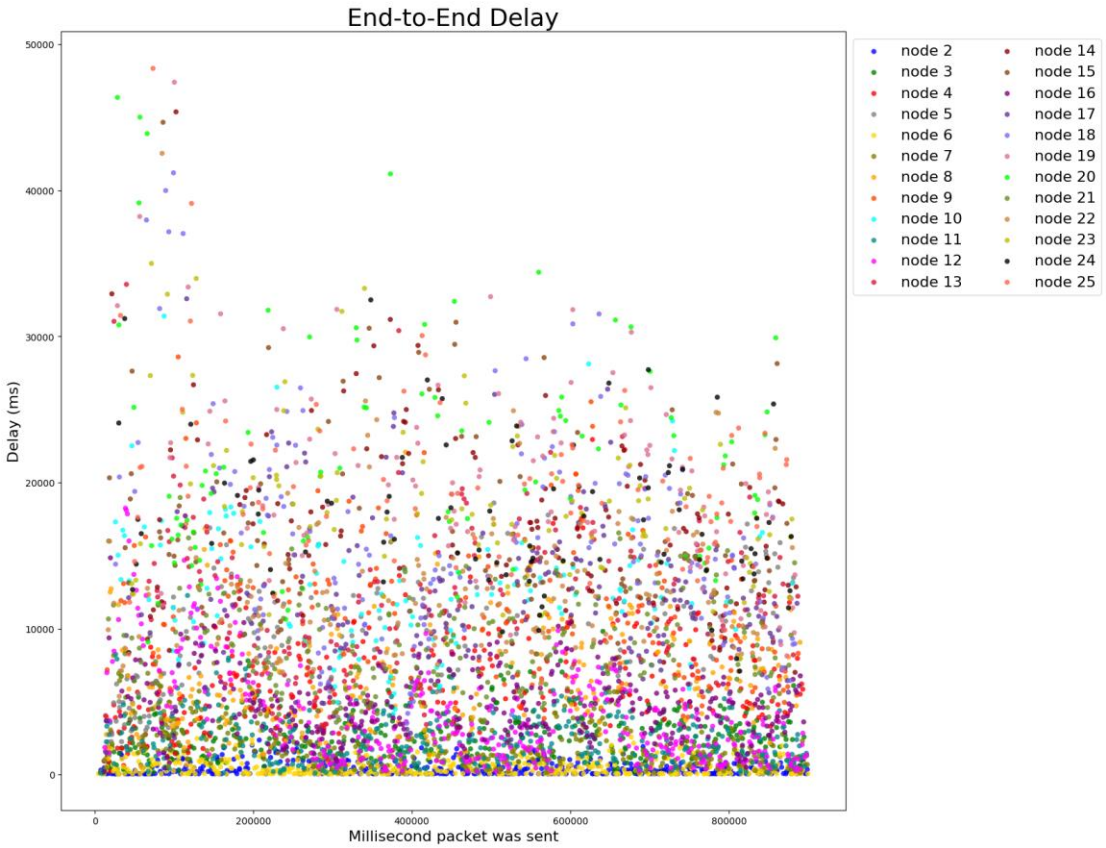
5.1.3 Τοπολογία Top

Σύμφωνα με τον Πίνακα 5.3 ο αριθμός των πακέτων που λαμβάνονται και που προωθούνται είναι ο ίδιος στα 845 πακέτα κατά μέσο όρο. Κάθε κόμβος στέλνει περίπου 449 πακέτα κατά μέσο όρο και απορρίπτει 55. Ακόμη στέλνει περίπου 15 DIO μηνύματα μέσα σε 15 λεπτά. Κατά την εξομοίωση έχει throughput 5.89 πακέτα το δευτερόλεπτο και απώλεια 6 πακέτα το δευτερόλεπτο.

	Πακέτα			
	Ελάχιστα	Μέγιστα	Μέσος Όρος	Sink
Received	0	2710	845.21	5304
Forwarded	0	2710	845.17	0
Sent	443	457	448.71	2
Dropped	37	83	55.21	5419
DIOs	7	38	14.75	7

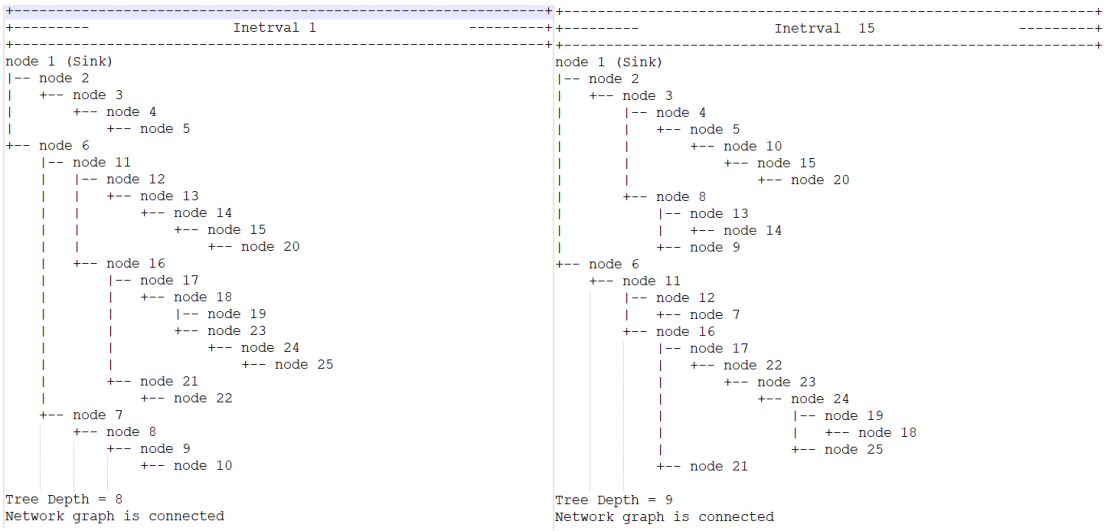
Πίνακας 5.3 Σύνολα Top

Από την γραφική διασποράς (Σχήμα 5.3) βλέπουμε ότι τα πλείστα πακέτα χρειάζονται λιγότερο από 5s για να φτάσουν στο sink, ενώ τα υπόλοιπα κατά το μέσο όρο χρειάζονται 10s. Σύμφωνα με το αρχείο για το forwarding delay, καθώς και την γραφική του, οι τιμές της καθυστέρησης κυμαίνονται μεταξύ των 10ms στο μέγιστο και των 7ms στο ελάχιστο.



Σχήμα 5.5 Γραφική Διασποράς για τοπολογία Top

Επιπλέον, όπως βλέπουμε στο Σχήμα 5.6, το πρώτο RPL δέντρο που δημιουργείται έχει βάθος 8, ενώ το τελευταίο έχει βάθος 9. Αυτό συμβαίνει, διότι οι κόμβοι δεν διαφημίζουν σταθερή απόσταση από τον sink, παρόλο που δεν μετακινούνται. Η διακύμανση στην απόσταση που διαφημίζεται προκαλεί αλλαγή επιλογής του κόμβου-πατέρα σε διαφορετικές χρονικές στιγμές, και έτσι μεταβάλλει το σχήμα του δέντρου.



Σχήμα 5.6 Πρώτο και τελευταίο δέντρο τοπολογίας Top

5.2 Malicious από την αρχή

Τα σενάρια που ανήκουν σε αυτή την κατηγορία έχουν έναν κόμβο που είναι μολυσμένο με ιό. Ο ιός μπορεί να είναι ένας εκ των τεσσάρων (Black-hole, Flooding, Selective Forwarding, Sinkhole). Για κάθε ένα από τα σενάρια σε αυτή την κατηγορία ο ιός αρχίζει την κακόβουλη συμπεριφορά του από την αρχή του σεναρίου. Κάθε σενάριο έχει μόνο ένα μολυσμένο κόμβο, ο οποίος δεν είναι ποτέ ο κόμβος sink. Κάθε μολυσμένος κόμβος φέρει μόνο ένα είδος ιού.

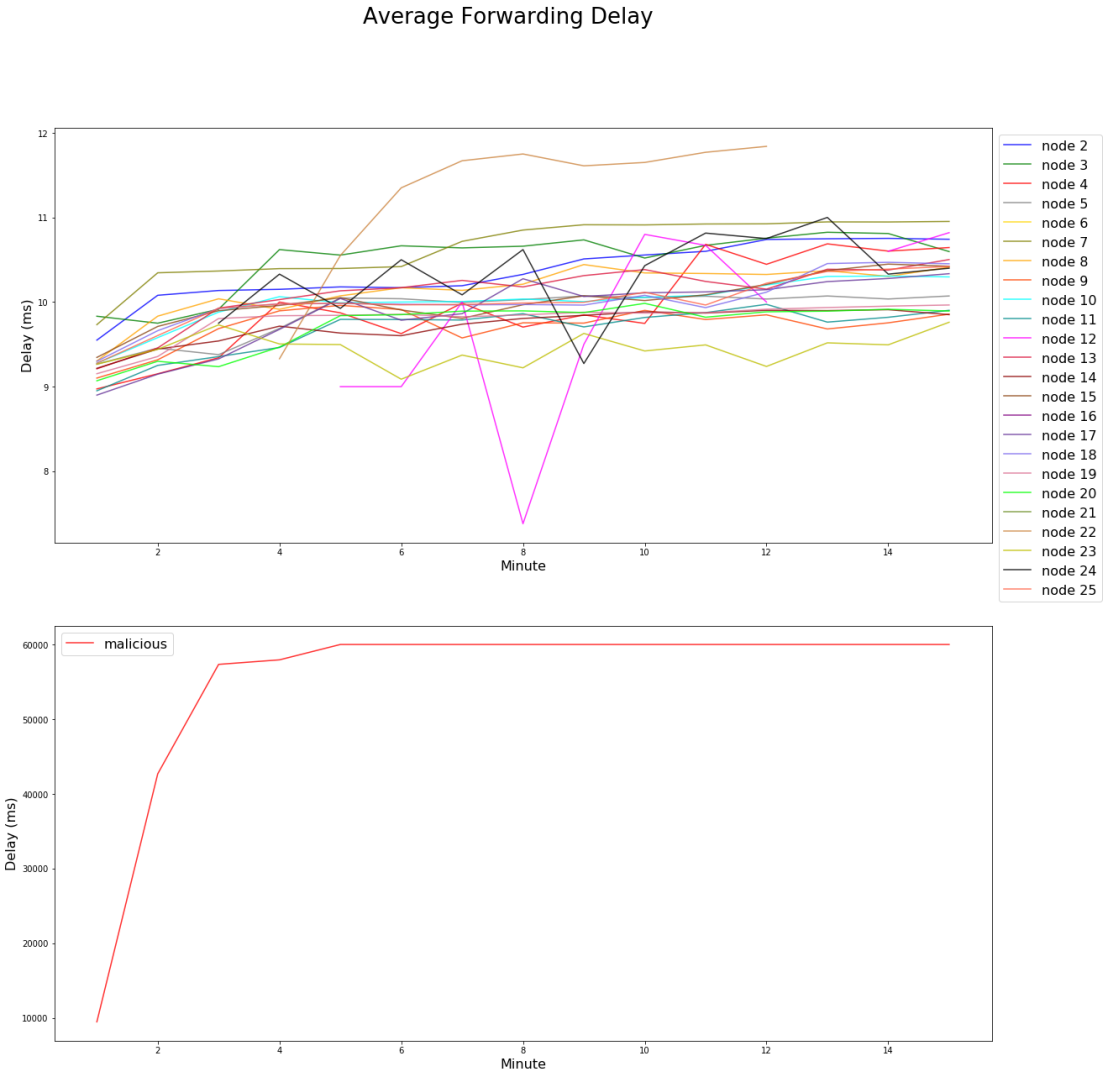
5.2.1 Black-hole

Αυτός ο ιός διαφημίζει μικρότερες αποστάσεις προς το sink με σκοπό να ελκύσει περισσότερους κόμβους να τον επιλέξουν ως πατέρα. Όσο περισσότερα πακέτα περνούν από τον μολυσμένο κόμβο τόσο μεγαλύτερο πρόβλημα μπορεί να δημιουργήσει στο δίκτυο.

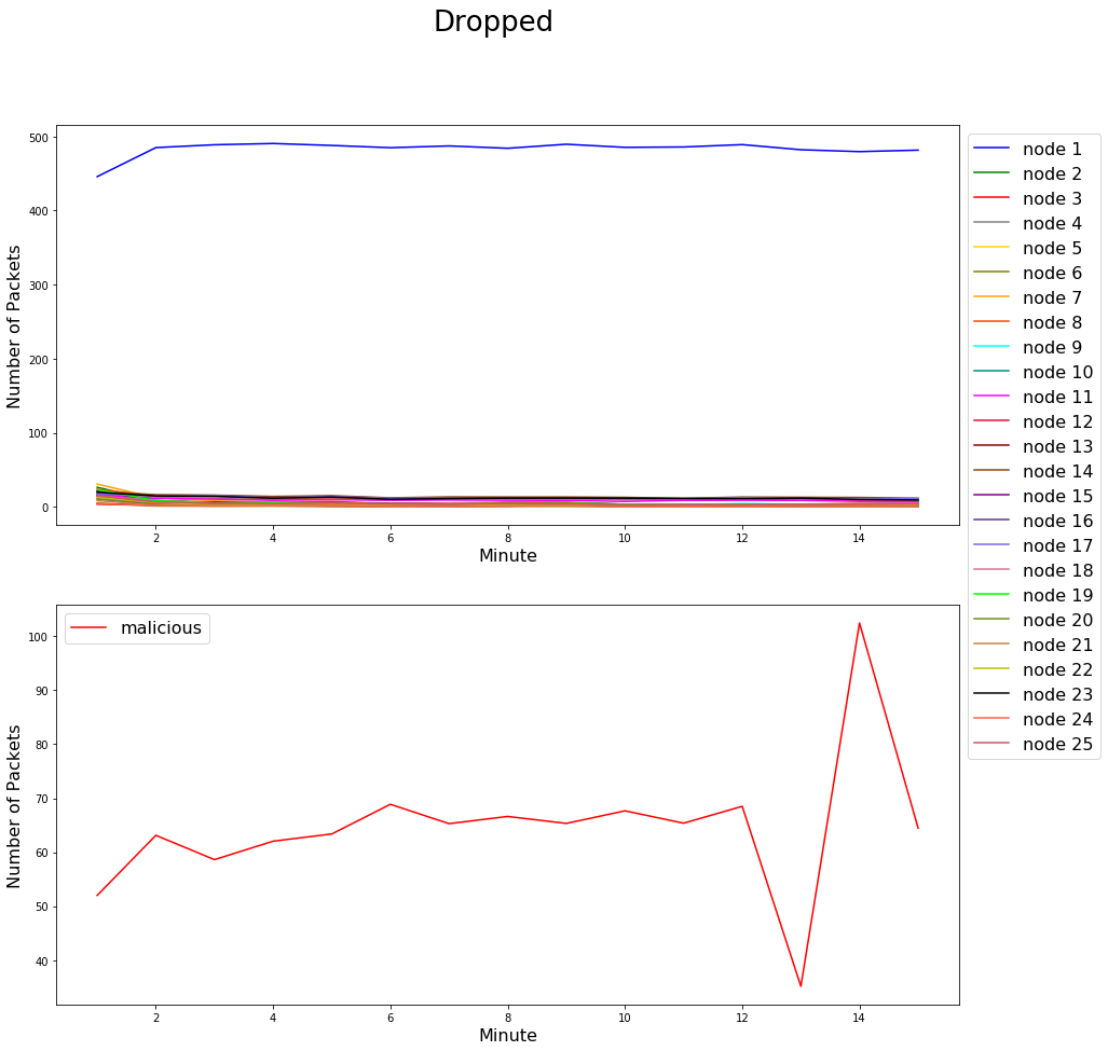
5.2.1.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.7 ο ιός δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό) και απορρίπτει περισσότερα πακέτα σε σχέση με τους άλλους κόμβους. Η πιο πάνω συμπεριφορά του ιού έχει ως αποτέλεσμα την μείωση της αποδοτικότητας του δικτύου. Ακόμη παρατηρούμε ότι οι benign κόμβοι έχουν μεγαλύτερη καθυστέρηση στην προώθηση πακέτων σε αυτά τα

σενάρια απ’ ότι στα υγιή σενάρια. Επίσης, το throughput έχει μειωθεί στα 8 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 4.3 πακέτα το δευτερόλεπτο. Επίσεις, παρατηρούμε ότι στα πλείστα σενάρια το αρχικό δέντρο δεν παρουσιάζει μεγάλη διαφορά βάθους με το τελευταίο.



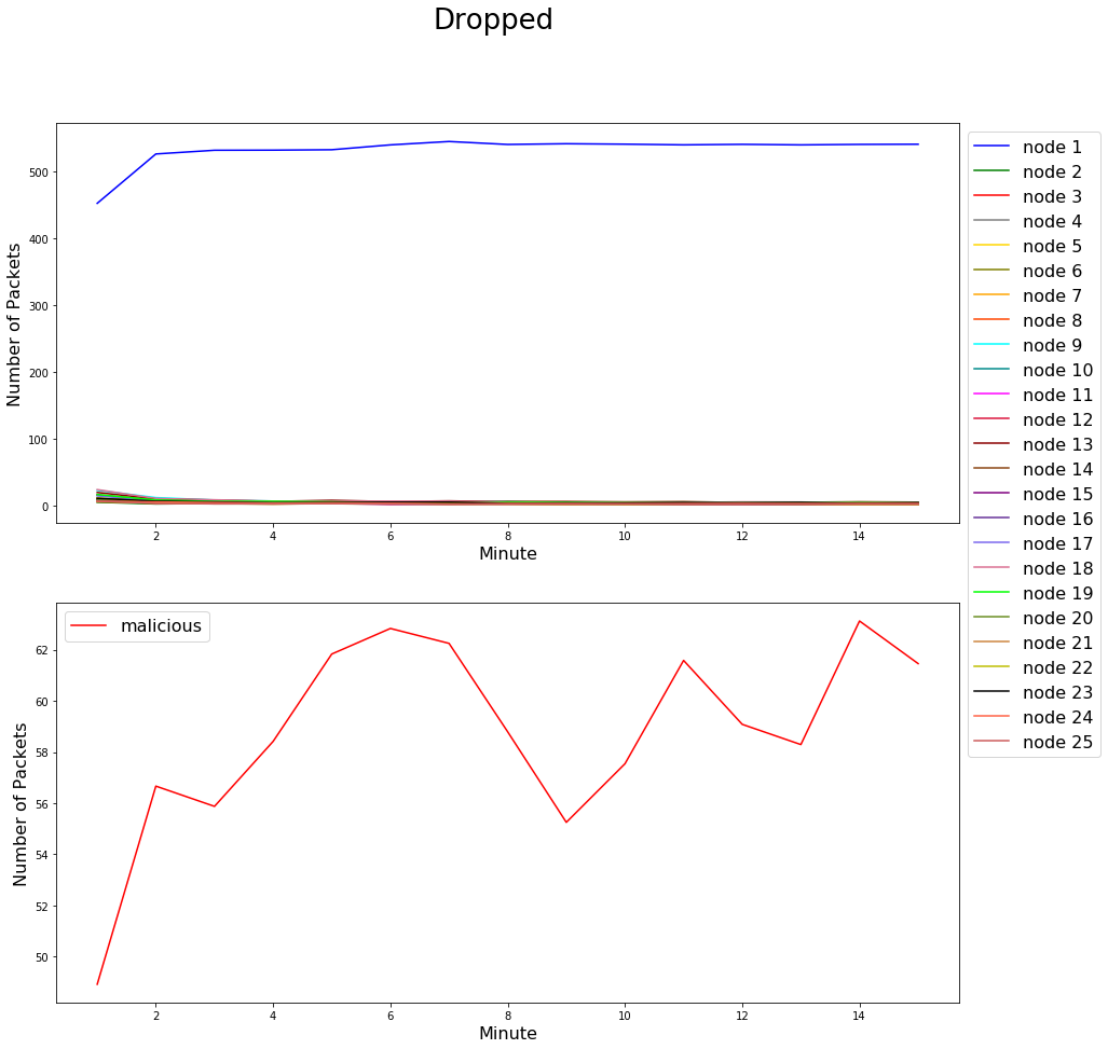
Σχήμα 5.7.α Γραφική Forwarding Delay για τοπολογία Black-hole Random



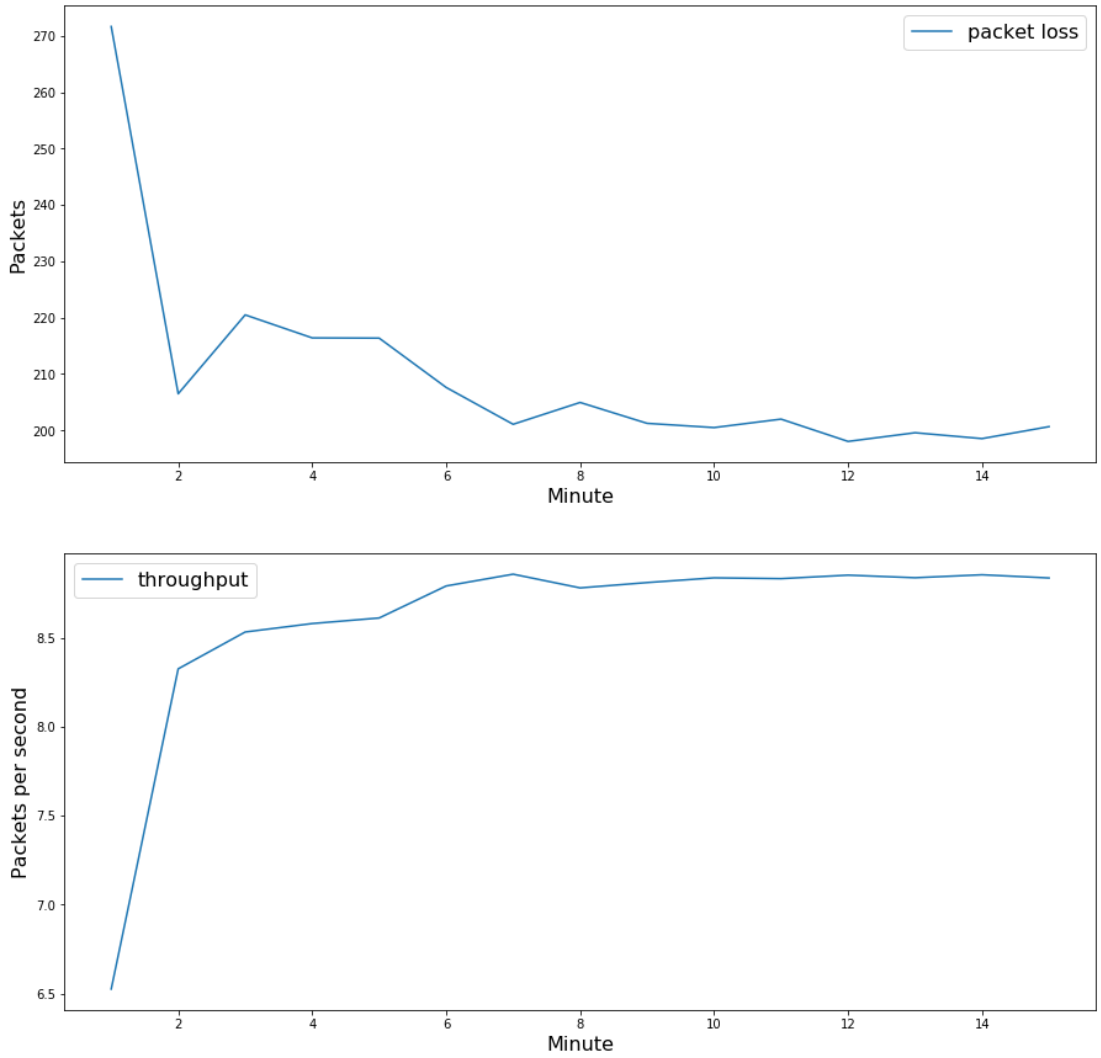
Σχήμα 5.7.b Γραφική Packets Dropped για τοπολογία Black-hole Random

5.2.1.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.8 ο κακόβουλος κόμβος απορρίπτει περισσότερα πακέτα σε σχέση με τους άλλους κόμβους. Επίσης δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Ακόμη παρατηρούμε ότι οι benign κόμβοι έχουν μεγαλύτερη καθυστέρηση στην προώθηση πακέτων σε αυτά τα σενάρια. Επιπλέον, το throughput έχει μειωθεί στα 9 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.33 πακέτα το δευτερόλεπτο. Έπειτα, παρατηρούμε ότι στα πλείστα σενάρια το αρχικό δέντρο δεν παρουσιάζει μεγάλη διαφορά βάθους με το τελευταίο.



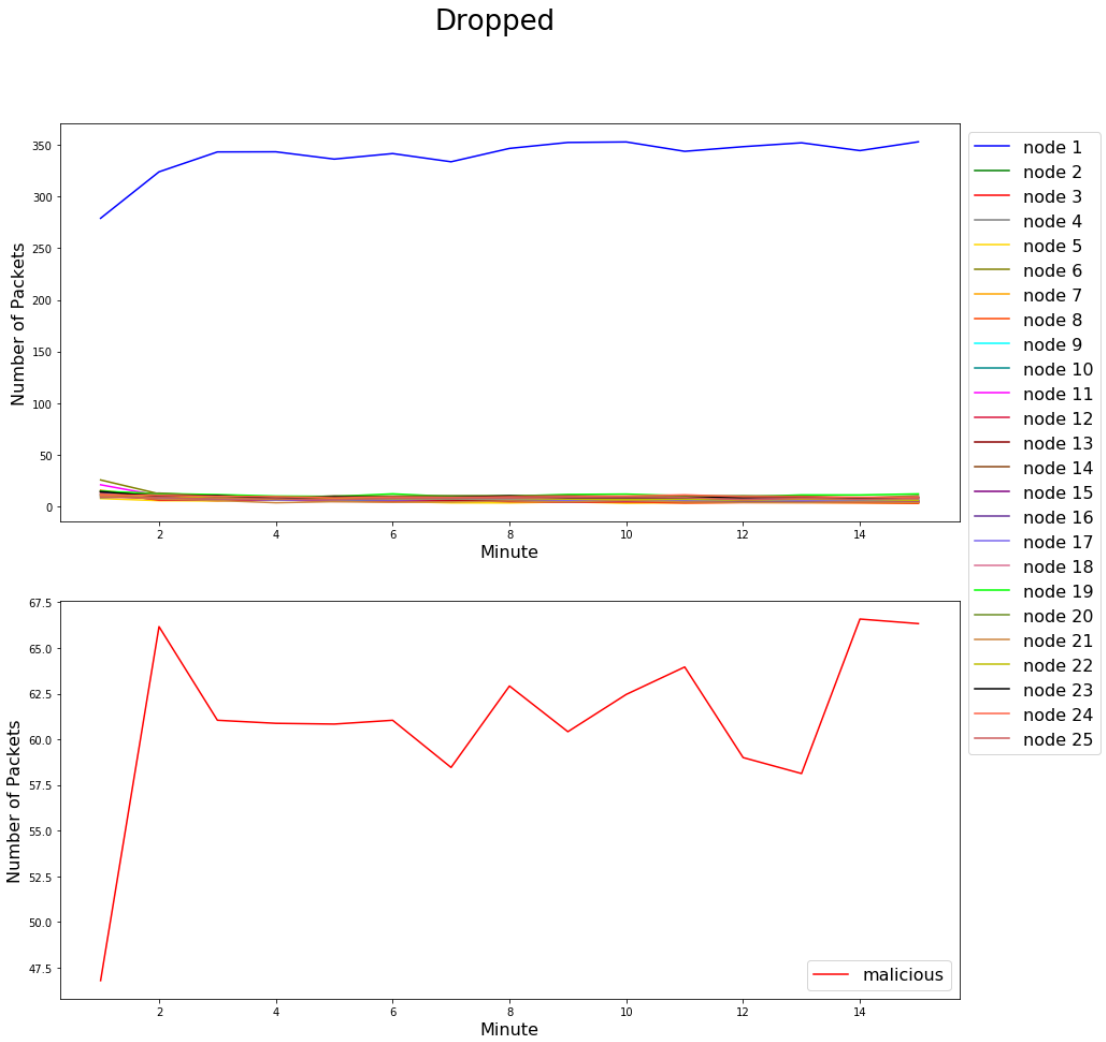
Σχήμα 5.8.a Γραφική Packets Dropped για τοπολογία Black-hole Middle



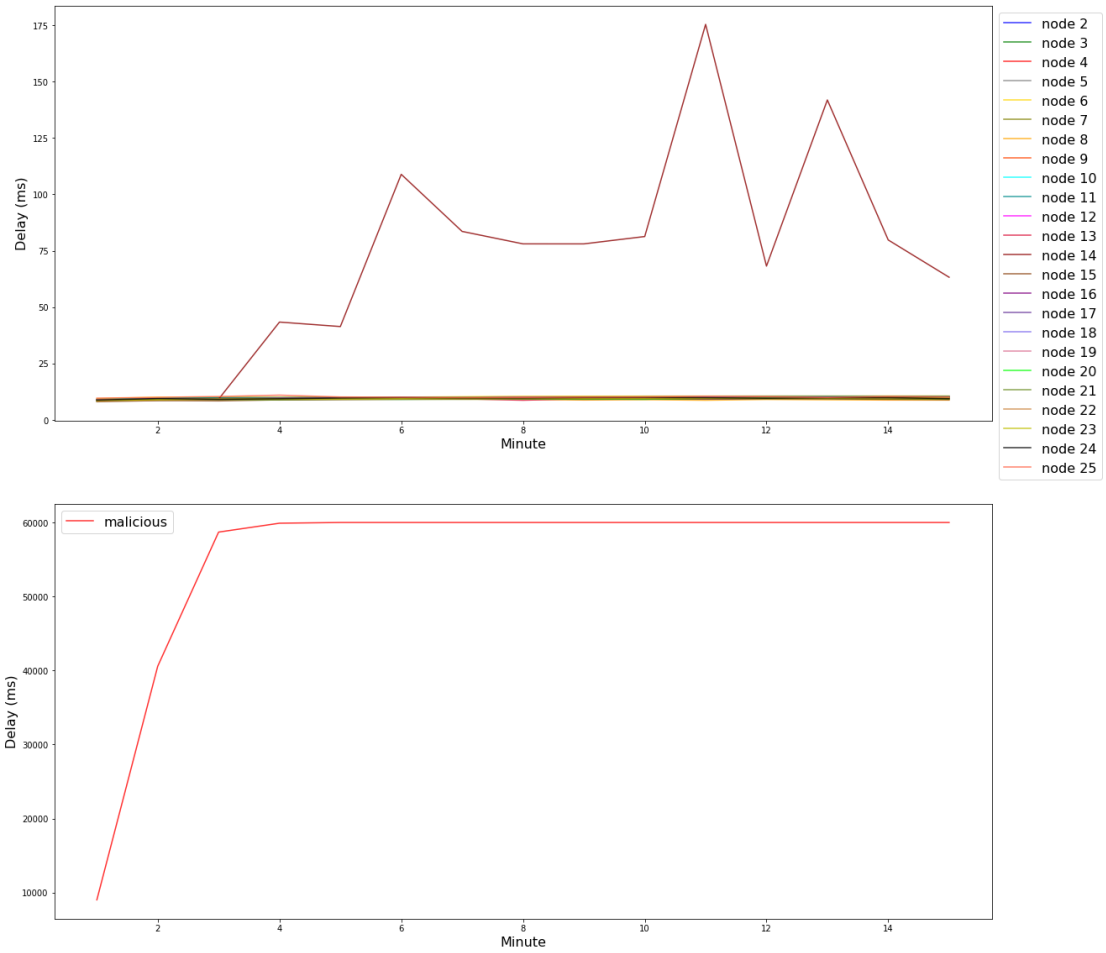
Σχήμα 5.8.b Γραφικές Packet Loss - Throughput για τοπολογία Black-hole Middle

5.2.1.3 Τοπολογία Top

Όπως παρατηρούμε στα σχήματα 5.9 ο κακόβουλος κόμβος απορρίπτει περισσότερα πακέτα σε σχέση με τους άλλους κόμβους και δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Ακόμη παρατηρούμε ότι οι benign κόμβοι έχουν μεγαλύτερη καθυστέρηση στην προώθηση πακέτων σε αυτά τα σενάρια. Επιπλέον, το throughput έχει μειωθεί στα 5.7 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 6.67 πακέτα το δευτερόλεπτο. Έπειτα παρατηρούμε ότι στα πλείστα σενάρια το αρχικό δέντρο δεν παρουσιάζει μεγάλη διαφορά βάθους με το τελευταίο.



Σχήμα 5.9.a Γραφική Packets Dropped για τοπολογία Black-hole Top



Σχήμα 5.9.b Γραφική Forwarding Delay για τοπολογία Black-hole Top

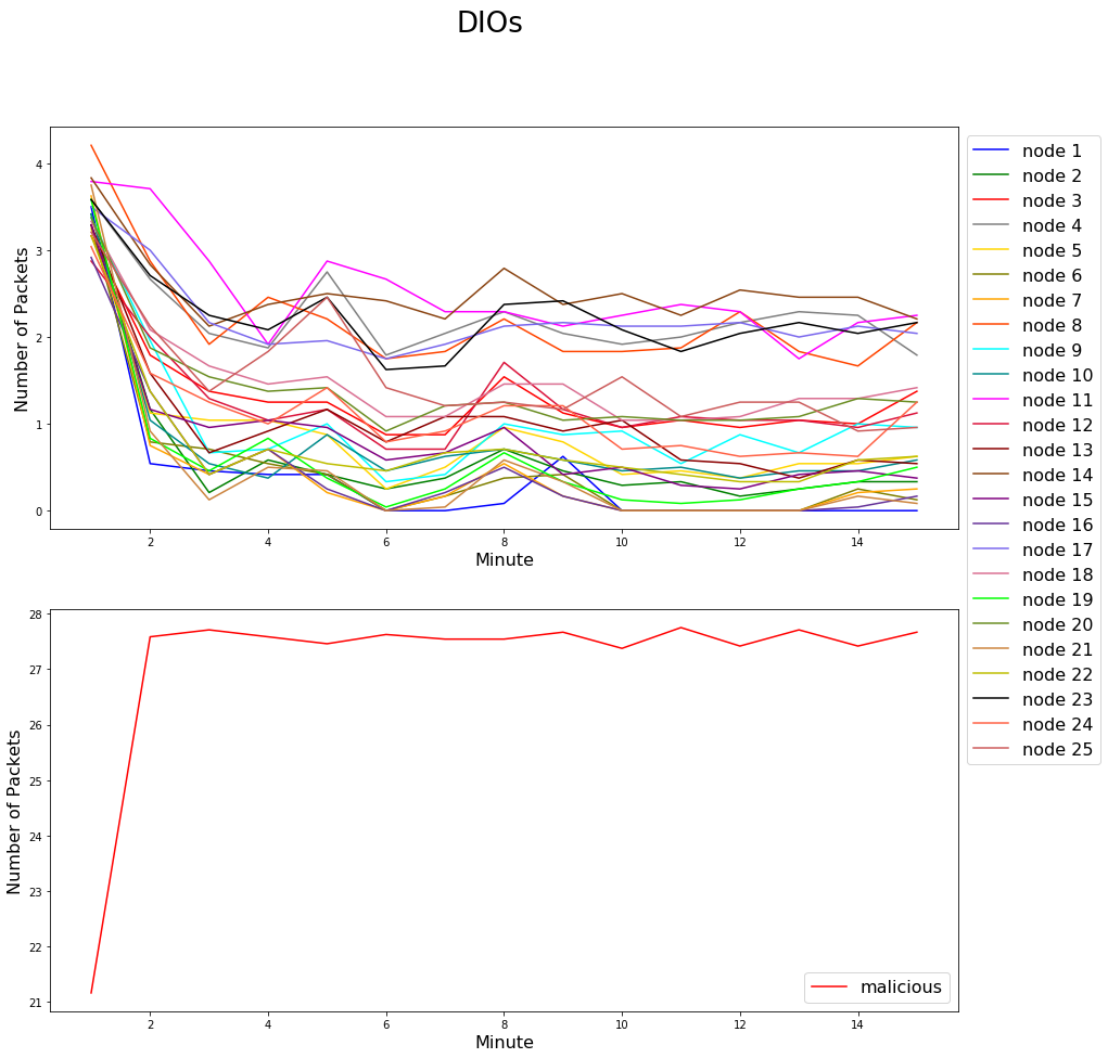
5.2.2 Flooding

Ο ιός αναγκάζει τον κακόβουλο κόμβο να στέλνει περισσότερα DIO μηνύματα, τα οποία είναι περιττά. Σκοπός του ιού είναι να αυξήσει τόσο τα περιττά μηνύματα που στέλνει έτσι ώστε να δημιουργήσει συμφόρηση στο δίκτυο, με αποτέλεσμα να προκαλέσει δυσλειτουργία.

5.2.2.1 Τοπολογία Random

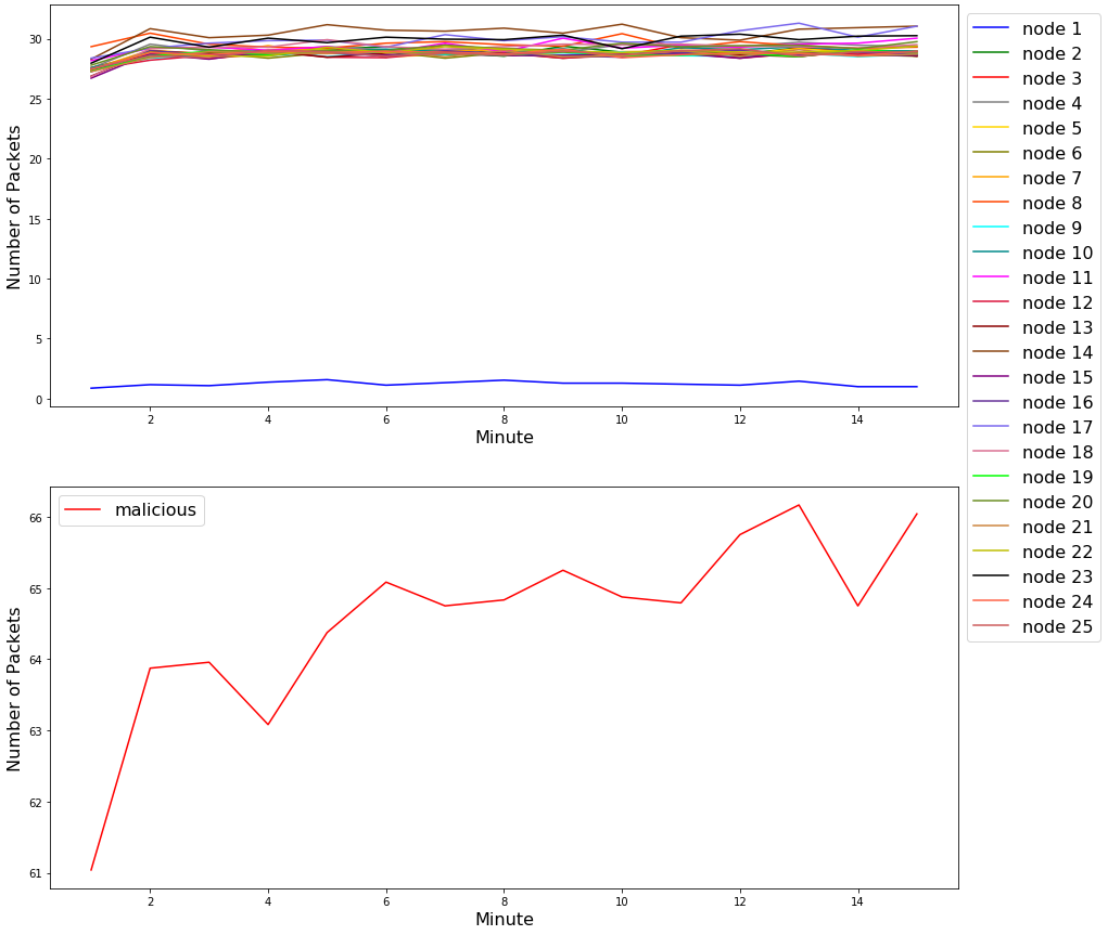
Όπως παρατηρούμε στα σχήματα 5.10 ο κακόβουλος κόμβος στέλνει τα επταπλάσια DIO μηνύματα ενώ δημιουργεί και τα διπλάσια πακέτα σε σχέση με τους άλλους κόμβους. Επιπλέον, το throughput έχει μειωθεί στα 8.2 πακέτα το δευτερόλεπτο, ενώ η απώλεια

έχει αυξηθεί στα 4.5 πακέτα το δευτερόλεπτο. Ακόμη παρατηρούμε ότι ενώ οι υγιείς κόμβοι τείνουν να μειώνουν τα DIO μηνύματα που στέλνουν, ο κακόβουλος κόμβος τείνει να τα αυξάνει. Παρομοίως, βλέπουμε πως ενώ ο αριθμός των πακέτων που στέλνουν τείνει να είναι σταθερός, ο αριθμός των πακέτων που στέλνει ο κακόβουλος κόμβος τείνει να αυξάνεται.



Σχήμα 5.10.a Γραφική DIO Packets για τοπολογία Flooding Random

Sent

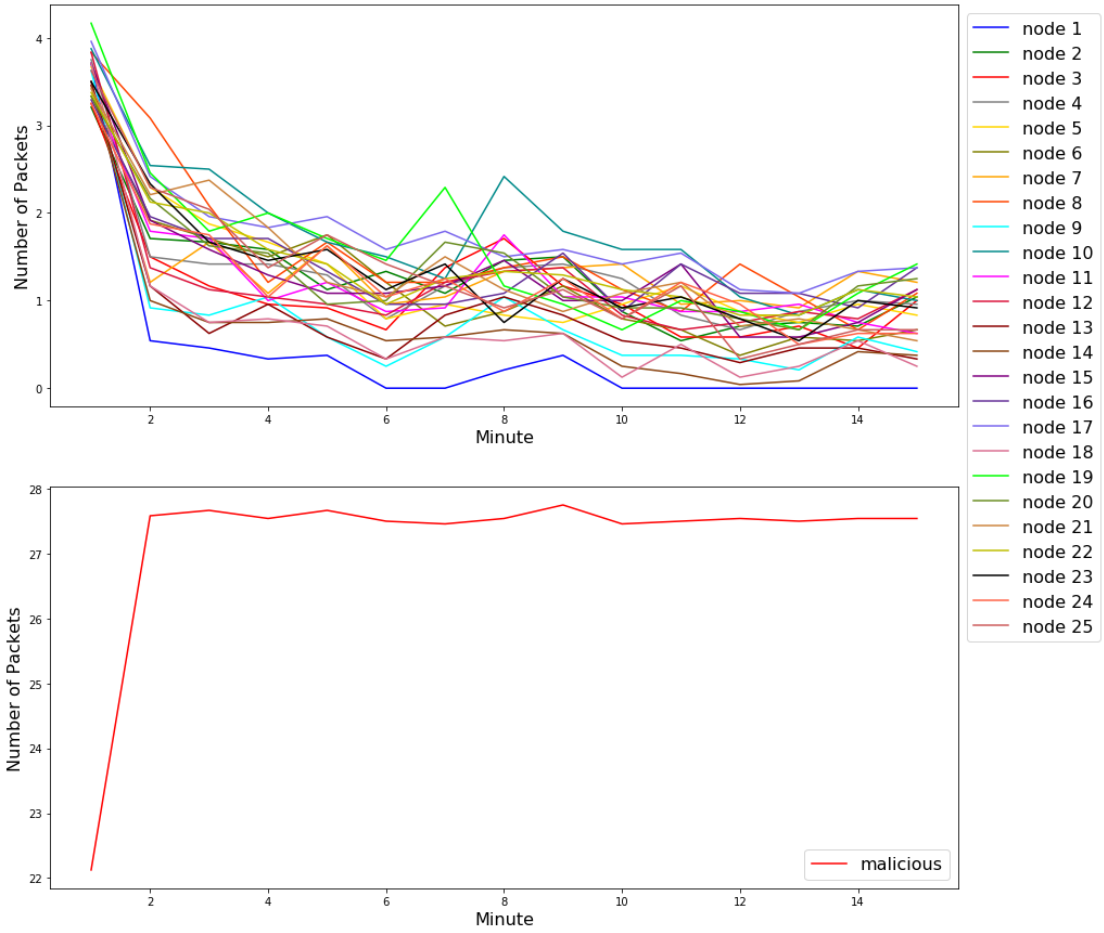


Σχήμα 5.10.b Γραφική Packets Sent για τοπολογία Flooding Random

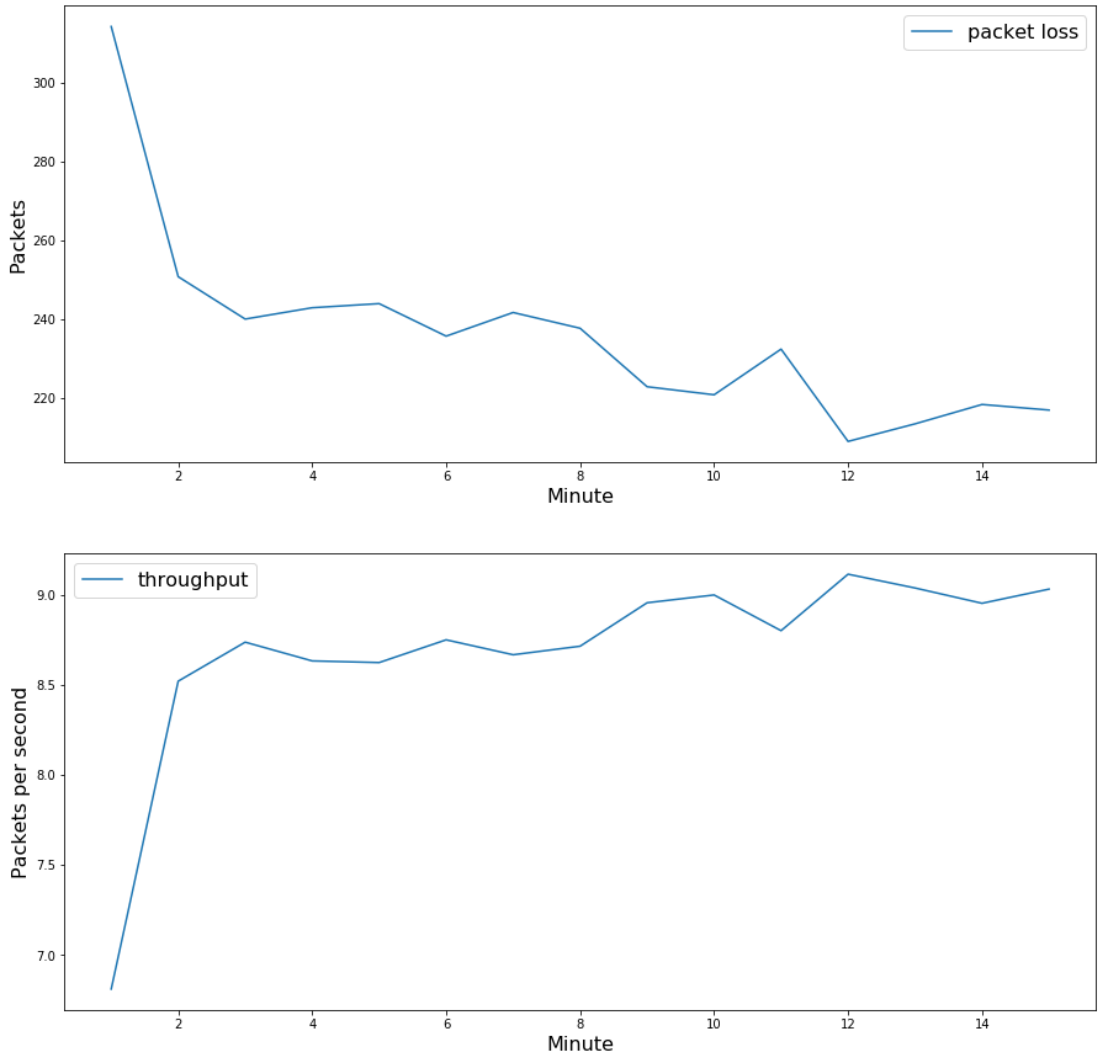
5.2.2.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.11 ο κακόβουλος κόμβος στέλνει τα επταπλάσια DIO μηνύματα και δημιουργεί και στέλνει τα διπλάσια πακέτα σε σχέση με τους άλλους κόμβους. Το throughput έχει μειωθεί στα 9 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.67 πακέτα το δευτερόλεπτο. Παρατηρούμε ότι καθώς μειώνονται τα μηνύματα DIO που στέλνουν οι υγιείς κόμβοι, τα αντίστοιχα μηνύματα του κακόβουλου κόμβου αυξάνονται.

DIOs



Σχήμα 5.11.α Γραφική DIO Packets για τοπολογία Flooding Middle

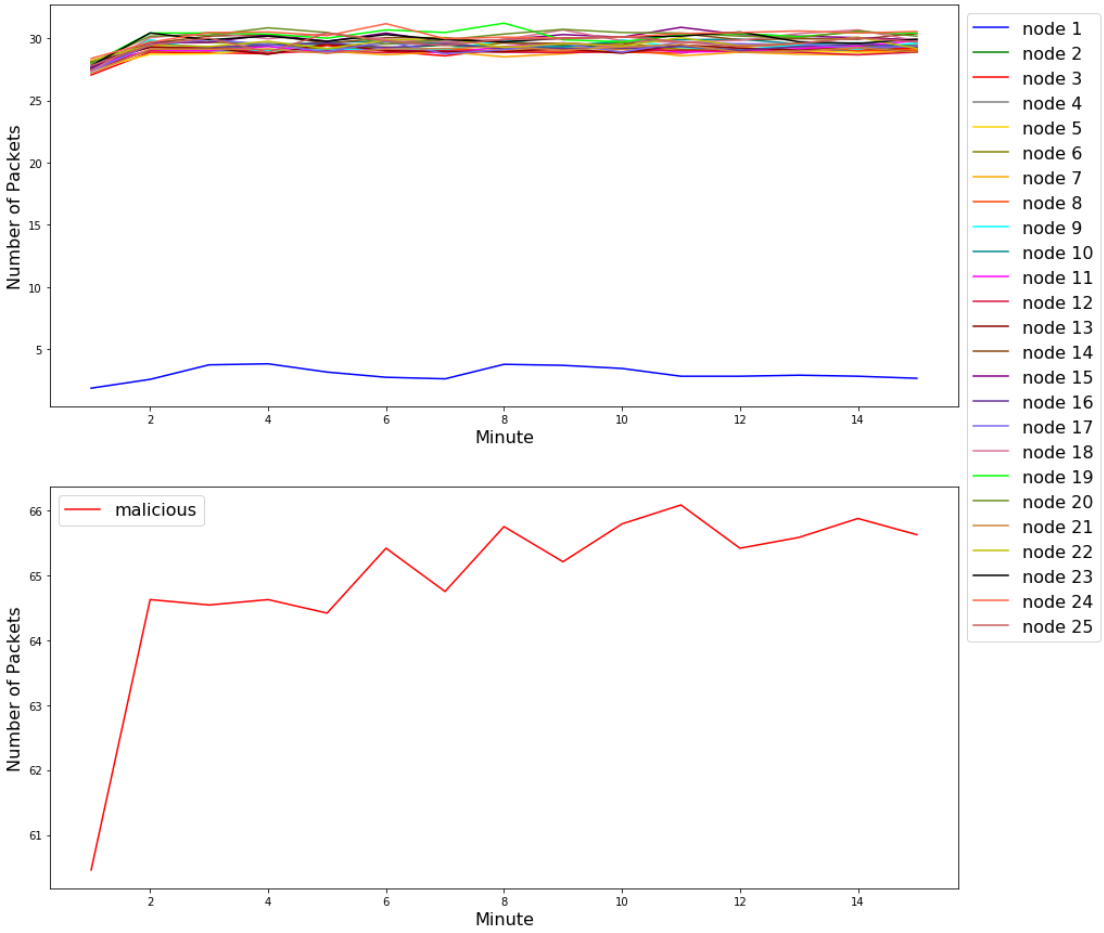


Σχήμα 5.11.b Γραφικές Packet Loss - Throughput για τοπολογία Flooding Middle

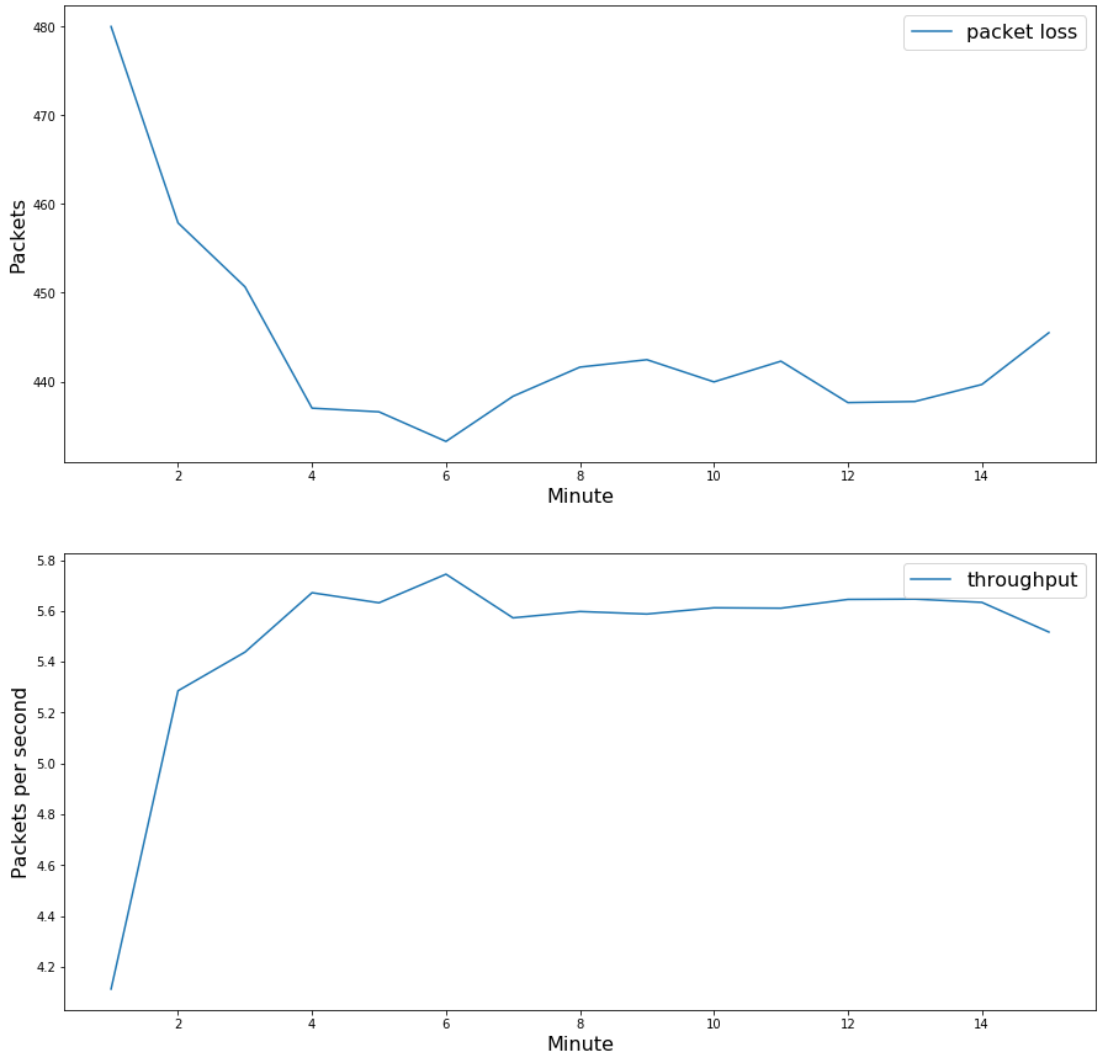
5.2.2.3 Τοπολογία Top

Όπως παρατηρούμε στα σχήματα 5.12 ο κακόβουλος κόμβος δημιουργεί και στέλνει τα διπλάσια πακέτα σε σχέση με τους άλλους κόμβους και στέλνει τα επταπλάσια DIO μηνύματα. Επιπλέον το throughput έχει μειωθεί στα 5.6 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 7.33 πακέτα το δευτερόλεπτο. Ακόμη παρατηρούμε ότι ενώ οι υπόλοιποι κόμβοι δημιουργούν και στέλνουν σχετικά σταθερό αριθμό πακέτων, ο κακόβουλος κόμβος τείνει να αυξάνει σταθερά τα πακέτα που στέλνει.

Sent



Σχήμα 5.12.b Γραφική Packets Sent για τοπολογία Flooding Top



Σχήμα 5.12.b Γραφικές Packet Loss - Throughput για τοπολογία Flooding Top

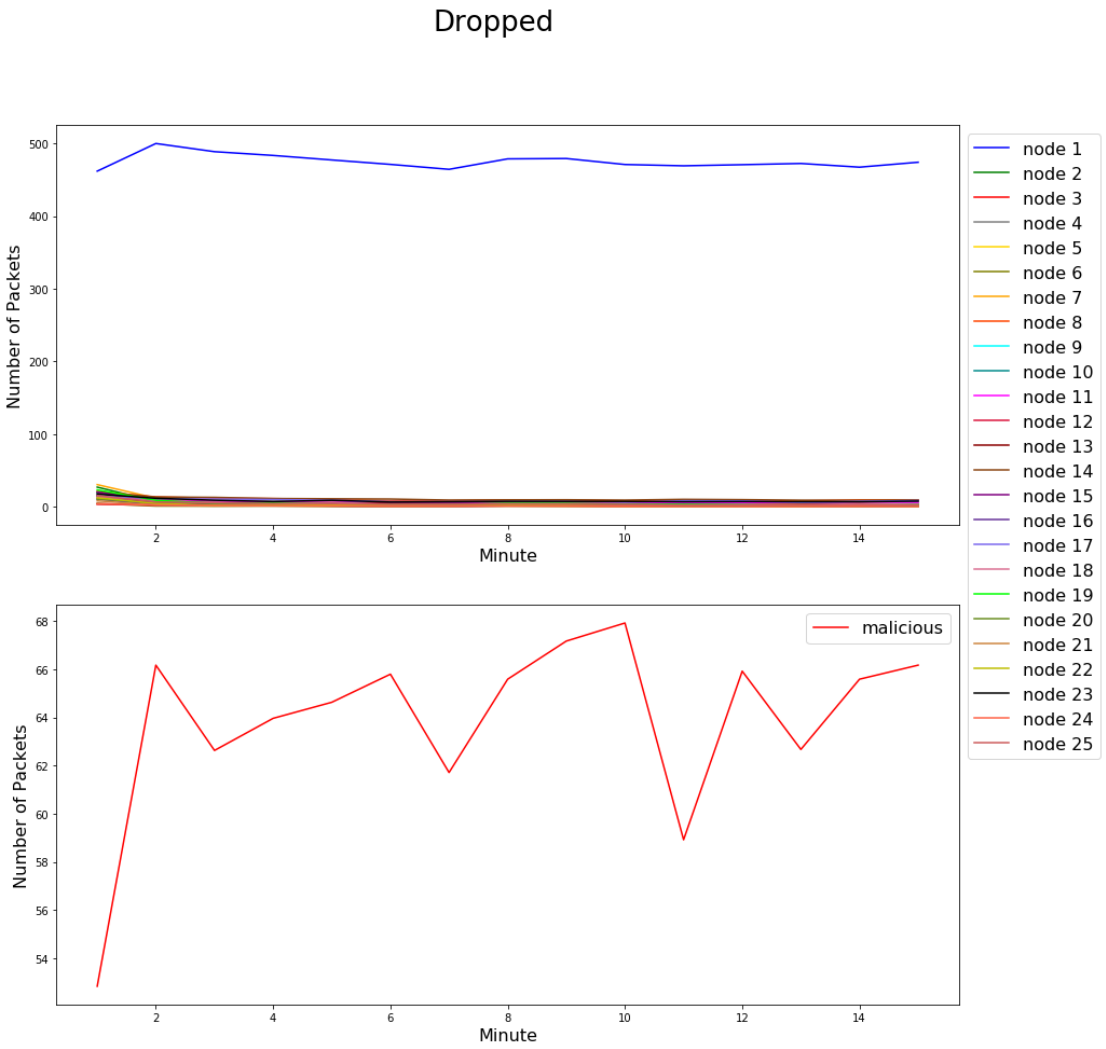
5.2.3 Selective Forwarding

Ο ιός επιλέγει να απορρίψει συγκεκριμένα πακέτα, τα οποία ένας benign κόμβος θα προωθούσε προς το sink. Σκοπός του να μειώσει την αποδοτικότητα του δικτύου αυξάνοντας την απώλεια πακέτων.

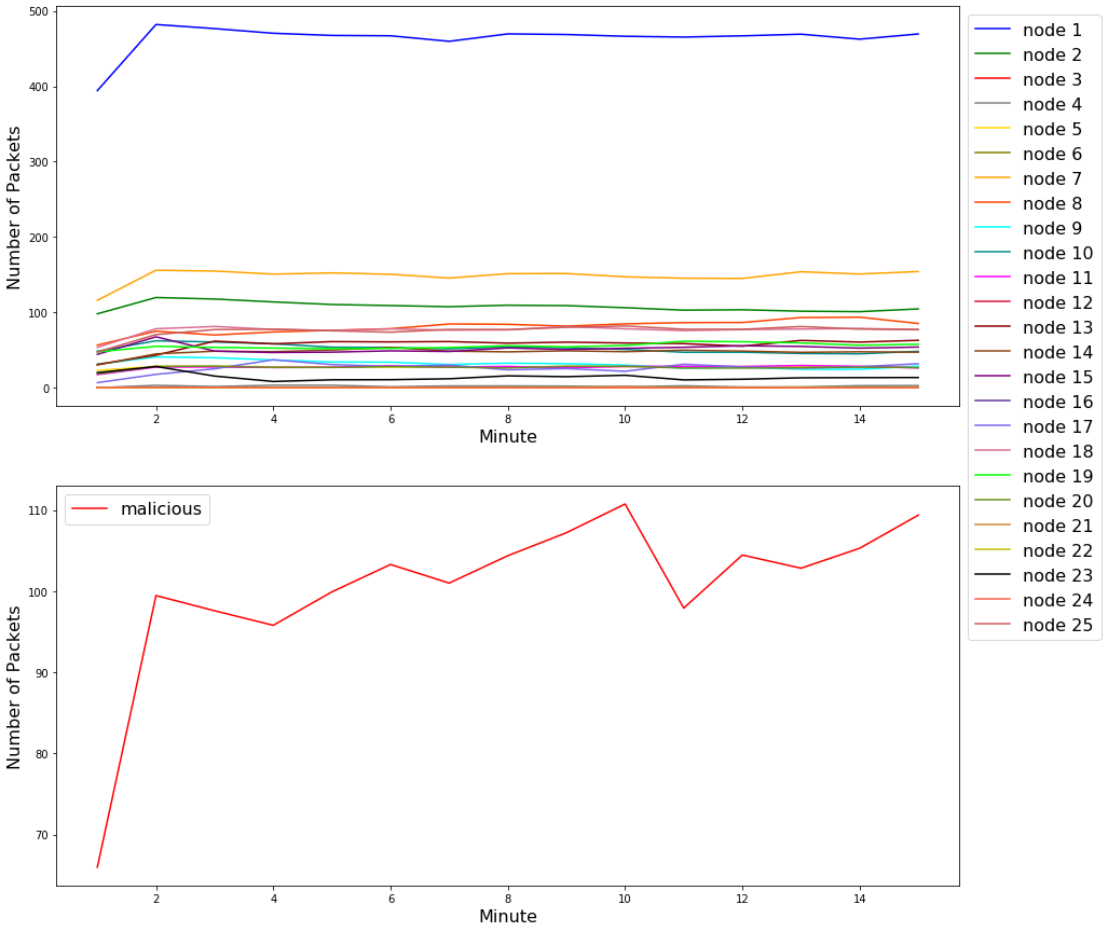
5.2.3.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.13 ο κακόβουλος κόμβος απορρίπτει τα μισά πακέτα που λαμβάνει και δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Επιπλέον, παρατηρούμε ότι ο αριθμός των

πακέτων που απορρίπτει είναι πολύ μεγαλύτερος από αυτό των υγιών κόμβων. Επίσεις, το throughput έχει μειωθεί στα 7.9 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 4.33 πακέτα το δευτερόλεπτο.



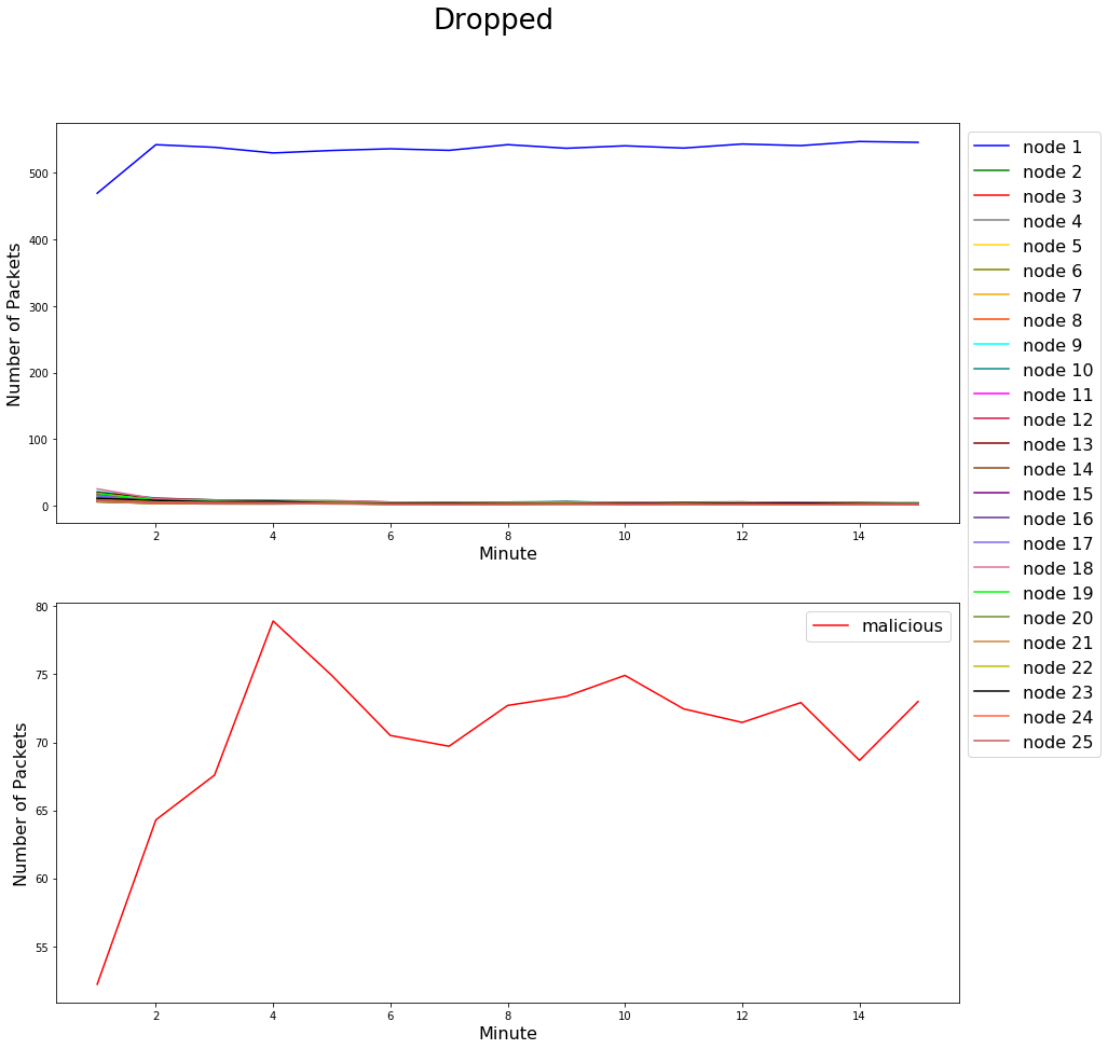
Σχήμα 5.13.α Γραφική Packets Dropped για τοπολογία Selective Forwarding Random



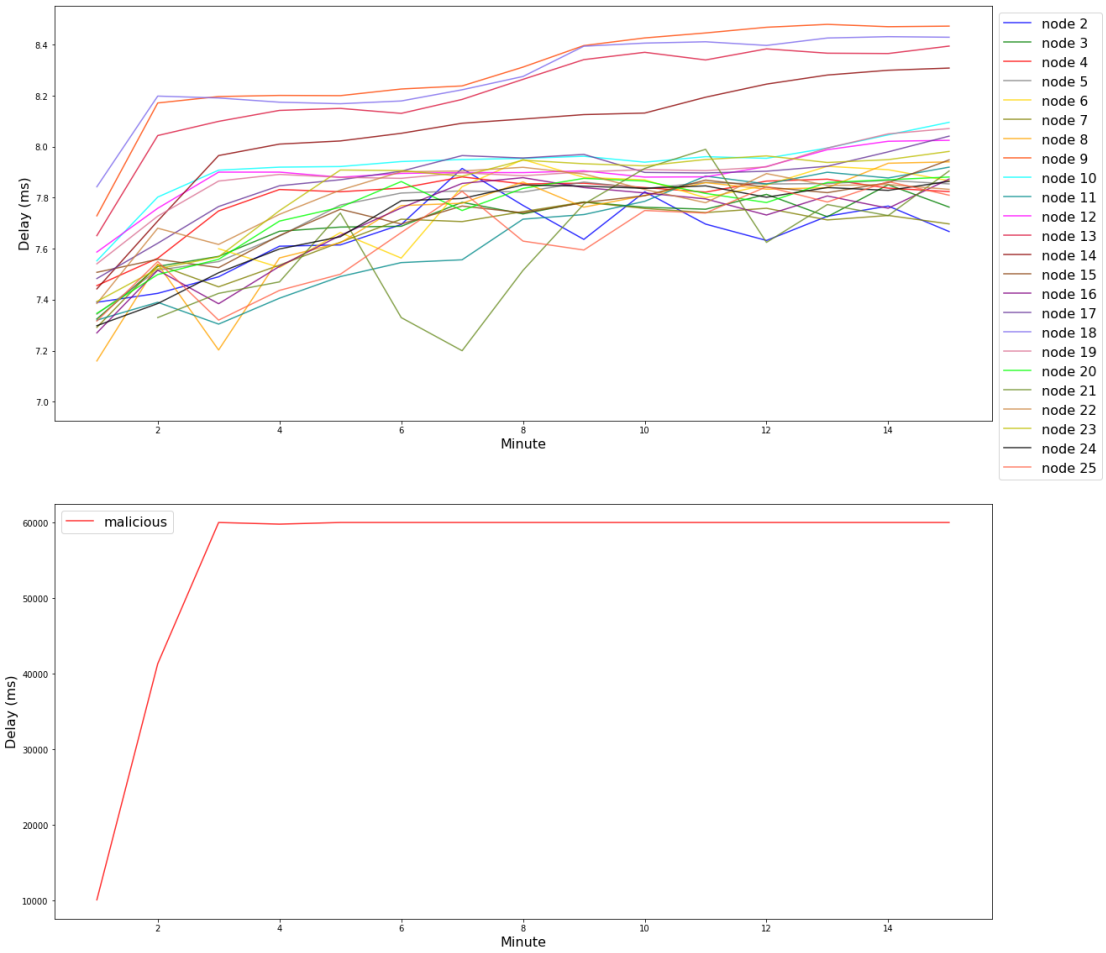
Σχήμα 5.13.b Γραφική Packets Received για τοπολογία Selective Forwarding Random

5.2.3.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.14 ο κακόβουλος κόμβος δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Επιπλέον, παρατηρούμε ότι απορρίπτει πολύ περισσότερα πακέτα απ' ότι οι υπόλοιποι κόμβοι, τα οποία αντιστοιχούν, τουλάχιστο, στα μισά πακέτα που λαμβάνει. Επίσεις, το throughput έχει μειωθεί στα 8.7 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.16 πακέτα το δευτερόλεπτο.



Σχήμα 5.14.α Γραφική Packets Dropped για τοπολογία Selective Forwarding Middle

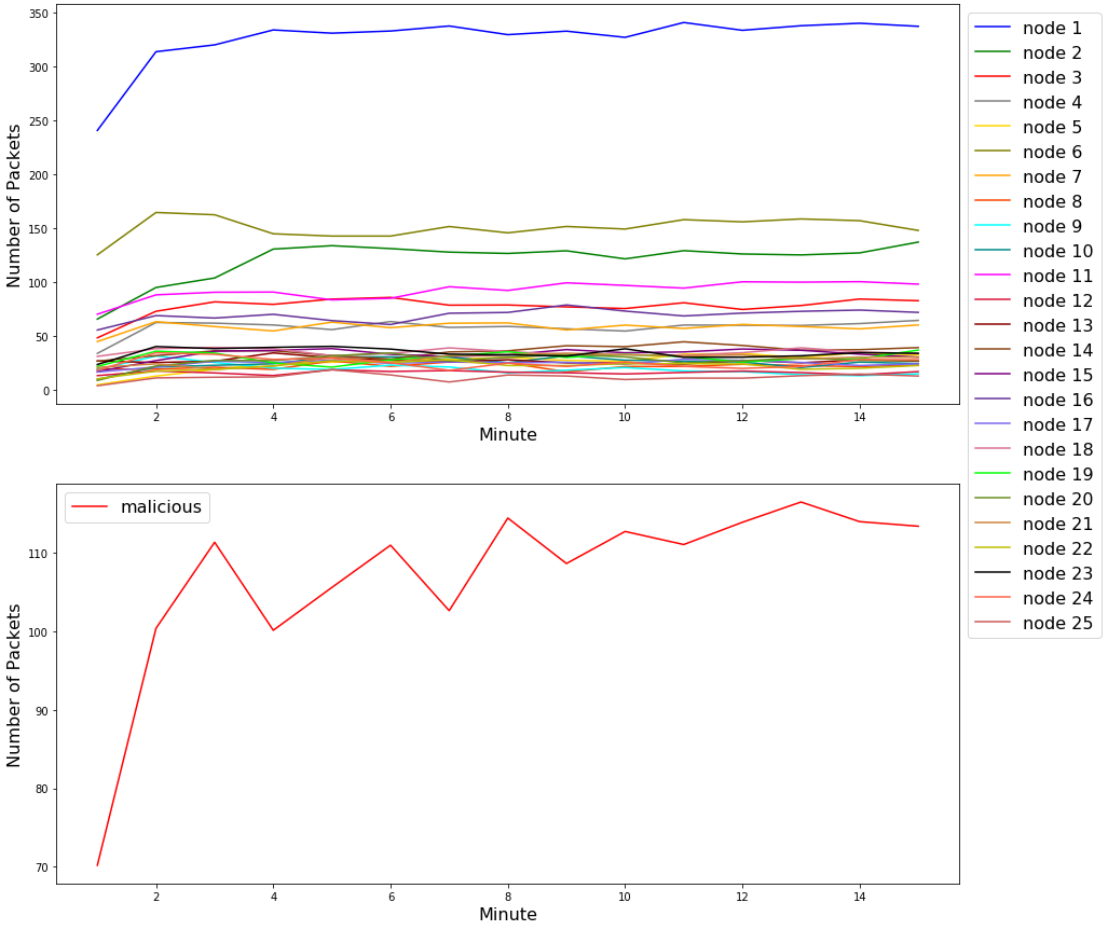


Σχήμα 5.14.b Γραφική Forwarding Delay για τοπολογία Selective Forwarding Middle

5.2.3.3 Τοπολογία Top

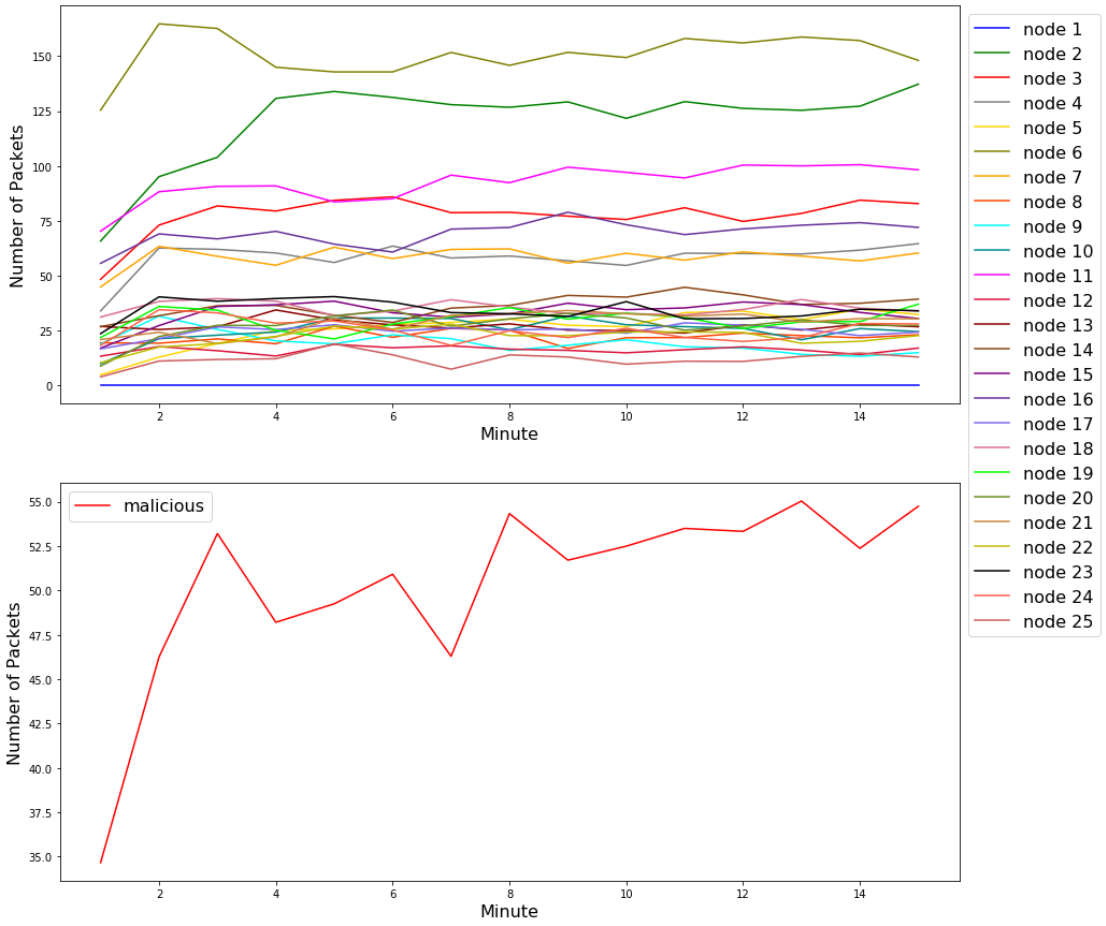
Όπως παρατηρούμε στα σχήματα 5.15 ο κακόβουλος κόμβος απορρίπτει τα μισά πακέτα που λαμβάνει (επομένως προωθούνται μόνο τα μισά πακέτα). Επιπλέον δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Επίσεις, το throughput έχει μειωθεί στα 5.6 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 6.58 πακέτα το δευτερόλεπτο.

Received



Σχήμα 5.15.α Γραφική Packets Received για τοπολογία Selective Forwarding Top

Forwarded



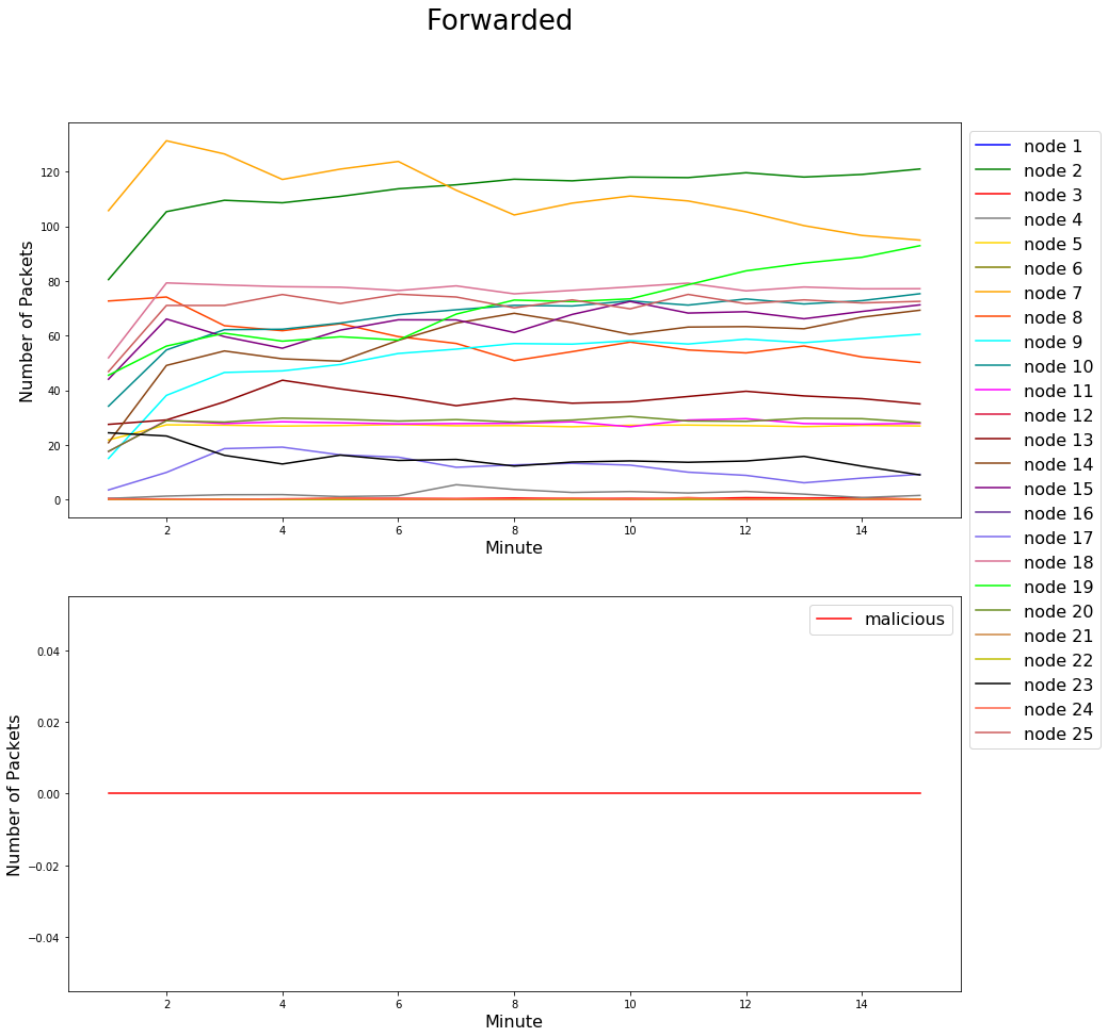
Σχήμα 5.15.b Γραφική Packets Forwarded για τοπολογία Selective Forwarding Top

5.2.4 Sinkhole

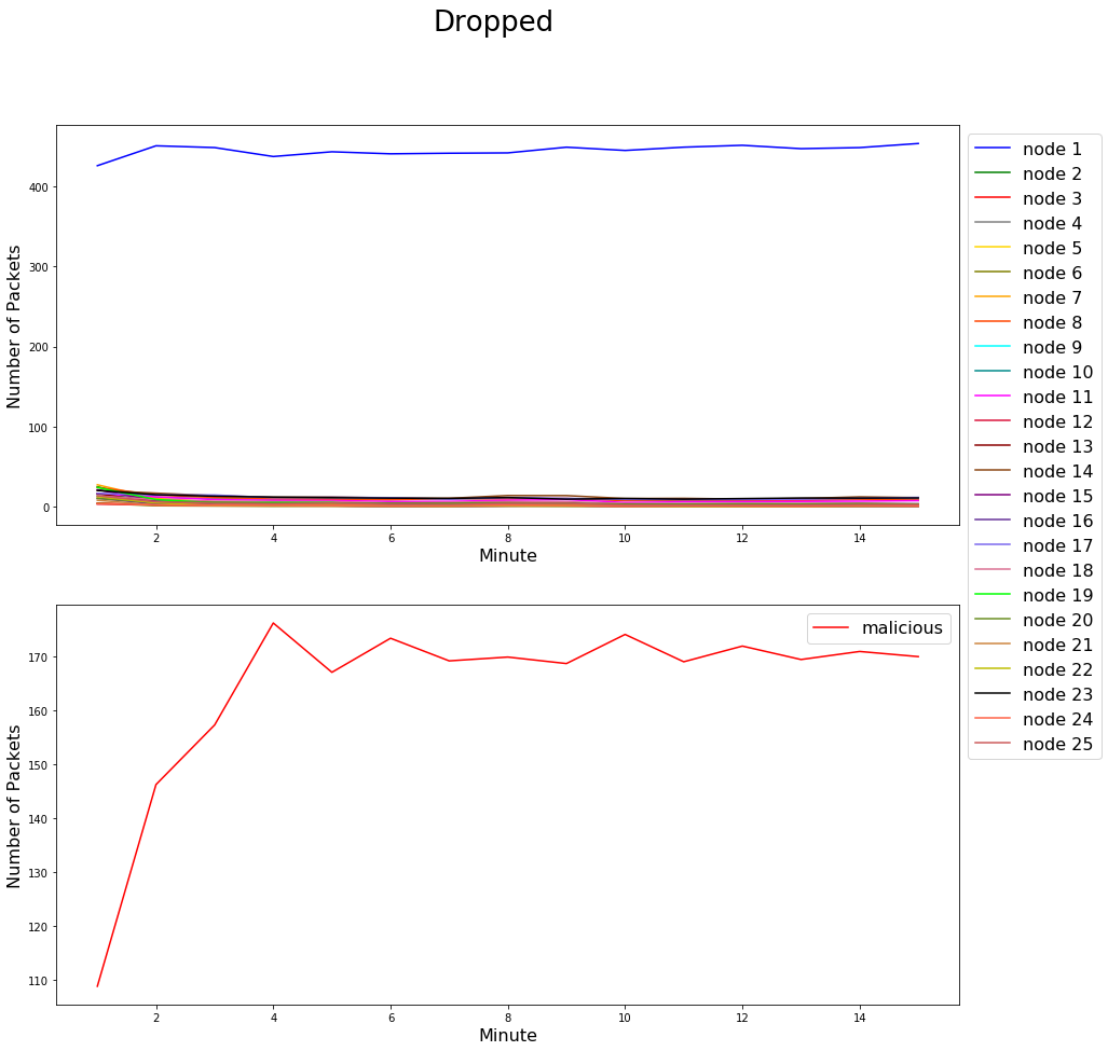
Ο ιός αναγκάζει τον κακόβουλο κόμβο να μιμείται τον κόμβο sink. Ο κόμβος το επιτυγχάνει αυτό διαφημίζοντας μικρό μονοπάτι προς το sink ή ακόμη και το ίδιο με αυτό του sink. Σκοπός του να πείσει τους γειτονικούς κόμβους ότι αυτός είναι ο sink και να παραλάβει αυτός τα πακέτα που προορίζονται για τον sink.

5.2.4.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.16 ο κακόβουλος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει, επομένως δεν προωθεί κανένα. Επιπλέον, το throughput έχει μειωθεί στα 7.4 πακέτα το δευτερόλεπτο, καθώς η απώλεια έχει αυξηθεί στα 4.42 πακέτα το δευτερόλεπτο.



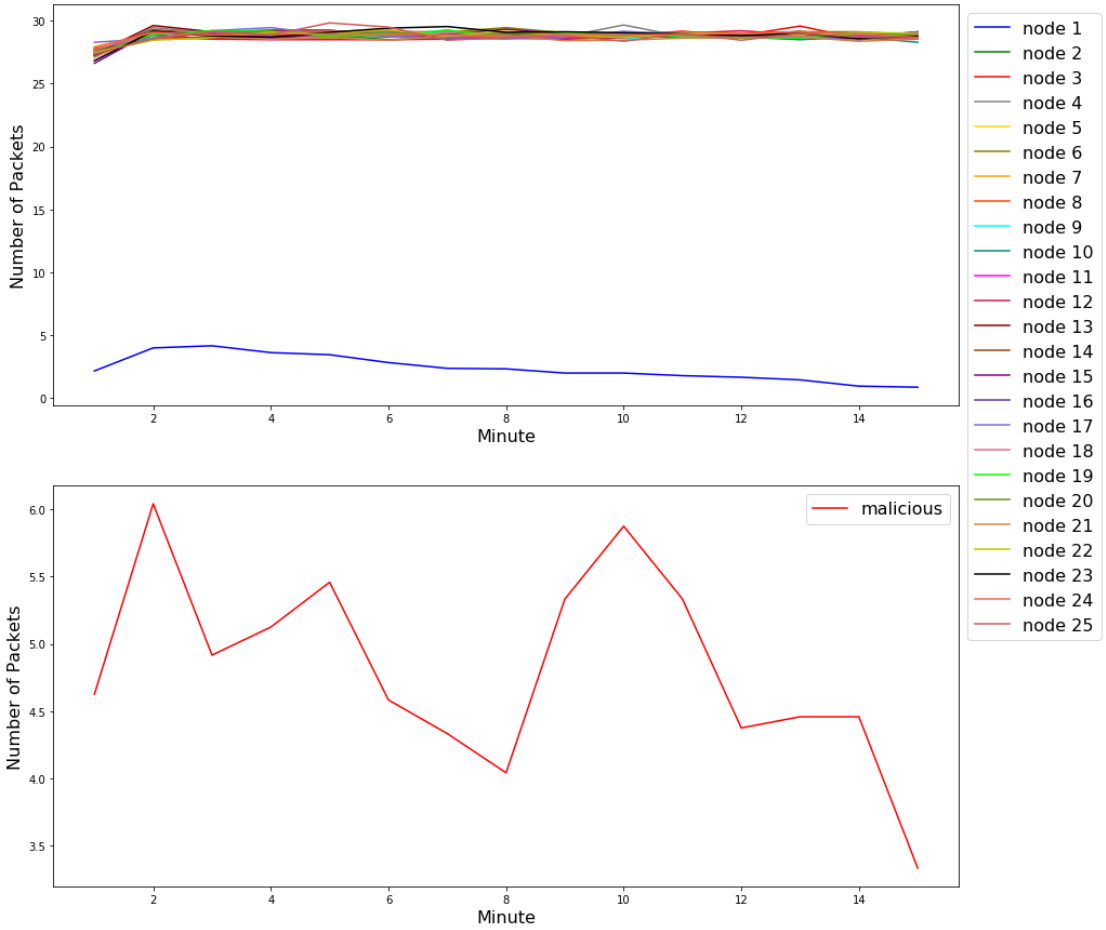
Σχήμα 5.16.a Γραφική Packets Forwarded για τοπολογία Sinkhole Random



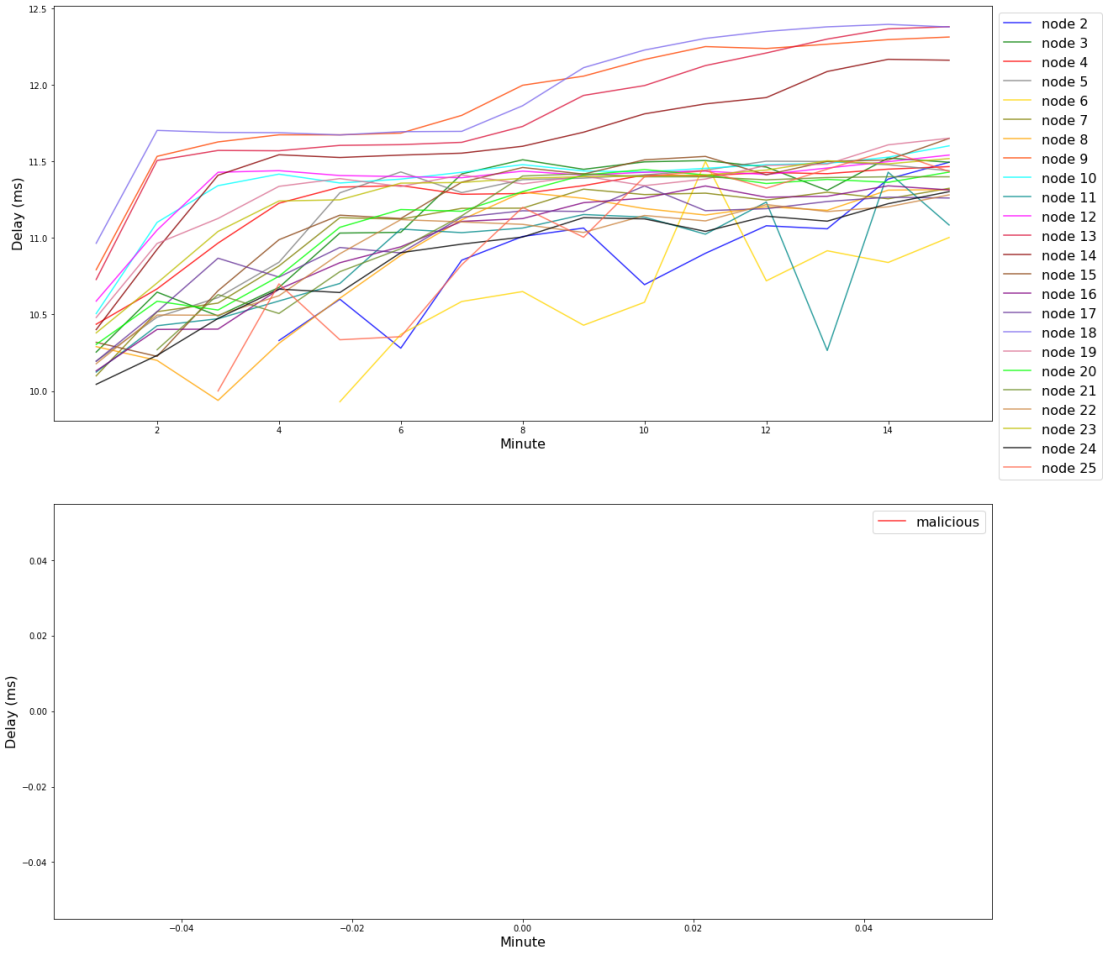
Σχήμα 5.16.b Γραφική Packets Dropped για τοπολογία Sinkhole Random

5.2.4.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.17 ο κακόβουλος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει, με αποτέλεσμα να μην προωθεί κανένα και έτσι να μην έχει καθόλου καθυστέρηση προώθησης. Επιπλέον, παρατηρούμε ότι δημιουργεί και στέλνει πολύ λιγότερα πακέτα απ' ότι οι άλλοι κόμβοι. επίσεις, το throughput έχει μειωθεί στα 8.5 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.17 πακέτα το δευτερόλεπτο.



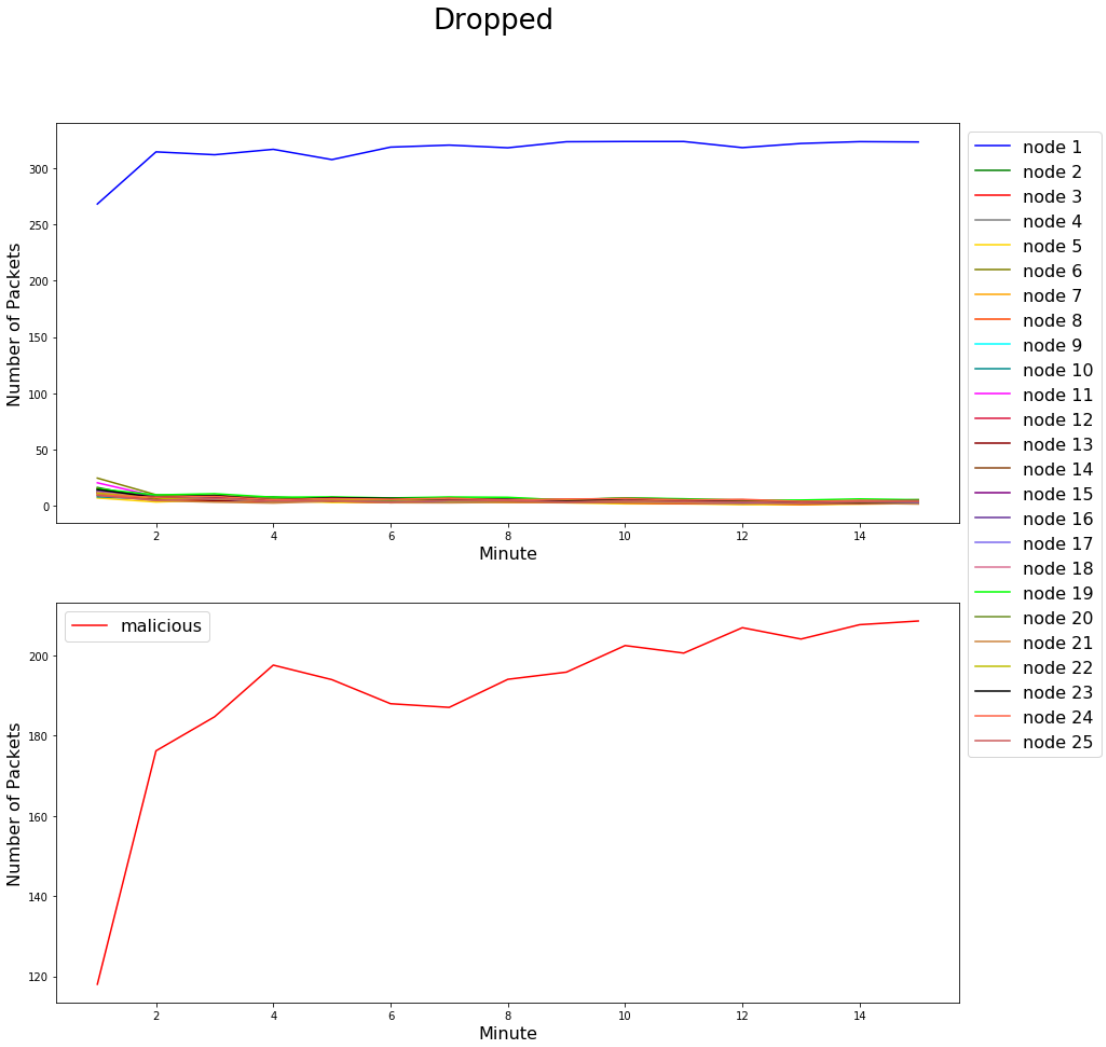
Σχήμα 5.17.α Γραφική Packets Sent για τοπολογία Sinkhole Middle



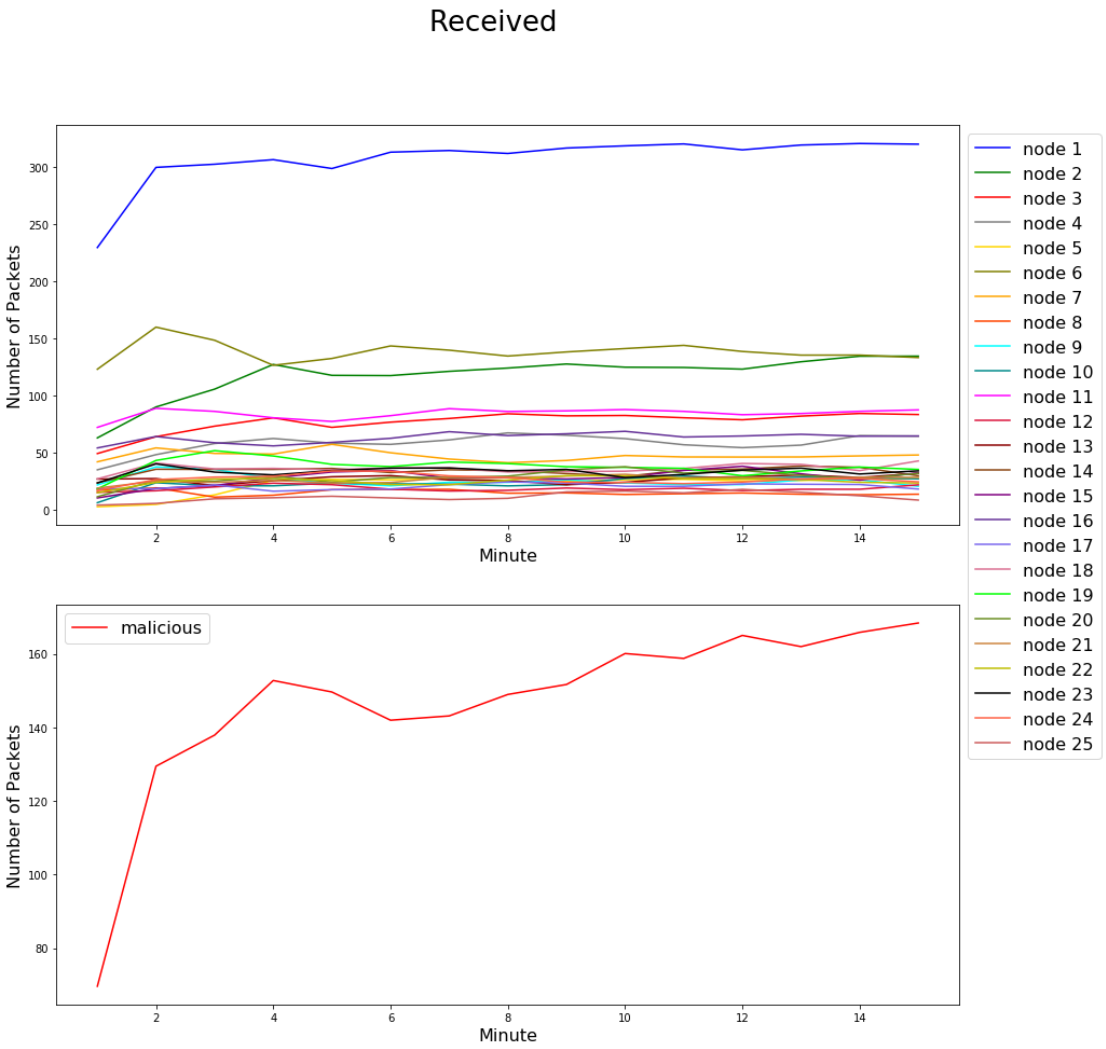
Σχήμα 5.17.b Γραφική Forwarding Delay για τοπολογία Sinkhole Middle

5.2.4.3 Τοπολογία Top

Όπως παρατηρούμε στα σχήματα 5.18 ο κακόβουλος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει και δεν προωθεί κανένα. Επομένως, έχει μηδενική καθυστέρηση προώθησης. Ακόμη, το throughput έχει μειωθεί στα 5.3 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 6.42 πακέτα το δευτερόλεπτο.



Σχήμα 5.18.α Γραφική Packets Dropped για τοπολογία Sinkhole Top



Σχήμα 5.18.b Γραφική Packets Received για τοπολογία Sinkhole Top

5.3 Malicious μετά από ένα χρονικό διάστημα

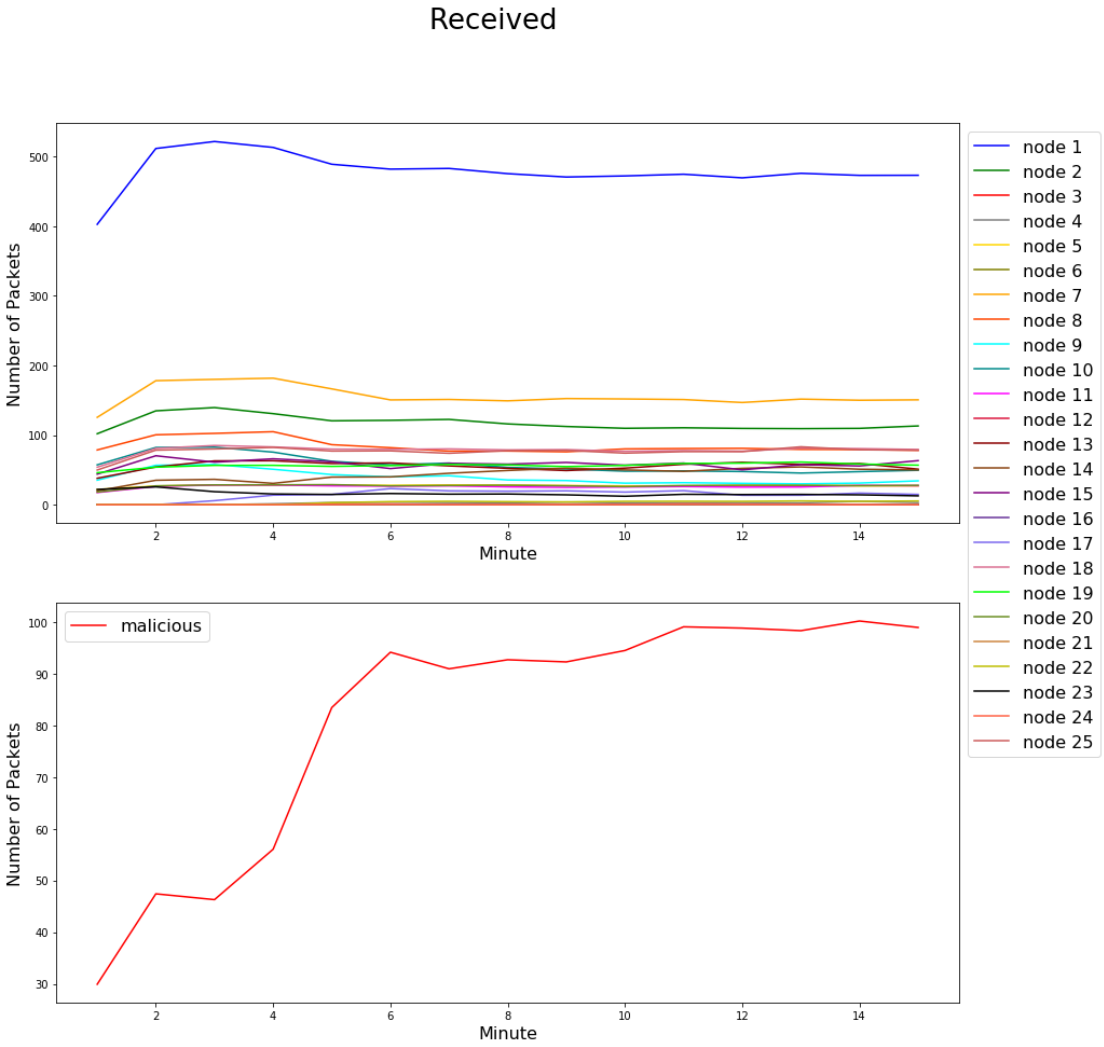
Τα σενάρια που ανήκουν σε αυτή την κατηγορία έχουν ένα κόμβο που είναι μολυσμένος με ιό. Ο ιός μπορεί να είναι ένας εκ των τεσσάρων (Black-hole, Flooding, Selective Forwarding, Sinkhole), και κάθε μολυσμένος κόμβος μπορεί να φέρει μόνο ένα ιό. Για κάθε ένα από τα σενάρια σε αυτή την κατηγορία ο ιός αρχίζει την κακόβουλη συμπεριφορά του μετά από ένα συγκεκριμένο χρονικό διάστημα αναλόγως των σεναρίων. Το διάστημα αυτό καθορίζεται από το χρόνο που χρειάζονται οι κόμβοι σε ένα σενάριο για να ενωθούν με τον sink μέσω του δέντρου.

5.3.1 Black-hole

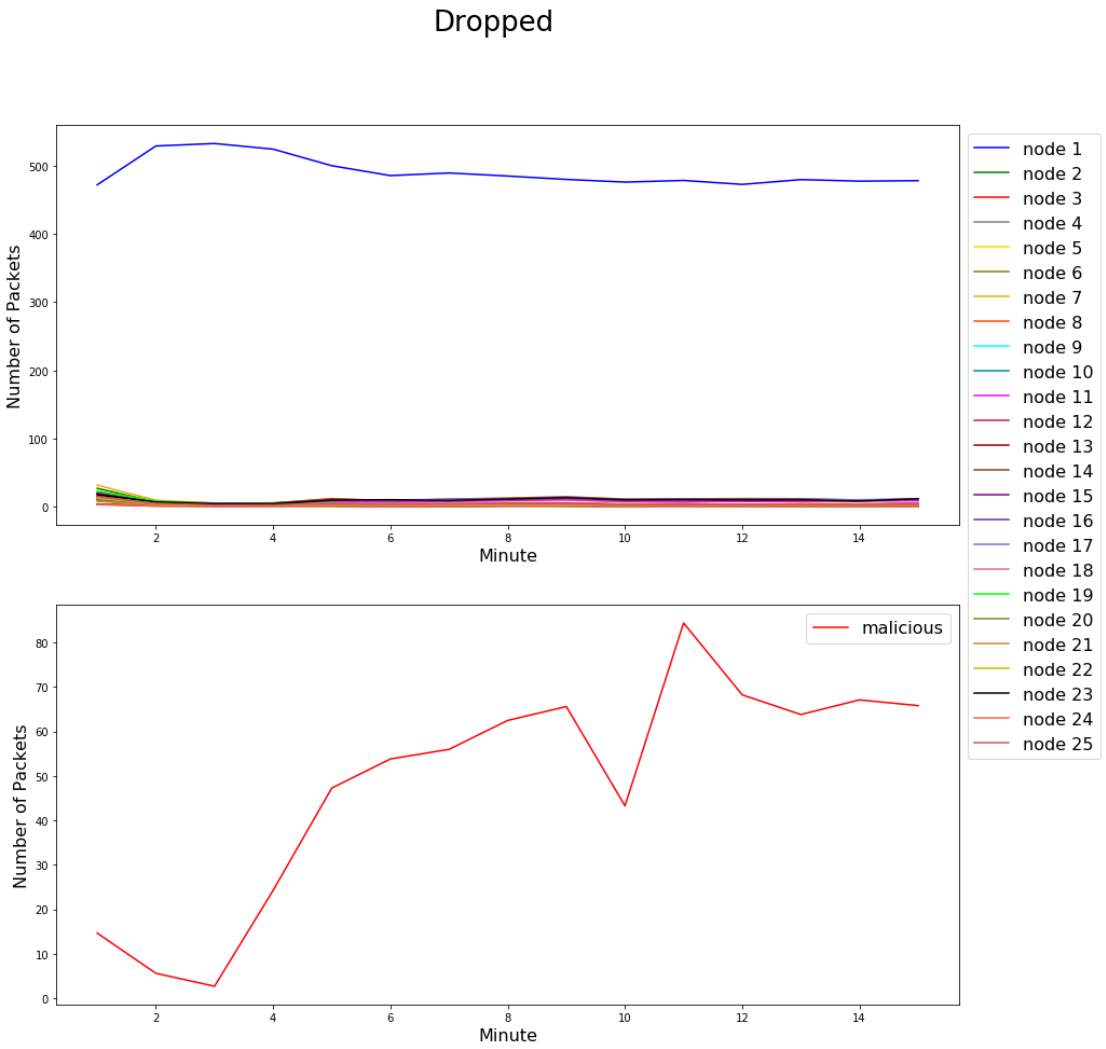
Σε αυτά τα σενάρια ο ιός αρχίζει την κακόβουλη συμπεριφορά μετά το δεύτερο λεπτό της προσομοίωσης.

5.3.1.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.19 ο κακόβουλος κόμβος απορρίπτει περισσότερα πακέτα σε σχέση με τους άλλους κόμβους, τα οποία αποτελούν μεγάλο μέρος των πακέτων που λαμβάνει (περίπου τα τρία τέταρτα). Επιπλέον, δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό), με αποτέλεσμα οι υγιείς κόμβοι να έχουν μεγαλύτερη καθυστέρηση στην προώθηση πακέτων σε αυτά τα σενάρια. Επίσεις, το throughput έχει μειωθεί στα 8 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 4.33 πακέτα δευτερόλεπτο.



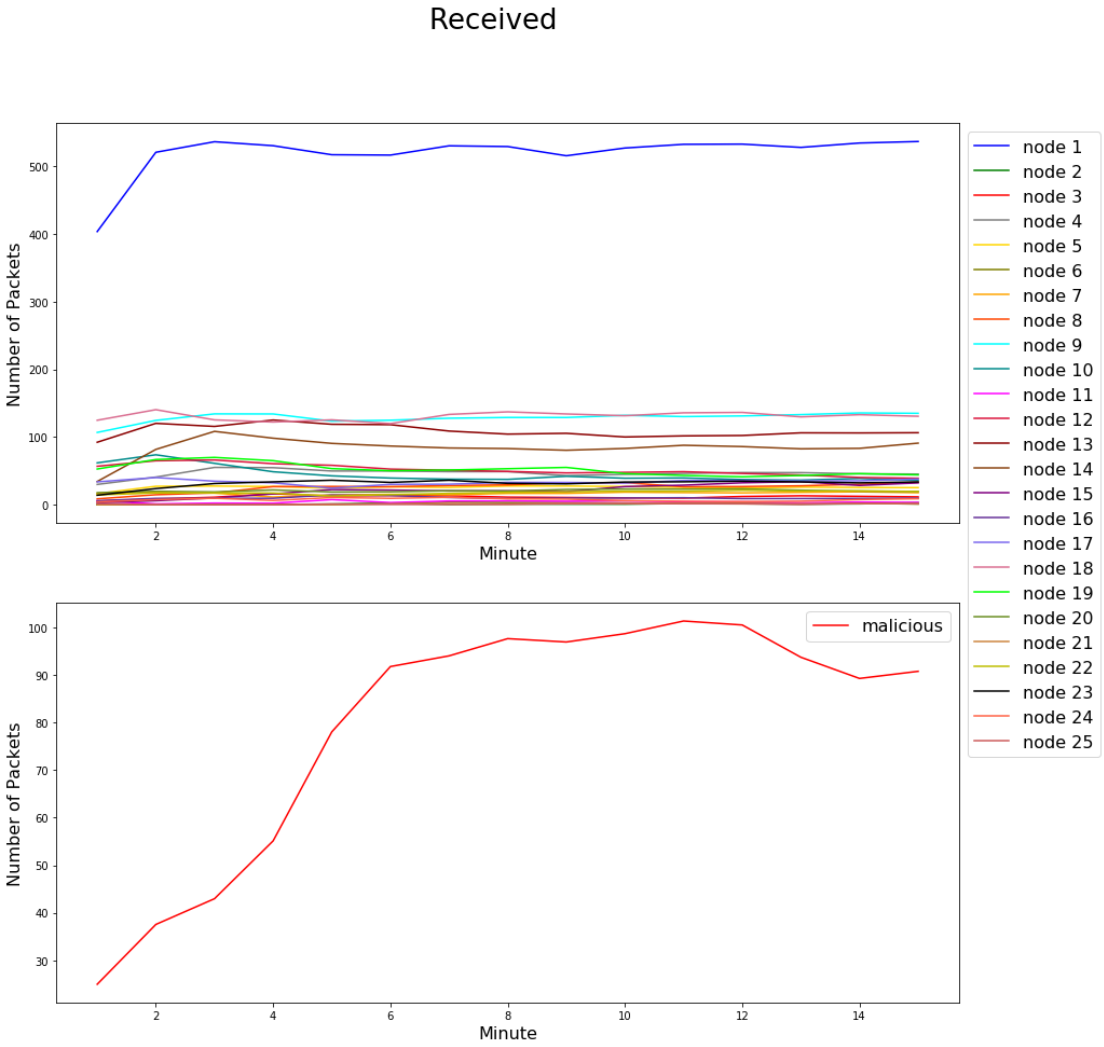
Σχήμα 5.19.a Γραφική Packets Received για τοπολογία Black-hole Random



Σχήμα 5.19.b Γραφική Packets Dropped για τοπολογία Black-hole Random

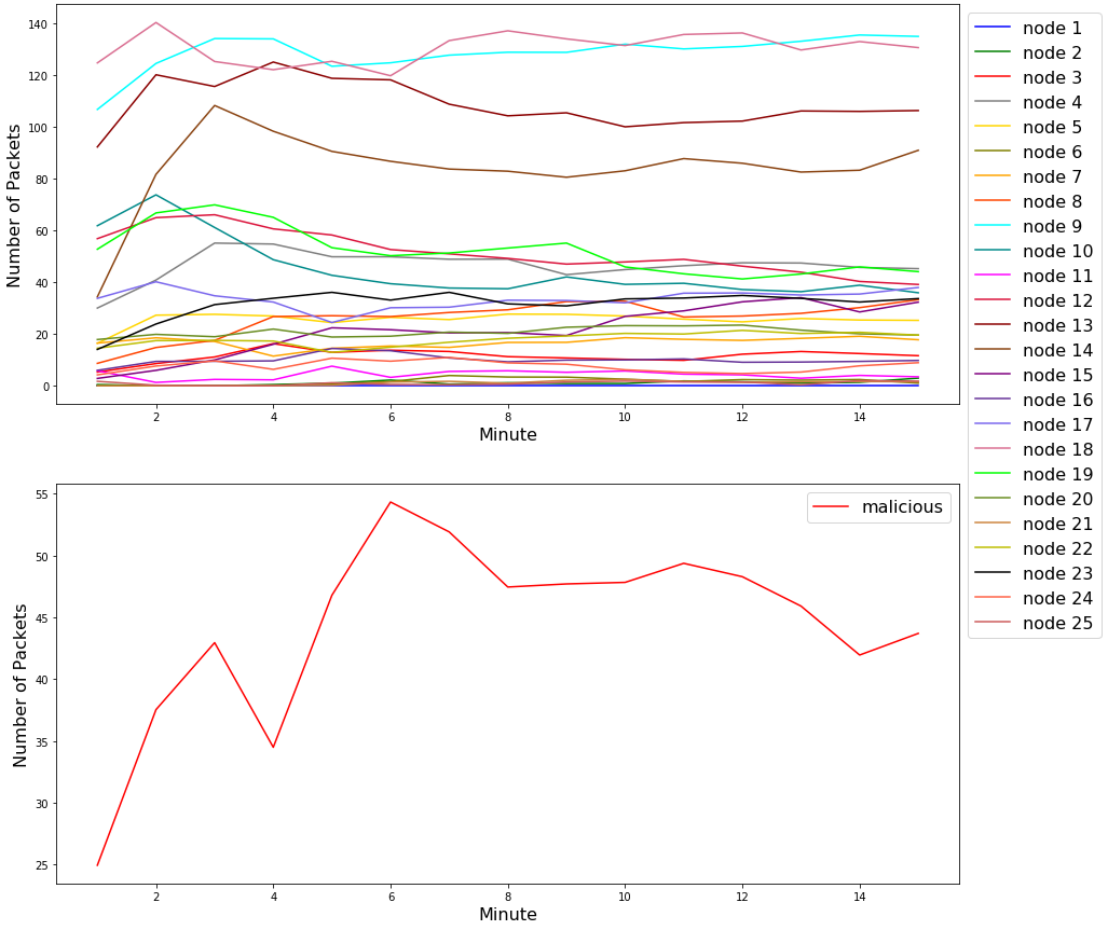
5.3.1.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.20 ο κακόβουλος κόμβος προωθεί λιγότερα από τα μισά πακέτα που λαμβάνει και απορρίπτει τα υπόλοιπα. Ακόμη, δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό) και προκαλεί τους υγιές κόμβους να έχουν μεγαλύτερη καθυστέρηση στην προώθηση πακέτων. Επίσης, το throughput έχει μειωθεί στα 8.7 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.33 πακέτα το δευτερόλεπτο.



Σχήμα 5.20.α Γραφική Packets Received για τοπολογία Black-hole Middle

Forwarded

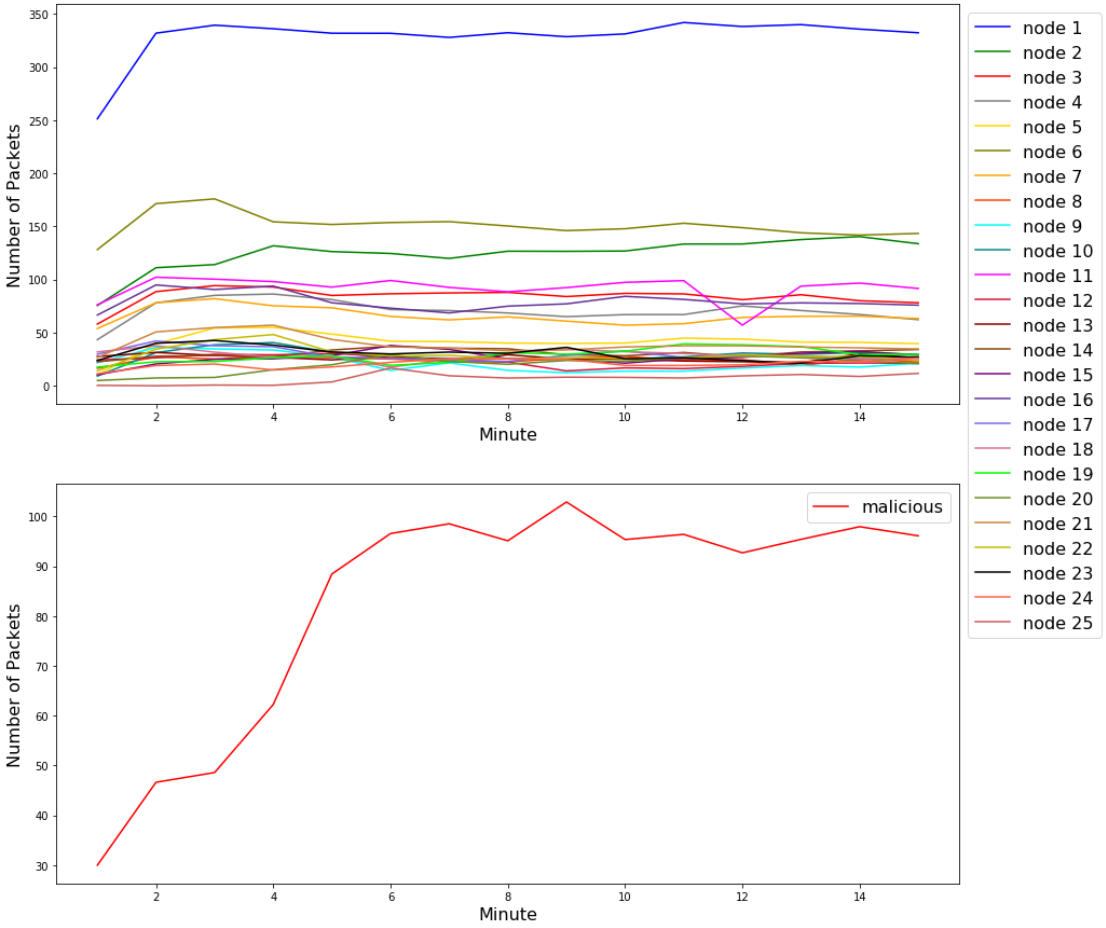


Σχήμα 5.20.b Γραφική Packets Forwarded για τοπολογία Black-hole Middle

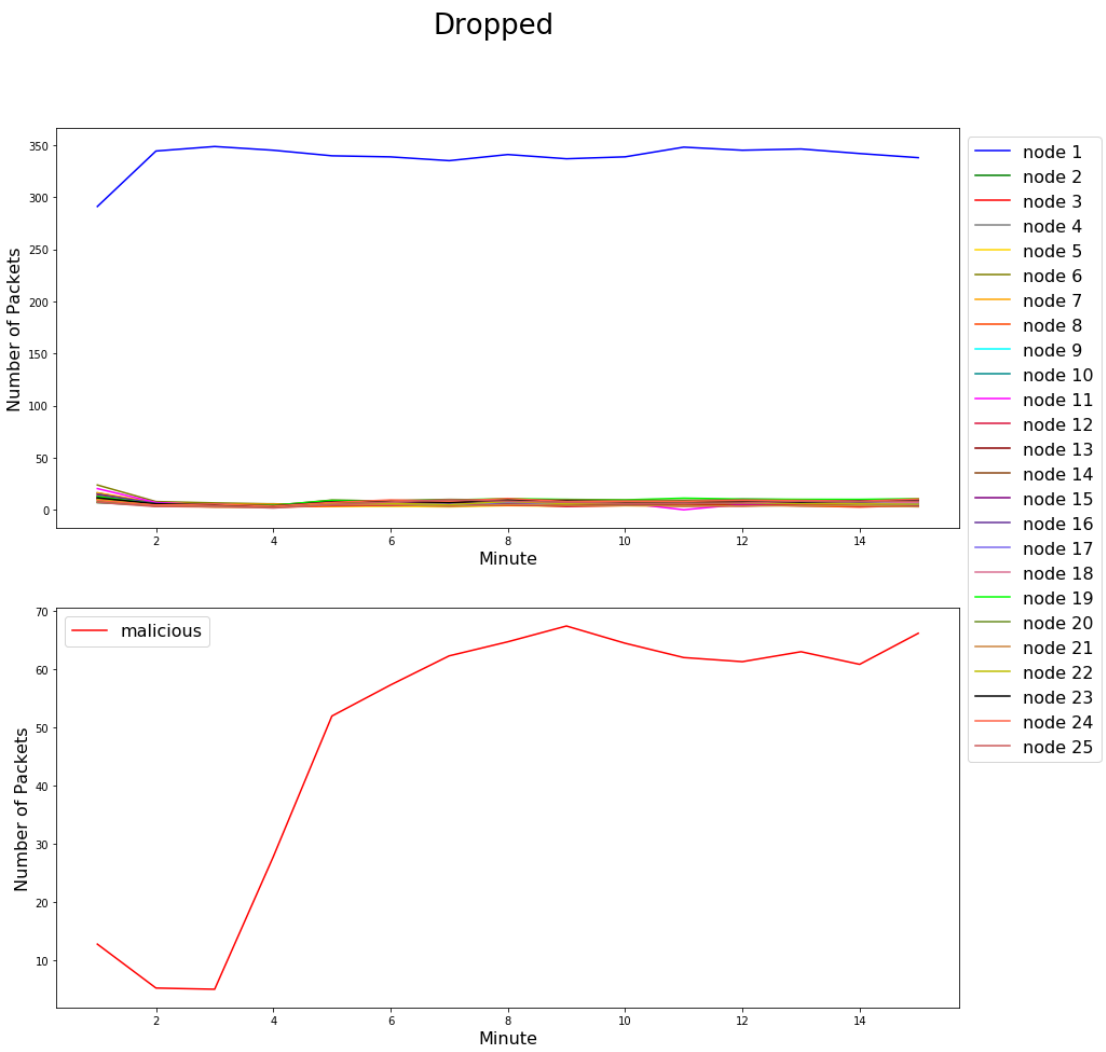
5.3.1.3 Τοπολογία Top

Όπως παρατηρούμε στα σχήματα 5.21 ο κακόβουλος κόμβος απορρίπτει μεγάλο μέρος των πακέτων που λαμβάνει (σχεδόν τα τρία τέταρτα) και δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Ακόμη παρατηρούμε ότι οι υγιείς κόμβοι έχουν μεγαλύτερη καθυστέρηση στην προώθηση πακέτων και το throughput έχει μειωθεί στα 5.6 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 6.83 πακέτα το δευτερόλεπτο.

Received



Σχήμα 5.21.a Γραφική Packets Received για τοπολογία Black-hole Top



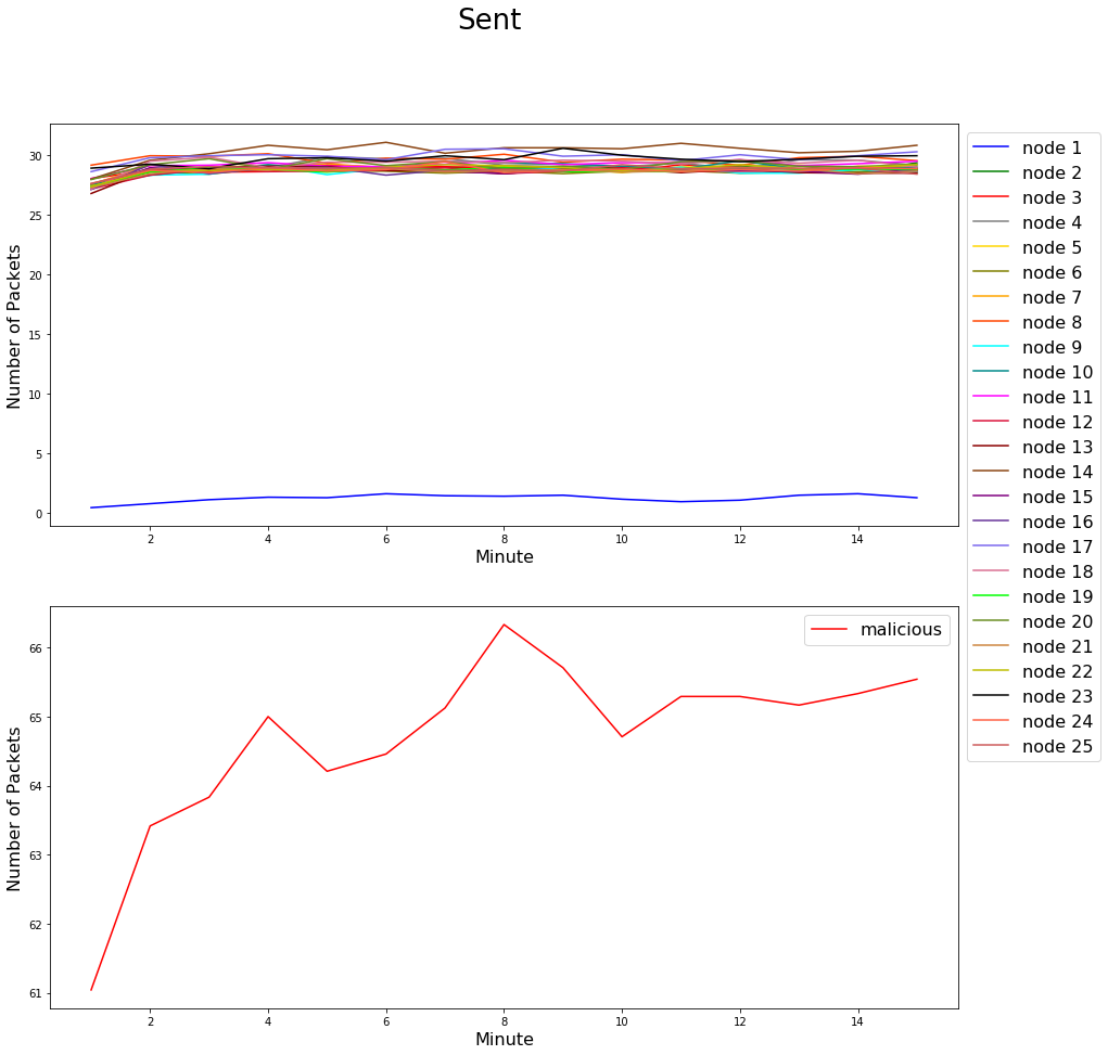
Σχήμα 5.21.b Γραφική Packets Dropped για τοπολογία Black-hole Top

5.3.2 Flooding

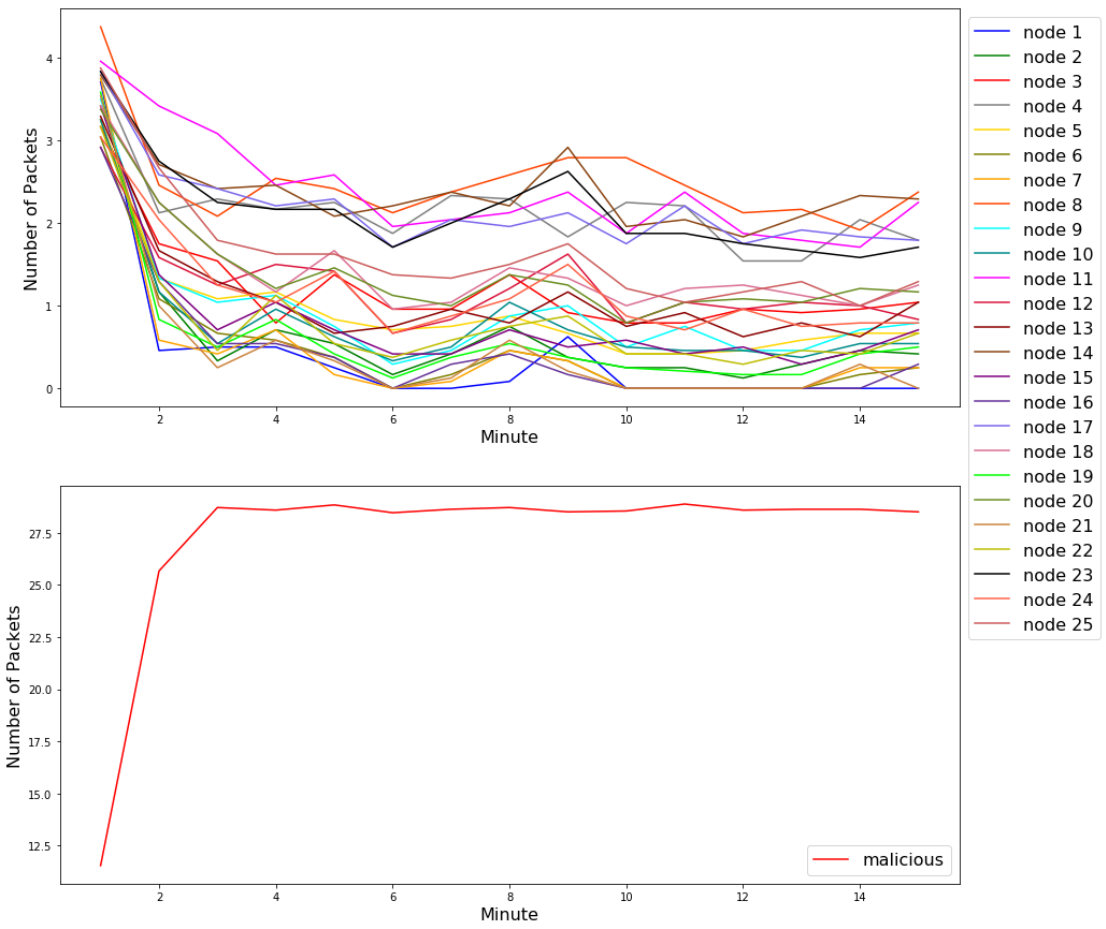
Σε αυτά τα σενάρια ο ιός αρχίζει την κακόβουλη συμπεριφορά μετά το πρώτο λεπτό της προσομοίωσης.

5.3.2.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.22 ο κακόβουλος κόμβος στέλνει τα επταπλάσια DIO μηνύματα ενώ δημιουργεί και στέλνει τα διπλάσια πακέτα σε σχέση με τους άλλους κόμβους. Επίσης, το throughput έχει μειωθεί στα 8.2 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 4.5 πακέτα το δευτερόλεπτο.



Σχήμα 5.22.a Γραφική Packets Sent για τοπολογία Flooding Random

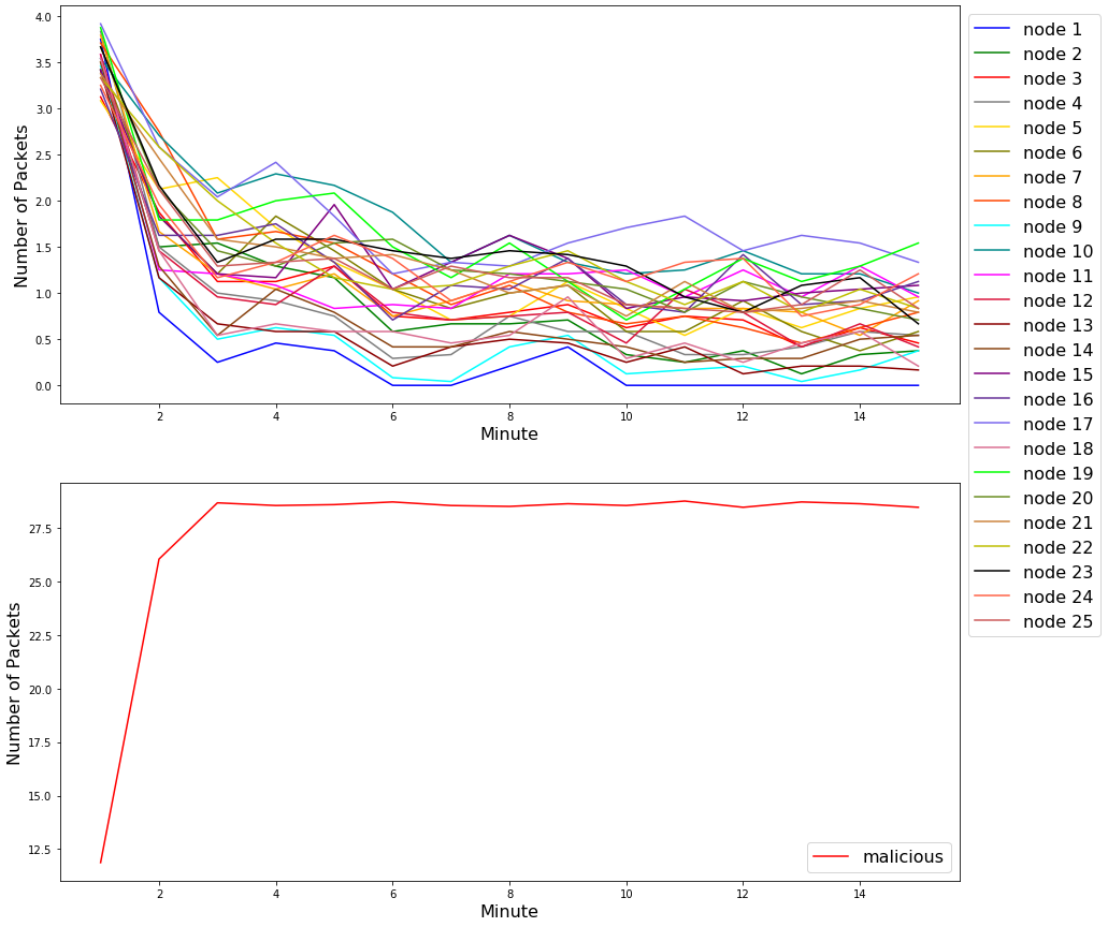


Σχήμα 5.22.b Γραφική DIO Packets για τοπολογία Flooding Random

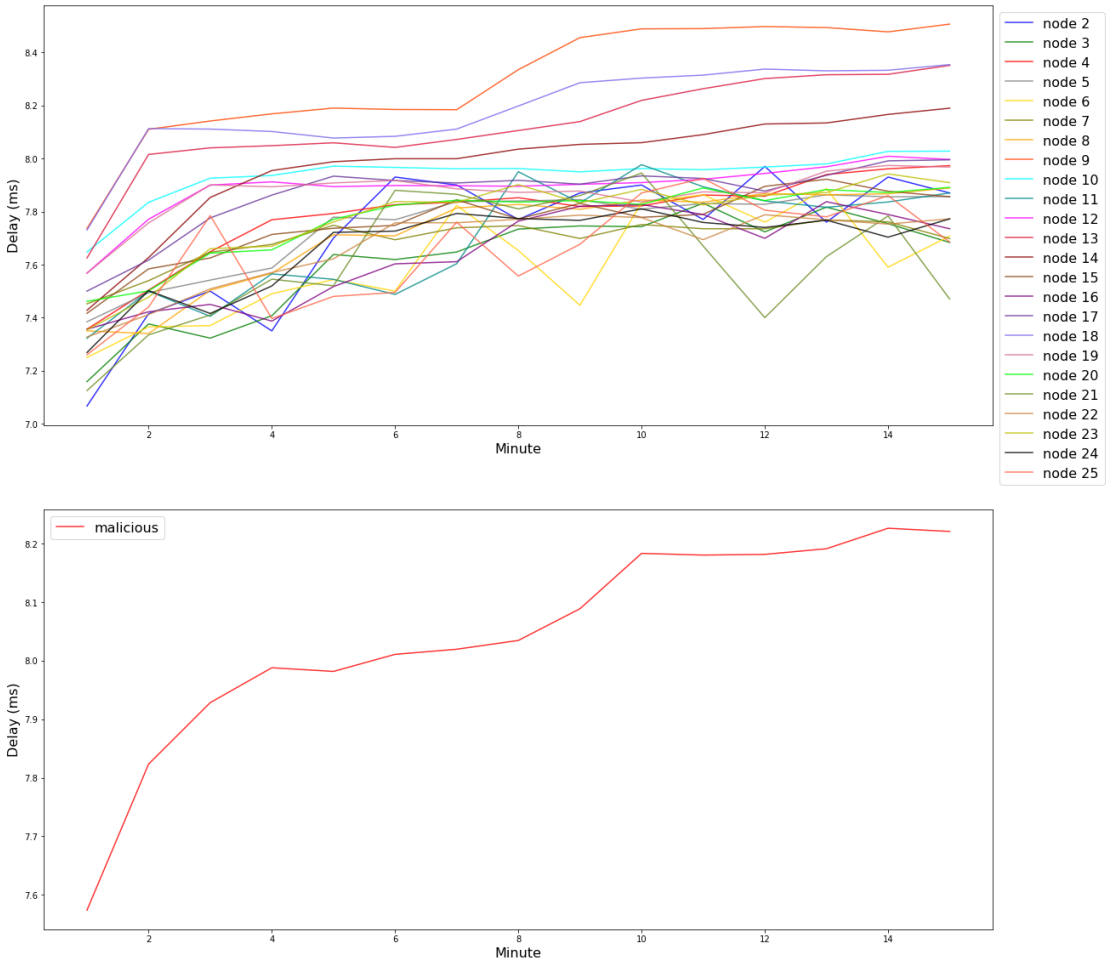
5.3.2.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.23 ο κακόβουλος κόμβος στέλνει τα επταπλάσια DIO μηνύματα ενώ δημιουργεί και στέλνει τα διπλάσια πακέτα σε σχέση με τους άλλους κόμβους. Ακόμη παρατηρούμε ότι η καθυστέρηση προώθησης τείνει να αυξάνεται σταθερά για όλους τους κόμβους. Επιπλέον, το throughput έχει μειωθεί στα 9 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.83 πακέτα το δευτερόλεπτο.

DIOs



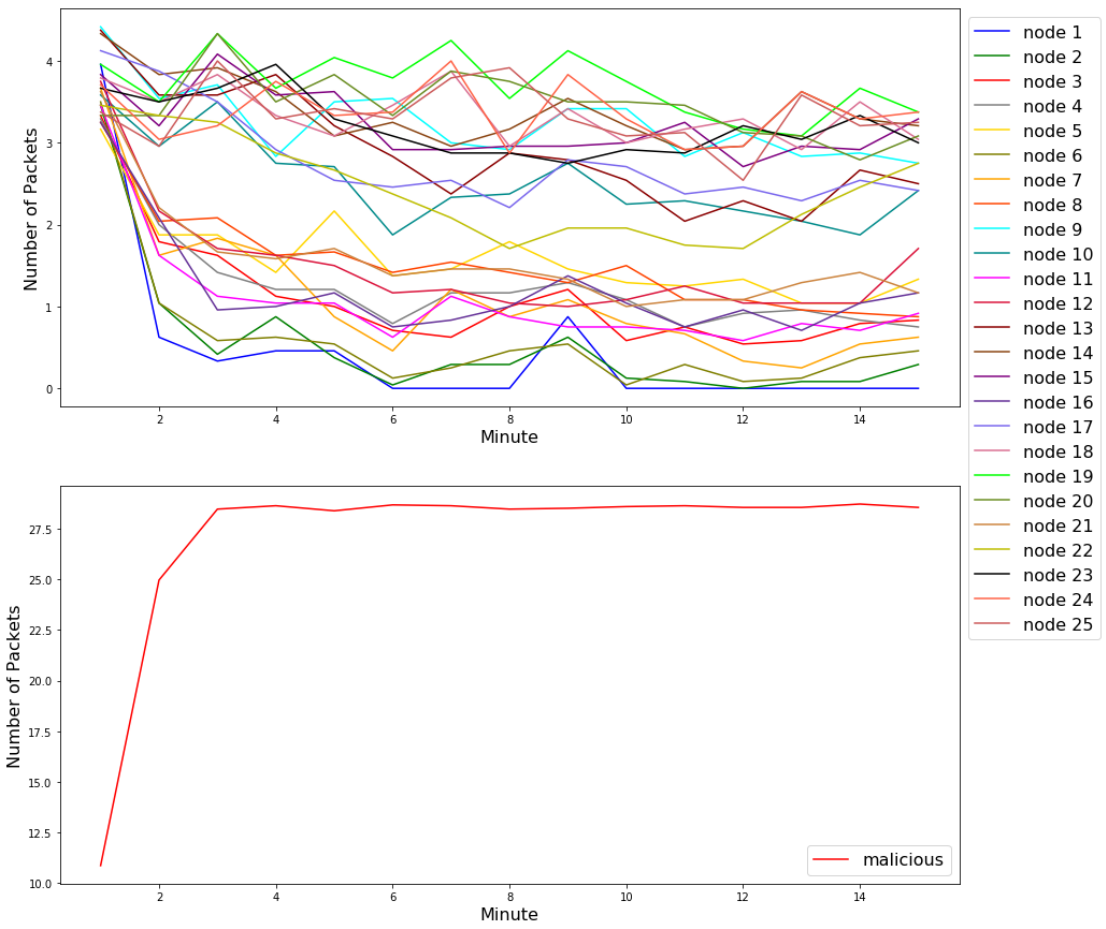
Σχήμα 5.23.α Γραφική DIO Packets για τοπολογία Flooding Middle



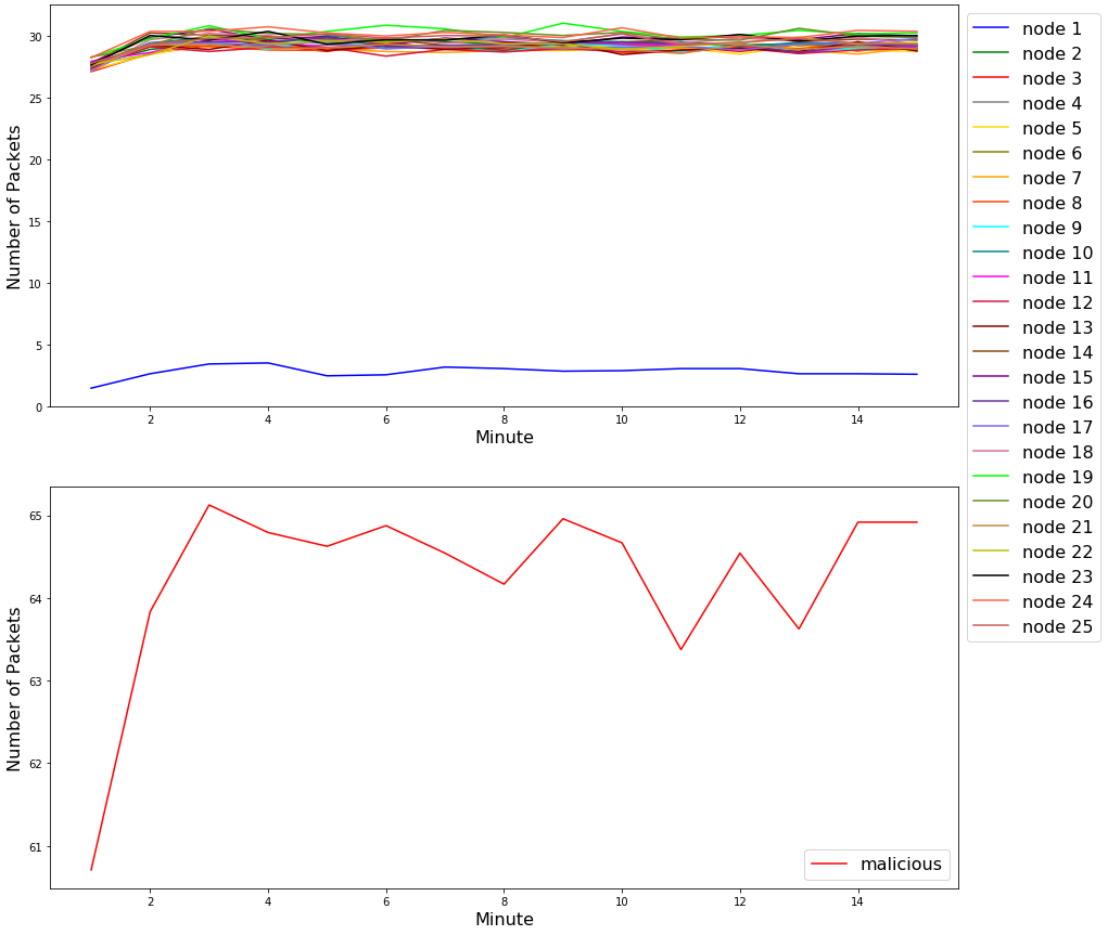
Σχήμα 5.23.b Γραφική Forwarding Delay για τοπολογία Flooding Middle

5.3.2.3 Τοπολογία Top

Όπως παρατηρούμε στα σχήματα 5.24 ο κακόβουλος κόμβος στέλνει τα επταπλάσια DIO μηνύματα ενώ δημιουργεί και στέλνει τα διπλάσια πακέτα σε σχέση με τους άλλους κόμβους. Επιπλέον, το throughput έχει μειωθεί στα 5.4 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 7.3 πακέτα το δευτερόλεπτο.



Σχήμα 5.24.α Γραφική DIO Packets για τοπολογία Flooding Top



Σχήμα 5.24.b Γραφική Packets Sent για τοπολογία Flooding Top

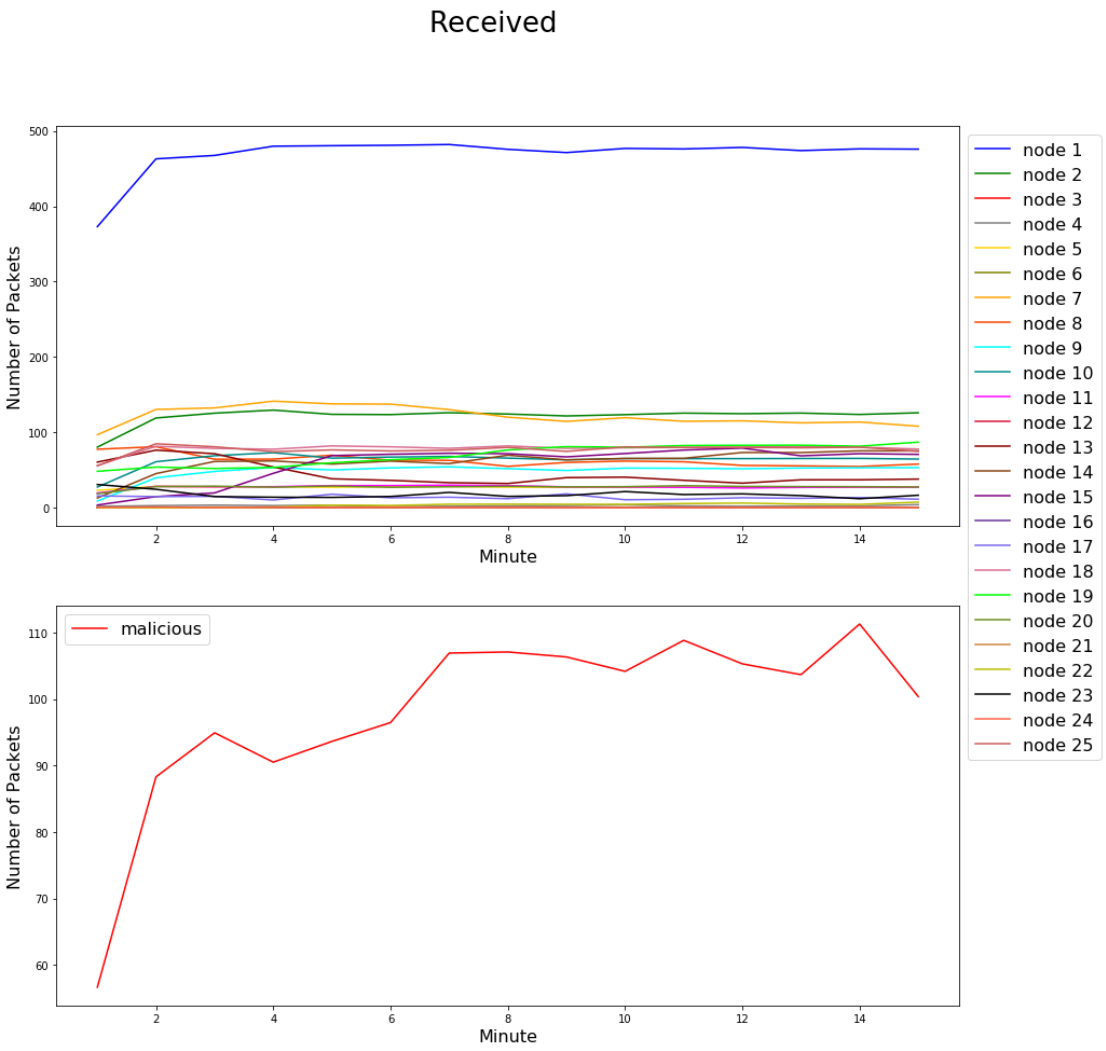
5.3.3 Selective Forwarding

Σε αυτά τα σενάρια ο ιός αρχίζει την κακόβουλη συμπεριφορά του μετά το πρώτο λεπτό της προσομοίωσης.

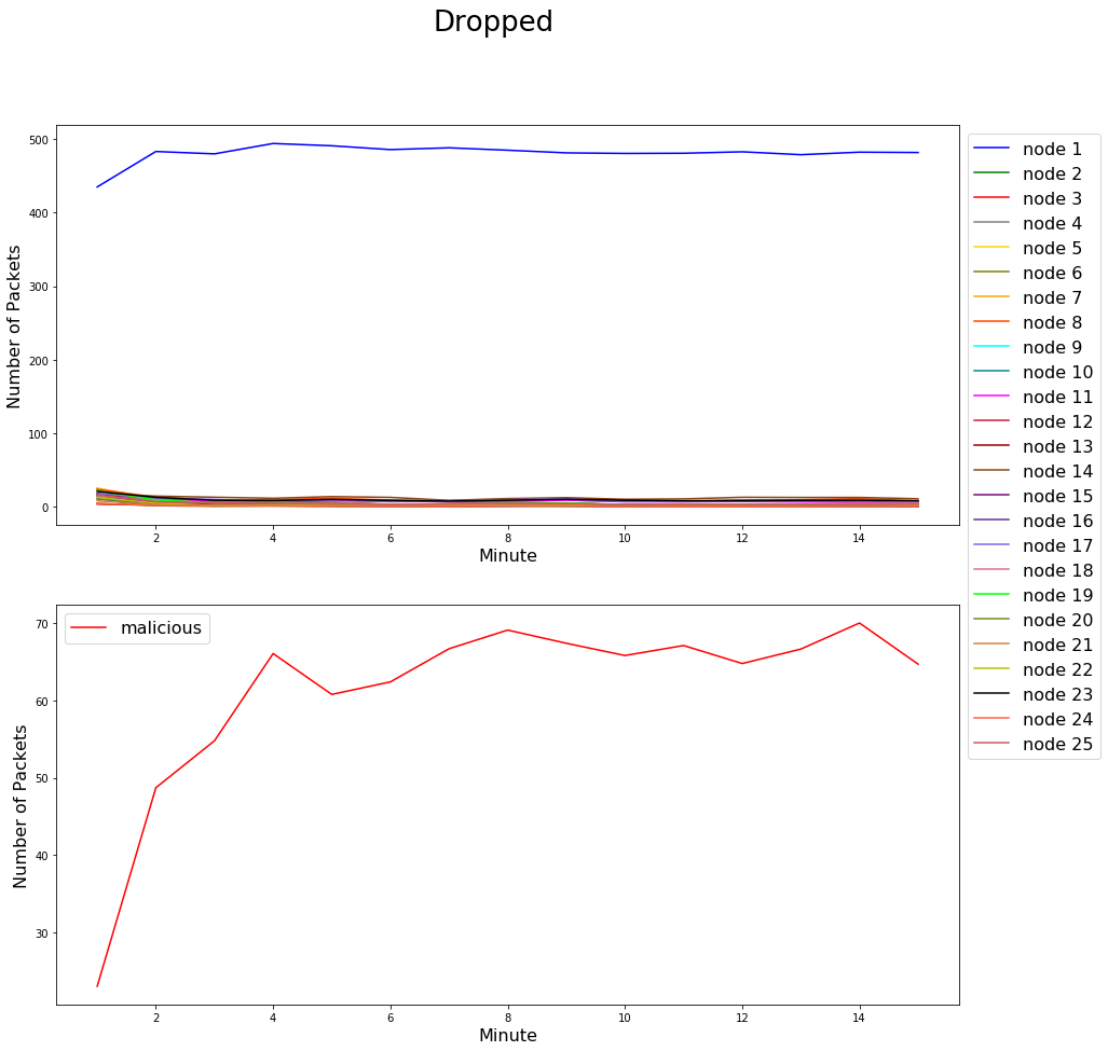
5.3.3.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.25 ο κακόβουλος κόμβος απορρίπτει τα μισά πακέτα που λαμβάνει, αριθμός πολύ μεγαλύτερος από τον αντίστοιχο για τους υπόλοιπους κόμβους. Επιπλέον, δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του

διαστήματος που χρησιμοποιούμε (1 λεπτό) και το throughput έχει μειωθεί στα 8 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 4.33 πακέτα το δευτερόλεπτο.



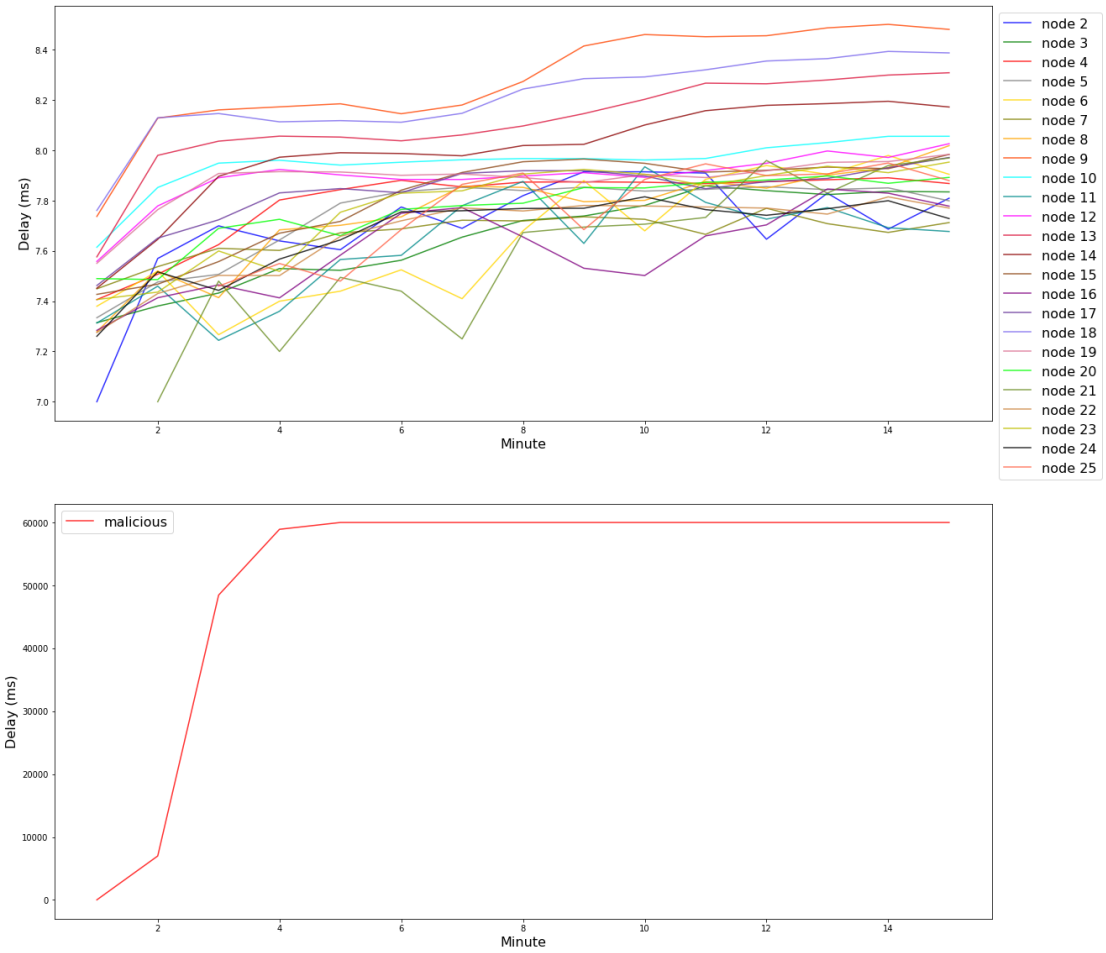
Σχήμα 5.25.α Γραφική Packets Received για τοπολογία Selective Forwarding Random



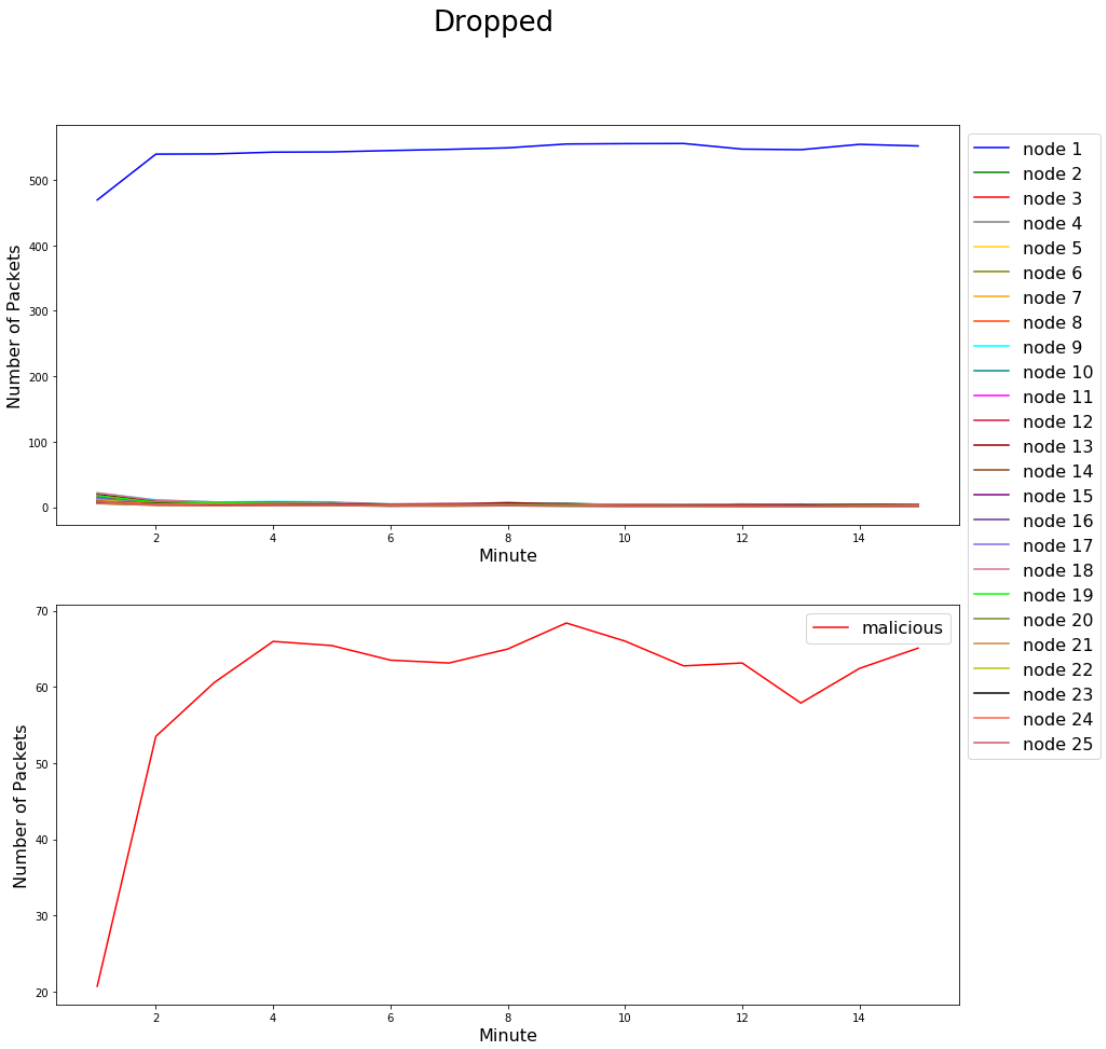
Σχήμα 5.25.b Γραφική Packets Dropped για τοπολογία Selective Forwarding Random

5.3.3.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.26 ο κακόβουλος κόμβος απορρίπτει τα μισά πακέτα που λαμβάνει, αριθμός πολύ μεγαλύτερος από τον αντίστοιχο για τους υπόλοιπους κόμβους, και δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Ακόμη, το throughput έχει μειωθεί στα 9 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.17 πακέτα το δευτερόλεπτο.



Σχήμα 5.26.α Γραφική Forwarding Delay για τοπολογία Selective Forwarding Middle

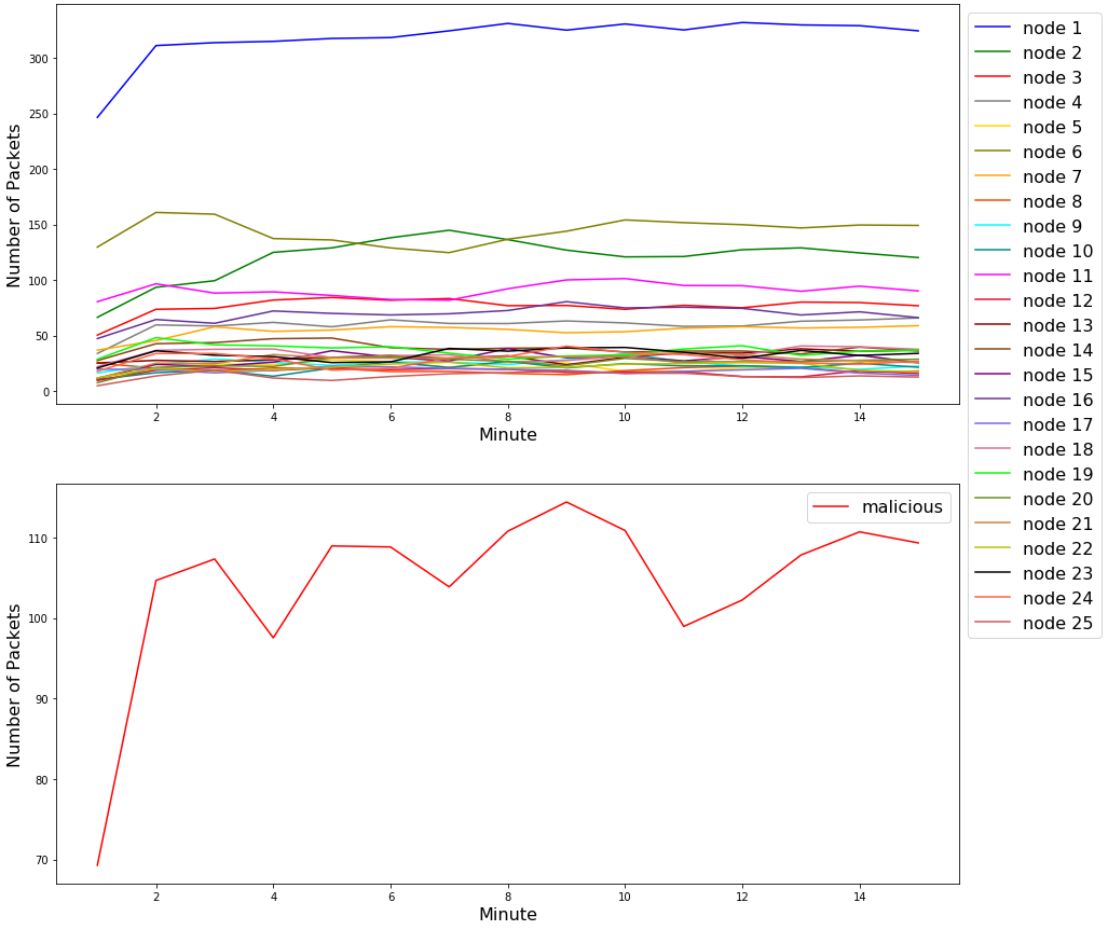


Σχήμα 5.26.b Γραφική Packets Dropped για τοπολογία Selective Forwarding Middle

5.3.3.3 Τοπολογία Top

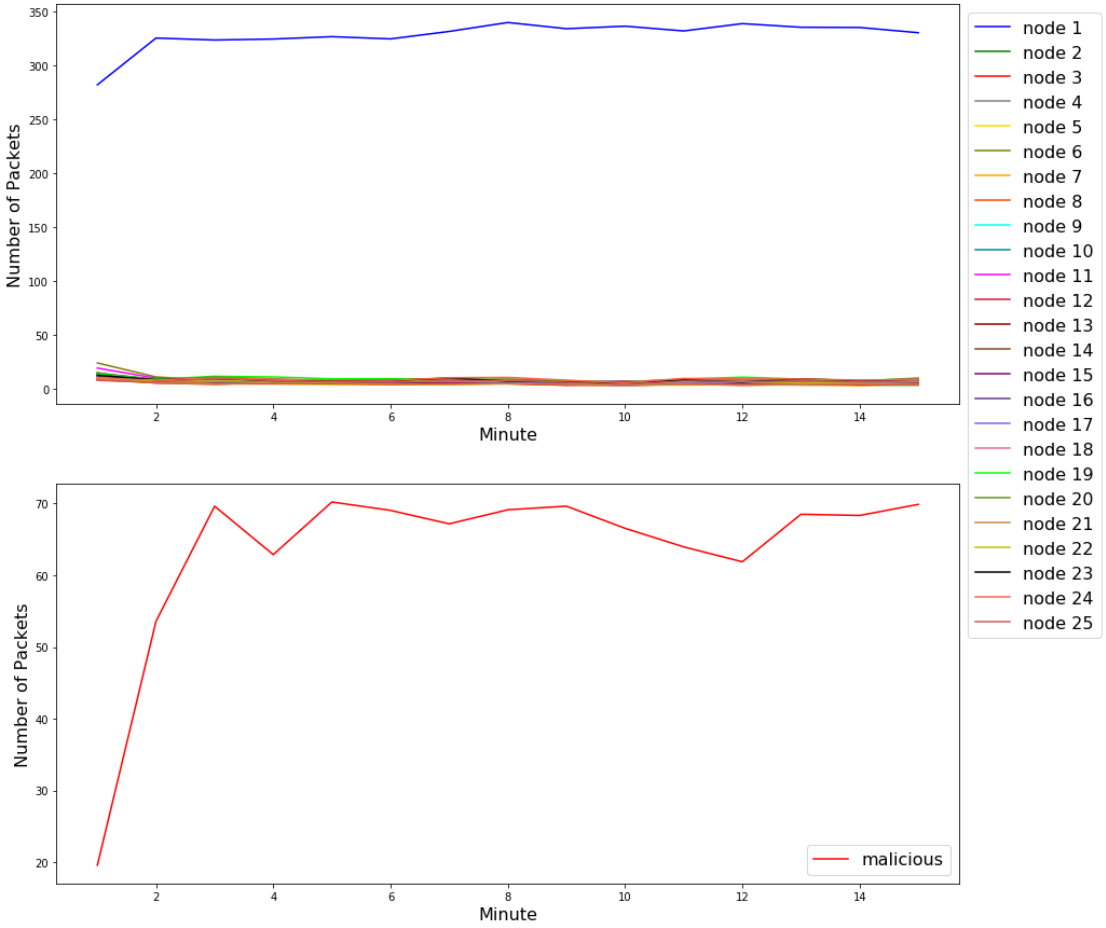
Όπως παρατηρούμε στα σχήματα 5.27 ο κακόβουλος κόμβος απορρίπτει πολύ περισσότερα πακέτα απ' ότι οι υπόλοιποι κόμβοι. Συγκεκριμένα, παρατηρείται να απορρίπτει τα μισά πακέτα που λαμβάνει και να δημιουργεί καθυστέρηση προώθησης πακέτων μεγαλύτερη του διαστήματος που χρησιμοποιούμε (1 λεπτό). Ακόμη, το throughput έχει μειωθεί στα 5.5 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 6.83 πακέτα το δευτερόλεπτο.

Received



Σχήμα 5.27.α Γραφική Packets Received για τοπολογία Selective Forwarding Top

Dropped



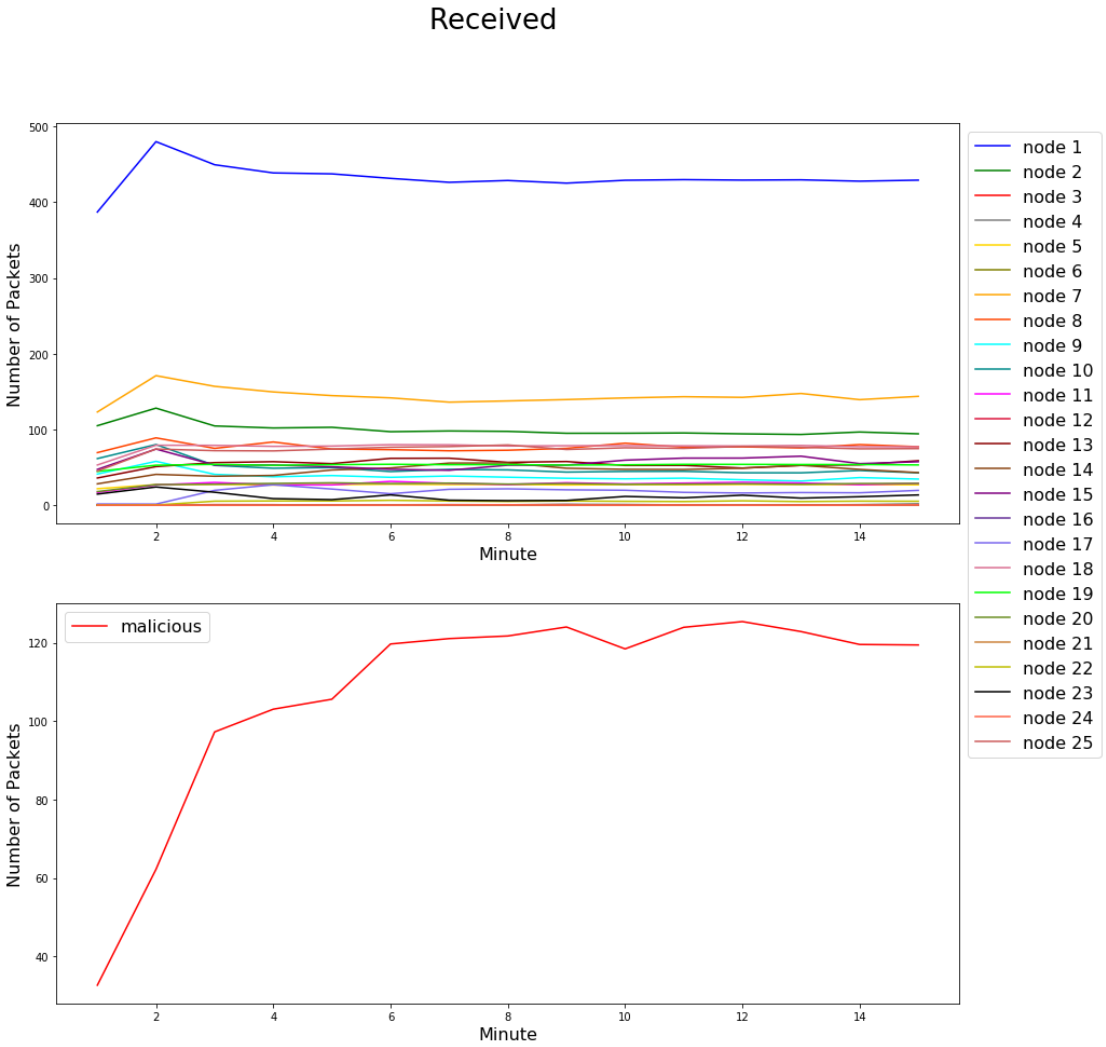
Σχήμα 5.27.b Γραφική Packets Dropped για τοπολογία Selective Forwarding Top

5.3.4 Sinkhole

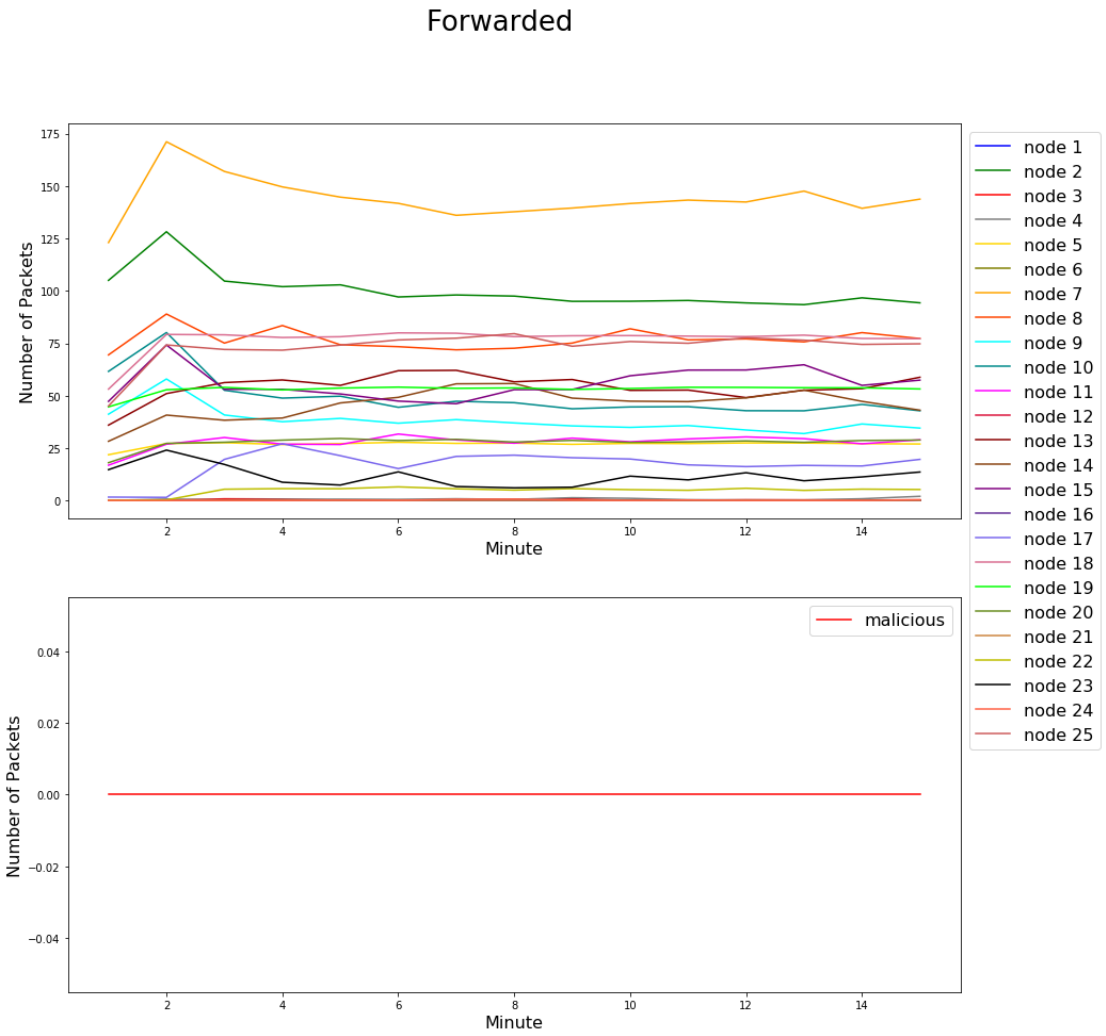
Σε αυτά τα σενάρια ο ιός αρχίζει την κακόβουλη συμπεριφορά του μετά το πρώτο λεπτό της προσομοίωσης.

5.3.4.1 Τοπολογία Random

Όπως παρατηρούμε στα σχήματα 5.28 ο κακόβουλος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει και δεν προωθεί κανένα, με αποτέλεσμα να έχει μηδενική καθυστέρηση προώθησης. Επιπλέον, το throughput έχει μειωθεί στα 7.1 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 4.7 πακέτα το δευτερόλεπτο.



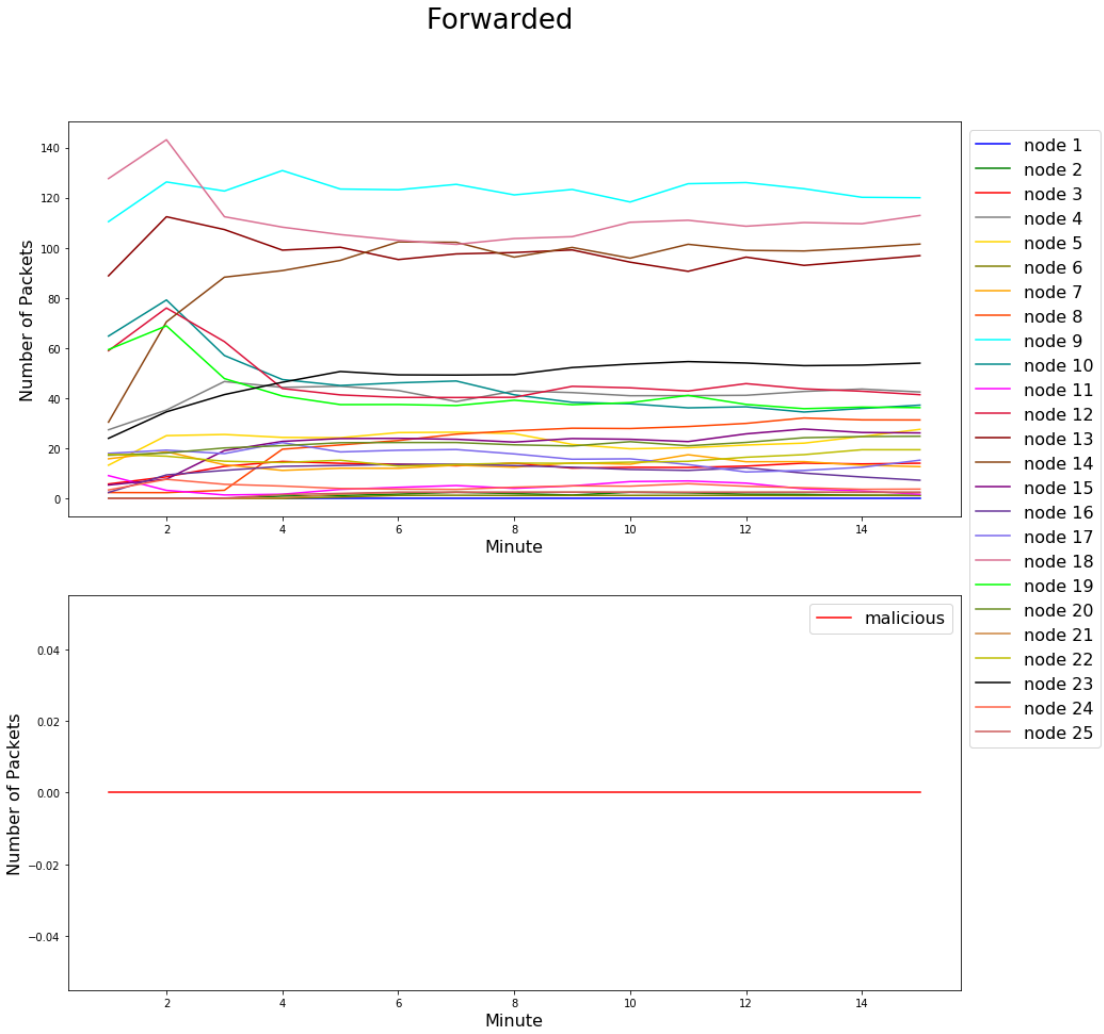
Σχήμα 5.28.α Γραφική Packets Received για τοπολογία Sinkhole Random



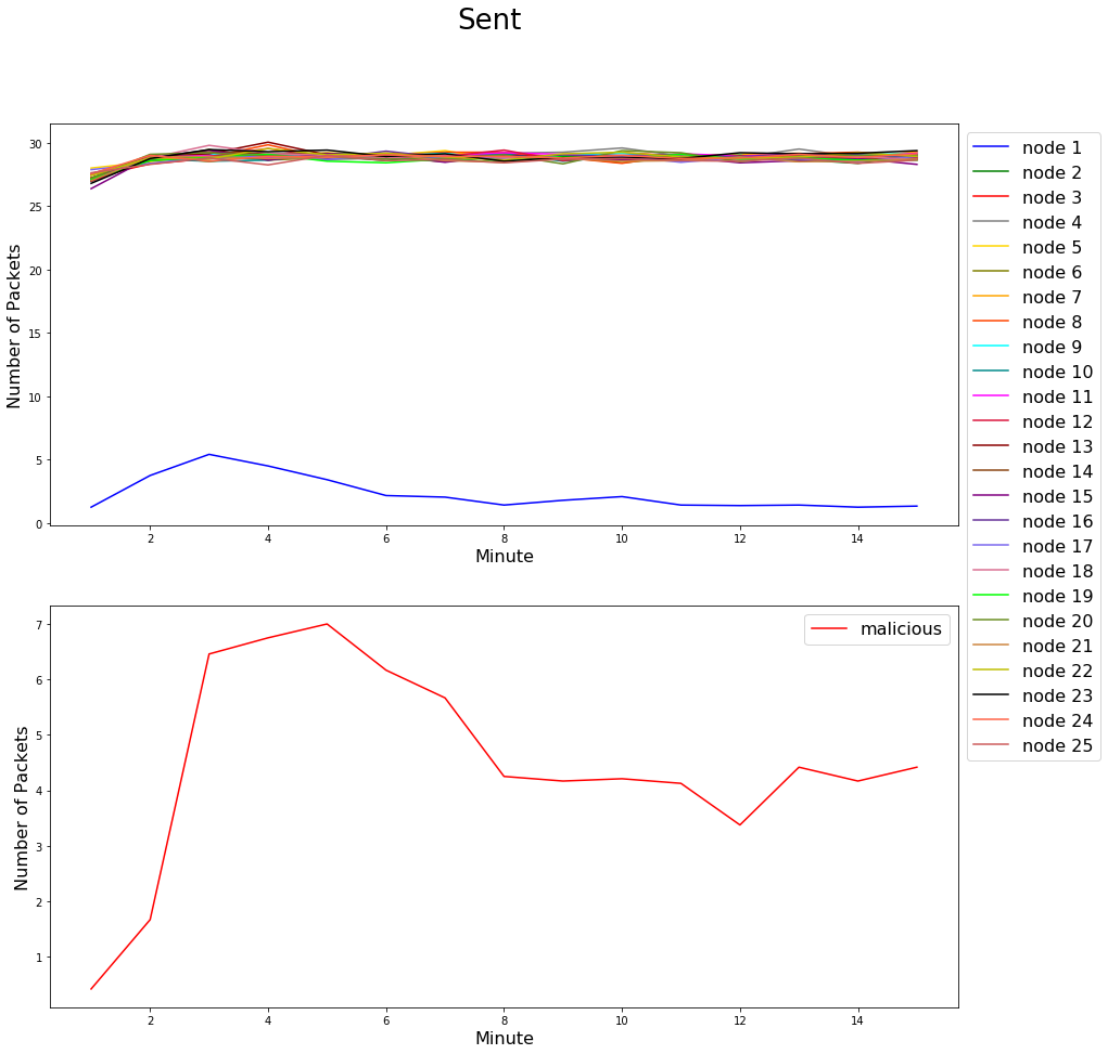
Σχήμα 5.28.b Γραφική Packets Forwarded για τοπολογία Sinkhole Random

5.3.4.2 Τοπολογία Middle

Όπως παρατηρούμε στα σχήματα 5.29 ο κακόβουλος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει, ενώ δημιουργεί και στέλνει πολύ λιγότερα πακέτα απ' ότι οι υπόλοιποι κόμβοι. Επιπλέον, παρατηρούμε ότι το throughput έχει μειωθεί στα 8.3 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 3.25 πακέτα το δευτερόλεπτο.



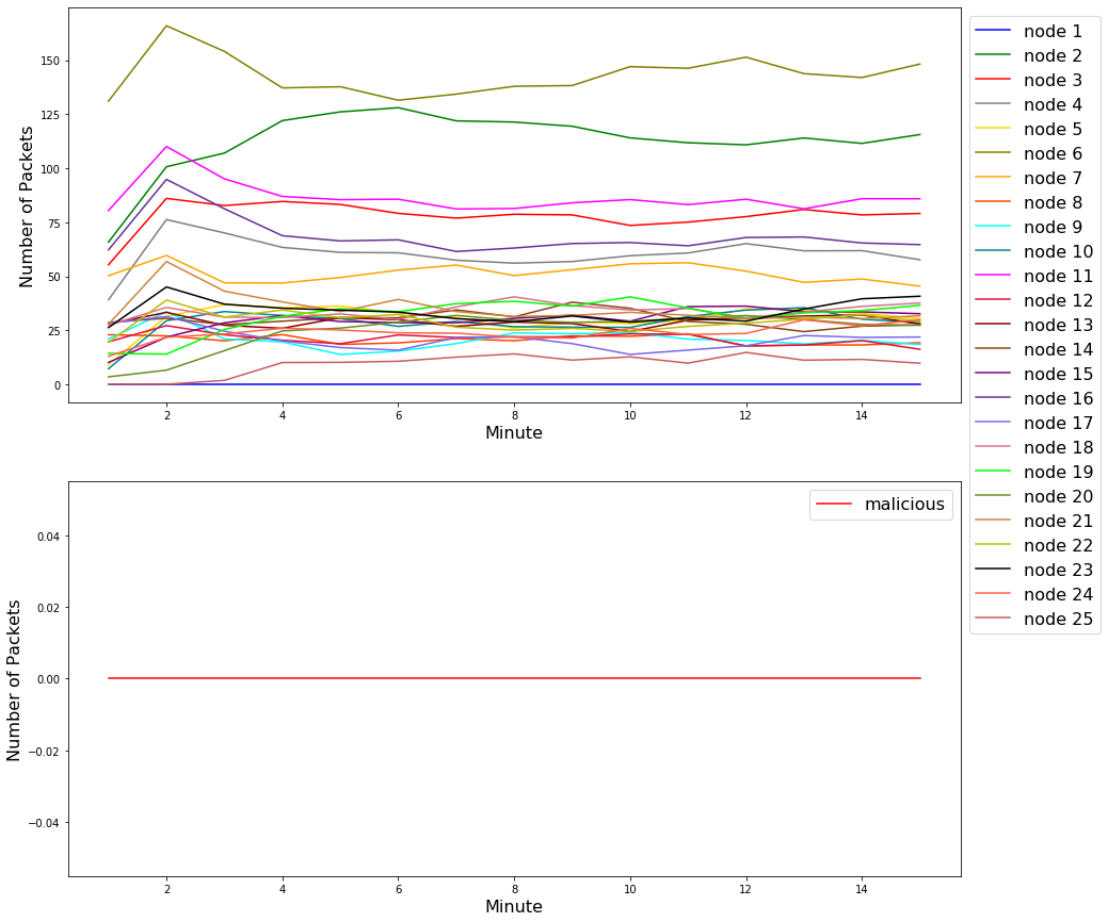
Σχήμα 5.29.a Γραφική Packets Forwarded για τοπολογία Sinkhole Middle



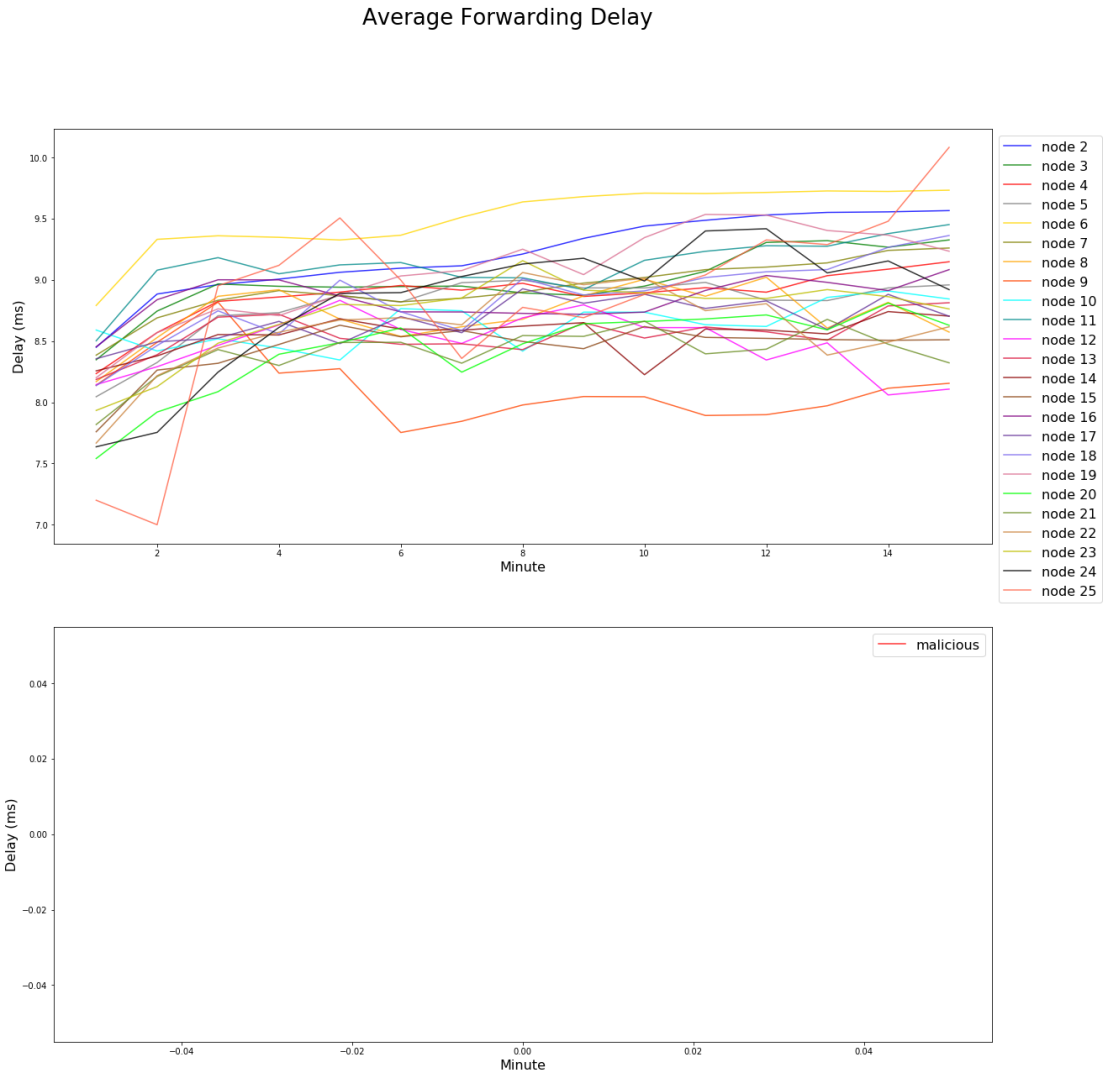
Σχήμα 5.29.b Γραφική Packets Sent για τοπολογία Sinkhole Middle

5.3.4.3 Τοπολογία Top

Όπως παρατηρούμε στα σχήματα 5.30 ο κακόβουλος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει και δεν προωθεί κανένα. Αυτό έχει ως αποτέλεσμα να παρουσιάζει μηδενική καθυστέρηση προώθησης. Ακόμη παρατηρούμε ότι δημιουργεί και στέλνει πολύ λιγότερα πακέτα σε σχέση με τους υπόλοιπους κόμβους. Επίσεις, το throughput έχει μειωθεί στα 5.1 πακέτα το δευτερόλεπτο, ενώ η απώλεια έχει αυξηθεί στα 8.25 πακέτα το δευτερόλεπτο.



Σχήμα 5.30.α Γραφική Packets Forwarded για τοπολογία Sinkhole Top



Σχήμα 5.30.b Γραφική Forwarding Delay για τοπολογία Sinkhole Top

Κεφάλαιο 6

Συμπεράσματα

6.1 Παρατηρήσεις

6.1.1 Black-hole

Ο ιός δημιουργεί προκαλεί κακόβουλη συμπεριφορά στον κόμβο, η οποία δεν επηρεάζει μόνο τον μολυσμένο κόμβο και τις λειτουργίες του, αλλά και όλο το υπόλοιπο δίκτυο. Έτσι παρατηρούμε, μεγάλη καθυστέρηση στην προώθηση πακέτων στον κακόβουλο κόμβο. Ο κακόβουλος κόμβος απορρίπτει μεγάλο μέρος των πακέτων που λαμβάνει και διαφημίζει πάντα μονοπάτι προς το sink ίσο με 0.50, ακόμη και όταν βρίσκεται πολύ μακριά από το sink. Αυτό προκαλεί μείωση της αποδοτικότητας του δικτύου (χαμηλότερο throughput) και μεγαλύτερη απώλεια πακέτων του δικτύου.

6.1.2 Flooding

Ο ιός προκαλεί συμπεριφορά στον κόμβο η οποία έχει ως αποτέλεσμα ο κόμβος να στέλνει μέχρι και τον επταπλάσιο όγκο μηνυμάτων DIO σε σχέση με τα μηνύματα που στέλνουν οι υπόλοιποι κόμβοι. Ακόμη φαίνεται να αναγκάζει τον κόμβο να δημιουργεί και να στέλνει τον διπλάσιο αριθμό πακέτων. Με αυτό τον τρόπο αυξάνεται η συμφόρηση στο δίκτυο, γεγονός που γίνεται πιο εμφανές παρατηρώντας την σταθερή αύξηση στην καθυστέρηση προώθησης σε όλους τους κόμβους. Έτσι ο ιός μειώνει την αποδοτικότητα (throughput) του δικτύου και αυξάνει την απώλεια πακέτων.

6.1.3 Selective Forwarding

Αυτός ο ιός επιβάλλει κακόβουλη συμπεριφορά στον κόμβο, η οποία έχει ως αποτέλεσμα ο κόμβος να απορρίπτει τουλάχιστο τα μισά πακέτα που λαμβάνει. Η κακόβουλη, αυτή, λειτουργία του κόμβου προκαλεί μεγάλη καθυστέρηση στην προώθηση πακέτων στον ίδιο τον κόμβο. Με αυτό τον τρόπο ο ιός επιτυγχάνει μείωση της αποδοτικότητας (throughput) του δικτύου και αύξηση στην απώλεια πακέτων.

6.1.4 Sinkhole

Ο ιός επιβάλλει συμπεριφορά στον κόμβο, η οποία μιμείται τη συμπεριφορά του κόμβου sink. Ως αποτέλεσμα παρατηρούμε ότι ο μολυσμένος κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει και διαφημίζει πάντα μονοπάτι προς το sink ίσο με 0.50, ακόμη και όταν

βρίσκεται πολύ μακριά από το sink. Ακόμη, δημιουργεί και στέλνει πολύ μικρό αριθμό πακέτων, όπως ο sink. Με αυτό τον τρόπο ο ιός επιτυγχάνει μείωση της αποδοτικότητας (throughput) του δικτύου και αύξηση στην απώλεια πακέτων. Επιπλέον παρατηρείται ότι ο sink δεν λαμβάνει ποτέ πακέτα από τον κόμβο που έχει μολυνθεί από αυτόν τον ιό.

6.2 Συμπεράσματα

Για τα πιο κάτω συμπεράσματα θεωρούμε ότι βρισκόμαστε σε δίκτυο όπου μπορούμε να εξάγουμε πληροφορίες από το δίκτυο για όλους τους κόμβους που να περιέχουν τα δεδομένα που χρησιμοποιούμε στο *Κεφάλαιο 3*. Ακόμη θεωρούμε ότι το δίκτυο δεν επηρεάζεται από άλλους ιούς και δεν έχει περισσότερο από ένα μολυσμένο κόμβο.

6.2.1 Black-hole

Μέσα από την ανάλυση των δεδομένων, γραφικών παραστάσεων και παρατηρήσεων που έχουμε εξάγει, μπορούμε να συμπεράνουμε ότι η ανίχνευση του ιού μπορεί να επιτευχθεί όταν ισχύουν όλες οι ακόλουθες καταστάσεις:

- Ο κόμβος αργεί να προωθήσει τα πακέτα που λαμβάνει, σε σχέση με τους υπόλοιπους κόμβους στο δίκτυο στο οποίο βρίσκεται.
- Ο κόμβος απορρίπτει μεγάλο μέρος (μέχρι και το 70%) των πακέτων που λαμβάνει.
- Ο κόμβος διαφημίζει συνεχώς μονοπάτι ETX path ίσο με 0.50, το οποίο δεν παίρνει άλλες τιμές ποτέ.

6.2.2 Flooding

Μέσα από την ανάλυση των δεδομένων, γραφικών παραστάσεων και παρατηρήσεων που έχουμε εξάγει, μπορούμε να συμπεράνουμε ότι η ανίχνευση του ιού μπορεί να επιτευχθεί, με μεγάλη πιθανότητα επιτυχίας, όταν ισχύουν όλες οι ακόλουθες καταστάσεις:

- Ο κόμβος στέλνει υπερβολικά μεγαλύτερο αριθμό μηνυμάτων DIO σε σχέση με τους υπόλοιπους κόμβους του δικτύου. Ο αριθμός αυτός μπορεί να είναι μέχρι και επταπλάσιος από τον αντίστοιχο αριθμό μηνυμάτων DIO των άλλων κόμβων.
- Ο κόμβος δημιουργεί και στέλνει τουλάχιστο το διπλάσιο αριθμό πακέτων σε σχέση με τους υπόλοιπους κόμβους του δικτύου.

6.2.3 Selective Forwarding

Μέσα από την ανάλυση των δεδομένων, γραφικών παραστάσεων και παρατηρήσεων που έχουμε εξάγει, μπορούμε να συμπεράνουμε ότι η ανίχνευση του ιού μπορεί να επιτευχθεί όταν ισχύουν όλες οι ακόλουθες καταστάσεις:

- Ο κόμβος αργεί να προωθήσει τα πακέτα που λαμβάνει, σε σχέση με τους υπόλοιπους κόμβους στο δίκτυο στο οποίο βρίσκεται.
- Ο κόμβος απορρίπτει τουλάχιστο τα μισά πακέτα που λαμβάνει.

Σημαντικό είναι να σημειώσουμε πως σε αντίθεση με τον ιό Black-hole, ο μολυσμένος κόμβος δεν διαφημίζει λανθασμένο ETX path.

6.2.4 Sinkhole

Μέσα από την ανάλυση των δεδομένων, γραφικών παραστάσεων και παρατηρήσεων που έχουμε εξάγει, μπορούμε να συμπεράνουμε ότι η ανίχνευση του ιού μπορεί να επιτευχθεί, με μεγάλη πιθανότητα επιτυχίας, όταν ισχύουν όλες οι ακόλουθες καταστάσεις:

- Ο κόμβος απορρίπτει όλα τα πακέτα που λαμβάνει.
- Ο κόμβος διαφημίζει συνεχώς μονοπάτι ETX path ίσο με 0.50, το οποίο δεν παίρνει άλλες τιμές ποτέ.
- Ο κόμβος δημιουργεί και στέλνει πολύ μικρό αριθμό πακέτων, σε σχέση με τους υπόλοιπους κόμβους.
- Ο κόμβος sink δεν λαμβάνει ποτέ πακέτα από αυτό τον κόμβο.

6.3 Μελλοντική Εργασία

Είναι επιθυμητό να αυξήσουμε την ακρίβεια των αποτελεσμάτων που παίρνουμε από τις προσομοιώσεις. Για αυτό θα ήταν καλύτερα να υλοποιηθεί λειτουργία στους κόμβους που να επιτρέπει από το εργαλείο *collect view* του προσομοιωτή cooja να εξάγει πληροφορίες από τους κόμβους κατά την προσομοίωση. Με τη χρήση του εργαλείου αυτού μπορούμε να έχουμε πιο ακριβές αποτελέσματα σε σχέση με τη χρονική στιγμή που συμβαίνουν οι ενέργειες των κόμβων. Ακόμη μπορούμε να μελετήσουμε με μεγαλύτερη ακρίβεια τις ενέργειες των κόμβων (π.χ. τον λόγο που απορρίφθηκε το πακέτο: σύγκρουση, έλλειψη χώρου στο queue του κόμβου κτλ.) και να εξετάσουμε σε πραγματικό χρόνο τη δημιουργία και διαμόρφωση του γράφου του δικτύου. Με αυτό τον τρόπο μπορούμε να εξετάσουμε και την κατανάλωση ενέργειας των κόμβων, γεγονός που

θα μπορούσε να μας δώσει ακόμη μία επιπλέον διάσταση καταμέτρησης και ανίχνευσης του ιού.

Έπειτα θα θέλαμε να μπορεί να γίνει η ανίχνευση των ιών σε πραγματικά δίκτυα, καθώς αυτά είναι σε λειτουργία. Για να επιτευχθεί αυτό, είναι απαραίτητο να αναβαθμιστούν τα script έτσι ώστε να μπορούν να διαβάζουν την πληροφορία μέσω ενός ζωντανού data stream και να την αναλύουν σε πραγματικό χρόνο. Για να υλοποιηθεί η πιο πάνω αναβάθμιση θα ήταν βοηθητικό πρώτα να βελτιστοποιηθούν τα script ώστε να μπορούν να λειτουργούν με τα αποτελέσματα από το *collect view*.

Βιβλιογραφία

[1] Christina Solomou, Elena Theophanous and Georgia Kalli, “Attacks to RPL Protocol”, University of Cyprus, pp 1-9, 2018.

[2] Winter et al., “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks”, The IETF Trust, RFC 6550, pp. 1-28, 2012.

[3] Winter et al., “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks”, The IETF Trust, RFC 6550, pp. 38-45, 2012.

Παράρτημα Α

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας υλοποίησης των script που αναφέρονται στο Κεφάλαιο 3.

A.1 totals.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""
import sys
numofnodes = 25



def TOTprocess(dt):
   nodes = []
   i = 0
   for i in range(0, numofnodes, 1):
       ni = []
       nodes.append(ni)
   for line in dt:
       line = line.strip('\n')
       z = int(line[13:15].strip(' '))
       nodes[z-1].append(line)
   i = 0
   totgen = 0
   totrec = 0
   for i in range(0, numofnodes, 1):
       x = 0
       node_info = ["sent", "forwarded", "received", "dropped",
                    "dio", "etx path"]
       rcv = 0
       fwd = 0
       for x in nodes[i]:
           if (x.find("Sent:") >= 0):
               node_info[0] = x
           if (x.find("Forwarded:") >= 0):
```

```
                node_info[1] = x
            if (x.find("Received:") >= 0):
                node_info[2] = x
            if (x.find("received") >= 0 and
                  x.find("to aaaa::ff:fe00:1") >= 0):
                rcv += 1
            if (x.find("RPL Option Error") >= 0):
                rcv -= 1
            if (x.find("dropping") >= 0):
                node_info[3] = x
            if (x.find("DIO") >= 0 and x.find("(sent)") >= 0):
                node_info[4] = x
            if (x.find("root") >= 0):
                node_info[5] = x
        print("ID:", i+1, end = '\t')
        if (node_info[2].find('\t') < 0):
            print('Received:', rcv, end = ', ')
        else:
            print(node_info[2].split('\t')[2], end = ', ')
        if (node_info[1].find('\t') < 0):
            print("Forwarded: 0", end = ', ')
        else:
            print(node_info[1].split('\t')[2], end = ', ')
            fwd = int(node_info[1].split('\t')[2].split()[1])
        if (node_info[0].find('\t') < 0):
            print("Sent: 0", end = ', ')
        else:
            sent = int(node_info[0].split('\t')[2].split()[1]) - fwd
            print("Sent:", sent, end = ', ')
            totgen += sent
        if (node_info[3].find('\t') < 0):
            print("Dropped: 0.")
        else:
            print("Dropped:", node_info[3].split('!')[1], end = '.\n')
        print("    ", end = '\t')
        if (node_info[4].find('\t') < 0):
            print('DIO packets: 0', end = ', ')
        else:
            dio = int(node_info[4].split(') ')[1])
            print("DIO packets:", dio, end = ', ')
        if (node_info[5].find('\t') < 0):
            print('No ETX path to root.')
        else:
            path = node_info[5].split('is ')[1]
            print('ETX path to root:', path, end = '.\n')
```

```
        if (i == 0):
            totrec = rcv
    print("*************************************************************", end
= '')
    print("*************")
    secs = mins * 60
    print("    \tThroughput: %.2f packets per second." % (totrec / secs))
    print("    \tPacket loss: %d packets." % (totgen - totrec))
    print("*************************************************************", end
= '')
    print("*************")


def printman():
    print("*************************************************************", end
= '')
    print("***********************************")
    print("To run script use the following:")
    print("*************************************************************", end
= '')
    print("***********************************")
    print("$> python3 totals.py <logfile> <mins>")
    print("-> where <logfile> put the name of the log file,")
    print("-> where <mins> put the number of maximum miniutes of", end =
' ')
    print("the simulation to take data from")
    print("*************************************************************", end
= '')
    print("***********************************")
    sys.exit(0)


def err():
    print("Wrong Arguments Given")
    print("If you need help please run script with the argumens -h or --
help")
    sys.exit(1)


l = len(sys.argv)
if (l <= 1): err()
if (l == 2):
    arg1 = sys.argv[1]
    if (arg1.find("-h") or arg1.find("--help")): printman()
    else: err()
```

```
if (l != 3): err()
logfile = sys.argv[1]
mins = 0
try:
   mins = int(sys.argv[2])
except ValueError as err:
   print("Give integer number of minutes (i.e. 1, 2, etc.)")
   err()
fp = None
try:
   fp = open(sys.argv[1], "r")
except FileNotFoundError as err:
   err()
data = fp.readlines()
fp.close()
uniformed = []
for line in data:
   time = line[0:9]
   if ((int(time[0:2]) < mins) or (int(time[0:2]) == mins
        and int(time[3:5]) == 0 and int(time[6:9]) == 0)):
       uniformed.append(line)
print("+-------------------------------------------------------------------
--+")
print("+----------                  In total:                  ---------
-+")
print("+-------------------------------------------------------------------
--+")
if (not uniformed): print("No logs found within the time given.")
else: TOTprocess(uniformed)
```

A.2 interval_totals.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""
import sys
numofnodes = 25



prevSnt = []
prevFwd = []
```

```
prevRcv = []
prevDpd = []
prevDIO = []
packetlosses = []
throughputs = []



for i in range(0, numofnodes, 1):
       prevSnt.append(0)
       prevFwd.append(0)
       prevRcv.append(0)
       prevDpd.append(0)
       prevDIO.append(0)



def process(dt):
   global prevSnt
   global prevFwd
   global prevRcv
   global prevDpd
   global prevDIO
   global packetlosses
   global throughputs
   nodes = []
   i = 0
   for i in range(0, numofnodes, 1):
       ni = []
       nodes.append(ni)
   for line in dt:
       line = line.strip('\n')
       z = int(line[13:15].strip(' '))
       nodes[z-1].append(line)
   i = 0
   totgen = 0
   totrec = 0
   for i in range(0, numofnodes, 1):
       x = 0
       node_info = ["sent", "forwarded", "received", "dropped", "dio",
"path"]
       rcv = 0
       fwd = 0
       for x in nodes[i]:
           if (x.find("Sent:") >= 0):
               node_info[0] = x
           if (x.find("Forwarded:") >= 0):
```

```
                node_info[1] = x
            if (x.find("Received:") >= 0):
                node_info[2] = x
            if (x.find("received") >= 0 and
                  x.find("to aaaa::ff:fe00:1") >= 0):
                rcv += 1
            if (x.find("dropping") >= 0):
                node_info[3] = x
            if (x.find("DIO") >= 0 and x.find("(sent)") >= 0):
                node_info[4] = x
            if (x.find("root") >= 0):
                node_info[5] = x
        print("ID:", i+1, end = '\t')
        if (node_info[2].find(' ') < 0):
            print('Received:', rcv, end = ', ')
            prevRcv[i] += rcv
        else:
            rec = int(node_info[2].split('\t')[2].split(' ')[1]) -
prevRcv[i]
            print('Received:', rec, end = ', ')
            prevRcv[i] += rec
        if (node_info[1].find('\t') < 0):
            print("Forwarded: 0", end = ', ')
        else:
            fwd = int(node_info[1].split('\t')[2].split(' ')[1]) -
prevFwd[i]
            print('Forwarded:', fwd, end = ', ')
            prevFwd[i] += fwd
        if (node_info[0].find('\t') < 0):
            print("Sent: 0", end = ', ')
        else:
            sent = int(node_info[0].split('\t')[2].split(' ')[1]) -
prevFwd[i]
            sent -= prevSnt[i]
            print("Sent:", sent, end = ', ')
            prevSnt[i] += sent
            totgen += sent
        if (node_info[3].find('\t') < 0):
            print("Dropped: 0", end = '.\n')
        else:
            dpd = int(node_info[3].split('!')[1]) - prevDpd[i]
            print("Dropped:",dpd, end = '.\n')
            prevDpd[i] += dpd
        print("    ", end = '\t')
        if (node_info[4].find('\t') < 0):
```

```
            print('DIO packets: 0', end = ', ')
        else:
            dio = int(node_info[4].split(') ')[1])
            dio -= prevDIO[i]
            print("DIO packets:", dio, end = ', ')
            prevDIO[i] += dio
        if (node_info[5].find('\t') < 0):
            print('no ETX path to root.')
        else:
            path = node_info[5].split('is ')[1]
            print('ETX path to root:', path, end = '.\n')
        if (i == 0):
            totrec += rcv
    print("************************************************************", end
= '')
    print("**********")
    secs = interval * 60
    throughput = (totrec / secs)
    throughputs.append(throughput)
    packetloss = totgen - totrec
    packetlosses.append(packetloss)
    print("    \tThroughput: %.2f packets per second." % throughput)
    print("    \tPacket loss: %d packets." % packetloss)
    print("************************************************************", end
= '')
    print("**********")


def splitData(d, t):
    sp = []
    for line in d:
        time = line[0:9]
        if ((int(time[0:2]) < t) or (int(time[0:2]) == t and
             int(time[3:5]) == 0 and int(time[6:9]) == 0)):
            sp.append(line)
    return sp


def printman():
    print("************************************************************", end
= '')
    print("**********************************")
    print("To run script use the following:")
    print("************************************************************", end
= '')
```

```
    print("***********************************")
    print("$> python3 intervals_totals.py <logfile> <mins> <intervals>")
    print("-> where <logfile> put the name of the log file,")
    print("-> where <mins> put the number of maximum miniutes of", end =
' ')
    print("the simulation to take data from,")
    print("-> where <intervals> put the length of each interval in
minutes")
    print("*******************************************************", end
= '')
    print("***********************************")
    sys.exit(0)


def err():
    print("Wrong Arguments Given")
    print("If you need help please run script with the argumens -h or --
help")
    sys.exit(1)


l = len(sys.argv)
if (l <= 1): err()
if (l == 2):
    arg1 = sys.argv[1]
    if (isinstance(arg1, str) and (arg1.find("-h") or arg1.find("--
help"))):
        printman()
    else: err()
if (l != 4): err()
logfile = sys.argv[1]
mins = 0
try:
    mins = int(sys.argv[2])
except ValueError as er:
    print("Give integer number of minutes (i.e. 1, 2, etc.)")
    err()
interval = 0
try:
    interval = int(sys.argv[3])
except ValueError as er:
    print("Give integer number for interval (i.e. 1, 2, etc.)")
    err()
fp = None
try:
```

```
    fp = open(sys.argv[1], "r")
except FileNotFoundError as er:
    err()
data = fp.readlines()
fp.close()
i = 1
print("**************************************************************
*")
print("Running for logs of upto %d minutes in intervals of %d minutes."
      % (mins, interval))
print("**************************************************************
*")
prevlen = 0
for i in range(interval, mins+1, interval):
    if (prevlen >= len(data)): break
    if (i/interval >= 10):
        print("+-----------------------------------------------", end =
'')
        print("----------------+")
        print("+---------                Inetrval  %d          "
              % (i/interval), end = '')
        print("       ---------+")
        print("+-----------------------------------------------", end =
'')
        print("----------------+")
    else:
        print("+-----------------------------------------------", end =
'')
        print("----------------+")
        print("+---------                 Inetrval %d          "
              % (i/interval), end = '')
        print("        ---------+")
        print("+-----------------------------------------------", end =
'')
        print("----------------+")
    sp = splitData(data[prevlen:], i)
    prevlen += len(sp)
    process(sp)


print("******************************************************", end =
'')
print("**********")
minThrough, maxThrough = min(throughputs), max(throughputs)
print(" \tMinimum throughput: %.2f packets per second."
```

```
      % minThrough, end=' ')
print("(interval %d)" % throughputs.index(minThrough))
print(" \tMaximum throughput: %.2f packets per second."
      % maxThrough, end = ' ')
print("(interval %d)" % throughputs.index(maxThrough))
minPackloss, maxPackloss = min(packetlosses), max(packetlosses)
print(" \tMinimum packet loss: %d packets." % minPackloss, end = ' ')
print("(interval %d)" % packetlosses.index(minPackloss))
print(" \tMaximum packet loss: %d packets." % maxPackloss, end = ' ')
print("(interval %d)" % packetlosses.index(maxPackloss))
print("********************************************************", end =
'')
print("**********")
```

A.3 end_to_end_delay.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import sys



numofnodes = 25



sinkrec = []
psent = []
for i in range(0, numofnodes, 1):
   li1 = []
   sinkrec.append(li1)
   li2 = []
   psent.append(li2)



def process(dt):
   global sinkrec; global psent
   for ln in dt:
       if (ln.find("DATA send") >= 0):
           pid = ln.split('\'')[-2]
           z = int(ln[13:15].strip(' '))
```

```
            psent[z-1].append("%s\t%s" % (ln[0:9], pid))
        if (ln.find("DATA recv") >= 0):
            pid = ln.split('\'')[1].split(' from ')[0]
            z = int(ln.split(' ')[-1].strip('\n'))
            sinkrec[z-1].append("%s\t%s" % (ln[0:9], pid))


def delays():
    for i in range(1, numofnodes, 1):
        print("node %d" % (i+1))
        for p in sinkrec[i]:
            pid = p.split('\t')[-1]
            found = False
            s = None
            for x in psent[i]:
                if x.find(pid) >= 0:
                    found = True
                    s = x
                    break
            if found:
                ts = int(s[0:2]) * 60000
                ts += int(s[3:5]) * 1000
                ts += int(s[6:9])
                tr = int(p[0:2]) * 60000
                tr += int(p[3:5]) * 1000
                tr += int(p[6:9])
                tot = tr - ts
                print("%s, %d" % (s[0:9], tot))



def splitData(d, t):
    sp = []
    for line in d:
        time = line[0:9]
        if ((int(time[0:2]) < t) or (int(time[0:2]) == t and
             int(time[3:5]) == 0 and int(time[6:9]) == 0)):
            sp.append(line)
    return sp


def printman():
    print("*****************************************************", end
= '')
```

```
    print("************************************")
    print("To run script use the following:")
    print("*********************************************************", end
= '')
    print("************************************")
    print("$> python3 end2end.py <logfile> <mins>")
    print("-> where <logfile> put the name of the log file,")
    print("-> where <mins> put the number of maximum miniutes of", end =
' ')
    print("the simulation to take data from")
    print("*********************************************************", end
= '')
    print("************************************")
    sys.exit(0)


def err():
    print("Wrong Arguments Given")
    print("If you need help please run script with the argumens -h or --
help")
    sys.exit(1)


l = len(sys.argv)
if (l <= 1): err()
argv1 = sys.argv[1]
if (l == 2 and (argv1.find("-h") >= 0 or argv1.find("--help") >= 0)):
    printman()
if l != 3: err()


mins = 0
try: mins = int(sys.argv[2])
except: print("Give number of mins (i.e. 1, 2, 3)"); err()
fp = None
try: fp = open(argv1, "r")
except:
    print("File not Found")
    err()
data = fp.readlines()
fp.close()
uniformed = splitData(data, mins)
process(uniformed)
delays()
```

A.4 forwarding_delay.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""


import sys


numofnodes = 25
rcounts = [0]*numofnodes


def delay(data):
    global rcounts
    rcv = []
    fwd = []
    for i in range(0, numofnodes, 1):
        li1 = []
        li2 = []
        rcv.append(li1)
        fwd.append(li2)
    for ln in data:
        if ln.find("Forwarded:") >= 0:
            ln = ln.strip('\n')
            z = int(ln[13:15].strip(' '))
            fwd[z-1].append(ln)
            continue
        if ln.find("received ") >= 0 and ln.find("to aaaa::ff:fe00:1") >=
0:
            z = int(ln[13:15].strip(' '))
            rcounts[z-1] += 1
            rcv[z-1].append("%s\tReceived: %d" % (ln[0:9], rcounts[z-1]))
            continue
        if ln.find("RPL Option Error") >= 0:
            z = int(ln[13:15].strip(' '))
            if not rcv[z-1]: continue
            del rcv[z-1][-1]
            rcounts[z-1] -= 1
```

```
    for x in range(0, numofnodes, 1):
        if x == 0:
                print("ID:1\tThis is the sink.")
                continue
        if (not rcv[x]):
            print("ID:%d\tNo packets were Received." % (x + 1))
            continue
        if (not fwd[x]):
            print("ID:%d\tNo packets were Forwarded." % (x + 1))
            continue
        i = 0
        delay = 0
        minDelay = 60000*interval
        maxDelay = 0
        for p in fwd[x]:
            rcp = rcv[x][i]
            while (int(p.split(': ')[1]) < int(rcp.split(': ')[1])):
                del fwd[x][i]
                if len(fwd[x]) <= 0: break
                p = fwd[x][i]
            while (int(p.split(': ')[1]) > int(rcp.split(': ')[1])):
                del rcv[x][i]
                rcp = rcv[x][i]
            if (int(p.split(': ')[1]) == int(rcp.split(': ')[1])):
                f = p[0:9]
                r = rcp[0:9]
                f = (int(f[0:2]) * 60000) + (int(f[3:5]) * 1000) +
int(f[6:9])
                r = (int(r[0:2]) * 60000) + (int(r[3:5]) * 1000) +
int(r[6:9])
                d = (f - r)
                delay += d
                i += 1
                if (d < minDelay): minDelay = d
                if (d > maxDelay): maxDelay = d
        if (delay == 0):
            print("ID:%d\tDelay bigger than the interval (%dmins)."
                  % ((x + 1), interval))
        else:
            print("ID:%d\tAverage Delay: %.2f (min: %dms, max: %dms)."
              % (x+1, delay/i, minDelay, maxDelay))



def splitData(d, t):
    sp = []
```

```
    for line in d:
        time = line[0:9]
        if ((int(time[0:2]) < t) or (int(time[0:2]) == t and
             int(time[3:5]) == 0 and int(time[6:9]) == 0)):
            sp.append(line)
    return sp


def printman():
    print("*******************************************************", end
= '')
    print("*********************************")
    print("To run script use the following:")
    print("*******************************************************", end
= '')
    print("*********************************")
    print("$> python3 forwarding_delay.py <logfile> <mins> <interval>")
    print("-> where <logfile> put the name of the log file,")
    print("-> where <mins> put the number of maximum miniutes of", end =
'')
    print("the simulation to take data from,")
    print("-> where <intervals> put the length of each interval in
minutes")
    print("*******************************************************", end
= '')
    print("*********************************")
    sys.exit(0)


def err():
    print("Wrong Arguments Given")
    print("If you need help please run script with the argumens -h or --
help")
    sys.exit(1)


l = len(sys.argv)
if (l <= 1): err()
argv1 = sys.argv[1]
if (l == 2):
    if (argv1.find("-h") >= 0 or argv1.find("--help") >= 0): printman()
    err()
if (l != 4): err()
logfile = sys.argv[1]
mins = 0
```

```
try:
   mins = int(sys.argv[2])
except:
   print("Give integer number of minutes (i.e. 1, 2, etc.)")
   err()
interval = 0
try:
   interval = int(sys.argv[3])
except:
   print("Give integer number for interval (i.e. 1, 2, etc.)")
   err()
fp = None
try: fp = open(argv1, "r")
except:
   print("File not Found")
   err()
data = fp.readlines()
fp.close()
prevlen = 0
for i in range(interval, mins+1, interval):
   if (prevlen >= len(data)): break
   if (i/interval >= 10):
       print("+-----------------------------------------------", end =
'')
       print("-----------------+")
       print("+---------                Inetrval  %d          "
             % (i / interval), end = '')
       print("       ---------+")
       print("+-----------------------------------------------", end =
'')
       print("-----------------+")
   else:
       print("+-----------------------------------------------", end =
'')
       print("-----------------+")
       print("+---------                 Inetrval %d          "
             % (i / interval), end = '')
       print("        ---------+")
       print("+-----------------------------------------------", end =
'')
       print("-----------------+")
   sp = splitData(data[prevlen:], i)
   prevlen += len(sp)
   delay(sp)
```

A.5 setuptime.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import sys



numofnodes = 25
times = [-1]*numofnodes
ip = []
for i in range(0, numofnodes, 1): ip.append(None)



def findIP(dt):
    global ip
    for line in dt:
        if (line.find("addresses:") >= 0):
            line = line.strip('\n')
            z = int(line[13:15].strip(' '))
            ip[z-1] = line.split("addresses: ")[1]
    noIP = False
    for x in ip:
        if x == None: noIP = True
    if noIP:
        print("Wrong logfile given")
        err()
    #transform IPv6 address to IPv6 link-local address of each node
    for i in range(0, numofnodes, 1):
        ip[i] = ip[i].replace("aaaa", "fe80")



def find_start(dt):
    for i in range(1, numofnodes, 1):
        s = "from %s" % ip[i]
        for ln in dt:
            if (ln.find(s) >= 0 and ln.find("ff02") < 0):
                times[i] = (int(ln[0:2]) * 60000) + (int(ln[3:5]) * 1000)
                + int(ln[6:9])
```

```
                break
    print("Tree set up time is %.2fs." % ((max(times) - offset) / 1000))


def after(x):
    y = 0
    for ln in data:
        t = int(ln[0:2]) * 60000
        t += int(ln[3:5])* 1000
        t += int(ln[6:9])
        if (t < x): y += 1
        else: return y


def printman():
    print("************************************************************", end
= '')
    print("***********************************")
    print("To run script use the following:")
    print("************************************************************", end
= '')
    print("***********************************")
    print("$> python3 setup_time.py <logfile> <offset>")
    print("-> where <logfile> put the name of the log file,")
    print("-> where <offset> put the number of seconds after of", end =
'')
    print("which the script will start searching\n   for the tree set up
time")
    print("If no value is given for the offset it will be set to", end =
' ')
    print("zero by default")
    print("************************************************************", end
= '')
    print("***********************************")
    sys.exit(0)


def err():
    print("Wrong Arguments Given")
    print("If you need help please run script with the arguments -h or --
help")
    sys.exit(1)
```

```
offset = 0
l = len(sys.argv)
if (l <= 1): err()
argv1 = sys.argv[1]
fp = None
if (l > 3): err()
if (l == 2):
   if (argv1.find("-h") >= 0 or argv1.find("--help") >= 0): printman()
if (l == 3):
   try: offset = int(sys.argv[2])
   except:
       print("Give number of offset secconds (i.e. 1, 60, 90, 200)")
       err()
try: fp = open(argv1, "r")
except:
   print("File not Found")
   err()
data = fp.readlines()
fp.close()
findIP(data)
t = after(offset * 1000)
find_start(data[t:])
```

A.6 init_tree.py

```
# -*- coding: utf-8 -*-
"""
Script needs the following to run:
   python v2.7.x
   treelib to be installed.
   /* treelib documentation can be found here:
    * https://treelib.readthedocs.io/en/latest/
    **/
@author: Maria Efthymiou
"""


from treelib import Tree
import sys


numofnodes = 25
```

```
def printman():

print("**********************************************************************
*************************")
   print("To run script use the following:")

print("**********************************************************************
*************************")
   print("$> python init_tree.py <logfile>")
   print("-> where <logfile> put the name of the log file")

print("**********************************************************************
*************************")
   sys.exit(0)



def err():
   print("Wrong Arguments Given")
   print("If you need help please run script with the argumens -h or --
help")
   sys.exit(1)



ip = []
parent = []
for i in range(0, numofnodes, 1):
   ip.append(None)
   parent.append(-1)



def findIP(dt):
   global ip
   for line in dt:
       if (line.find("addresses:") >= 0):
           line = line.strip('\n')
           z = int(line[13:15].strip(' '))
           ip[z-1] = line.split("addresses: ")[1]
   noIP = False
   for x in ip:
       if x == None: noIP = True
   if noIP:
       print("Wrong logfile given")
       err()
```

```
    #transform IPv6 address to IPv6 link-local address of each node
    for i in range(0, numofnodes, 1):
        ip[i] = ip[i].replace("aaaa", "fe80")



if (len(sys.argv) != 2): err()
argv1 = sys.argv[1]
if (argv1.find("-h") >= 0 or argv1.find("--help") >= 0): printman()
try: fp = open(sys.argv[1], "r")
except FileNotFoundError as err:
    print("File not Found")
    err()
data = fp.readlines()
fp.close()
findIP(data)
for i in range(0, numofnodes, 1):
    for ln in data:
        str = "from %s" % ip[i]
        if ln.find(str) >= 0 and ln.find("ff02") < 0:
            z = int(ln[13:15].strip(' '))
            if parent[i] < 0:
                parent[i] = z
                break
tree = Tree()
tree.create_node("node 1 (Sink)", 0)
intree = [0]
for x in intree:
    for i in range(1, numofnodes, 1):
        if parent[i] == x+1:
            node = "node %d" % (i + 1)
            tree.create_node(node, i, parent=x)
            intree.append(i)
    if (len(intree) >= 25): break
print("************************************************")
print("****              Initial Tree:              ****")
print("************************************************")
tree.show(line_type="ascii")
print("************************************************")
print("*   Tree Depth = %d                            *" % tree.depth())
if len(intree) != numofnodes:
    print("*   Network graph not connected                *")
else: print("*   Network graph is connected                 *")
print("************************************************")
```

A.7 trees.py

```
# -*- coding: utf-8 -*-
"""
Script needs the following to run:
   python v2.7 or higher.
   treelib to be installed.
   /* treelib documentation can be found here:
    * https://treelib.readthedocs.io/en/latest/
    **/
@author: Maria Efthymiou
"""
from treelib import Tree
import sys



numofnodes = 25



tree = Tree()
tree.create_node("node 1 (Sink)", 0)
firstInterval = True
ip = []
for i in range(0, numofnodes, 1): ip.append(None)



def findIP(dt):
   global ip
   for line in dt:
       if (line.find("addresses:") >= 0):
           line = line.strip('\n')
           z = int(line[13:15].strip(' '))
           ip[z-1] = line.split("addresses: ")[1]
   noIP = False
   for x in ip:
       if x == None: noIP = True
   if noIP:
       print("Wrong logfile given")
       err()
   #transform IPv6 address to IPv6 link-local address of each node
   for i in range(0, numofnodes, 1):
       ip[i] = ip[i].replace("aaaa", "fe80")
```

```
def splitData(d, t):
    sp = []
    for line in d:
        time = line[0:9]
        if (((int(time[0:2]) * 60000) + (int(time[3:5]) * 1000) +
int(time[6:9])) <= t*1000):
            sp.append(line)
    return sp



def prntTree(dt):
    global firstInterval
    parent = [-1]*numofnodes
    for i in range(0, numofnodes, 1):
        for line in dt:
            str = "from %s" % ip[i]
            if (line.find(str) >= 0 and line.find("ff02") < 0):
                z = int(line[13:15].strip(' '))
                if parent[i] < 0:
                    parent[i] = z - 1
                    break
    if firstInterval:
        intree = [0]
        for x in intree:
            for i in range(1, numofnodes, 1):
                if parent[i] == x:
                    node = "node %d" % (i + 1)
                    tree.create_node(node, i, parent=x)
                    intree.append(i)
                    firstInterval = False
            if (len(intree) >= 25):
                firstInterval = False
                break
    else:
        for i in range(1, numofnodes, 1):
            if parent[i] >= 0:
                try: tree.move_node(i, parent[i])
                except: continue

    tree.show(line_type="ascii")
    print("Tree Depth = %d" % tree.depth())
    if tree.size() != numofnodes:
        print("Network graph not connected")
```

```
    else:
        print("Network graph is connected")


def printman():

print("**********************************************************************
************************")
    print("To run script use the following:")

print("**********************************************************************
************************")
    print("$> python trees.py <logfile> <mins> <interval>")
    print("-> where <logfile> put the name of the log file,")
    print("-> where <mins> put the number of maximum miniutes of the
simulation to take data from,")
    print("-> where <intervals> put the length of each interval in
seconds")

print("**********************************************************************
************************")
    sys.exit(0)


def err():
    print("Wrong Arguments Given")
    print("If you need help please run script with the arguments -h or --
help")
    sys.exit(1)
l = len(sys.argv)
if (l <= 1): err()
argv1 = sys.argv[1]
if (l == 2):
    if (argv1.find("-h") >= 0 or argv1.find("--help") >= 0): printman()
    err()
if (l != 4): err()
logfile = sys.argv[1]
mins = 0
try:
    mins = int(sys.argv[2])
except:
    print("Give integer number of minutes (i.e. 1, 2, etc.)")
    err()
interval = 0
try:
```

```
    interval = int(sys.argv[3])
except:
    print("Give integer number for interval (i.e. 1, 2, etc.)")
    err()
fp = None
try: fp = open(argv1, "r")
except:
    print("File not Found")
    err()
data = fp.readlines()
fp.close()
findIP(data)
prevlen = 0
simlen = mins * 60
i = interval
if (simlen % interval != 0): repeat = True
else: repeat = False
while i <= simlen:
    if (prevlen >= len(data)): break
    ii = i / interval
    if (ii >= 10):
        print("+-------------------------------------------------------------
------+")
        print("+---------                Inetrval  %d                  ---
------+" % ii)
        print("+-------------------------------------------------------------
------+")
    else:
        print("+-------------------------------------------------------------
------+")
        print("+---------                 Inetrval %d                  --
-------+" % ii)
        print("+-------------------------------------------------------------
------+")
    sp = splitData(data[prevlen:], i)
    prevlen += len(sp)
    prntTree(sp)
    dif = simlen - i
    if (dif < interval and repeat): i += dif; repeat = False
    else: i += interval
```

Παράρτημα Β

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας υλοποίησης των script που αναφέρονται στο Κεφάλαιο 4.

Β.1 csv_convert.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""


import sys


fp = open(sys.argv[1], "r")
data = fp.readlines()
fp.close()


for ln in data:
   if ln[0] == 'n':
       ln = ', ' + ln
       print(ln, end='')
   else:
       sp = ln.split(',')
       t = int(sp[0][0:2]) * 60000
       t += int(sp[0][3:5]) * 1000
       t += int(sp[0][6:9])
       print("%d,%s" % (t, sp[1]), end = '')
```

B.2 file_convert.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import os
import sys



path = sys.argv[1]
filename, outlos, outtot = 'interval_totals', "thrloss", "tots"
files, outploss, outt = [], [], []
for entry in os.listdir(path):
    if os.path.isdir(os.path.join(path, entry)):
        filepath = "%s%s/%s" % (path, entry, filename)
        files.append(filepath)
        filepath = "%s%s/%s" % (path, entry, outlos)
        outploss.append(filepath)
        filepath = "%s%s/%s" % (path, entry, outtot)
        outt.append(filepath)
#print(outploss, outt)



for f, fl, ft in zip(files, outploss, outt):
    fp = open(f, "r")
    data = fp.readlines()
    fp.close()
    flos, ftot = open(fl, "w"), open(ft, "w")
    nodes = []
    intervals = []
    nid = 0
    through, packloss = [], []
    for i in range(0, 25):
        nodes.append([])
        if i < 15: intervals.append(i+1)
    del data[-6:]
    for ln in data:
        if ln[0] == '+' or ln[0] == '*': continue
        if ln[0] == 'I':
```

```
            nid = int(ln[4:6].strip(' '))
            nodes[nid-1].append(ln.strip('\n'))
        d = ln.find("DIO")
        if ln[0] == ' ' and d >= 0:
            nodes[nid-1][-1] += " %s" % ln[d:]
        d = ln.find('Through')
        if d >= 0: through.append(float(ln[d:].split(' ')[1]))
        d = ln.find("loss")
        if d >= 0: packloss.append(int(ln[d:].split(' ')[1]))
    for t in through: flos.write("%s, " % t)
    flos.write("\n")
    for p in packloss: flos.write("%s, " % p)
    flos.write("\n")
    flos.close()
    for n in nodes:
        for ln in n:
            ftot.write("%s" % ln)
    ftot.close()
```

B.3 scatterplot.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import sys
import matplotlib.pyplot as plt



fp = open(sys.argv[1], "r")
data = fp.readlines()
fp.close()



x, y = [], []
for i in range(0, 24):
    li = []
    x.append(li)
    la = []
    y.append(la)
```

```
node = 0
for ln in data:
   if ln[0] == ',': node = int(ln.split(' ')[-1]) - 2; continue
   l = ln.strip('\n').split(', ')
   x[node].append(int(l[0]))
   y[node].append(int(l[1]))



sets = []
nodes = []
for i in range(0, 24):
   sets.append((x[i], y[i]))
   nodes.append("node %d" % (i+2))
sets = tuple(sets)
nodes = tuple(nodes)
color = ("blue", "green", "red", "grey", "gold", "olive", "orange",
"orangered",
        "cyan", "darkcyan", "magenta", "crimson", "darkred",
"saddlebrown",
        "purple", "rebeccapurple", "mediumslateblue", "palevioletred",
"lime",
        "olivedrab", "peru", "y", "black", "tomato")



plt.figure(1, figsize=(15,15))
plt.xlabel("Millisecond packet was sent", fontsize=16)
plt.ylabel("Delay (ms)", fontsize=16)
for dt, col, group in zip(sets, color, nodes):
   x, y = dt
   plt.scatter(x, y, alpha=0.8, c=col, edgecolors='none', s=30,
label=group)
plt.title('End-to-End Delay', fontsize=26)
plt.legend(loc="upper left", bbox_to_anchor=(1, 1), ncol=2,
fontsize=16)
plt.savefig("%s.png" % sys.argv[2], bbox_inches='tight')
```

B.4 totalsplot.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""
```

```
import sys
import matplotlib.pyplot as plt


fp = open(sys.argv[1], "r")
data = fp.readlines()
fp.close()


nodes = []
intervals = []
nid = 0
through, packloss = [], []
for i in range(0, 25):
   nodes.append([])
   if i < 15: intervals.append(i+1)
del data[-6:]
for ln in data:
   if ln[0] == '+' or ln[0] == '*': continue
   if ln[0] == 'I':
       nid = int(ln[4:6].strip(' '))
       nodes[nid-1].append(ln.strip('\n'))
   d = ln.find("DIO")
   if ln[0] == ' ' and d >= 0:
       nodes[nid-1][-1] += " %s" % ln[d:]
   d = ln.find('Through')
   if d >= 0: through.append(float(ln[d:].split(' ')[1]))
   d = ln.find("loss")
   if d >= 0: packloss.append(int(ln[d:].split(' ')[1]))


plt.figure(1, figsize=(15, 15))
plt.subplot(2, 1, 1)
plt.xlabel("Minute", fontsize=16)
plt.ylabel("Packets", fontsize=16)
plt.plot(tuple(intervals), tuple(packloss), label="packet loss")
plt.legend(fontsize=16)
plt.subplot(2, 1, 2)
plt.xlabel("Minute", fontsize=16)
plt.ylabel("Packets per second", fontsize=16)
plt.plot(tuple(intervals), tuple(through), label="throughput")
plt.legend(fontsize=16)
```

```
plt.savefig("throughputLoss.png", bbox_inches='tight')
plt.clf()
rcv, fwd, sent, drop, DIO  = [], [], [], [], []
for i in range(0, 25):
   rcv.append([])
   fwd.append([])
   sent.append([])
   drop.append([])
   DIO.append([])



for i in range(0, 25):
   for j in nodes[i]:
       ln = j.split(' ')
       rcv[i].append(ln[2].strip(','))
       fwd[i].append(ln[4].strip(','))
       sent[i].append(ln[6].strip(','))
       drop[i].append(ln[8].strip('.'))
       DIO[i].append(ln[11].strip(','))



sets = []
nd = []
for i in range(0, 25):
   sets.append((intervals, rcv[i]))
   nd.append("node %d" % (i+1))
nd = tuple(nd)
color = ("blue", "green", "red", "grey", "gold", "olive", "orange",
"orangered",
        "cyan", "darkcyan", "magenta", "crimson", "darkred",
"saddlebrown",
        "purple", "rebeccapurple", "mediumslateblue", "palevioletred",
"lime",
        "olivedrab", "peru", "y", "black", "tomato", "indianred")



plt.figure(1, figsize=(25, 25))
plt.suptitle("Totals", fontsize=26)
plt.subplot(3, 2, 1)
plt.ylabel("Number of Packets", fontsize=16)
plt.xlabel("Minute", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, c=col, label=group)
```

```
sets = []
for i in range(0, 25):
   sets.append((intervals, fwd[i]))
plt.subplot(3, 2, 2)
plt.ylabel("Number of Packets", fontsize=16)
plt.xlabel("Minute", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, c=col, label=group)
plt.legend(loc="upper left", bbox_to_anchor=(1, 1), ncol=1,
fontsize=14)
sets = []
for i in range(0, 25):
   sets.append((intervals, sent[i]))
plt.subplot(3, 2, 3)
plt.ylabel("Number of Packets", fontsize=16)
plt.xlabel("Minute", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, c=col, label=group)
sets = []
for i in range(0, 25):
   sets.append((intervals, drop[i]))
plt.subplot(3, 2, 4)
plt.ylabel("Number of Packets", fontsize=16)
plt.xlabel("Minute", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, c=col, label=group)
sets = []
for i in range(0, 25):
   sets.append((intervals, DIO[i]))
plt.subplot(3, 2, 5)
plt.ylabel("Number of Packets", fontsize=16)
plt.xlabel("Minute", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, c=col, label=group)
plt.savefig("totals.png", bbox_inches='tight')
plt.close()
```

B.5 forward_delay_plot.py

```
# -*- coding: utf-8 -*-
```

```
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import sys
import matplotlib.pyplot as plt



fp = open(sys.argv[1], "r")
data = fp.readlines()
fp.close()



nodes = []
intervals = []
for i in range(0, 25):
   li = []
   nodes.append(li)
   if i != 0 and i <= 15: intervals.append(i)



for ln in data:
   if ln[0] != 'I': continue
   nid = int(ln[3:5].strip(' '))
   if ln.find("Average") < 0: nodes[nid-1].append(None); continue
   val = ln.split(': ')[1]
   val = val.split(' (')[0]
   nodes[nid-1].append(float(val))



sets = []
nd = []
for i in range(0, 25):
   sets.append((intervals, nodes[i]))
   nd.append("node %d" % (i+1))
sets = tuple(sets)
nd = tuple(nd)
color = ("tomato", "green", "red", "grey", "gold", "olive", "orange",
"orangered",
        "cyan", "darkcyan", "magenta", "crimson", "darkred",
"saddlebrown",
```

```
        "purple", "rebeccapurple", "mediumslateblue", "palevioletred",
"lime",
        "olivedrab", "peru", "y", "black", "blue", "indianred")



plt.figure(1, figsize=(15,15))
plt.xlabel("Minute", fontsize=16)
plt.ylabel("Delay (ms)", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, alpha=0.8, c=col, label=group)
plt.title('Forwarding Delay', fontsize=26)
plt.legend(loc="upper left", bbox_to_anchor=(1, 1), ncol=2,
fontsize=16)
plt.savefig("fwding_Delay.png", bbox_inches='tight')
```

B.6 average_forwarding.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import os
import sys
import matplotlib.pyplot as plt



path = sys.argv[1]
filename = 'frwd_delay'
files = []
for entry in os.listdir(path):
   if os.path.isdir(os.path.join(path, entry)):
       filepath = "%s%s/%s" % (path, entry, filename)
       files.append(filepath)



nodes = []
for i in range(0, 25):
   li = []
   for j in range(0, 24):
```

```
        lj = []
        li.append(lj)
    nodes.append(li)



run = 0
for f in files:
    fp = open(f, "r")
    data = fp.readlines()
    fp.close()
    malid = int(f.split('/')[5][-2:])
    for ln in data:
        if ln[0] != 'I': continue
        nid = int(ln[3:5].strip(' '))
        if nid == 1: continue
        if nid == malid: nid = 1
        if ln.find("Average") < 0:
            if ln.find("bigger") < 0:
                nodes[nid-1][run].append(None); continue
            else:
                nodes[nid-1][run].append(60000); continue
        val = ln.split(': ')[1]
        val = val.split(' (')[0]
        nodes[nid-1][run].append(float(val))
    run += 1



node = []
for i in nodes:
    avg, count = [0]*15, [0]*15;
    for j in i:
        if not j: continue
        for k in range(0, 15):
            if not j[k]: continue
            if str(j[k]).find("None") < 0:
                avg[k] += j[k];
                count[k] += 1
    rez = []
    for x, y in zip(avg, count):
        if y == 0: rez.append(None); continue
        rez.append(x/y)
    node.append(rez)
intervals = []
for i in range(1, 16): intervals.append(i)
sets = []
```

```
nd = []
for i in range(1, 25):
   sets.append((intervals, node[i]))
   nd.append("node %d" % (i+1))
#nd[0] = "malicious"
sets = tuple(sets)
nd = tuple(nd)
color = ("blue", "green", "red", "grey", "gold", "olive", "orange",
"orangered",
        "cyan", "darkcyan", "magenta", "crimson", "darkred",
"saddlebrown",
        "purple", "rebeccapurple", "mediumslateblue", "palevioletred",
"lime",
        "olivedrab", "peru", "y", "black", "tomato", "indianred")



plt.figure(1, figsize=(20,20))
plt.subplot(2, 1, 1)
plt.suptitle('Average Forwarding Delay', fontsize=26)
plt.xlabel("Minute", fontsize=16)
plt.ylabel("Delay (ms)", fontsize=16)
for dt, col, group in zip(sets, color, nd):
  x, y = dt
  plt.plot(x, y, alpha=0.8, c=col, label=group)
plt.legend(loc="upper left", bbox_to_anchor=(1, 1), ncol=1,
fontsize=16)
plt.subplot(2, 1, 2)
plt.xlabel("Minute", fontsize=16)
plt.ylabel("Delay (ms)", fontsize=16)
plt.plot(intervals, node[0], alpha=0.8, c='red', label='malicious')
plt.legend(fontsize=16)
plt.savefig("%sfwding_Delay.png" % path, bbox_inches='tight')
plt.show()
```

B.7 average_totals.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import os
```

```
import sys
import matplotlib.pyplot as plt



def node():
   nodes = []
   for i in range(0, 25):
       li = []
       for j in range(0, 24):
           lj = []
           li.append(lj)
       nodes.append(li)
   return nodes



path = sys.argv[1]
filename = 'tots'
files = []
for entry in os.listdir(path):
   if os.path.isdir(os.path.join(path, entry)):
       filepath = "%s/%s/%s" % (path, entry, filename)
       files.append(filepath)
nd = [[], [], [], [], []]
for i in range(0, 5): nd[i] = node()
mal = []
for i in range(0, 5):
   li = []
   for j in range(0, 24):
       lj = []
       li.append(lj)
   mal.append(li)
intervals = []
for i in range(1, 16): intervals.append(i)
for f, mid in zip(files, range(2, 26)):
   run = mid - 2
   fp = open(f, "r")
   data = fp.readlines()
   fp.close()
   for ln in data:
       nid = int(ln[4:6].strip(' '))
       if (nid) == mid:
           mal[0][run].append(int(ln.split(' ')[2].strip(',')))
           mal[1][run].append(int(ln.split(' ')[4].strip(',')))
           mal[2][run].append(int(ln.split(' ')[6].strip(',')))
           mal[3][run].append(int(ln.split(' ')[8].strip('.')))
```

```
            mal[4][run].append(int(ln.split(' ')[11].strip(',')))
        else:
            nnid = nid - 1
            nd[0][nnid][run].append(int(ln.split(' ')[2].strip(',')))
            nd[1][nnid][run].append(int(ln.split(' ')[4].strip(',')))
            nd[2][nnid][run].append(int(ln.split(' ')[6].strip(',')))
            nd[3][nnid][run].append(int(ln.split(' ')[8].strip('.')))
            nd[4][nnid][run].append(int(ln.split(' ')[11].strip(',')))
packet_type = ['Received', 'Forwarded', 'Sent', 'Dropped', 'DIOs']
for ptype, t in zip(packet_type, range(0, 5)):
    avmal = [0]*15; cmal, cben = 24, 24
    av = []
    for i in range(0, 25): av.append([0]*15)
    for r in mal[t]:
        for i in range(0, 15):
            avmal[i] += r[i]
    for n, i in zip(nd[t], range(0, 25)):
        for r in n:
            if not r: continue
            for j in range(0, 15):
                av[i][j] += r[j]
    for i in range(0, 15): avmal[i] = avmal[i] / cmal
    for i in range(0, 25):
        for j in range(0, 15):
            av[i][j] = av[i][j] / cben
    sets = []
    n = []
    for i in range(0, 25):
        sets.append((intervals, av[i]))
        n.append("node %d" % (i+1))
    sets = tuple(sets)
    n = tuple(n)
    color = ("blue", "green", "red", "grey", "gold", "olive", "orange",
"orangered",
             "cyan", "darkcyan", "magenta", "crimson", "darkred",
"saddlebrown",
             "purple", "rebeccapurple", "mediumslateblue",
"palevioletred", "lime",
             "olivedrab", "peru", "y", "black", "tomato", "indianred")
    plt.figure(1, figsize=(15, 15))
    plt.subplot(2, 1, 1)
    plt.suptitle(ptype, fontsize=26)
    plt.xlabel("Minute", fontsize=16)
    plt.ylabel("Number of Packets", fontsize=16)
    for dt, col, group in zip(sets, color, n):
```

```
        x, y = dt
        plt.plot(x, y, c=col, label=group)
    plt.legend(loc="upper left", bbox_to_anchor=(1, 1), fontsize=16)
    plt.subplot(2, 1, 2)
    plt.xlabel("Minute", fontsize=16)
    plt.ylabel("Number of Packets", fontsize=16)
    plt.plot(intervals, avmal, label='malicious', c='red')
    plt.legend(fontsize=16)
    plt.savefig("%s/%s.png" % (path, ptype), bbox_inches='tight')
    plt.clf()
plt.close()
```

B.8 av_setuptime.py

```
# -*- coding: utf-8 -*-
"""
Script needs python v3.3 or higher to run
@author: Maria Efthymiou
"""



import os
import sys



path = sys.argv[1]
filename = 'setup_time'
files = []
for entry in os.listdir(path):
    if os.path.isdir(os.path.join(path, entry)):
        filepath = "%s/%s/%s" % (path, entry, filename)
        files.append(filepath)
t = 0
for f in files:
    fp= open(f, "r")
    t += float(fp.readlines()[1].split(' ')[-1].strip('s.\n'))
    fp.close()
print("%.2f" % (t/24))
```