ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι.
ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Ανάπτυξη εφαρμογής υπολογισμού και γραφικής αναπαράστασης αποτελεσμάτων του προσομοιωτή ns-2

Του φοιτητή

Παπαγιαννίδη Χρήστου – Χαράλαμπου

Αρ. Μητρώου : 06/3024

Επιβλέπων καθηγητής

Θωμάς Λάγκας

Θεσσαλονίκη 2011

1 ΠΡΟΛΟΓΟΣ

Στα πλαίσια των προπτυχιακών σπουδών στο τμήμα Πληροφορικής του Αλεξάνδρειου Ανώτατου Τεχνολογικού Ιδρύματος Θεσσαλονίκης και καθώς ο κανονισμός ορίζει, ανατέθηκε η εκπόνηση πτυχιακής εργασίας ως αναπόσπαστο μέρος της ολοκλήρωσης των σπουδών. Η παρούσα πτυχιακή εργασία έχει τίτλο “Ανάπτυξη εφαρμογής δημιουργίας γραφημάτων από τα Trace αρχεία που εξάγει ο προσομοιωτής δικτύων NS-2” και επιβλέπων καθηγητής για την εκπόνησή της είναι ο Δρ Θωμάς Λάγκας, επιστημονικός συνεργάτης του τμήματος πληροφορικής του ΑΤΕΙ Θεσσαλονίκης.

Η πτυχιακή εργασία αυτή έχει σαν στόχο την δημιουργία μιας εφαρμογής ανάλυσης και γραφικής απεικόνισης στατιστικών-μετρικών των σεναρίων προσομοίωσης που εκτελούνται στον NS-2. Η εφαρμογή έχει ως μοναδική πηγή πληροφοριών το Trace αρχείο που εξάγει ο προσομοιωτής δικτύων NS-2. Παρόμοιες εφαρμογές υπάρχουν , όμως όλες παρουσιάζουν κάποια προβλήματα ή ελλείψεις τις οποίες έρχεται να καλύψει αυτή η εργασία.

Οι πρώτες κινήσεις μετά την ανάθεση της πτυχιακής εργασίας περιελάμβαναν την αναζήτηση πληροφοριών σχετικά με τη μορφή των πληροφοριών που εξάγει ο NS-2, καθώς επίσης και των ιδιαιτεροτήτων που μπορεί να παρουσιάζουν ανάλογα με τις ρυθμίσεις που ορίζει ο χρήστης στον προσομοιωτή. Την ίδια στιγμή ξεκίνησε και η σχεδίαση της εφαρμογής και η εύρεση της καλύτερης αρχιτεκτονικής λογισμικού. Έπειτα στα τελευταία βήματα πραγματοποιήθηκαν όλοι οι έλεγχοι ποιότητας και η σύγκριση των αποτελεσμάτων της εφαρμογής με άλλες παρόμοιες που κυκλοφορούν ήδη στο διαδίκτυο.

Συνοπτικά η εργασία αυτή περιλαμβάνει το θεωρητικό υλικό που βρέθηκε σχετικά με τον NS-2 το οποίο χρησιμοποιήθηκε και ως βάση για την ανάπτυξη της εφαρμογής μας. Ακολουθεί η περιγραφή των λειτουργιών και του γραφικού περιβάλλοντος της εφαρμογής καθώς επίσης και των παραδοχών που ίσως δημιουργήθηκαν για την αντιμετώπιση διαφόρων προβλημάτων. Τέλος περιλαμβάνει ένα σενάριο προσομοίωσης μαζί με τα αποτελέσματα της εφαρμογής ώστε να γίνει εφικτή η απόδειξη της σωστής λειτουργίας. Στο παράρτημα Β μπορεί κανείς να βρει όλο τον κώδικα της εφαρμογής.

2 ΠΕΡΙΛΗΨΗ

Μετά την συγκέντρωση και μελέτη των απαραίτητων πληροφοριών σχετικά με τον προσομοιωτή δικτύων NS-2 και πιο συγκεκριμένα για τα Trace αρχεία που εξάγει, ξεκίνησε η ανάπτυξη μιας ανεξάρτητης και ανοιχτού κώδικα εφαρμογής για την ανάλυση και αξιολόγηση των αποτελεσμάτων που προκύπτουν κατά την προσομοίωση ενός δικτύου.

Αμέσως μετά ορίστηκαν οι απαιτήσεις και οι λειτουργίες της εφαρμογής και ξεκίνησε η φάση ανάπτυξης. Την ολοκλήρωση της εφαρμογής ακολούθησε μια διαδικασία δοκιμών και ελέγχου των αποτελεσμάτων της σε μια πληθώρα σεναρίων διαφορετικού τύπου, ενώ ταυτόχρονα υποβάλλονταν διορθώσεις στον κώδικα της εφαρμογής.

3 ABSTRACT

After collecting and studying the necessary information about the network simulator NS-2 and more specifically on the Trace files export, began to develop an independent and open source application for analyzing and evaluating the results obtained by simulating a network.

Immediately started the development phase , after requirements and functions of the application were defined. Completion the application followed by testings and in a variety of different types of scenarios, while undergoing corrections to code of the application.

4 Πίνακας περιεχομένων

5 ΠΕΡΙΕΧΟΜΕΝΑ ΕΙΚΟΝΩΝ

6 ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ

1 Εισαγωγή

1.1 Γενικά

Ζούμε σε μια εποχή που σημειώνεται τεράστια αύξηση της κοινότητας ανοιχτού κώδικα και της λίστας εφαρμογών της κοινότητας. Όλο και περισσότερη προσοχή εστιάζεται σε εφαρμογές ανοιχτού κώδικα τόσο από την πανεπιστημιακή κοινότητα όσο και από ιδιώτες και εταιρίες. Ο προσομοιωτής δικτύου NS-2 είναι αποτέλεσμα της κοινότητας αυτής και έχει κερδίσει ήδη θέση σε πολλά πανεπιστήμια και ερευνητικά κέντρα και ως εκπαιδευτικό αλλά και ως ερευνητικό λογισμικό. Κάθε πρόγραμμα τέτοιας πολυπλοκότητας όπως ενός προσομοιωτή δικτύων συνοδεύεται από μια σειρά λειτουργιών ή δυνατοτήτων που δεν υλοποιήθηκαν η δεν αποτελούσαν μέρος των απαιτήσεων του προγράμματος κατά την σχεδίασή του. Αυτά τα κενά καλείται η κοινότητα να καλύψει προσφέροντας ο κάθε ένας το έργο του.

1.2 Αντικείμενο και Σκοπός της Παρούσης Εργασίας

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε ως αναπόσπαστο κομμάτι των σπουδών μου στα πλαίσια του προπτυχιακού προγράμματος σπουδών του τμήματος Πληροφορικής του ΑΤΕΙ Θεσσαλονίκης κατά το ακαδημαϊκό έτος 2010-2011. Η εργασία έχει τίτλο “Ανάπτυξη εφαρμογής υπολογισμού και γραφικής αναπαράστασης αποτελεσμάτων του προσομοιωτή ns-2”.

Αντικείμενο τις εν λόγω εργασίας είναι η ανάπτυξη εφαρμογής που δέχεται ως είσοδο αρχεία αποτελεσμάτων του προσομοιωτή δικτύου ns-2 (trace files) και υπολογίζει στατιστικά αποτελέσματα, ενώ δημιουργεί και σχετικές γραφικές παραστάσεις. Τέλος η εργασία περιλαμβάνει την μελέτη δοκιμαστικών σεναρίων προσομοίωσης δικτύων στον ns-2, για την επίδειξη της εν λόγω εφαρμογής.

Σκοπός της παρούσης εργασίας είναι η προσθήκη μιας ανοιχτού-κώδικα εφαρμογής στην διάθεση των χρηστών του NS-2 για την κάλυψη λειτουργιών που δεν καλύπτονται από τον ίδιο τον προσομοιωτή, όπως και η συλλογή των κατάλληλων πληροφοριών σχετικά με τα trace αρχεία που εξάγονται από τον προσομοιωτή.

1.3 Μεθοδολογία και Δομή Εργασίας

Κύριο τμήμα της εργασίας αποτελεί η εφαρμογή που αναπτύχθηκε στα πλαίσια της. Για την σχεδίαση της εφαρμογής έπρεπε να παρθούν κάποιες αποφάσεις σχετικά με την υποστήριξη των τύπων των Trace αρχείων που εξάγει ο NS-2. Ο προσομοιωτής είναι ανοιχτού κώδικα με δυνατότητα ανάπτυξης και εγκατάστασης module ή την τροποποίηση του βασικού κώδικα του, και αφήνει στην ευχέρεια του χρήστη την μορφή των trace αρχείων που εξάγει η προσομοίωσή του. Το γεγονός αυτό δεν μπορούσε να προβλεφθεί και για το λόγο αυτό η εφαρμογή σχεδιάστηκε βάση των βασικών τύπων trace τα οποία εξάγει ο προσομοιωτής πριν γίνει οποιαδήποτε μετατροπή ή προσθήκη από κάποιον χρήστη, χωρίς αυτό να αποκλείει την ενδεχόμενη σωστή λειτουργία και σε προσομοιώσεις που έχουν γίνει αλλαγές.

Στην συνέχεια με σενάρια που δημιουργήθηκαν ή βρέθηκαν στο διαδίκτυο με τυχαία επιλογή, πραγματοποιήθηκαν έλεγχοι καλής λειτουργίας στην εφαρμογή που αναπτύχθηκε. Οι διορθώσεις και οι αλλαγές στον κώδικα ήταν συχνές καθότι τόσο το documentation του προσομοιωτή όσο και οι πληροφορίες που αντλήθηκαν κατά τη διάρκεια της εργασίας δεν πρόσφεραν ένα εύκολο και αξιόπιστο περιβάλλον ανάπτυξης.

2 Προσομοιωτής δικτύων NS-2

2.1 Εισαγωγή

Ο NS-2 είναι ένας προσομοιωτής δικτύων διακριτών γεγονότων ο οποίος αναπτύχθηκε στο πανεπιστήμιο της Καλιφόρνιας Berkeley (UCB). Υποστηρίζει τα πρωτόκολλα δικτύων TCP και UDP, όπως επίσης πρωτόκολλα multicast και δρομολόγησης τόσο σε ενσύρματα όσο και σε ασύρματα δίκτυα. Ο NS-2 εκμεταλλεύεται πλήρως τις δυνατότητες του αντικειμενοστραφούς προγραμματισμού και είναι γραμμένος σε C++ και TCL. Η TCL χρησιμοποιείται για την επικοινωνία του χρήστη με τον προσομοιωτή και την σχεδίαση του δικτύου, ενώ η C++ υλοποιεί όλες τις λειτουργίες πυρήνα για να εκτελεστεί μια προσομοίωση και να εξαχθούν τα αποτελέσματα της. Οι δύο αυτές γλώσσες επιλέχθηκαν καθαρά για λόγους επιδόσεων από την πλευρά της C++ και από τη μη ανάγκη για recompile για την TCL. Αν και ο NS-2 δεν εγγυάται την εξαγωγή πιστών αποτελεσμάτων σε σχέση με τον πραγματικό κόσμο, προσπαθεί να μοντελοποιήσει με ακρίβεια την συμπεριφορά των περισσότερων πρωτοκόλλων και να μπορεί να χρησιμοποιηθεί για την μελέτη πολλών πρωτοκόλλων στα διάφορα επίπεδα του OSI. Ο NS-2 δεν επεξεργάζεται τα αποτελέσματα των προσομοιώσεων, παρά μόνο εξάγει ένα αρχείο το οποίο ονομάζεται trace file και περιλαμβάνει όλα τα διακριτά γεγονότα που συνέβησαν κάθε χρονική στιγμή κατά τη διάρκεια της προσομοίωσης. [3],[4]

2.2 Είσοδος δεδομένων

Ο χρήστης του NS-2 πρέπει να παρέχει τα δεδομένα του δικτύου που επιθυμεί να προσομοιώσει χρησιμοποιώντας την γλώσσα προγραμματισμού TCL. Η σύνδεση του πυρήνα του προσομοιωτή με τον TCL κώδικα που παρέχει ο χρήστης γίνεται με την χρήση TclCL (Tcl with classes) και OTcl (MIT Object TCL). Η μορφή που δίνει τα δεδομένα του σεναρίου προσομοίωσης ο χρήστης φαίνεται στον παρακάτω κώδικα. [3]

```
#Create an instance of the Simulator class

set ns [new Simulator]
#Add four nodes n0, n1, n2 and n3
set n0 [$ns node]
```

```
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Export packet traces
set f [open d2_l1.tr w]
$ns trace-all $f
#Create 2 (duplex) links (n0-n2, n1-n2, n3-n2)
#with 1.5Mb bandwidth and 10ms delay per link,
#including a DropTail queue for each link

$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
```

2.3 Κόμβοι

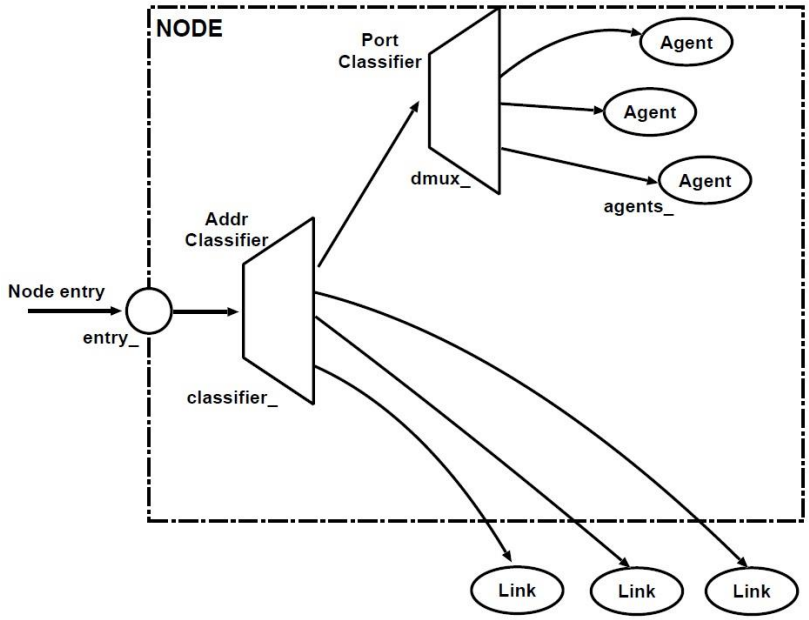
Κόμβος (Node) σε ένα δίκτυο ονομάζεται ένα σημείο σύνδεσης, είτε ένα σημείο αναδιανομής ή ένα τελικό σημείο επικοινωνίας. Ο ορισμός ενός κόμβου εξαρτάται από το δίκτυο και το επίπεδο στο οποίο το πρωτόκολλο αναφέρεται. Ένας φυσικός κόμβος δικτύου είναι μία ενεργή ηλεκτρονική συσκευή που είναι συνδεδεμένη σε ένα δίκτυο και έχει τη δυνατότητα αποστολής, λήψης ή διαβίβασης δεδομένων μέσω ενός καναλιού επικοινωνίας. [12]

Στον NS-2 η έννοια του κόμβου δεν έχει μεγάλες διαφορές με τον ορισμό που δόθηκε παραπάνω. Στον προσομοιωτή ως κόμβος ορίζεται κάθε αντικείμενο της κλάσης node.cc/node.h.

Όλοι οι κόμβοι μιας προσομοίωσης περιλαμβάνουν τουλάχιστον τα παρακάτω πεδία [3],[5] :

- Μία διεύθυνση ή κάποιο αναγνωριστικό (id) μονοτονικά αυξανόμενο κατά 1 (ξεκινώντας από την τιμή 0) σε όλο το εύρος της προσομοίωσης καθώς δημιουργείται ο κάθε κόμβος,
- Μια λίστα από γειτονικούς κόμβους(neighbor_),
- Μία λίστα από πράκτορες (agent_),
- Ένα αναγνωριστικό για τον τύπο του κόμβου(nodetype_)
- Το μοντέλο δρομολόγησης

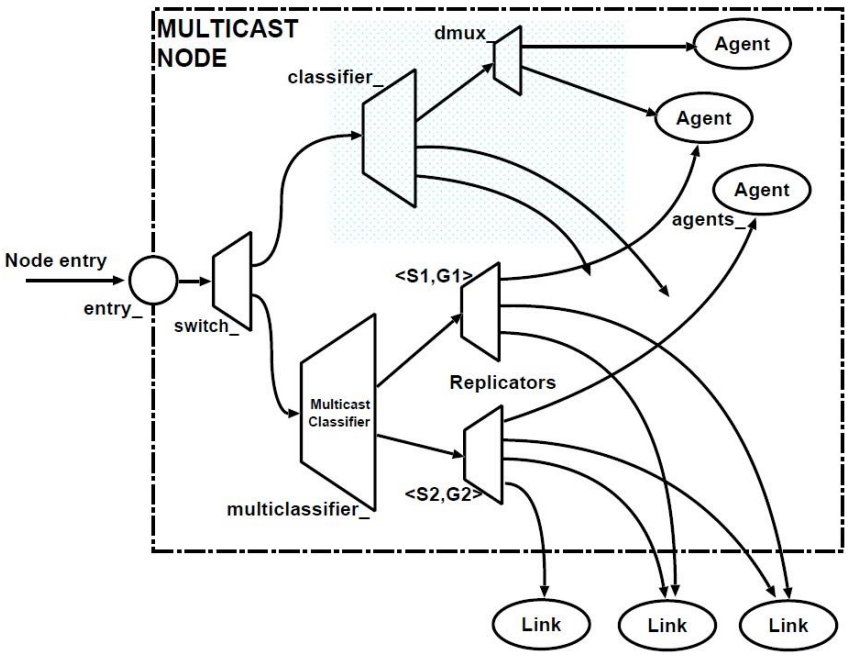
Η εσωτερική δομή ενός Unicast κόμβου φαίνεται στην εικόνα 1.



ΕΙΚΟΝΑ 1: Η ΕΣΩΤΕΡΙΚΗ ΔΟΜΗ ΕΝΟΣ UNICAST ΚΟΜΒΟΥ

Ως προεπιλογή ο προσομοιωτής δημιουργεί κόμβους μονοδιανομής (unicast nodes). Για να δημιουργηθεί μια προσομοίωση που να υποστηρίζει πολυδιανομή θα πρέπει να χρησιμοποιηθεί η ρύθμιση “-multicast on”.

Η εσωτερική δομή ενός βασικού κόμβου πολυδιανομής φαίνεται στην εικόνα 2.



ΕΙΚΟΝΑ 2 : Η ΕΣΩΤΕΡΙΚΗ ΔΟΜΗ ΕΝΟΣ MULTICAST ΚΟΜΒΟΥ

2.4 Αρχεία εξόδου

Υπάρχουν διάφοροι τρόποι συλλογής των δεδομένων που εξάγονται σε μια προσομοίωση. Σε γενικές γραμμές τα γεγονότα (Trace Events) είτε εμφανίζονται άμεσα κατά τη διάρκεια εκτέλεσης της προσομοίωσης, ή (πιο συχνά) αποθηκεύονται σε ένα αρχείο ώστε να επεξεργαστεί και να αναλυθεί αργότερα. Υπάρχουν δύο βασικοί αλλά διαφορετικοί τρόποι παρακολούθησης των δεδομένων που παράγονται κατά τη διάρκεια μιας προσομοίωσης και υποστηρίζονται από τον προσομοιωτή. Ο πρώτος, που ονομάζεται Trace καταγράφει κάθε πακέτο μόλις φθάσει, αποσταλεί ή απορριφθεί από μια σύνδεση ή μια ουρά. Τα αντικείμενα Trace πρέπει να ρυθμιστούν ως κόμβοι εντός της τοπολογίας του δικτύου, συνήθως συνδέοντας τους με ένα Tcl αντικείμενο τύπου “Channel”, το οποίο αντιπροσωπεύει έναν προορισμό για τα δεδομένα που συλλέγονται (συνήθως ένα αρχείο κειμένου). Ο άλλος τρόπος παρακολούθησης των δεδομένων χρησιμοποιεί αντικείμενα γνωστά με την ονομασία monitors. Τα αντικείμενα αυτά καταγράφουν μετρήσεις από διάφορα ενδιαφέροντα συμβάντα, όπως αφίξεις, αποστολές, απορρίψεις τόσο πακέτων όσο και bytes. Επίσης τα αντικείμενα αυτά έχουν την δυνατότητα να καταγράφουν μετρήσεις που σχετίζονται με το σύνολο των πακέτων των ροών, ή με τα πακέτα κάποιας συγκεκριμένης ροής δεδομένων χρησιμοποιώντας ένα flow monitor. [3]

Ο NS-2 για να υποστηρίξει τα Traces, χρησιμοποιεί ένα ειδικό header το οποίο περιλαμβάνεται σε κάθε πακέτο. Τοποθετεί επίσης ένα μοναδικό αναγνωριστικό σε κάθε πακέτο, ένα πεδίο τύπου για το πακέτο το οποίο ορίζουν οι agents που παράγουν τα πακέτα, ένα πεδίο για το μέγεθος του πακέτου σε μονάδες bytes και ένα πεδίο που χρησιμοποιείται για τον υπολογισμό multicast δέντρων διανομής. [3]

Αντίστοιχα για να υποστηρίξει τα Monitors χρησιμοποιεί ένα σύνολο αντικειμένων που δημιουργούνται και εισάγονται στην τοπολογία του δικτύου γύρω από τις ουρές δεδομένων. Με αυτό τον τρόπο αποτελούν το σημείο στο οποίο οι χρόνοι και τα στατιστικά στοιχεία που λαμβάνονται συλλέγονται για να χρησιμοποιηθούν στον υπολογισμό στατιστικών σε διάφορα χρονικά διαστήματα.[3]

2.4.1 Trace και OTcl

Η υποστήριξη των Trace επεκτείνεται και στην OTcl. Αυτό γίνεται δυνατό με ένα σύνολο ειδικών κλάσεων ορατών στην OTcl αλλά υλοποιημένες με C++, σε συνδυασμό με ένα σύνολο από βοηθητικές Tcl μεθόδους και κλάσεις ορισμένες στην βιβλιοθήκη του NS-2.Τα αντικείμενα των τύπων που ακολουθούν εισάγονται αυτούσια στην ίδια γραμμή στην τοπολογία του δικτύου. [2]

- Trace/Hop trace a “hop” (XXX what does this mean exactly; it is not really used XXX)
- Trace/Enque a packet arrival (usually at a queue)
- Trace/Deque a packet departure (usually at a queue)
- Trace/Drop packet drop (packet delivered to drop-target)
- Trace/Recv packet receive event at the destination node of a link

Όπως αναφέρθηκε παραπάνω υπάρχει και ένα σύνολο βοηθητικών μεθόδων που μπορούν να χρησιμοποιηθούν κατά την σχεδίαση της τοπολογίας του δικτύου και αποτελούν αυτόνομες διαδικασίες της κλάσης Simulator. [2]

flush-trace {} flush buffers for all trace objects in simulation

create-trace { type file src dst } create a trace object of type between the given src and dest nodes. If file is non-null, it is interpreted as a Tcl channel and is attached to the newly-created trace object.

trace-queue { n1 n2 file } arrange for tracing on the link between nodes n1 and n2. This function calls create-trace, so the same rules apply with respect to the file argument.

trace-callback{ ns command } arranges to call command when a line is to be traced. The procedure treats command as a string and evaluates it for every line traced.

monitor-queue { n1 n2 qtrace [Sample_interval]} calls the init-monitor function on the link between nodes n1 and n2.

drop-trace { n1 n2 trace } the given trace object is made the drop-target of the queue associated with the link between nodes n1 and n2.

2.5 Trace Formats

Υπάρχουν τρείς βασική τύποι αρχείων Trace που υποστηρίζει και εξάγει ο NS-2 σύμφωνα με το κεφάλαιο “Trace and Monitoring Support Trace File Format” του NS-2 Manual. Διατηρεί ένα τύπο για ενσύρματα δίκτυα, και δύο για ασύρματα τα οποία όμως έχουν μεγάλες διαφορές μεταξύ τους ως προς την μορφή των πληροφοριών και την διάταξή τους.

2.5.1 Normal Trace Format

Αυτός ο τύπος Trace χρησιμοποιείτε για την καταγραφή των γεγονότων που συμβαίνουν σε ενσύρματες συνδέσεις ή δίκτυα. Διατηρεί την πιο απλή και κατανοητή διάταξη στις πληροφορίες που περιλαμβάνει. Κάθε διακριτό γεγονός που καταγράφεται ξεκινά με ένα από τους τέσσερις πιθανούς χαρακτήρες(Abbreviation) και στη συνέχεια ακολουθούν τα υπόλοιπα πεδία όπως αυτά φαίνονται στον πίνακα 1 , και αναφέρονται στα [1],[3].

ΠΙΝΑΚΑΣ 1: ΠΕΔΙΑ ΚΑΙ ΜΟΡΦΟΠΟΙΗΣΗ ΤΟΥ NORMAL TRACE FORMAT

Abbreviation	Type	Value
r: Receive d: Drop e: Error +: Enqueue -: Dequeue	%g %d %d %s %d %s %d %d.%d %d.%d %d %d	
	double	Time
	int	(Link-layer) Source Node
	int	(Link-layer) Destination Node
	string	Packet Name
	int	Packet Size
	string	Flags
	int	Flow ID
	int	(Network-layer) Source Address
	int	Source Port
	int	(Network-layer) Destination Address
	int	Destination Port
	int	Sequence Number
	int	Unique Packet ID

Το πεδίο Flags αν είναι κενό έχει την τιμή “-------“. Αντίθετα αν κάποια σημαία είναι ενεργοποιημένη στο συγκεκριμένο πακέτο, τότε οι χαρακτήρες που φαίνονται στον πίνακα 2,τοποθετούνται στις αντίστοιχες θέσεις αντί για τον χαρακτήρα “-“.[1]

ΠΙΝΑΚΑΣ 2:ΤΙΜΕΣ ΤΟΥ ΠΕΔΙΟΥ FLAG ΣΤΟ NORMAL TRACE FORMAT

Value	Meaning
C	ECN-echo
P	Pri_(supposedly unused)
-	
A	Congestion Action
E	Congestion Experienced
F	Fast Start
N	ECN-capable

Ανάλογα με τον τύπο του πακέτου μπορεί το Trace να έχει επιπλέον πληροφορίες όπως φαίνονται στον πίνακα 3.[1]

ΠΙΝΑΚΑΣ 3: ΠΛΗΡΟΦΟΡΙΕΣ ΑΝΑΛΟΓΑ ΤΟΥ ΤΥΠΟΥ ΠΑΚΕΤΟΥ ΣΕ NORMAL TRACE FORMAT

Event	Type	Value
TCP	%d 0x%x %d %d	
	int	Ack Number
	hexadecimal	Flags (Used by FullTCP) FIN=0x01, SYN=02, PUSH=08, ACK=10, ECE=40, CWR=80
	int	Header Length

Event	Type	Value
	int	Socket Address Length
Satelite	%.2f %.2f %.2f %.2f	
	double	Source Latitude
	double	Source Longitude
	double	Destination Latitude
	double	Destination Longitude

2.5.2 Old Wireless Trace Format

Αυτός ο τύπος Trace χρησιμοποιείται όπως δηλώνει και το όνομά του για ασύρματα γεγονότα που συμβαίνουν κατά τη διάρκεια της προσομοίωσης. Υπάρχουν και εδώ τέσσερις ειδικοί χαρακτήρες με τους οποίους ξεκινάει κάθε καταγεγραμένο γεγονός και ακολουθεί μία από τις δύο πιθανές μορφοποιήσεις, ανάλογα με το αν καταγράφονται στο γεγονός οι συντεταγμένες X και Y του ασύρματου κόμβου.Η ιδιαιτερότητα που υπάρχει στο σημείο αυτό είναι ότι αν στο γεγονός καταγράφονται και οι συντεταγμένες του κόμβου τότε το αναγνωριστικό του κόμβου θα εσωκλύεται με το σύμβολο “_”, ενώ στην περίπτωση που δεν καταγράφονται οι συντεταγμένες του τότε θα υπάρχει μόνο το αναγνωριστικό του κόμβου χωρίς “-“. Ο πίνακας 4 δείχνει αναλυτικά την μορφή των γεγονότων αυτού του τύπου. [1],[3]

Πινακας 4: Πεδια και μορφοποιηση του Old Wireless Trace Format

Abbreviation	Type	Value
s: Sent r: Receive d: Drop f: Forward	%.9f %d (%6.2f %6.2f) %3s %4s %d %s %d [%x %x %x %x]	
	%.9f _%d_ %3s %4s %d %s %d [%x %x %x %x]	
	double	Time
	int	Node ID
	double	X Coordinate (If Logging Position)

Abbreviation	Type	Value
	double	Y Coordinate (If Logging Position)
	string	Trace Name
	string	Reason
	int	Event Identifier
	string	Packet Type
	int	Packet Size
	hexadecimal	Time To Sent Data
	hexadecimal	Destination MAC Address
	hexadecimal	Source MAC Address
	hexadecimal	Type (ARP, IP)

Κάποιες παλιότερες versions του NS-2 (όπως η 2.1b5) έχουν 5 δεκαεξαδικές τιμές μεταξύ των αγκυλών αντί για τέσσερις. Η πρώτη δεκαεξαδική τιμή αναπαριστά στοιχεία ελέγχου πλαισίου MAC, ενώ οι υπόλοιπες τιμές παραμένουν όπως φαίνονται στον παραπάνω πίνακα.[1]

Ανάλογα με τον τύπο του πακέτου, το γεγονός που καταγράφεται μπορεί να συμπεριλαμβάνει και επιπλέον πληροφορίες όπως αυτές φαίνονται στον πίνακα 5.

ΠΙΝΑΚΑΣ 5: ΠΛΗΡΟΦΟΡΙΕΣ ΑΝΑΛΟΓΑ ΤΟΥ ΤΥΠΟΥ ΠΑΚΕΤΟΥ ΣΕ OLD WIRELESS TRACE FORMAT

Event	Type	Value
ARP Trace	------- [%s %d/%d %d/%d]	
	string	Request or Reply
	int	Source MAC Address
	int	Source Address
	int	Destination MAC Address
	int	Destination Address
DSR Trace	%d [%d %d] [%d %d %d %d->%d] [%d %d %d %d->%d]	
	int	Number Of Nodes Traversed
	int	Routing Request Flag
	int	Route Request Sequence Number
	int	Routing Reply Flag
	int	Route Request Sequence Number
	int	Reply Length

Event	Type	Value
	int	Source Of Source Routing
	int	Destination Of Source Routing
	int	Error Report Flag (?)
	int	Number Of Errors
	int	Report To Whom
	int	Link Error From
	int	Link Error To
AODV Trace	[0x%x %d %d [%d %d] [%d %d]] (REQUEST)	
	hexadecimal	Type
	int	Hop Count
	int	Broadcast ID
	int	Destination
	int	Destination Sequence Number
	int	Source
	int	Source Sequence Number
	[0x%x %d [%d %d] %f] (%s)	
	hexadecimal	Type
	int	Hop Count
	int	Destination
	int	Destination Sequence Number
	double	Lifetime
	string	Operation (REPLY, ERROR, HELLO)
TORA Trace	[0x%x %d] (QUERY)	
	hexadecimal	Type
	int	Destination
	0x%x %d (%f %d %d %d %d) (UPDATE)	
	hexadecimal	Type
	int	Destination
	double	Tau
	int	Oid
	int	R
	int	Delta
	int	ID
	[0x%x %d %f %d] (CLEAR)	
	hexadecimal	Type

Event	Type	Value
	int	Destination
	double	Tau
	int	Oid
IP Trace	------- [%d:%d %d:%d %d %d]	
	int	Source IP Address
	int	Source Port Number
	int	Destination IP Address
	int	Destination Port Number
	int	TTL Value
	int	Next Hop Address, If Any
TCP Trace	[%d %d] %d %d	
	int	Sequence Number
	int	Acknowledgment Number
	int	Number Of Times Packet Was Forwarded
	int	Optimal Number Of Forwards
CBR Trace	[%d] %d %d	
	int	Sequence Number
	int	Number Of Times Packet Was Forwarded
	int	Optimal Number Of Forwards
IMEP Trace	[%c %c %c 0x%04x]	
	char	Acknowledgment Flag
	char	Hello Flag
	char	Object Flag
	hexadecimal	Length
RCA Trace (from MIT Leach Code)	------- [%c %d %d %d]	
	char	Operation (A, R, D)
	int	RCA Source
	int	RCA Link Destination
	int	RCA MAC Destination

2.5.3 New Wireless Trace Format

Σε μια προσπάθεια να συγχωνευτεί το OldWireless format, το οποίο χρησιμοποιεί τα cmu-trace αντικείμενα, με το tracing του NS-2, δημιουργήθηκε ένας νέος βελτιωμένος τύπος καταγραφής γεγονότων. Η δομή κάθε γεγονότος ακολουθεί εντελώς διαφορετική δομή από τους δύο προηγούμενους τύπους καταγραφής γεγονότων, όπως θα φανεί παρακάτω. Το New Wireless Format είναι συμβατό-προς τα πίσω με το Old Wireless Format και μπορεί να ενεργοποιηθεί απλώς καλώντας την παρακάτω εντολή :

$ns use-newtrace

Αυτή η εντολή πρέπει να κληθεί πριν την γενική εντολή $ns trace-all <trace-filedirectory>.

Κάθε γεγονός που καταγράφεται χωρίζεται σε πεδία τιμών, όπου προηγείται της κάθε τιμής ένα αναγνωριστικό της μορφής –X,–XX ή -XXX που μας βοηθάει να καταλαβαίνουμε τι αντιπροσωπεύει η κάθε τιμή. Έτσι κάθε γεγονός χωρίζεται στα παρακάτω πεδία [1]

- **Τύπος γεγονότος**

  s: sent

  r: receive

  d: drop

  f: forward

- **Γενική ετικέτα**

  Το δεύτερο πεδίο ξεκινάει με ‘-t’ και μπορεί να αναπαριστά χρονική στιγμή ή κάποια γενική ρύθμιση.

  -t time

  -t * (Γενική ρύθμιση)

- **Ετικέτες ιδιοτήτων κόμβου**

  -Ni: node id
  -Nx: node’s x-coordinate

-Ny: node's y-coordinate

-Nz: node's z-coordinate

-Ne: node energy level

-Nl: trace level, such as AGT, RTR, MAC

-Nw: reason for the event. The different reasons for dropping a packet are given below:

"END" DROP_END_OF_SIMULATION

"COL" DROP_MAC_COLLISION

"DUP" DROP_MAC_DUPLICATE

"ERR" DROP_MAC_PACKET_ERROR

"RET" DROP_MAC_RETRY_COUNT_EXCEEDED

"STA" DROP_MAC_INVALID_STATE

"BSY" DROP_MAC_BUSY

"NRTE" DROP_RTR_NO_ROUTE i.e no route is available.

"LOOP" DROP_RTR_ROUTE_LOOP i.e there is a routing loop

"TTL" DROP_RTR_TTL i.e TTL has reached zero.

"TOUT" DROP_RTR_QTIMEOUT i.e packet has expired.

"CBK" DROP_RTR_MAC_CALLBACK

"IFQ" DROP_IFQ_QFULL i.e no buffer space in IFQ.

"ARP" DROP_IFQ_ARP_FULL i.e dropped by ARP

"OUT" DROP_OUTSIDE_SUBNET i.e dropped by base stations on receiving routing updates from nodes outside its domain.

- **Πληροφορίες πακέτου σε επίπεδο IP**

  Οι ετικέτες αυτής της κατηγορίας ξεκινούν με –I(κεφαλαίο i) και περιγράφονται παρακάτω:

  -Is: source address.source port number

  -Id: dest address.dest port number

  -It: packet type

  -Il: packet size

  -If: flow id

  -Ii: unique id

  -Iv: ttl value

- **Πληροφορίες Επόμενου Packet Hop**

  Οι ετικέτες αυτής της κατηγορίας ξεκινούν με –H και περιγράφονται παρακάτω:

  -Hs: id for this node

  -Hd: id for next hop towards the destination, -1,-2 . Οι τιμές -1 και -2 δηλώνουν ότι το πακέτο είναι broadcast, και ότι ο κόμβος προορισμού δεν έχει οριστεί ακόμα αντίστοιχα. Η τιμή -2 συνήθως εμφανίζεται σε πακέτα που περνούν από Επίπεδο Εφαρμογής(AGT) στο Επίπεδο δρομολόγησης(RTR).

- **Πληροφορίες πακέτου σε επίπεδο MAC**

  Οι ετικέτες αυτής της κατηγορίας ξεκινούν με –M και περιγράφονται παρακάτω:

  -Ma: duration

  -Md: destination’s Ethernet address

  -Ms: source’s Ethernet address

  -Mt: Ethernet type

- **Πληροφορίες πακέτου σε επίπεδο Εφαρμογής**

  Οι πληροφορίες που καταγράφονται σε αυτή την κατηγορία αναφέρονται σε τύπους εφαρμογών όπως ARP,TCP καθώς και πρωτόκολλα δρομολόγησης(adhoc) όπως PUMA,DSR,AODV κ.α. Οι ετικέτες αυτής της κατηγορίας ξεκινούν με –P και αναφέρονται παρακάτω σε σειρά ανάλογα με τον τύπο Εφαρμογής:

  ARP

  -Po: Request or Reply

  -Pms: Source Mac Address

  -Ps: Source Address

  -Pmd: Destination MAC Address

-Pd: Destination Address

DSR

-Ph: Number Of Nodes Traversed

-Pn: Route Request Sequence Number

-Pq: Routing Request Flag

-Pp: Routing Reply Flag

-Pl: Reply length

-Pe: Source->Destination Of Source Routing

-Pw: Error Report Flag

-Pm: Number Of Errors

-Pc: Report To Whom

-Pb: Link Error From LinkA->LinkB

AODV

-Pt: Type

-Ph: Hop Count

-Pb: Broadcast ID

-Pd: Destination

-Pds: Destination Sequence Number

-Ps: Source

-Pss: Source Sequence Number

-Pl: Lifetime

-Pc: Operation(REQUEST,REPLY,ERROR,HELLO)

TORA

-Pt: Type

-Pd: Destination

-Pa: Time

-Po: Creator ID

-Pr: R

-Pe: Delta

-Pi: ID

-Pc: Operation(QUERY,UPDATE,CLEAR)

IP

-Is: Source Address.Port

-Id: Destination Address.port

-It: Packet Type

-Il(κεφαλαίο i,πεζό L): Packet Size

-If: Flow ID

-Ii: Unique Packet ID

-Iv: TTL Value

CBR

-Pi: Sequence Number

-Pf: Number Of Times Packet Was Forwarded

-Po: Optimal Number Of Forwards

TCP

-Ps: Sequence Number

-Pa: Acknowledgment Number

-Pf: Number Of Times Packet Was Forwarded

-Po: Optimal Number Of Forwards

IMEP

-Pa: Acknowledgment Flag

-Ph: Hello Flag

-Po: Object Flag

-Pl(πεζό L): Length

Θα πρέπει να αναφερθεί ότι τα πεδία αυτά μπορεί να μεταβάλλονται να αφαιρούνται ή να προστίθενται καινούργια καθώς το Format αυτό είναι ακόμα σε στάδιο ανάπτυξης και δεν έχει οριστικοποιηθεί η μορφή και οι προδιαγραφές του. Τέλος πρέπει να τονιστεί ότι ο NS-2 είναι ένα ανοιχτού κώδικα προσομοιωτής και συνηθίζεται να πειράζουν οι έμπειροι χρήστες τις κλάσεις που εξάγου τα Traces ή ακόμα και να προσθέτουν επιπλέον πληροφορίες, κάτι που οδηγεί στην αδυναμία της εφαρμογής μας να προσαρμοστεί στις τυχόν αλλαγές που πραγματοποιούν οι χρήστες του NS-2.

3 jTrcezer

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί η εφαρμογή που αναπτύχθηκε στα πλαίσια αυτής της πτυχιακής εργασίας. Το όνομα της εφαρμογής είναι jTrcezer και προήλθε από το συνδυασμό των λέξεων Trace και Analyzer. Ο στόχος της εφαρμογής αυτής είναι η δημιουργία γραφικών παραστάσεων της πραγματικής κατάστασης ενός δικτύου προσομοίωσης για την εύκολη ανάλυση και αξιολόγηση του δικτύου αυτού. Η εφαρμογή αποτελείται από δύο βασικές λειτουργίες, η μία είναι η ανάγνωση και κατανόηση των δεδομένων που περιέχονται σε ένα Trace αρχείο που εξάγεται από τον προσομοιωτή δικτύων NS-2, όπως περιγράφεται σε προηγούμενο κεφάλαιο και η δεύτερη είναι η ανάλυση, ο υπολογισμός και η εμφάνιση γραφημάτων που αφορούν κάποια μέτρα αξιολόγησης της κατάστασης

και της αποδοτικότητας ενός δικτύου. Για την ανάπτυξη της εφαρμογής επιλέχθηκε η γλώσσα προγραμματισμού Java λόγω του αντικειμενοστραφούς χαρακτήρας της, αλλά κυρίως λόγω της φορητότητας που προσφέρει. Επίσης για την δημιουργία των διαγραμμάτων χρησιμοποιήθηκε η βιβλιοθήκη jFreeChart η οποία συνοδεύεται από GNU άδεια χρήσης. Ο κώδικας της εφαρμογής είναι διαθέσιμος στο παράρτημα Β.

3.2 Αρχιτεκτονική Σχεδίασης

Ένας από τους κύριους στόχους που είχε οριστεί πριν ξεκινήσει η ανάπτυξη της εφαρμογής ήταν η εύκολη τροποποίηση και επέκταση της εφαρμογής για την υποστήριξη περισσότερων τύπων Trace αρχείων όπως επίσης και νέων μετρήσεων που δεν συμπεριλαμβάνονται στην τρέχουσα έκδοση της εφαρμογής. Για το λόγο αυτό έγινε ένας βασικός διαχωρισμός των λειτουργιών σε 4 επίπεδα.

- Ανάγνωση αρχείου (I/O λειτουργία)
- Κατανόηση τύπου δεδομένων ανά γραμμή
- Αναζήτηση και επεξεργασία των δεδομένων
- Εμφάνιση διαγραμμάτων

Κάθε μία από τις παραπάνω λειτουργίες είναι ανεξάρτητη από τις υπόλοιπες ώστε να είναι δυνατή η αλλαγή οποιουδήποτε τμήματος της εφαρμογής χωρίς να απαιτείτε η ενημέρωση των άλλων κλάσεων.

Πιο συγκεκριμένα μία κλάση(Util/TrcReader.java) είναι υπεύθυνη για την ανάγνωση του αρχείου, η οποία γίνεται με τη βοήθεια ενός BufferedInputStream που επιταχύνει τη διαδικασίας ανάγνωσης του αρχείου κειμένου. Στην συνέχεια τα δεδομένα περνάνε στην κλάση Global/Trace.java, η οποία αναλαμβάνει να αντιστοιχίσει κάθε γεγονός σε ένα τύπο αντικειμένου και να συλλέξει όλα τα αντικείμενα σε μία δομή δεδομένων η οποία αποτελεί από αυτό το σημείο και μετά την πηγή πληροφοριών για την ανάλυση των αποτελεσμάτων της προσομοίωσης. Τέλος μια τρίτη κλάση έχει όλη την ευθύνη του φιλτραρίσματος των δεδομένων και την διανομή μόνο των πληροφοριών που χρειάζεται κάθε μία από τις κλάσεις που δημιουργούν τα διαγράμματα.

3.2.1 Τύποι Δεδομένων

Για κάθε ένα από τους τρείς τύπους γεγονότων που αναφέρονται στο κεφάλαιο 2 σχεδιάστηκε μία κλάση αντικειμένων αυτού του τύπου. Όλες οι κλάσεις αντικειμένων που αναπαριστούν τύπους γεγονότων πρέπει να συνδέονται με κληρονομικότητα με μια γενική κλάση Event που έχει μοναδική χρησιμότητα στην αποθήκευση όλων των αντικειμένων αυτών σε μία δομή δεδομένων τύπου Event όπως φαίνεται στην εικόνα 3. Κάθε κλάση έχει στο δομητή της όλες της πληροφορίες για την ανάλυση και την κατανομή των πληροφοριών αυτών στις ανάλογες μεταβλητές αφού δέχεται αυτούσιο το Trace ως String, έτσι εξασφαλίζεται εν μέρει, η αρχή της Μοναδικής Αρμοδιότητας(Single-Responsibility Principle).

- Normal Event(Data/NormalEvent.java)

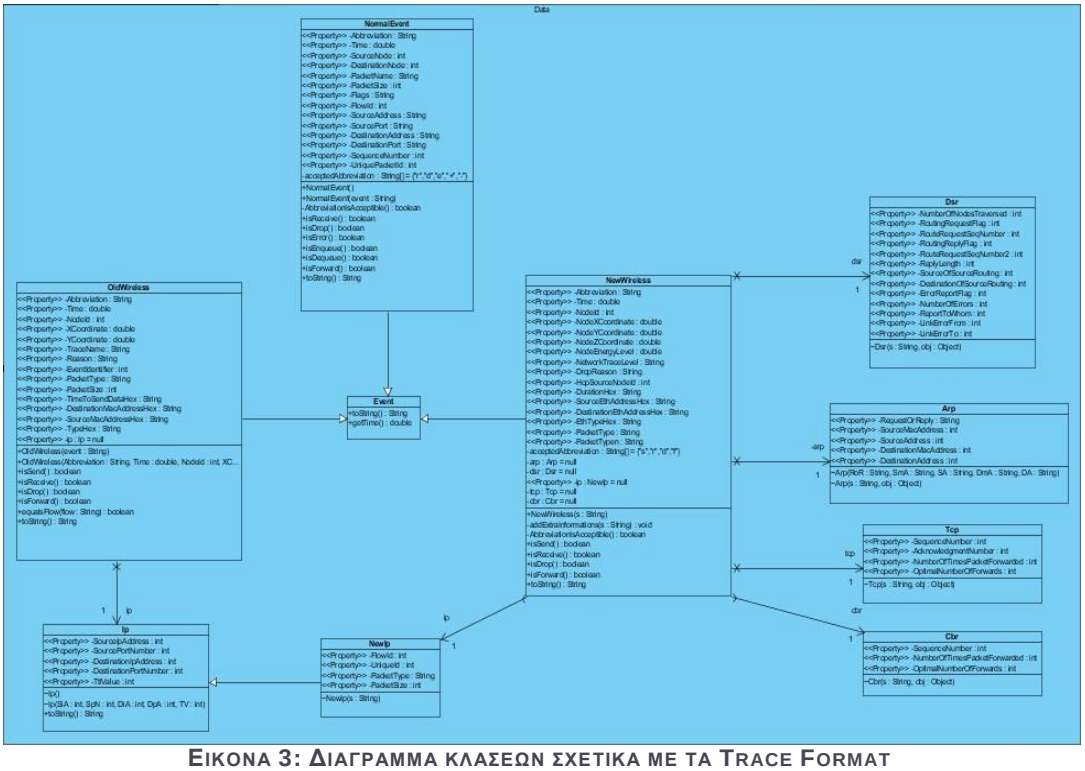
```
+ 0.5 0 1 cbr 500 ------- 1 0.0 3.0 0 64
```

- Old Wireless(Data/OldWireless.java)

```
s 29.031284241 _9_ AGT  --- 0 cbr 512 [0 0 0 0] -------
[9:2 11:0 32 0]
```

- New Wireless(Data/NewWireless.java)

```
s -t 39.100000000 -Hs 2 -Hd -2 -Ni 2 -Nx 43.69 -Ny 17.12
-Nz 0.0 0 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0
-Mt 0 -Is 2.0 -Id 1 .0 -It cbr -Il 50 -If 0 -Ii 195 -Iv 32
-Pn cbr -Pi 195 -Pf 0 -Po 0
```

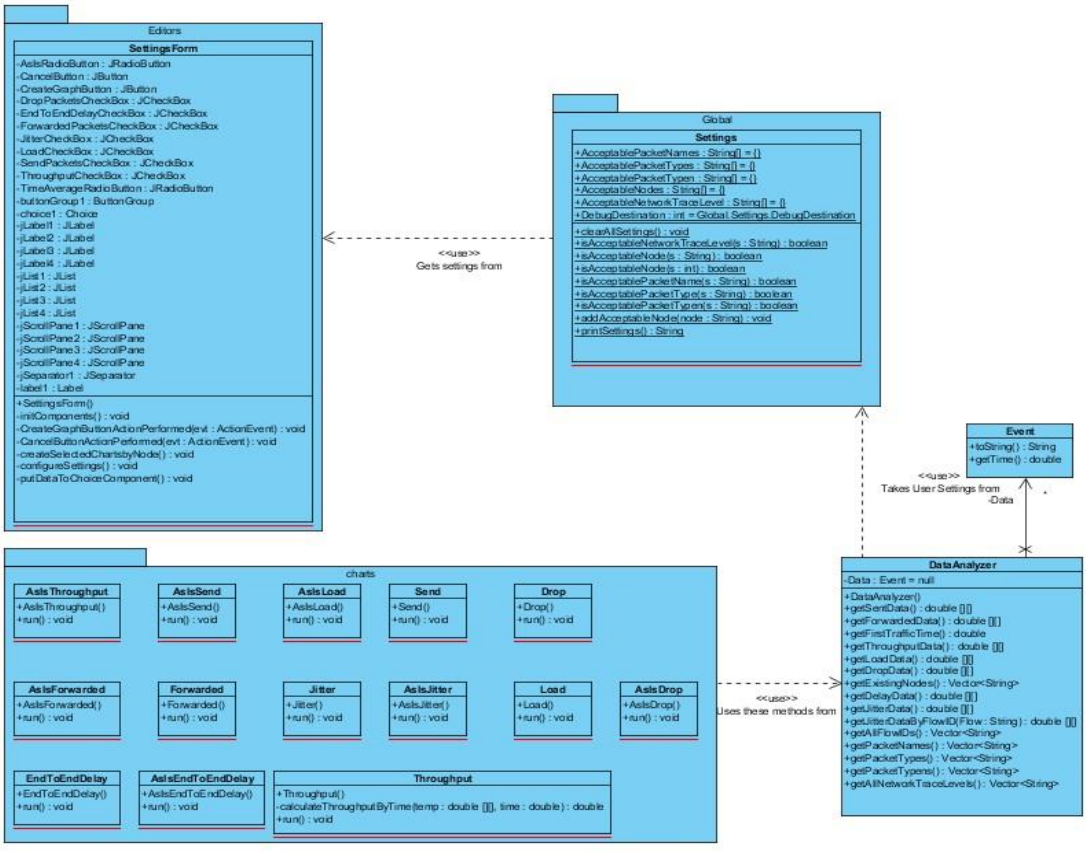


ΕΙΚΟΝΑ 3: ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ ΣΧΕΤΙΚΑ ΜΕ ΤΑ TRACE FORMAT

3.2.2 Φιλτράρισμα και επεξεργασία των δεδομένων

Κάθε σενάριο προσομοίωσης που εκτελούν οι χρήστες του NS είναι διαφορετικό και οι εξαγόμενες πληροφορίες είναι δυναμικές ανάλογα με την τοπολογία, τα πρωτόκολλα, τις εφαρμογές, τον τρόπο διευθυνσιοδότησης κ.α. Επίσης ο χρήστης της εφαρμογής μας έχει θα πρέπει να έχει τη δυνατότητα να εξάγει κάποια διαγράμματα όποια επιθυμεί και για συγκεκριμένους τύπους πακέτων ,επίπεδα OSI και κόμβους. Έτσι είναι αναγκαίο το φιλτράρισμα των Trace ανάλογα με τις ρυθμίσεις που έχει ορίσει ο χρήστης. Για το λόγο αυτό υλοποιήθηκε μια κλάση(Data/Settings.java) υπεύθυνη να ενημερώνει τις υπόλοιπες λειτουργίες της εφαρμογής με τις τρέχουσες ρυθμίσεις του κάθε χρήστη όπως φαίνεται και στην εικόνα 4. Οι πληροφορίες που συλλέγει είναι οι εξής :

- Αποδεκτά ονόματα πακέτων
- Αποδεκτούς τύπους πακέτων
- Αποδεκτούς τύπους IP πακέτων
- Επιλεγμένοι κόμβοι δικτύου για την εξαγωγή μετρήσεων
- Επιλεγμένο Επίπεδο OSI στο οποίο θα γίνουν οι μετρήσεις



ΕΙΚΟΝΑ 4: ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ ΣΧΕΤΙΚΑ ΜΕ ΤΑ ΓΡΑΦΗΜΑΤΑ ΚΑΙ ΤΙΣ ΡΥΘΜΙΣΕΙΣ ΤΟΥ ΧΡΗΣΤΗ

Για το φιλτράρισμα, την συγκέντρωση και την μετατροπή των Trace σε μια πιο ευκολονόητη μορφή ανάλογα με την μέτρηση που επιδιώκεται, υπεύθυνη είναι αποκλειστικά μια κλάση(Util/DataAnalyzer.java). Κάθε αίτημα που φτάνει στην κλάση αυτή ξεκινά μια κοινή διαδικασία για όλες τις μετρήσεις, φιλτράρισμα των Trace ανάλογα με τις πληροφορίες που δέχεται από την Data/Settings.java, αποθήκευση μόνο των δεδομένων που είναι απαραίτητες για τον υπολογισμό της κάθε μέτρησης σε μια δομή δεδομένων και επιστροφή της δομής αυτής στην κλάση που έστειλε το ανάλογο αίτημα. Μια παραδοχή που έχει οριστεί για την περίπτωση που ο χρήστης δεν επιλέξει κανένα όνομα, τύπου πακέτου ή τύπου IP πακέτων, είναι ότι δεν έχει εξαιρέσει κανένα πακέτο το οποίο ανήκει σε οποιαδήποτε από τις κατηγορίες που υπάρχουν στο τρέχον Trace αρχείο. Αντίθετα αν ο χρήστης δεν έχει επιλέξει κανένα κόμβο του δικτύου δεν εξάγεται κανένα αποτέλεσμα, ενώ η μη επιλογή κάποιου κόμβου δεν σημαίνει την εξαίρεση του από τους υπολογισμούς αλλά την μη εξαγωγή αποτελεσμάτων αποκλειστικά για τον συγκεκριμένο κόμβο. Τέλος όσο αναφορά τα επίπεδα OSI, εμφανίζονται στο χρήστη όλα τα πιθανά επίπεδα που αναγνωρίστηκαν(με διαδοχική σειρά) στο τρέχον Trace αρχείο και ως προεπιλογή ορίζεται το πρώτο αναγνωρισθέν επίπεδο.

3.3 Μετρήσεις και Διαγράμματα

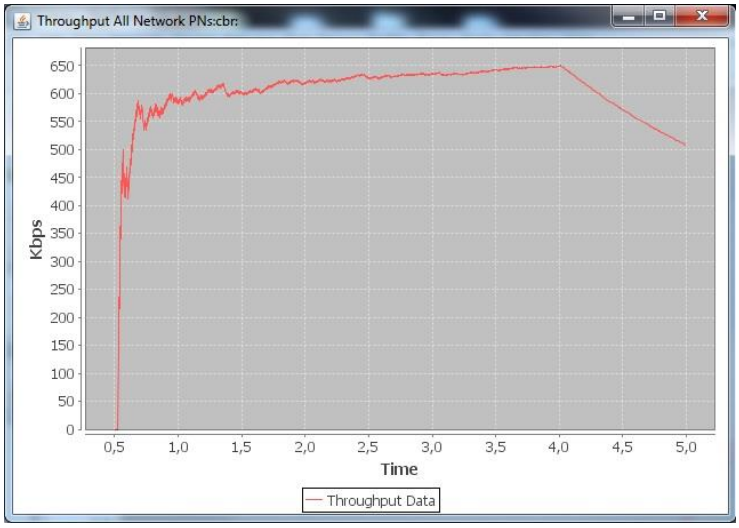
3.3.1 Throughput

Στην επιστήμη δικτύων το throughput είναι ένας αριθμός απόδοσης που χαρακτηρίζει το σύνολο των δεδομένων που στάλθηκαν από ένα σημείο του δικτύου, μεταφέρθηκαν και παρελήφθησαν χωρίς σφάλμα από κάποιο άλλο σημείο. Με λίγα λόγια είναι μία μέτρηση της απόδοσης ενός δικτύου. Η μονάδα μέτρησης του είναι bits ανά δευτερόλεπτο (bps,Kbps,Mbps). [9],[10]

Υπάρχουν δύο τύποι διαγραμμάτων που μπορεί να εμφανίσει η εφαρμογή jTrcezer σχετικά με το throughput. Ο πρώτος ονομάζεται Time Average Throughput και απεικονίζει κάθε χρονική στιγμή το μέσο όρο της απόδοσης του δικτύου από την αρχή της προσομοίωσης έως την κάθε χρονική στιγμή. Σε προσομοιώσεις δικτύων με χρήση πρωτοκόλλων μεταβλητού ρυθμού μετάδοσης δεδομένων αυτό το διάγραμμα δεν είναι αξιόπιστο διότι υπάρχει η πιθανότητα

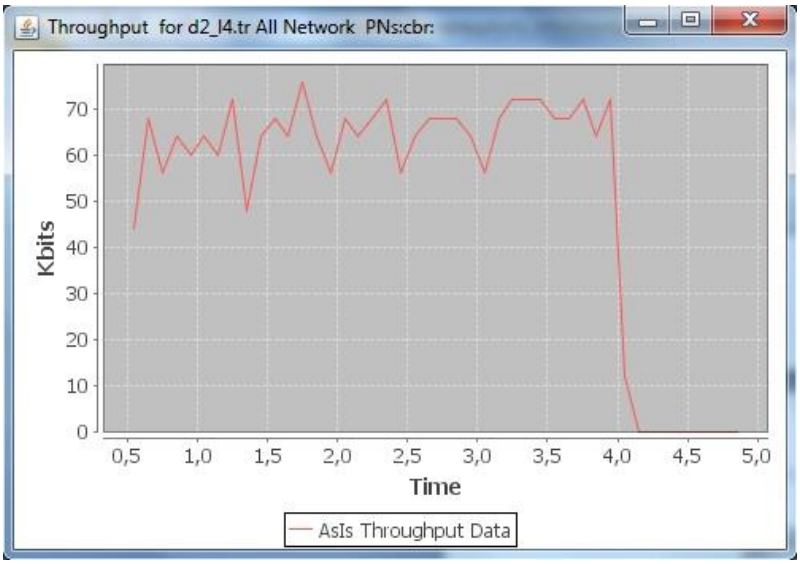
μικρές αλλαγές στην τιμή του να μην γίνονται αντιληπτές όπως αναφέρεται στο [10]. Ο υπολογισμός που πραγματοποιεί η εφαρμογή είναι ο εξής:

Για κάθε χρονική στιγμή t που συνέβη οποιοδήποτε γεγονός στο δίκτυο προσομοίωσης υπολογίζεται το άθροισμα των bytes που στάλθηκαν και παρελήφθησαν επιτυχώς, μετατρέπεται το αποτέλεσμα σε μονάδες bits και στη συνέχεια διαιρείται με το χρόνο t – $t_{έναρξης\ αποστολής\ δεδομένων}$. Στη συνέχεια αυτές οι τιμές εισάγονται σε ένα διάγραμμα συνεχούς γραμμής(Line Chart) όπως αυτό της εικόνας 5.



ΕΙΚΟΝΑ 5: ΠΑΡΑΔΕΙΓΜΑ ΓΡΑΦΗΜΑΤΟΣ ΤΟΥ THROUGHPUT

Ο δεύτερος τύπος διαγράμματος ονομάζεται AsIs Throughput και απεικονίζει την πραγματική τιμή του Throughput υπολογίζοντας το μέσο όρο της απόδοσης του δικτύου ανά 0.1 δευτερόλεπτα. Ο υπολογισμός που εκτελείτε ξεκινάει από το $t_{έναρξης\ αποστολής\ δεδομένων}$ έως το $t_{έναρξης\ αποστολής\ δεδομένων}$ +0.1s και για αυτό το διάστημα υπολογίζεται το σύνολο των bits που στάλθηκαν και διαιρείτε με το 0.1s, αυτό συνεχίζεται έως την τελευταία χρονική στιγμή της προσομοίωσης. Η κάθε μία από αυτές τις τιμές εισάγεται σε ένα διάγραμμα συνεχούς γραμμής αλλά αντιστοιχίζεται στη μέση της χρονικής στιγμής έναρξης και λήξης της μέτρησης(t + t + 0.1 / 2). Στην εικόνα 6 φαίνεται ένα παράδειγμα του AsIs διαγράμματος για το throughput.



ΕΙΚΟΝΑ 6: ΠΑΡΑΔΕΙΓΜΑ ΓΡΑΦΗΜΑΤΟΣ ΤΟΥ ASIS THROUGHPUT

Για την εμφάνιση των διαγραμμάτων αυτών για κάποιο συγκεκριμένο κόμβο του δικτύου συγκεντρώνονται όλα τα διακριτά τα οποία δείχνουν λήψη πακέτων για τα οποία τελικός προορισμός είναι ο συγκεκριμένος κόμβος.

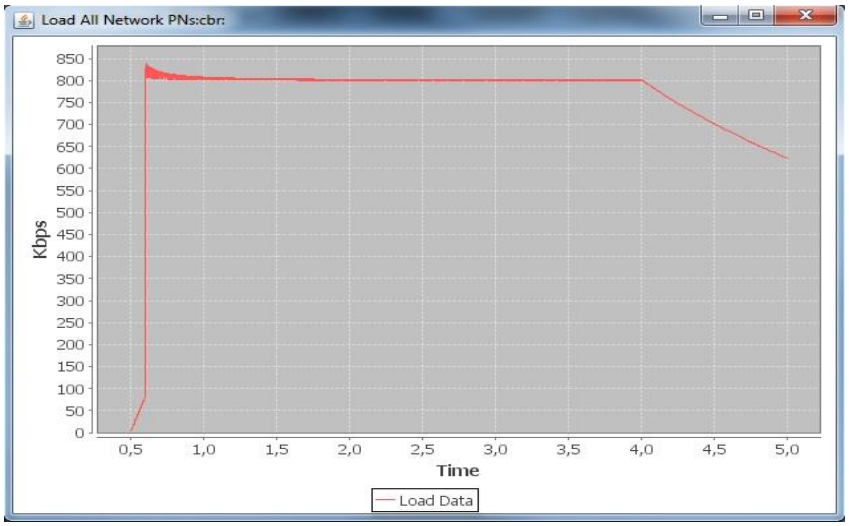
3.3.2 Load

Το Load ή στα ελληνικά φόρτος στην επιστήμη δικτύων είναι μία μέτρηση που αναφέρεται στο σύνολο των δεδομένων που καλείται ένα δίκτυο να μεταφέρει. Σε αντίθεση με την διεκπαιραιωτική ικανότητα ενός δικτύου(Throughput) που αναλύσαμε στην προηγούμενη ενότητα ως φόρτο συμπεριλαμβάνονται όλα τα δεδομένα που εισήλθαν στο δίκτυο άσχετα αν αυτά στάλθηκαν επιτυχώς ή αν απορρίφτηκαν ή χάθηκαν κάποια στιγμή κατά τη μεταφορά.

Και εδώ η εφαρμογή προσφέρει τη δυνατότητα εμφάνισης δύο τύπων διαγραμμάτων. Για τον υπολογισμό του Time Average και του AsIs φόρτου του δικτύου η εφαρμογή ακολουθεί ακριβώς την ίδια μαθηματική συνάρτηση με τις αντίστοιχη που αναφέρθηκε στην προηγούμενη ενότητα (Throughput) με τη μόνη διαφορά τα δεδομένα πάνω στα οποία γίνονται οι υπολογισμοί. Για τον υπολογισμό του φόρτου μετράμε το σύνολο των δεδομένων που προστέθηκαν(+ Enqueue) στις ουρές των κόμβων, σε διακριτά γεγονότα που ακολουθούν το Normal Trace Format, καθώς επίσης και όλα τα αποσταλμένα πακέτα(s Sent), για τα διακριτά γεγονότα που ακολουθούν κάποιο από τα OldWireless και

NewWireless Formats. Για τον υπολογισμό του Load δεν συνυπολογίζονται τα πακέτα που προωθούνται σε ενδιάμεσους κόμβους, και αυτό επιτιγχάνεται με ένα έλεγχο σε κάθε διακριτό γεγονός εάν το node είναι ίδιο με την Ip address του αποστολέα.

Η διαφορετική αυτή προσέγγιση στο φιλτράρισμα των διακριτών γεγονότων για τον υπολογισμό της ίδιας μέτρησης καθίσταται αναγκαία καθώς υπάρχουν βασικές διαφορές μεταξύ τους. Ενώ το Normal Trace Format καταγράφει όλα τα πακέτα και τις χρονικές στιγμές που αυτά εισάγονται σε μία ουρά αποστολής τα άλλα δύο Wireless Formats καταγράφουν μόνο τη χρονική στιγμή που αυτά στάλθηκαν στο δίκτυο. Στην εικόνα 7 φαίνεται ένα παράδειγμα του Time Average Load γραφήματος.



ΕΙΚΟΝΑ 7: ΠΑΡΑΔΕΙΓΜΑ ΓΡΑΦΗΜΑΤΟΣ ΤΟΥ LOAD

Για την εμφάνιση των διαγραμμάτων αυτών για κάποιο συγκεκριμένο κόμβο του δικτύου, συγκεντρώνονται όλα τα διακριτά γεγονότα τα οποία δείχνουν είσοδο στην ουρά αποστολής (Normal Format) και αποστολή πακέτων (Old Wireless/ New Wireless Format), και ο κόμβος αποστολέας είναι ο συγκεκριμένος κόμβος.

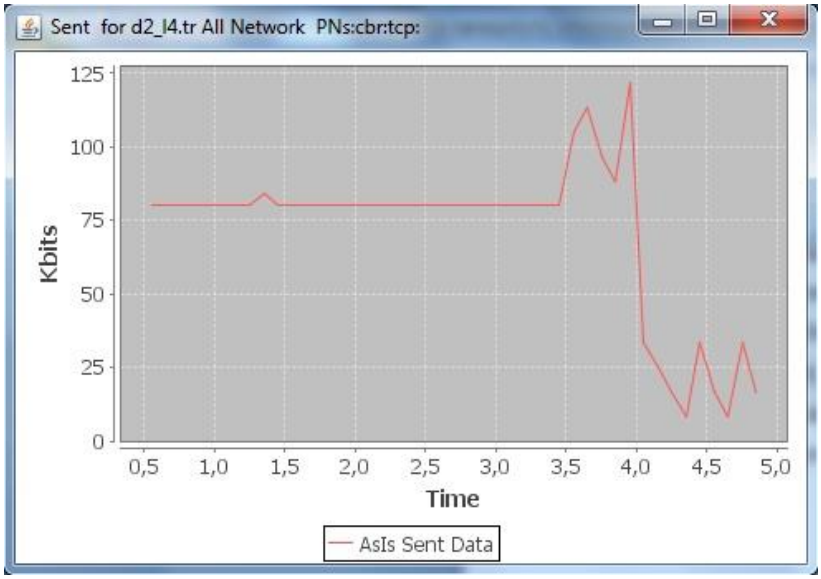
3.3.3 Sent Packets

Μια άλλη ενδιαφέρουσα μέτρηση είναι αυτή των πακέτων που στάλθηκαν από οποιονδήποτε κόμβο του δικτύου κατά τη διάρκεια της προσομοίωσης. Και εδώ

δεν έχει καμία σημασία ο έλεγχος αν αυτά τα πακέτα στάλθηκαν επιτυχώς ή αν απορρίφτηκαν ή χάθηκαν κάποια στιγμή κατά τη μεταφορά.

Η μέτρηση αυτή χαρακτηρίζεται ενδιαφέρουσα καθώς σε συνδυασμό με την μέτρηση του φόρτου δικτύου όπως αυτός αναλύθηκε στην προηγούμενη ενότητα, μπορούν να εξαχθούν συμπεράσματα σχετικά με την κατάσταση των ουρών αποστολής του κάθε κόμβου κατά τη διάρκεια της προσομοίωσης. Στην πραγματικότητα λόγο της διαφορετικής προσέγγισης που ακολουθούν τα Trace Formats σχετικά με την καταγραφή ή μη καταγραφή ως διακριτό γεγονός την είσοδο ενός πακέτου στην ουρά αποστολής ενός κόμβου, σημαντικά συμπεράσματα μπορούν να εξαχθούν μόνο για τοπολογίες ενσύρματων δικτύων ή για συγκεκριμένους κόμβους ενός δικτύου που διατηρούν ενσύρματες συνδέσεις άσχετα αν το δίκτυο συμπεριλαμβάνει και ασύρματες τοπολογίες.

Για τον υπολογισμό και την εμφάνιση των Time Average και AsIs διαγραμμάτων η εφαρμογή χρησιμοποιεί ακριβώς την ίδια μαθηματική συνάρτηση με τις δύο προηγούμενες ενότητες. Και εδώ την διαφορά κάνει το διαφορετικό σύνολο διακριτών γεγονότων που χρησιμοποιούνται για τον υπολογισμό της τιμής αυτής σε κάθε χρονική στιγμή. Συγκεκριμένα λαμβάνονται υπόψη μόνο τα διακριτά γεγονότα τα οποία αντιστοιχούν σε αποστολή πακέτου, δηλαδή Dequeue(-) για το Normal Trace Format, και Sent (s) για τα δύο ασύρματα Trace Formats αντίστοιχα. Ένα διάγραμμα αυτού του τύπου φαίνεται στην εικόνα 8.



Εικονα 8: Παραδειγμα γραφηματος των sent packets

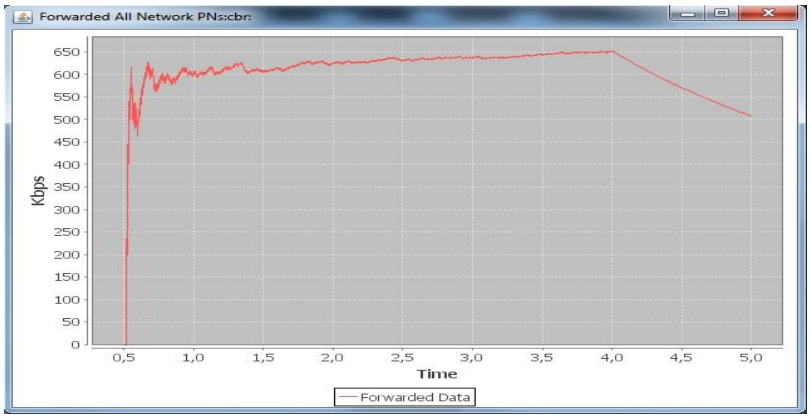
3.3.4 Forwarded Packets

Στα δίκτυα υπολογιστών προωθούμενα πακέτα ονομάζονται τα πακέτα αυτά που μεταβιβάζονται από έναν κόμβο σε έναν ή περισσότερους άλλους κόμβους μέσω των συνδέσεων του τρέχοντος δικτύου. Υπάρχουν διάφορα μοντέλα προώθησης πακέτων με τα πιο σημαντικά να είναι το μοντέλο μονοδιανομής(unicast) το μοντέλο πολυδιανομής(multicast) και τέλος η εκπομπή(broadcast). [9],[10]

Το Μοντέλο Μονοδιανομής (unicast) είναι το απλούστερο εκ των τριών μοντέλων και περιλαμβάνει τη μεταγωγή ενός πακέτου από σύνδεση σε σύνδεση, με αλυσιδωτό τρόπο, από τον αποστολέα έως τον μοναδικό παραλήπτη. [10]

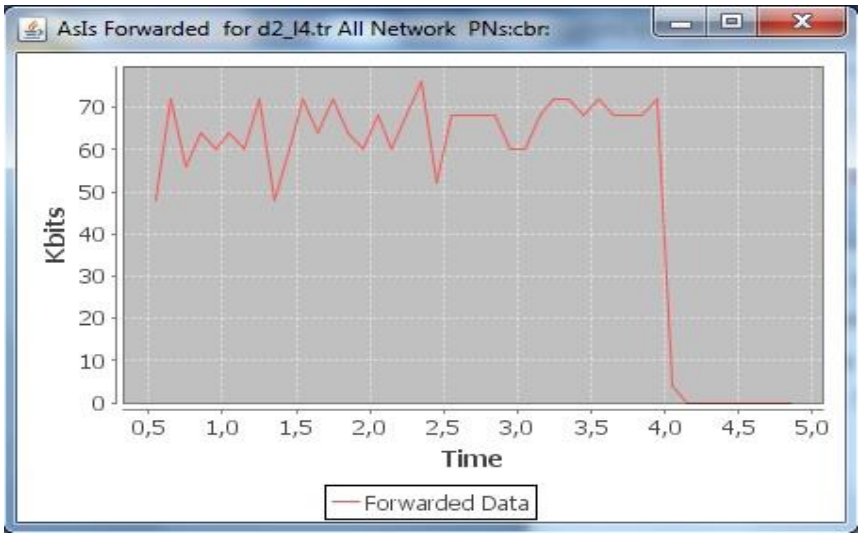
Το Μοντέλο πολυδιανομής (multicast) χρησιμοποιείται όταν ένα πακέτο πρέπει να αποσταλεί σε ένα υποσύνολο-ομάδα κόμβων εντός του τρέχοντος δικτύου. Αν το υποσύνολο των κόμβων αυτών είναι ίσο με το σύνολο των κόμβων του δικτύου, ονομάζεται εκπομπή (Broadcast). [10]

Ο υπολογισμός των Time Average και AsIs τιμών που εμφανίζονται στα αντίστοιχα διαγράμματα γίνεται ακριβώς με τον ίδιο τρόπο που υπολογίζεται και το Throughput,Sent Packets και Load των προηγούμενων ενοτήτων. Και εδώ την διαφορά κάνει το διαφορετικό σύνολο διακριτών γεγονότων που χρησιμοποιούνται για τον υπολογισμό της τιμής αυτής σε κάθε χρονική στιγμή. Συγκεκριμένα λαμβάνονται υπόψη μόνο τα διακριτά γεγονότα τα οποία αντιστοιχούν σε αποστολή πακέτου, δηλαδή Dequeue (-) για το Normal Trace Format όπου το αναγνωριστικό του κόμβου αποστολέα είναι διαφορετικό από την διεύθυνση του αποστολέα του πακέτου, και Forward (f) για τα δύο ασύρματα Trace Formats αντίστοιχα. Ένα παράδειγμα Time Average διαγράμματος φαίνεται στην εικόνα 9.



ΕΙΚΟΝΑ 9: ΠΑΡΑΔΕΙΓΜΑ TIME AVERAGE ΓΡΑΦΗΜΑΤΟΣ ΤΩΝ FORWARDED PACKETS

Ενώ το αντίστοιχο AsIs διάγραμμα είναι αυτό της εικόνας 10.



ΕΙΚΟΝΑ 10 : ΠΑΡΑΔΕΙΓΜΑ ASIS ΓΡΑΦΗΜΑΤΟΣ ΤΩΝ FORWARDED PACKETS

Το διάγραμμα αυτό εμφανίζει αξιόπιστα συμπεράσματα και για τα τρία μοντέλα προώθησης πακέτων.

3.3.5 Dropped Packets

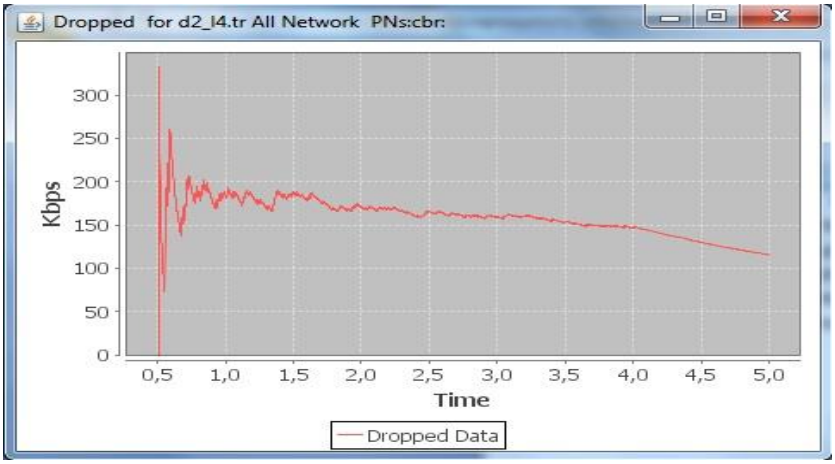
Όταν ένα πακέτο αποστέλλεται από ένα κόμβο του δικτύου σε κάποιον άλλο, υπάρχει η πιθανότητα το πακέτο αυτό να απορριφθεί από κάποιο ενδιάμεσο κόμβο του δικτύου ή ακόμα και από τον ίδιο τον κόμβο προορισμού. Οι αιτίες που μπορούν να προκαλέσουν το φαινόμενο αυτό είναι πολλές [12]. Κάποιες από τις αιτίες αυτές είναι οι εξής:

- ✓ υποβάθμιση του σήματος, φυσικό επίπεδο
- ✓ συμφόρηση
- ✓ κατεστραμμένα πακέτα
- ✓ βλάβη στο υλικό δικτύωσης
- ✓ απλών πρωτοκόλλων δρομολόγησης, όπως το DSR(Dynamic Source Routing)

Εάν το φαινόμενο αυτό είναι έντονο σε ένα δίκτυο τότε επηρεάζεται η αποδοτικότητα του δικτύου καθώς μπορεί να εμφανιστεί μείωση του Throughput, και αύξηση του Jitter που αποτελεί σημαντικό πρόβλημα για ευαίσθητες εφαρμογές όπως streaming εφαρμογές, voice over IP, online gaming, συνδιασκέψεις κ.α. [10]

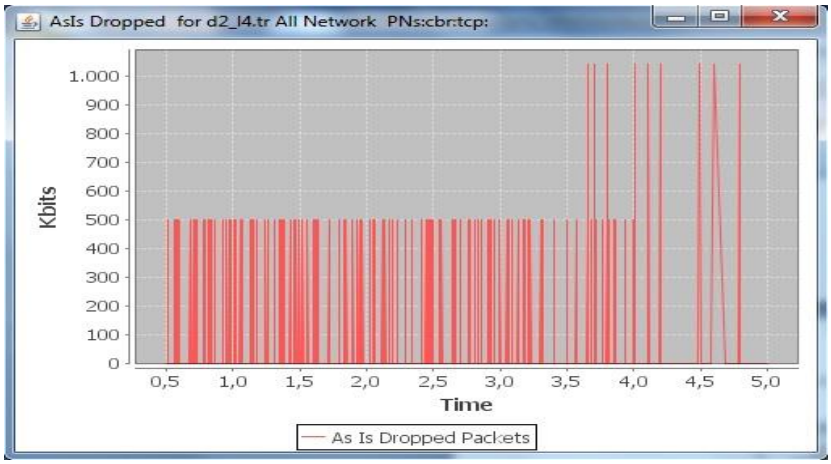
Η μονάδα μέτρησης που χρησιμοποιείται είναι kbps και εμφανίζεται σε γράφημα συνεχούς γραμμής τύπου Time Average και AsIs. Για τον υπολογισμό αυτής της μέτρησης για κάθε χρονική στιγμή της προσομοίωσης ακολουθείτε η ίδια διαδικασία με τα προηγούμενα μέτρα αξιολόγησης του δικτύου που αναλύθηκαν.

Τα διακριτά γεγονότα του Trace αρχείου που λαμβάνονται υπόψη στον υπολογισμό αυτό είναι αυτά των οποίων το πεδίο Abbreviation έχει τιμή d ή D για όλους τους τύπους Trace που υποστηρίζει η εφαρμογή. Στην εικόνα 11 φαίνεται ως παράδειγμα το γράφημα Time Average Dropped Packets ενός δικτύου.



Εικονα 11: Παραδειγμα γραφηματος των dropped packets

Ενώ στην εικόνα 12 φαίνεται ένα παράδειγμα του AsIs Dropped γραφήματος ενός δικτύου.



Εικονα 12: Παραδειγμα γραφηματος των asis dropped packets

3.3.6 EndToEnd Delay

Με τον όρο EndToEnd Delay ορίζεται ο χρόνος που χρειάστηκε κάποιο πακέτο να φτάσει από τον αποστολέα στον παραλήπτη [12]. Υπολογίζεται σύμφωνα με την παρακάτω μαθηματική σχέση :

$$d_{end\text{-}end} = N[\, d_{trans} + d_{prop} + d_{proc}] \ ,$$

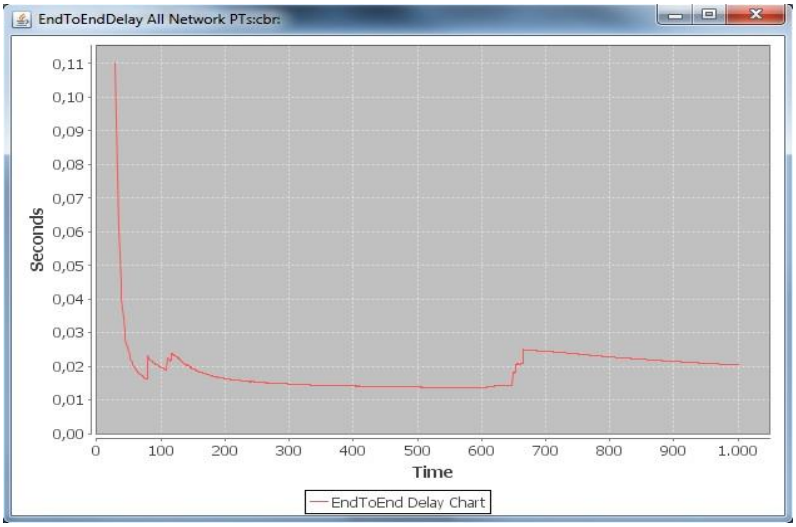
όπου

$d_{end\text{-}end}$= end-to-end delay,
d_{trans}= transmission delay,
d_{prop}= propagation delay,
d_{proc}= processing delay,
N= number of links (Number of routers + 1).

Η εφαρμογή jTrcezer για να υπολογίσει την καθυστέρηση ενός πακέτου χρησιμοποιεί τη χρονική στιγμή αποστολής του πακέτου από τον αρχικό αποστολέα και τη χρονική στιγμή λήψης του από τον τελικό προορισμό και τα αφαιρεί. Η καθυστέρηση αυτή μπορεί να είναι μόνο θετική ή ίση του μηδενός.

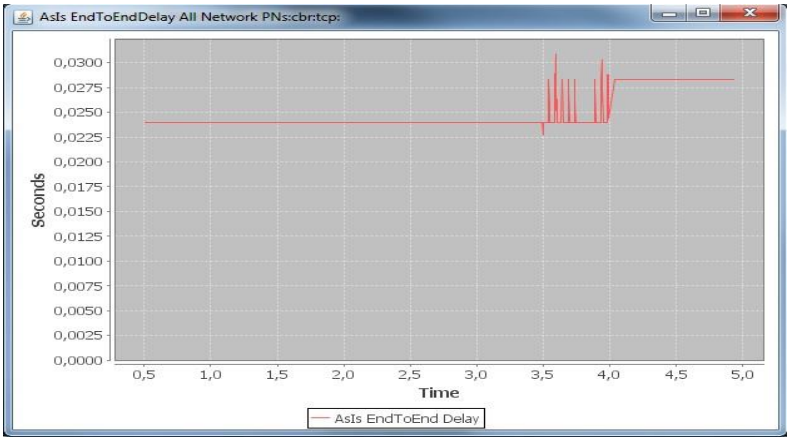
Η τιμή της καθυστέρησης κάθε πακέτου εισάγεται στο AsIs γράφημα, στην χρονική στιγμή αποστολής του κάθε πακέτου και όχι την χρονική στιγμή λήψης του. Η αντιστοίχηση της καθυστέρησης με το χρόνο αποστολή γίνεται για να είναι πιο εύκολη η εξαγωγή συμπερασμάτων από το διάγραμμα καθυστέρησης.

Όσο αναφορά το Time Average γράφημα αυτού του μέτρου αξιολόγησης, υπολογίζεται και εμφανίζεται με τον ίδιο τρόπο όπως τα Time Average γραφήματα των προηγούμενων μέτρων αξιολόγησης που αναλύθηκαν. Στην εικόνα 13 φαίνεται ένα παράδειγμα αυτού του γραφήματος.



Εικονα 13: Παραδειγμα γραφηματος της καθυστερηση ληψης των πακετων

Μια ιδιαιτερότητα που υπάρχει στο μέτρο αυτό, είναι οι περιπτώσεις που κάποια πακέτα δεν λαμβάνονται ποτέ από τον τελικό προορισμό. Στην περίπτωση αυτή δεν μπορούμε να υπολογίσουμε την καθυστέρηση του πακέτου αφού δεν υπάρχει χρόνος παραλαβής. Τις περιπτώσεις αυτές τις αγνοούμε και δεν τις εμφανίζουμε στο γράφημα του άκρου προς άκρου καθυστέρησης. Στην εικόνα 14 φαίνεται ένα παράδειγμα αυτού του γραφήματος.



ΕΙΚΟΝΑ 14: ΠΑΡΑΔΕΙΓΜΑ ΓΡΑΦΗΜΑΤΟΣ ΤΗΣ ASIS ΚΑΘΥΣΤΕΡΗΣΗΣ ΛΗΨΗΣ ΤΩΝ ΠΑΚΕΤΩΝ

3.3.7 Jitter

Ως Jitter ορίζεται μια μέτρηση που δηλώνει την μεταβολή της καθυστέρησης λήψης των πακέτων. Πιο συγκεκριμένα η διαφορά της καθυστέρησης δύο διαδοχικών πακέτων της ίδιας ροής δεδομένων ονομάζεται jitter.Μπορούν να προκύψουν μεταβολές στην τιμή αυτή κυρίως λόγο συμφόρησης του δικτύου, λάθος εισαγωγής πακέτου σε ουρά, ή κάποιας αλλαγής στις παραμέτρους του δικτύου σε οποιοδήποτε επίπεδο. [10]

Για τον προσδιορισμό της ροής δεδομένων στην οποία ανήκει κάθε πακέτο, υπάρχει ανάλογο πεδίο σε κάθε διακριτό γεγονός του Trace αρχείου τύπου Normal ή NewWireless. Για τον υπολογισμό του jitter σε Trace αρχεία τύπου OldWireless όπου δεν καταγράφεται κάποιο αναγνωριστικό της ροής δεδομένων στην οποία ανήκει κάθε πακέτο, ορίστηκε ότι μια ροή δεδομένων σύμφωνα με τα παρακάτω:

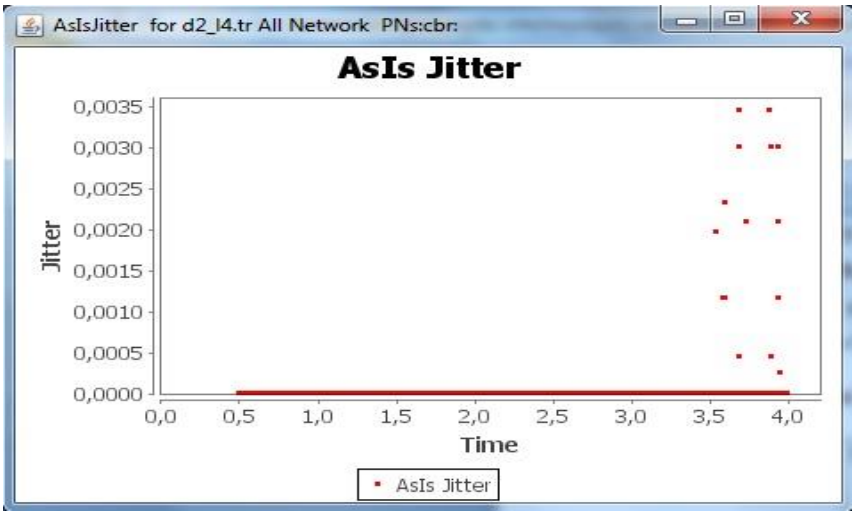
- ✓ Διεύθυνση IP αποστολέα
- ✓ Αριθμός πόρτας αποστολέα
- ✓ Διεύθυνση IP προορισμού

- ✓ Αριθμός πόρτας προορισμού
- ✓ Πρωτόκολλο πακέτου

Το jitter είναι πάντα θετικό, άλλωστε δηλώνει μεταβολή τιμών. Το γράφημα που εμφανίζεται είναι απλών σημείων(Dot plot) και όχι συνεχούς γραμμής όπως τα προηγούμενα γραφήματα που περιγράφτηκαν. Η διαδικασία υπολογισμού των τιμών αυτών των σημείων ακολουθεί την εξής ροή:

1. Εύρεση όλων των διαφορετικών ροών δεδομένων που υπάρχουν στο τρέχων δίκτυο και ισχύουν όλοι περιορισμοί που πρόσθεσε ο χρήστης κατά την διαδικασία ρύθμισης των φίλτρων.
2. Για κάθε μία από τις παραπάνω ροές δεδομένων υπολογίζονται τα EndToEnd Delay των πακέτων της εκάστοτε ροής, και στη συνέχεια προστίθενται σε μια δομή δεδομένων οι διάφορες τιμές jitter.
3. Τέλος εμφανίζονται όλες οι τιμές jitter όλων των ροών δεδομένων του δικτύου στο ίδιο γράφημα απλών σημείων.

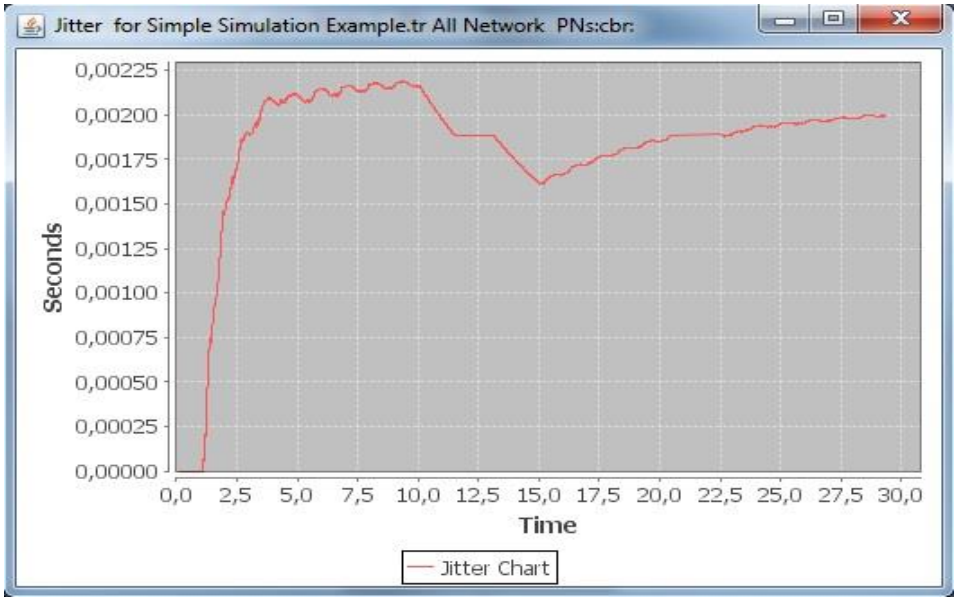
Στην εικόνα 15 φαίνεται ένα παράδειγμα αυτού του γραφήματος.



ΕΙΚΟΝΑ 15: ΠΑΡΑΔΕΙΓΜΑ ΓΡΑΦΗΜΑΤΟΣ ΤΟΥ ASIS JITTER

Το αντίστοιχο Time Average Jitter εμφανίζεται σε διάγραμμα ευθείας συνεχούς γραμμής και σε κάθε χρονική στιγμή το jitter υπολογίζεται αθροίζοντας όλα τα jitter όλων των επιλεγμένων ροών δεδομένων μέχρι εκείνη τη στιγμή και διαιρόντας το με το πλήθος των jitter που προστέθηκαν.

Στην εικόνα 16 φαίνεται ένα παράδειγμα αυτού του γραφήματος.



ΕΙΚΟΝΑ 16 : ΠΑΡΑΔΕΙΓΜΑ ΓΡΑΦΗΜΑΤΟΣ TIME AVERAGE JITTER

3.4 Βιβλιοθήκη Γραφημάτων

3.4.1 jFreeChart

Για την δημιουργία των γραφημάτων, χρησιμοποιήθηκε η ανοιχτού κώδικα βιβλιοθήκη jFreeChart 1.0.13 που συνοδεύεται με άδεια χρήσης GNU Lesser General Public.

Η βιβλιοθήκη αυτή επιλέχθηκε για δύο λόγους:

1. Κώδικας γραμμένος σε καθαρή Java
2. Λειτουργίες που υποστηρίζει

Συγκεκριμένα η δυνατότητα zoom In/Out σε επιλεγμένες από το χρήστη περιοχές που παρέχει η βιβλιοθήκη αποτελεί βασική απαίτηση για την ευκολότερη ανάλυση των διαγραμμάτων που εμφανίζει η εφαρμογή. Επιπρόσθετα προσφέρει την δυνατότητα επεξεργασίας του LookAndFeel των γραφημάτων δηλαδή των χρωμάτων των τίτλων που εμφανίζονται στους άξονες ή τον γενικό τίτλο του γραφήματος κ.α.

4 Σενάριο Καλής λειτουργίας

4.1 Εισαγωγή

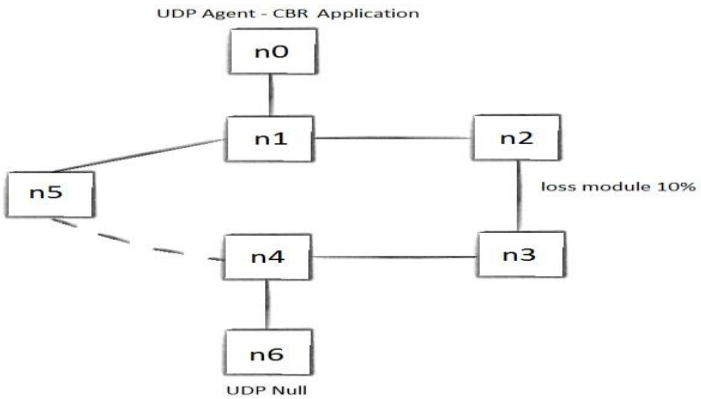
Στο κεφάλαιο αυτό θα αναλύσουμε τα αποτελέσματα της εφαρμογής jTrcezer σε ένα δοκιμαστικό σενάριο. Με τον τρόπο αυτό θα δοθεί ένα δείγμα για το πώς μπορεί να χρησιμοποιηθεί η εφαρμογή για την εξαγωγή χρήσιμων συμπερασμάτων καθώς επίσης θα αποδειχθεί και η εγκυρότητα των αποτελεσμάτων της εφαρμογής. Ο κώδικας TCL που εκφράζει το σενάριο της προσομοίωσης είναι διαθέσιμος στο παράρτημα Α.

4.2 Σενάριο Προσομοίωσης

Σχεδιάσαμε ένα δίκτυο που αποτελείται από 7 κόμβους n0,n1,n2,n3,n4,n5,n6, και έξι αρχικές συνδέσεις χωρητικότητας 2Mb οι οποίες συνδέουν τους παρακάτω κόμβους μεταξύ τους :

- ✓ Κόμβος n0 – Κόμβος n1
- ✓ Κόμβος n1 – Κόμβος n2
- ✓ Κόμβος n2 – Κόμβος n3
- ✓ Κόμβος n3 – Κόμβος n4
- ✓ Κόμβος n4 – Κόμβος n5
- ✓ Κόμβος n4 – Κόμβος n6

Στην σύνδεση n2-n3 προστέθηκε και ένα μοντέλο σφαλμάτων με ποσοστό 10%. Ο αλγόριθμος δρομολόγησης που επιλέχθηκε για το δίκτυο είναι ο rtProto/DV Το δίκτυο φαίνεται στην εικόνα 17.



ΕΙΚΟΝΑ 17 : ΤΟΠΟΛΟΓΙΑ ΔΙΚΤΥΟΥ ΓΙΑ ΤΟ ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Επίσης συνδέσαμε έναν UDPAgent στον κόμβο n0 για την παραγωγή κίνησης στο δίκτυο, καθώς επίσης και έναν UDP null Agent στον κόμβο n6 για την αποδοχή της κίνησης που παράγεται από τον κόμβο n0. Τέλος δημιουργήθηκε και μία εφαρμογή παραγωγής CBR(Constant bit rate) κίνησης από τον κόμβο n0 προς τον κόμβο n6 που επικοινωνεί πάνω στο UDP, με μέγεθος πακέτων 200 bytes.

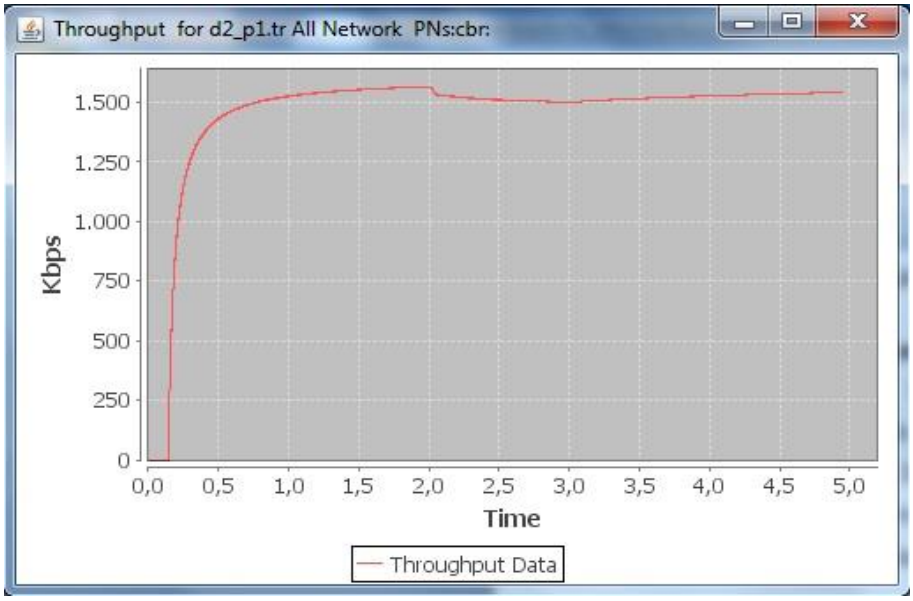
Για να βγάλουμε πιο εμφανή και εύκολα συγκρίσιμα αποτελέσματα προγραμματίσαμε το δίκτυο να εκτελέσει τα παρακάτω :

- ✓ Στη χρονική στιγμή 0.1 να ξεκινήσει την κίνηση CBR
- ✓ Στη χρονική στιγμή 2.0 να πέσει η σύνδεση Κόμβου n4 - Κόμβου n5
- ✓ Στη χρονική στιγμή 3.0 να επανέλθει η σύνδεση Κόμβου n4 - Κόμβου n5
- ✓ Στη χρονική στιγμή 4.9 να σταματήσει την κίνηση CBR

Ο κώδικας του σεναρίου είναι διαθέσιμος στο Παράρτημα 1.

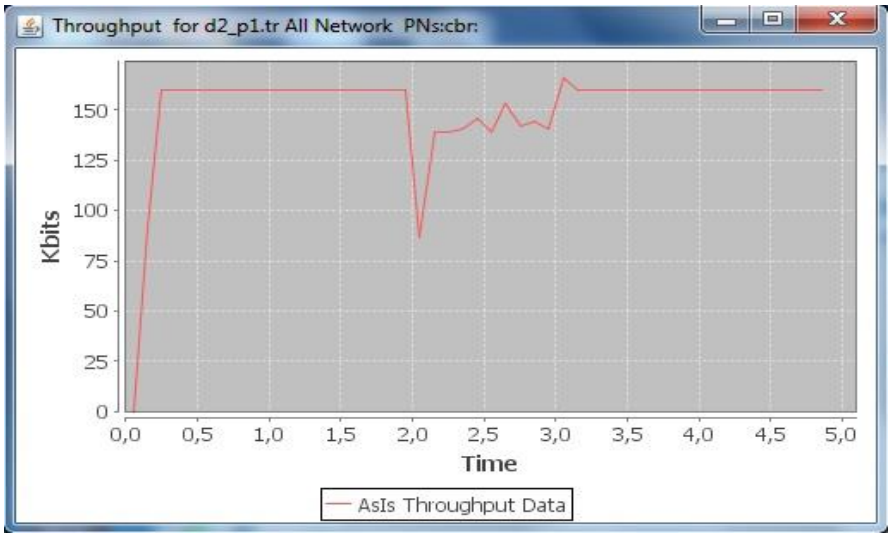
4.2.1 Εμφάνιση Γραφημάτων

Σύμφωνα με την τοπολογία και τις ρυθμίσεις που ορίσαμε στο δίκτυο προσομοίωσης είναι αναμενόμενες κάποιες αλλαγές των μετρήσεων κατά το διάστημα 2.0 sec – 3.0 sec της διάρκειας προσομοίωσης. Στην εικόνα 18 φαίνεται μια μικρή πτώση του Throughput κατά τι διάρκεια που η σύνδεση μεταξύ των κόμβων n4-n5 είναι μη διαθέσιμη.



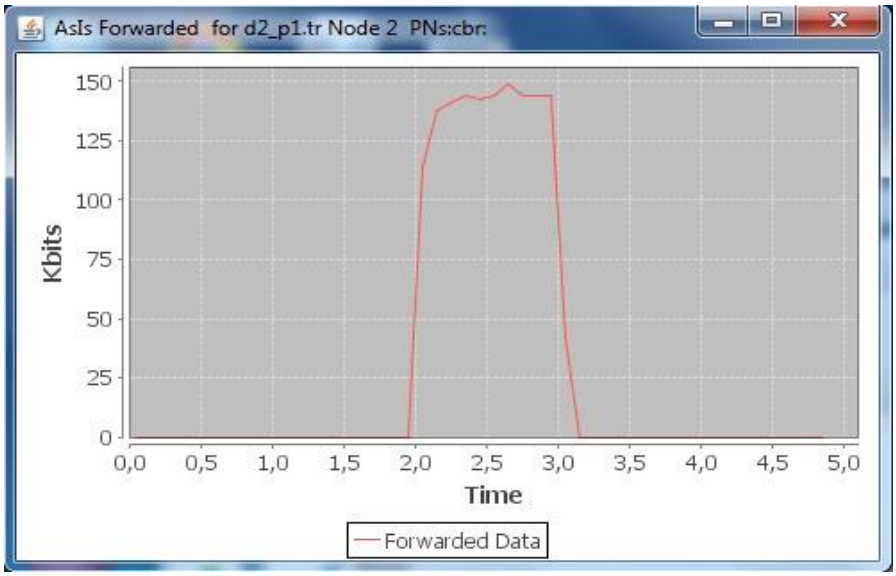
ΕΙΚΟΝΑ 18 : TIME AVERAGE THROUGHPUT - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Για να δούμε την μεταβολή του Throughput με μεγαλύτερη ακρίβεια καλύτερα ας κοιτάξουμε το ASIS διάγραμμα του Throughput που φαίνεται στην εικόνα 19.



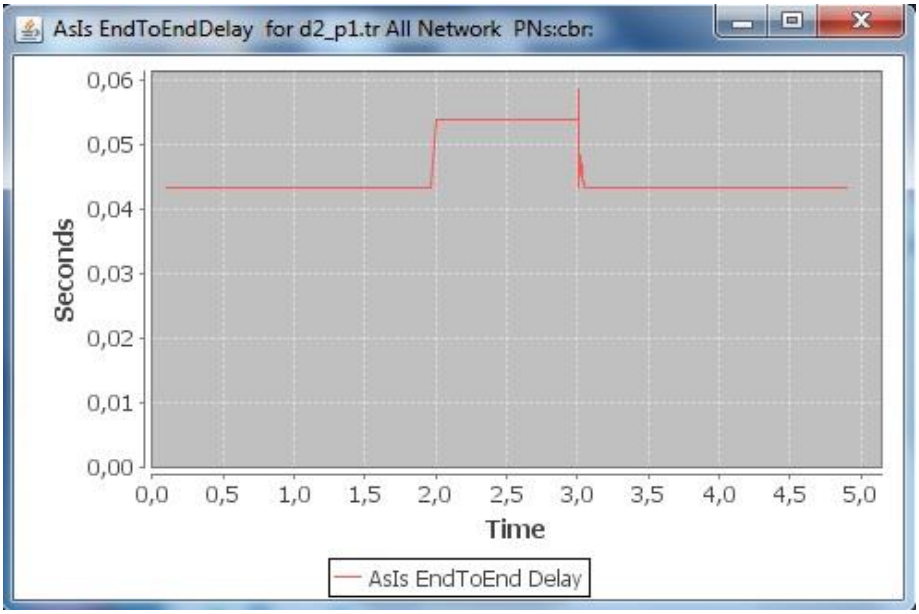
ΕΙΚΟΝΑ 19 : ASIS THROUGHPUT - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Βλέπουμε ότι όλη το δίκτυο χρησιμοποιούσε τη διαδρομή n0-n5-n4-n6 για την αποστολή δεδομένων, κάτι που μπορούμε να το επαληθεύσουμε αν κοιτάξουμε τι δεδομένα προωθούσε ο κόμβος n2 πριν πέσει η σύνδεση n4-n5 κάτι που όντως επιβεβαιώνεται από την εικόνα 20.



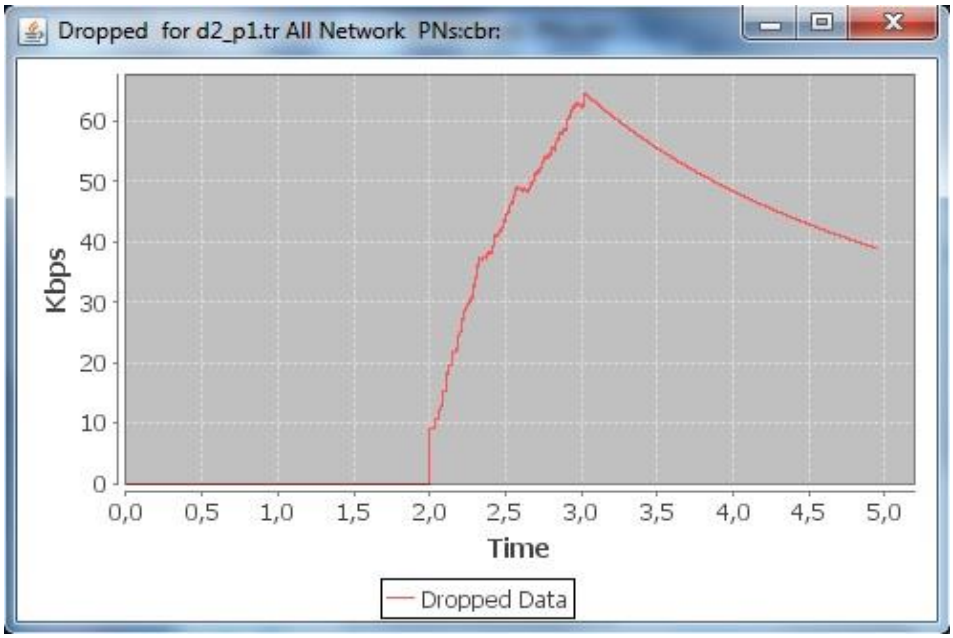
ΕΙΚΟΝΑ 20 : TIME AVERAGE FORWARDED PACKETS - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Μια αλλαγή όπως αυτή της πτώσης μιας σύνδεσης και αλλαγής της διαδρομής που ακολουθούν τα πακέτα περιμένουμε να προκαλέσει και κάποιες αλλαγές στην καθυστέρηση λήψης των πακέτων αυτών, κάτι που συνεπάγεται και αύξηση του jitter. Κατά το διάστημα 2.0-3.0 τα πακέτα πρέπει να ακολουθήσουν την μοναδική διαδρομή για να φτάσουν στον προορισμό τους κάτι που σημαίνει ότι πρέπει να αποσταλούν από τον κόμβο n0 και να προωθηθούν από τους κόμβους n1,n2,n3,n4 ώστε τελικά να παραληφτούν από τον κόμβο n6 κάτι που αυξάνει το από άκρο σε άκρο καθυστέρηση των πακέτων όπως δείχνει η εικόνα 21.



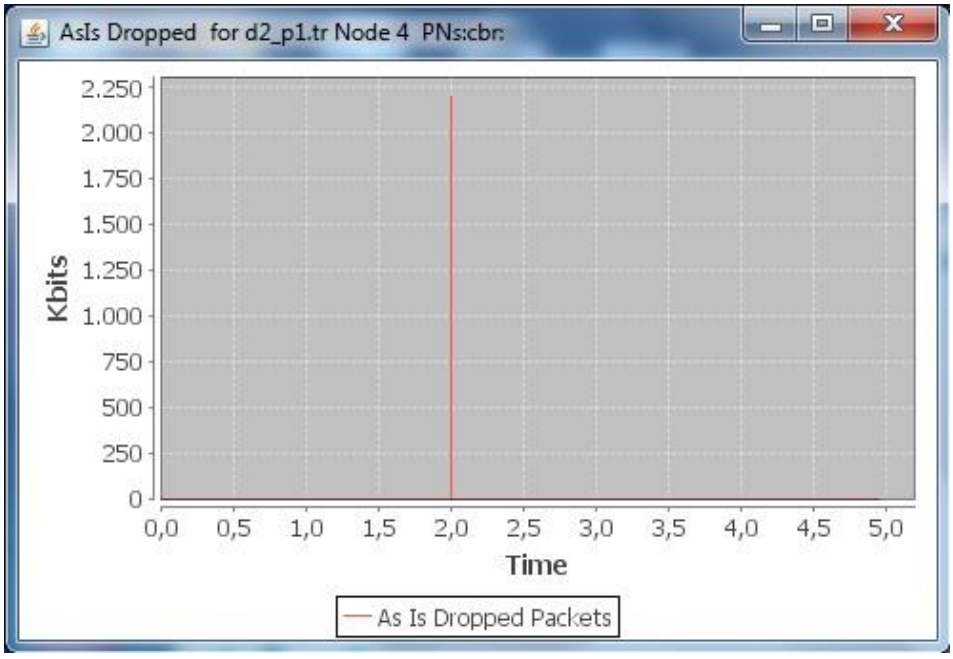
Εικονα 21 : Asis end-to-end delay - σεναριο καλης λειτουργιας

Μια άλλη συνέπεια της πτώσης της σύνδεσης αυτής είναι η απώλεια κάποιων πακέτων που είτε βρίσκονταν ήδη στο δίκτυο κατά τη διάρκεια της πτώσης της σύνδεσης είτε κατά το χρόνο ενημέρωσης του κόμβου n1 σχετικά με το γεγονός αυτό και προώθησης των πακέτων προς τον κόμβο 2. Επίσης στην νέα διαδρομή τον πακέτων περιλαμβάνεται και η σύνδεση n2-n3 στην οποία έχουμε προσθέσει ένα μοντέλο εμφάνισης σφαλμάτων με ποσοστό 10%, επομένως περιμένουμε να αυξηθούν οι απώλειες όσο τα πακέτα χρησιμοποιούν αυτή τη σύνδεση. Στην εικόνα 22, φαίνεται το Time Average διάγραμμα απορριφθέντων πακέτων για όλο το δίκτυο.



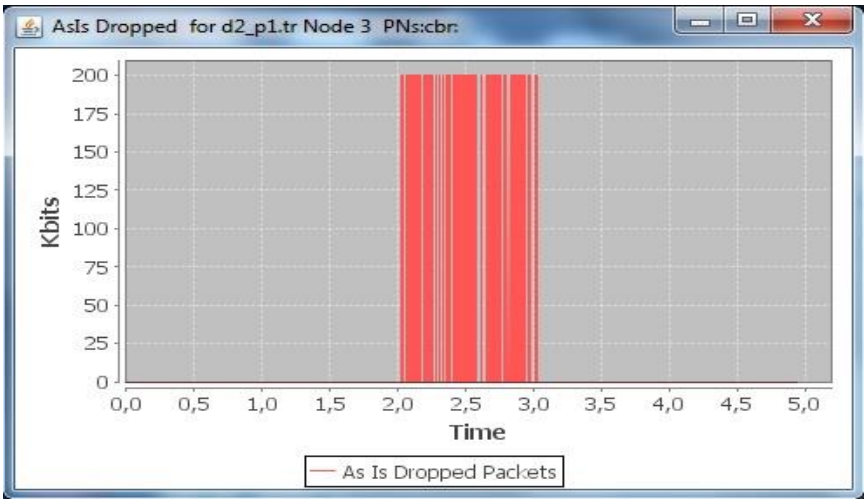
ΕΙΚΟΝΑ 22 : TIME AVERAGE DROPPED PACKETS - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Ενδιαφέρον έχει και το ποιος κόμβος απέρριπτε αυτά τα πακέτα. Στην εικόνα 23 γίνεται εμφανές το γεγονός ότι τη χρονική στιγμή 2.0 απορρίφθηκαν 2.250Kbits από τον κόμβο 4, κάτι που ήταν αναμενόμενο αφού την ίδια χρονική στιγμή το σενάριο είναι προγραμματισμένο να ρίξει την σύνδεση n5-n4.



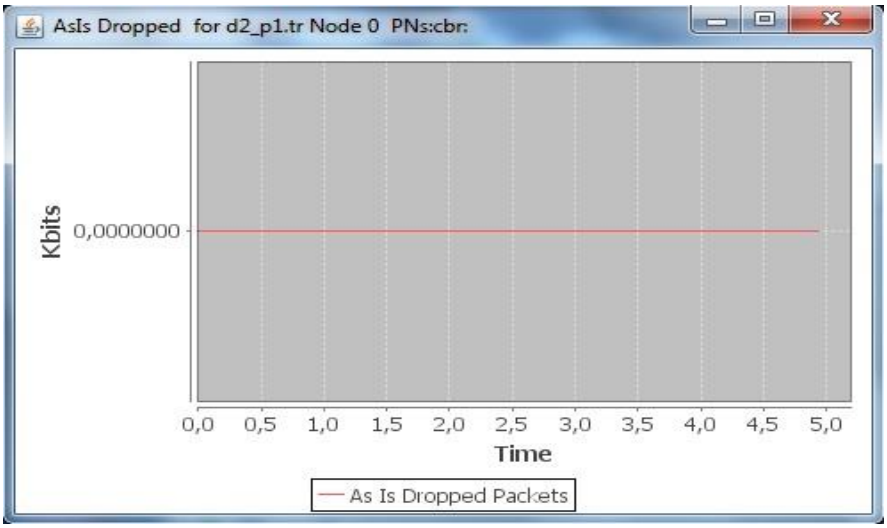
ΕΙΚΟΝΑ 23 : ASIS DROPPED PACKETS ΓΙΑ ΤΟΝ ΚΟΜΒΟ 4 - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Αφού η σύνδεση n5-n4 δεν είναι πλέον διαθέσιμη μετά την χρονική στιγμή 2.0 τα πακέτα αναδρομολογούνται προς την δεύτερη διαθέσιμη διαδρομή, δηλαδή την n0-n1-n2-n3-n4-n6. Όπως όμως αναφέρθηκε και στην προηγούμενη ενότητα στο σενάριο έχει τοποθετηθεί ένα μοντέλο εμφάνισης σφαλμάτων 10% στην σύνδεση n2-n3 την οποία για το διάστημα 2.0 έως 3.0 χρησιμοποιούν τα δεδομένα για να φτάσουν στον προορισμό του. Έτσι το γράφημα της εικόνας 24 σωστά εμφανίζει μόνο για την χρονική αυτή περίοδο αρκετά απορριφθέντα πακέτα από τον κόμβο 3



ΕΙΚΟΝΑ 24 : ASIS DROPPED PACKETS ΓΙΑ ΤΟΝ ΚΟΜΒΟ 3 - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

Αντίθετα για όλους τους υπόλοιπους κόμβους του δικτύου τα γραφήματα που δείχνουν τα απορριφθέντα πακέτα έχουν την τιμή μηδέν καθόλη τη διάρκεια του σεναρίου όμοίως με το γράφημα για τον κόμβο 0 που φαίνεται στην εικόνα 25.



ΕΙΚΟΝΑ 25 : ASIS DROPPED PACKETS ΓΙΑ ΤΟΝ ΚΟΜΒΟ 0 - ΣΕΝΑΡΙΟ ΚΑΛΗΣ ΛΕΙΤΟΥΡΓΙΑΣ

5 Σύγκριση με υπάρχουσες εφαρμογές

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούν εφαρμογές που αναπτύχθηκαν από άλλα πανεπιστήμια και παρέχουν τις ίδιες ή παρόμοιες λειτουργίες με την εφαρμογή της παρούσας πτυχιακής. Θα τονιστούν κάποια πλεονεκτήματα και μειονεκτήματα της κάθε εφαρμογής και θα δοθούν κάποιες σημαντικές πληροφορίες για την κάθε μία.

5.2 Εφαρμογή jTrana

Μία από τις δύο standalone εφαρμογές ανάλυσης και γραφικής αναπαράστασης μέτρων αξιολόγησης δικτύων που προσομοιώθηκαν στον προσομοιωτή NS-2 και υπάρχουν ήδη στο διαδίκτυο είναι το jTrana. Κύριο χαρακτηριστικό της εφαρμογής αυτής είναι η χρήση μιας βάσης δεδομένων, και συγκεκριμένα Mysql που χρησιμοποιείτε ως ενδιάμεσο της ανάγνωση-κατανόησης των δεδομένων με την ανάλυση και γραφική απεικόνιση των μετρικών του δικτύου. Κατά την εγκατάσταση της εφαρμογής δεν εγκαθίσταται αυτόματα η Mysql ούτε δημιουργείτε αυτόματα η βάση δεδομένων την οποία χρησιμοποιεί για την αποθήκευση των διακριτών γεγονότων που διαβάζει από ένα αρχείο Trace.

Η χρήση μιας βάσης δεδομένων ως ενδιάμεσο μπορεί να χαρακτηριστεί μειονέκτημα αλλά συνάμα αποτελεί και πλεονέκτημα. Πολύ μεγάλα ή πολύπλοκα σενάρια προσομοίωσης μπορούν να παράγουν χιλιάδες γραμμές διακριτών γεγονότων, που στην περίπτωση της εφαρμογής jTrcezer τα γεγονότα αυτά πρέπει να αποθηκευτούν ως Objects στην μνήμη RAM του υπολογιστή. Στην περίπτωση αυτή είναι πιθανό να χρειάζεται αύξηση της μνήμης που χρησιμοποιεί το Runtime environment της java για να καλύψει τους πόρους μνήμης που απαιτεί κάποιο Trace αρχείο, καθώς επίσης σε εξαιρετικά μεγάλα σενάρια ενδέχεται να μην μπορούν να καλυφθούν οι πόροι αυτοί. Αυτό το πρόβλημα δεν υπάρχει στην εφαρμογή jTrana καθώς σε μία βάση δεδομένων δεν υπάρχει κάποιο φυσικό μέγιστο όριο στον μέγεθος του Trace αρχείου, αν και όσο μεγαλύτερο είναι αυτό τόσο πιο πολύ επιβαρύνεται η εφαρμογή και η απόδοσή της.

Την εφαρμογή την υλοποίησαν οι Hengheng Qian και ο Weiwei Fang το 2000. Όπως αναφέρουν οι προγραμματιστές στην ιστοσελίδα της εφαρμογής [11], δεν υποστηρίζεται ούτε ενημερώνεται από κάποιο άτομο ή ομάδα, που σημαίνει πως

τυχόν αλλαγές και προσθήκες σε νεότερες εκδόσεις του NS-2 δεν εφαρμόζονται και στην εφαρμογή jTrana, και το σημαντικότερο είναι πως δεν υπάρχει κοινότητα για την επίλυση ή την αναφορά προβλημάτων.

5.3 Εφαρμογή Tracegraph

Μία ακόμη παρόμοια εφαρμογή είναι γνωστή ως Tracegraph. Αναπτύχθηκε από τον Jaroslaw Malek και είναι υλοποιημένη στη γλώσσα προγραμματισμού Matlab. Υποστηρίζει και τα τρία Trace Formats όπως και η εφαρμογή jTrana και η εφαρμογή της πτυχιακής αυτής εργασίας, jTrcezer. Ένα από τα αρνητικά της εφαρμογής είναι ότι δεν υπάρχει μεγάλη ευελιξία στην διεπαφή του χρήστη με τα διαγράμματα καθώς υπάρχουν προβλήματα τόσο υπερκάλυψης πληροφοριών από άλλα component του γραφικού περιβάλλοντος της εφαρμογής όσο και δυνατοτήτων χειρισμού(zoom in/out,move) των διαγραμμάτων που εμφανίζονται. Επίσης μεγάλο μειονέκτημα της εφαρμογής αυτής είναι η δυνατότητα εμφάνισης ενός μόνο διαγράμματος κάθε φορά και η τοποθέτησή του στο main panel της εφαρμογής.

Το Tracegraph δεν περιορίζεται μόνο στις βασικές μετρικές, και προσφέρει ένα σύνολο συνδυασμών των μετρικών στα ίδια διαγράμματα, κάτι που μερικές φορές αλλοιώνει τα αποτελέσματα ή τα αποτελέσματα αυτά είναι άχρηστα για τον χρήστη και δεν τον βοηθούν να βγάλει ασφαλή συμπεράσματα για την κατάσταση του δικτύου του οποιαδήποτε χρονική στιγμή. Επίσης τα 3D διαγράμματα που προσφέρει έχουν κάποια σημασία σε μικρά μόνο δίκτυα με μικρό πλήθος κόμβων. Το βασικό όμως μειονέκτημα είναι ότι έχει τη δυνατότητα να διαβάζει Trace Αρχεία που έχουν διακριτά γεγονότα ενός τύπου μόνο, ενώ αρχεία με συνδυασμό ασύρματων και ενσύρματων συνδέσεων δεν μπορεί να τα φορτώσει.

Τέλος ένα από τα πλεονεκτήματα της εφαρμογής αυτής είναι τα frequency distribution γραφήματα που εξάγει, όπως για παράδειγμα την συχνότητα απόρριψης πακέτων σε συνάρτηση με το μέγεθος των πακέτων.

6 Βιβλιογραφία

1. NS-2 Trace Formats. Retrieved from Wiki : http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats
2. Drakos N. & Moore R. (1992-1999). The Network Simulator ns-2: Documentation. Retrieved from http://www.isi.edu/nsnam/ns/ns-documentation.html
3. Kevin Fall & Kannan Vardhan (May 9 2010). The ns Manual –The VINT Project. Retrieved from http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf
4. Teerawat I. & Ekram H. (2008). Introduction to Network Simulator NS2. Springer.
5. NS by Example Jae Chung & Mark Claypool. Located at http://nile.wpi.edu/NS/
6. Mohid P. Tahiliani. Trace analysis awk scripts. Retrieved from http://mohittahiliani.blogspot.com/
7. NS-2 Learning Guide. Located at http://hpds.ee.ncku.edu.tw/~smallko/ns2/ns2.htm
8. Larry L. Peterson, Bruce S. Davie (2011). Computer Networks: A Systems Approach *Morgan Kaufmann Series in Networking Computer Networks: A Systems Approach*.
9. Abdelhamid Mellouk (2008). End-to-end quality of service: engineering in next generation heterogenous networks.
10. Randy L. Melton, Nova Southeastern University. Computer Information Systems (MCIS, DCIS) (2008). Monitoring QoS in wireless ad hoc networks.
11. Weiwei Fang & Hengheng Qian. jTrana application. Retrieved from http://sites.google.com/site/ns2trana/
12. Wikipedia The Free Encyclopedia Retrieved from http://en.wikipedia.org/wiki/Wiki

7 ΠΑΡΑΡΤΗΜΑ Α

```
#Create an instance of the Simulator class

set ns [new Simulator]

#Add dynamic routing

$ns rtproto DV

#Define blue color for the first data flow

$ns color 1 Blue

#Add seven nodes n0, to n6 with a for loop

for {set i 0} {$i < 7} {incr i} {
set n($i) [$ns node]
}


#Export packet traces

set f [open d2_p1.tr w]
$ns trace-all $f

#Export nam traces

set nf [open d2_p1.nam w]
$ns namtrace-all $nf

#Create a duplex link (n0-n1)
#with 2Mb bandwidth and 10ms delay per link,
#including a DropTail queue for each link

$ns duplex-link $n(0) $n(1) 2Mb 10ms DropTail

#Create 5 (duplex) circular links (n1 to n5)
#with 2Mb bandwidth and 10ms delay per link,
#including a DropTail queue for each link

$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(1) 2Mb 10ms DropTail

#Create a duplex link (n4-n6)
#with 2Mb bandwidth and 10ms delay per link,
#including a DropTail queue for each link
```

```
$ns duplex-link $n(4) $n(6) 2Mb 10ms DropTail

#Monitor the queue for link 4-6 for NAM

$ns duplex-link-op $n(4) $n(6) queuePos 0.5

#Set error model on link 2-3 equal to 10%

set loss_module [new ErrorModel]
$loss_module set rate_ 0.1
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)

#Create a UDP connection (udp0) and attach it to node n(0)

set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n(0) $udp0

#Create a CBR traffic source (cbr0) and attach it to udp0
#Set packetSize = 200 bytes
#Set interval time = 0.001 secs (stored in int variable)

set cbr0 [new Application/Traffic/CBR]
set int 0.001
$cbr0 set packetSize_ 200
$cbr0 set interval_ $int
$cbr0 attach-agent $udp0

#Create a Null agent (null0) and attach it to node n(6)

set null0 [new Agent/Null]
$ns attach-agent $n(6) $null0

#Connect udp0 and null0 agents

$ns connect $udp0 $null0

#Set the link n(4)-n(5) down for 1 sec between 2.0 and 3.0 sec

$ns rtmodel-at 2.0 down $n(4) $n(5)
$ns rtmodel-at 3.0 up $n(4) $n(5)

#Monitor queue 4-6 and store traces in queue.tr for each 0.1 sec

set qmon [$ns monitor-queue $n(4) $n(6) [open queue_p1.tr w] 0.1]
[$ns link $n(4) $n(6)] queue-sample-timeout

#Schedule event cbr0 to start at 0.1 secs and to stop at 4.9 secs

$ns at 0.1 "$cbr0 start"
$ns at 4.9 "$cbr0 stop"

#Call the procedure "finish" at 5.0 secs
```

```
$ns at 5.0 "finish"

#Add a procedure, called "finish"

proc finish {} {
        global ns f nf
      $ns flush-trace
      close $nf
        close $f
      exit 0
}

#Start the simulation

$ns run
```

8 ΠΑΡΑΡΤΗΜΑ Β

```
package Data;


import util.StringUtils;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Arp {


    private String RequestOrReply;
    private int SourceMacAddress;
    private int SourceAddress;
    private int DestinationMacAddress;
    private int DestinationAddress;


    Arp(String RoR,String SmA,String SA,String DmA,String DA) {
        RequestOrReply = RoR;
        SourceMacAddress = Integer.parseInt(SmA);
        SourceAddress = Integer.parseInt(SA);
        DestinationMacAddress = Integer.parseInt(DmA);
        DestinationAddress = Integer.parseInt(DA);
    }


    Arp(String s,Object obj) {
        if(obj instanceof NewWireless) {
            RequestOrReply = StringUtils.NwFlagValue(s, "-Po");
            SourceMacAddress = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pms"));
            SourceAddress = Integer.parseInt(StringUtils.NwFlagValue(s, "-Ps"));
            DestinationMacAddress = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pmd"));
            DestinationAddress = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pd"));
```

```
        }
        else {
            //888 Unsuported Constructor : Add here code for Arp constructor for OldWireless
Trace Format;
        }
    }

    public int getDestinationAddress() {
        return DestinationAddress;
    }

    public int getDestinationMacAddress() {
        return DestinationMacAddress;
    }

    public String getRequestOrReply() {
        return RequestOrReply;
    }

    public int getSourceAddress() {
        return SourceAddress;
    }

    public int getSourceMacAddress() {
        return SourceMacAddress;
    }
}

package Data;

import util.StringUtils;
```

```
/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Cbr {

    private int SequenceNumber;
    private int NumberOfTimesPacketForwarded;
    private int OptimalNumberOfForwards;

    Cbr(String s,Object obj) {
        if(obj instanceof NewWireless) {
            SequenceNumber = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pi"));
NumberOfTimesPacketForwarded= Integer.parseInt(StringUtils.NwFlagValue(s, "-Pf"));
            OptimalNumberOfForwards = Integer.parseInt(StringUtils.NwFlagValue(s, "-Po"));
        }
    }

    public int getNumberOfTimesPacketForwarded() {
        return NumberOfTimesPacketForwarded;
    }

    public int getOptimalNumberOfForwards() {
        return OptimalNumberOfForwards;
    }

    public int getSequenceNumber() {
        return SequenceNumber;
    }
}
```

```
package Data;

import util.StringUtils;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Dsr {

    private int NumberOfNodesTraversed;
    private int RoutingRequestFlag;
    private int RouteRequestSeqNumber;
    private int RoutingReplyFlag;
    private int RouteRequestSeqNumber2;
    private int ReplyLength;
    private int SourceOfSourceRouting;
    private int DestinationOfSourceRouting;
    private int ErrorReportFlag;
    private int NumberOfErrors;
    private int ReportToWhom;
    private int LinkErrorFrom;
    private int LinkErrorTo;

    Dsr (String s, Object obj) {
        if(obj instanceof NewWireless) {
            NumberOfNodesTraversed = Integer.parseInt(StringUtils.NwFlagValue(s, "-Ph"));
            RoutingRequestFlag = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pq"));
            RouteRequestSeqNumber = Integer.parseInt(StringUtils.NwFlagValue(s, "-Ps"));
            RoutingReplyFlag = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pp"));
```

```
            RouteRequestSeqNumber2 = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pn"));
            ReplyLength = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pl"));
            SourceOfSourceRouting = Integer.parseInt(StringUtils.NwFlagValue(s, "-
Pe").split("->")[0]);
            DestinationOfSourceRouting = Integer.parseInt(StringUtils.NwFlagValue(s, "-
Pe").split("->")[1]);
            ErrorReportFlag = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pw"));
            NumberOfErrors = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pm"));
            ReportToWhom = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pc"));
            LinkErrorFrom = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pb").split("->")[0]);
            LinkErrorTo = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pb").split("->")[1]);
        }
    }

    public int getDestinationOfSourceRouting() {
        return DestinationOfSourceRouting;
    }

    public int getErrorReportFlag() {
        return ErrorReportFlag;
    }

    public int getLinkErrorFrom() {
        return LinkErrorFrom;
    }

    public int getLinkErrorTo() {
        return LinkErrorTo;
    }

    public int getNumberOfErrors() {
        return NumberOfErrors;
```

```
}

public int getNumberOfNodesTraversed() {
    return NumberOfNodesTraversed;
}

public int getReplyLength() {
    return ReplyLength;
}

public int getReportToWhom() {
    return ReportToWhom;
}

public int getRouteRequestSeqNumber() {
    return RouteRequestSeqNumber;
}

public int getRouteRequestSeqNumber2() {
    return RouteRequestSeqNumber2;
}

public int getRoutingReplyFlag() {
    return RoutingReplyFlag;
}

public int getRoutingRequestFlag() {
    return RoutingRequestFlag;
}

public int getSourceOfSourceRouting() {
```

```
        return SourceOfSourceRouting;
    }
}

package Data;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Event {

    @Override
    public String toString(){
        return "Object Type: Event\n";
    }

    public double getTime(){
        return 0.0;
    }

}

package Data;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */

public class Ip {
```

```
private int SourceIpAddress;
private int SourcePortNumber;
private int DestinationIpAddress;
private int DestinationPortNumber;
private int TtlValue;

Ip(){};
Ip(int SiA,int SpN,int DiA,int DpA,int TV) {
    SourceIpAddress = SiA;
    SourcePortNumber = SpN;
    DestinationIpAddress = DiA;
    DestinationPortNumber = DpA;
    TtlValue = TV;
}

public int getDestinationIpAddress() {
    return DestinationIpAddress;
}

public int getDestinationPortNumber() {
    return DestinationPortNumber;
}

public int getSourceIpAddress() {
    return SourceIpAddress;
}

public int getSourcePortNumber() {
    return SourcePortNumber;
}
```

```
    public int getTtlValue() {
        return TtlValue;
    }

    @Override
    public String toString() {
        return "Ip{" + "SourceIpAddress=" + SourceIpAddress + "SourcePortNumber=" +
SourcePortNumber + "DestinationIpAddress=" + DestinationIpAddress +
"DestinationPortNumber=" + DestinationPortNumber + "TtlValue=" + TtlValue + '}';
    }

}

package Data;

import util.StringUtils;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class NewIp extends Ip{
    private int FlowId;
    private int UniqueId;
    private String PacketType;
    private int PacketSize;

    NewIp(String s){
        super(Integer.parseInt(StringUtils.NwFlagValue(s, "-Is").split("\\.")[0]),
            Integer.parseInt(StringUtils.NwFlagValue(s, "-Is").split("\\.")[1]),
            Integer.parseInt(StringUtils.NwFlagValue(s, "-Id").split("\\.")[0]),
            Integer.parseInt(StringUtils.NwFlagValue(s, "-Id").split("\\.")[1]),
```

```
            Integer.parseInt(StringUtils.NwFlagValue(s, "-Iv"))
            );

    FlowId = Integer.parseInt(StringUtils.NwFlagValue(s, "-If"));
    UniqueId = Integer.parseInt(StringUtils.NwFlagValue(s, "Ii"));
    PacketType = StringUtils.NwFlagValue(s, "-It");
    PacketSize = Integer.parseInt(StringUtils.NwFlagValue(s, "-Il"));
}

public int getFlowId() {
    return FlowId;
}

public int getPacketSize() {
    return PacketSize;
}

public String getPacketType() {
    return PacketType;
}

public int getUniqueId() {
    return UniqueId;
}

public void setFlowId(int FlowId) {
    this.FlowId = FlowId;
}

public void setUniqueId(int UniqueId) {
    this.UniqueId = UniqueId;
```

```
    }

}

package Data;

import util.StringUtils;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class NewWireless extends Event {

    private String Abbreviation;
    private double Time;
    private int NodeId;
    private double NodeXCoordinate;
    private double NodeYCoordinate;
    private double NodeZCoordinate;
    private double NodeEnergyLevel;
    private String NetworkTraceLevel;
    private String DropReason;
    private int HopSourceNodeId;
    private String DurationHex;
    private String SourceEthAddressHex;
    private String DestinationEthAddressHex;
    private String EthTypeHex;
    private String PacketType;
    private String PacketTypen;
```

```
//Extra Informations according to Packet Type
private Arp arp = null;
private Dsr dsr = null;
//private Aodv aodv = null;
//private Tora tora = null;
private NewIp ip = null;
private Tcp tcp = null;
private Cbr cbr = null;
//private Imep imep = null;



private String[] acceptedAbbreviation = {"s","r","d","f"};

public NewWireless(String s) {
   Abbreviation=StringUtils.getAbbreviation(s);
   Time=Double.parseDouble( StringUtils.NwFlagValue(s, "-t") );
   NodeId=Integer.parseInt( StringUtils.NwFlagValue(s, "-Ni") );
   NodeXCoordinate=Double.parseDouble( StringUtils.NwFlagValue(s, "-Nx"));
   NodeYCoordinate=Double.parseDouble( StringUtils.NwFlagValue(s, "-Ny"));
   NodeZCoordinate=Double.parseDouble( StringUtils.NwFlagValue(s, "-Nz"));
   NodeEnergyLevel=Double.parseDouble( StringUtils.NwFlagValue(s, "-Ne"));
   NetworkTraceLevel=StringUtils.NwFlagValue(s, "-Nl");
   DropReason=StringUtils.NwFlagValue(s, "-Nw");
   HopSourceNodeId=Integer.parseInt(StringUtils.NwFlagValue(s, "-Hs"));
   DurationHex=StringUtils.NwFlagValue(s, "-Ma");
   SourceEthAddressHex=StringUtils.NwFlagValue(s, "-Ms");
   DestinationEthAddressHex=StringUtils.NwFlagValue(s, "-Md");
   EthTypeHex=StringUtils.NwFlagValue(s, "-Mt");
   PacketType=StringUtils.NwFlagValue(s, "-P");
   PacketTypen=StringUtils.NwFlagValue(s, "-Pn");
```

```
    addExtraInformations(s);

}

private void addExtraInformations (String s) {
    /*if(PacketType.equals("arp"))
        arp = new Arp(s,this);
    else if(PacketType.equals("dsr"))
        dsr= new Dsr(s,this);
    else if(PacketType.equals("imep"))
        imep= new Imep(s,this);
    else if(PacketType.equals("tora"))
        tora= new Tora(s,this);*/
    if(StringUtils.NwFlagExists(s, "-ll"))
        ip= new NewIp(s);
    /*if(PacketTypen.equals("tcp"))
        tcp=new Tcp(s,this);*/
    /*if(PacketTypen.equals("cbr"))
        cbr=new Cbr(s,this);*/

}

private boolean AbbreviationIsAcceptible() {
    for(String acceptible : acceptedAbbreviation){
        if(acceptible.equals(Abbreviation))
            return true;
    }
    return false;
}
```

```
public String getDestinationEthAddressHex() {
    return DestinationEthAddressHex;
}

public String getDropReason() {
    return DropReason;
}

public String getDurationHex() {
    return DurationHex;
}

public String getEthTypeHex() {
    return EthTypeHex;
}

public int getHopSourceNodeId() {
    return HopSourceNodeId;
}

public String getNetworkTraceLevel() {
    return NetworkTraceLevel;
}

public double getNodeEnergyLevel() {
    return NodeEnergyLevel;
}

public int getNodeId() {
    return NodeId;
}
```

```
public double getNodeXCoordinate() {
    return NodeXCoordinate;
}

public double getNodeYCoordinate() {
    return NodeYCoordinate;
}

public double getNodeZCoordinate() {
    return NodeZCoordinate;
}

public String getPacketType() {
    return PacketType;
}

public String getPacketTypen() {
    return PacketTypen;
}

public String getSourceEthAddressHex() {
    return SourceEthAddressHex;
}

public String getAbbreviation() {
    return Abbreviation;
}

public double getTime() {
    return Time;
```

```
}

public NewIp getIp(){
    return ip;
}

public boolean isSent(){
    if(Abbreviation.equals("s"))
        return true;
    return false;
}

public boolean isReceive(){
    if(Abbreviation.equals("r"))
        return true;
    return false;
}

public boolean isDrop(){
    if(Abbreviation.equalsIgnoreCase("d"))
        return true;
    return false;
}

public boolean isForward(){
    if(Abbreviation.equals("f"))
        return true;
    return false;
}

@Override
```

```
    public String toString() {

        return "NewWireless{" + "\nAbbreviation=" + Abbreviation + "\nTime=" + Time +
"\nNodeId=" + NodeId + "\nNodeXCoordinate=" + NodeXCoordinate +
"\nNodeYCoordinate=" + NodeYCoordinate + "\nNodeZCoordinate=" + NodeZCoordinate
+ "\nNodeEnergyLevel=" + NodeEnergyLevel + "\nNetworkTraceLevel=" +
NetworkTraceLevel + "\nDropReason=" + DropReason + "\nHopSourceNodeId=" +
HopSourceNodeId + "\nDurationHex=" + DurationHex + "\nSourceEthAddressHex=" +
SourceEthAddressHex + "\nDestinationEthAddressHex=" + DestinationEthAddressHex +
"\nEthTypeHex=" + EthTypeHex + "\nPacketType=" + PacketType + "\nPacketTypen=" +
PacketTypen + "\narp=" + arp + "\ndsr=" + dsr + "\nip=" + ip + "\ntcp=" + tcp + "\ncbr=" +
cbr + '}';

    }

}


package Data;


/*

 * @author Chrison

 *

 */


import Global.Debug;

import Global.Settings;


public class NormalEvent extends Event {


    private String Abbreviation;

    private double Time;

    private int SourceNode;

    private int DestinationNode;

    private String PacketName;

    private int PacketSize;

    private String Flags;

    private int FlowId;

    private String SourceAddress;
```

```
private String SourcePort;
private String DestinationAddress;
private String DestinationPort;
private int SequenceNumber;
private int UniquePacketId;

private String[] acceptedAbbreviation = {"r","d","e","+","-"};

public NormalEvent() {}

public NormalEvent(String event) {
   String TempEvent[] = event.split(" ");

   Abbreviation = TempEvent[0];
   Time = Double.parseDouble(TempEvent[1]);
   SourceNode = Integer.parseInt(TempEvent[2]);
   DestinationNode = Integer.parseInt(TempEvent[3]);
   PacketName = TempEvent[4];
   PacketSize = Integer.parseInt(TempEvent[5]);
   Flags = TempEvent[6];
   FlowId = Integer.parseInt(TempEvent[7]);



   String Temp[]=TempEvent[8].split("\\.");

   SourceAddress = Temp[0];
   SourcePort = Temp[1];

   Temp = TempEvent[9].split("\\.");

   DestinationAddress = Temp[0];
```

```
        DestinationPort = Temp[1];
        SequenceNumber = Integer.parseInt(TempEvent[10]);
        UniquePacketId = Integer.parseInt(TempEvent[11]);


    }


    private boolean AbbreviationIsAcceptible() {
        for(String acceptible : acceptedAbbreviation){
            if(acceptible.equals(Abbreviation))
                if(Debug.analyze) {
                    Debug.print("AbbreviationIsAcceptible() returned :
true",Settings.DebugDestination);
                    return true;
                }
        }
        if(Debug.analyze)
            Debug.print("AbbreviationIsAcceptible() returned :
false",Settings.DebugDestination);
        return false;
    }


    public String getDestinationAddress() {
        return DestinationAddress;
    }


    public int getDestinationNode() {
        return DestinationNode;
    }


    public String getDestinationPort() {
        return DestinationPort;
    }
```

```
public String getFlags() {
    return Flags;
}

public int getFlowId() {
    return FlowId;
}

public String getPacketName() {
    return PacketName;
}

public int getPacketSize() {
    return PacketSize;
}

public int getSequenceNumber() {
    return SequenceNumber;
}

public String getSourceAddress() {
    return SourceAddress;
}

public int getSourceNode() {
    return SourceNode;
}

public String getSourcePort() {
    return SourcePort;
```

```
}

public int getUniquePacketId() {
    return UniquePacketId;
}

public String getAbbreviation() {
    return Abbreviation;
}

public double getTime() {
    return Time;
}

public boolean isReceive(){
    if(Abbreviation.equals("r"))
        return true;
    return false;
}

public boolean isDrop(){
    if(Abbreviation.equalsIgnoreCase("d"))
        return true;
    return false;
}

public boolean isError(){
    if(Abbreviation.equals("e"))
        return true;
    return false;
}
```

```
    public boolean isEnqueue(){
        if(Abbreviation.equals("+"))
            return true;
        return false;
    }

    public boolean isDequeue(){
        if(Abbreviation.equals("-"))
            return true;
        return false;
    }

    public boolean isForward() {
        if( !SourceAddress.equals(String.valueOf(SourceNode)) )
            return true;
        return false;
    }

    @Override
    public String toString() {
        return "NormalEvent:\n" + "abbreviation=" + Abbreviation + ",\ntime=" + Time +
",\nSourceNode=" + SourceNode + ",\nDestinationNode=" + DestinationNode +
",\nPacketName=" + PacketName + ",\nPacketSize=" + PacketSize + ",\nFlags=" + Flags
+ ",\nFlowId=" + FlowId + ",\nSourceAddress=" + SourceAddress + ",\nSourcePort=" +
SourcePort + ",\nDestinationAddress=" + DestinationAddress + ",\nDestinationPort=" +
DestinationPort + ",\nSequenceNumber=" + SequenceNumber + ",\nUniquePacketId=" +
UniquePacketId + ";\n";
    }
}


package Data;
```

```
import Global.Debug;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class OldWireless extends Event {


    private String Abbreviation;
    private double Time;
    private int NodeId;
    private double XCoordinate;
    private double YCoordinate;
    private String TraceName;
    private String Reason;
    private int EventIdentifier;
    private String PacketType;
    private int PacketSize;
    private String TimeToSentDataHex;
    private String DestinationMacAddressHex;
    private String SourceMacAddressHex;
    private String TypeHex;
    private Ip ip = null;


    public OldWireless(String event) {
        //event = event.replace("  ", " ");
        String TempEvent[] = event.split(" ");


        Abbreviation = TempEvent[0];
        Time = Double.parseDouble(TempEvent[1]);
        if (TempEvent[2].contains("_")) {
```

```
            NodeId = Integer.parseInt(TempEvent[2].replaceAll("_", ""));
            TraceName = TempEvent[3];
            Reason = TempEvent[5];
            EventIdentifier = Integer.parseInt(TempEvent[6]);
            PacketType = TempEvent[7];
            PacketSize = Integer.parseInt(TempEvent[8]);


            TimeToSentDataHex = TempEvent[9].replace("[", "");
            DestinationMacAddressHex = TempEvent[10];
            SourceMacAddressHex = TempEvent[11];
            TypeHex = TempEvent[12].replace("]", "");


            if (TempEvent.length > 13 && TempEvent[13].equals("-------") &&
TempEvent[15].contains(":")) {
                try {
                    String TempValue = TempEvent[14];
                    TempValue = TempValue.replace("[", "");
                    String splitTemp[] = TempValue.split(":");
                    int SourceIpAddress = Integer.parseInt(splitTemp[0]);
                    int SourcePortNumber = Integer.parseInt(splitTemp[1]);
                    splitTemp = TempEvent[15].split(":");
                    int DestinationIpAddress = Integer.parseInt(splitTemp[0]);
                    int DestinationPortNumber = Integer.parseInt(splitTemp[1]);
                    int TTL = Integer.parseInt(TempEvent[16]);


                    ip = new Ip(SourceIpAddress, SourcePortNumber, DestinationIpAddress,
DestinationPortNumber, TTL);
                } catch (Exception e) {
                }


            }
        } else {
```

```
        NodeId = Integer.parseInt(TempEvent[2]);


        String temp = TempEvent[3].replace("(", "").replace(" ", "");
        XCoordinate = Double.parseDouble(temp);


        temp = TempEvent[4].replace(")", "").replace(" ", "");
        YCoordinate = Double.parseDouble(temp);
        TraceName = TempEvent[5];
        Reason = TempEvent[6];
        EventIdentifier = Integer.parseInt(TempEvent[7]);
        PacketType = TempEvent[8];
        PacketSize = Integer.parseInt(TempEvent[9]);
        TimeToSentDataHex = TempEvent[10].replace("[", "");
        DestinationMacAddressHex = TempEvent[11];
        SourceMacAddressHex = TempEvent[12];
        TypeHex = TempEvent[13].replace("]", "");


        if (TempEvent[14].equals("-------")) {
            String TempValue;
            TempValue = TempEvent[15];
            TempValue = TempValue.replace("[", "");
            String splitTemp[] = TempValue.split(":");
            int SourceIpAddress = Integer.parseInt(splitTemp[0]);
            int SourcePortNumber = Integer.parseInt(splitTemp[1]);
            splitTemp = TempEvent[16].split(":");
            int DestinationIpAddress = Integer.parseInt(splitTemp[0]);
            int DestinationPortNumber = Integer.parseInt(splitTemp[1]);
            int TTL = Integer.parseInt(TempEvent[17]);


            ip = new Ip(SourceIpAddress, SourcePortNumber, DestinationIpAddress,
DestinationPortNumber, TTL);
```

```
        }


    }
}


    public OldWireless(String Abbreviation, double Time, int NodeId, double XCoordinate,
double YCoordinate, String TraceName, String Reason, int EventIdentifier, String
PacketType, int PacketSize, String TimeToSentDataHex, String
DestinationMacAddressHex, String SourceMacAddressHex, String TypeHex) {
        this.Abbreviation = Abbreviation;
        this.Time = Time;
        this.NodeId = NodeId;
        this.XCoordinate = XCoordinate;
        this.YCoordinate = YCoordinate;
        this.TraceName = TraceName;
        this.Reason = Reason;
        this.EventIdentifier = EventIdentifier;
        this.PacketType = PacketType;
        this.PacketSize = PacketSize;
        this.TimeToSentDataHex = TimeToSentDataHex;
        this.DestinationMacAddressHex = DestinationMacAddressHex;
        this.SourceMacAddressHex = SourceMacAddressHex;
        this.TypeHex = TypeHex;
    }


    public String getAbbreviation() {
        return Abbreviation;
    }


    public String getDestinationMacAddressHex() {
        return DestinationMacAddressHex;
```

```
}

public int getEventIdentifier() {
    return EventIdentifier;
}

public int getNodeId() {
    return NodeId;
}

public int getPacketSize() {
    return PacketSize;
}

public String getPacketType() {
    return PacketType;
}

public String getReason() {
    return Reason;
}

public String getSourceMacAddressHex() {
    return SourceMacAddressHex;
}

public double getTime() {
    return Time;
}

public String getTimeToSentDataHex() {
```

```
    return TimeToSentDataHex;
}

public String getTraceName() {
    return TraceName;
}

public String getTypeHex() {
    return TypeHex;
}

public double getXCoordinate() {
    return XCoordinate;
}

public double getYCoordinate() {
    return YCoordinate;
}

public Ip getIp() {
    return ip;
}

public boolean isSent() {
    if (Abbreviation.equals("s")) {
        return true;
    }
    return false;
}

public boolean isReceive() {
```

```
    if (Abbreviation.equals("r")) {
        return true;
    }
    return false;
}

public boolean isDrop() {
    if (Abbreviation.equalsIgnoreCase("d")) {
        return true;
    }
    return false;
}

public boolean isForward() {
    if (Abbreviation.equals("f")) {
        return true;
    }
    return false;
}

public boolean equalsFlow(String flow) {
    try {
        String[] Temp = flow.split(":");

        if (String.valueOf(this.getIp().getSourceIpAddress()).equals(Temp[0])
                && String.valueOf(this.getIp().getSourcePortNumber()).equals(Temp[1])
                && String.valueOf(this.getIp().getDestinationIpAddress()).equals(Temp[2])
                && String.valueOf(this.getIp().getDestinationPortNumber()).equals(Temp[3])
                && this.getPacketType().equals(Temp[4])) {
            return true;
        }
```

```
            if (Debug.analyze) {

                System.out.println("OldWireless equalsFlow() at event\n" + this.toString());

                System.out.println("The Flow Id which attempting to compare" + flow);

                System.out.println("The event's real flow Id: " +
String.valueOf(this.getIp().getSourceIpAddress()) + ":" +
String.valueOf(this.getIp().getSourcePortNumber()) + ":" +
String.valueOf(this.getIp().getDestinationIpAddress()) + ":" +
String.valueOf(this.getIp().getDestinationPortNumber()) + ":" + this.getPacketType());

            }

        } catch (ArrayIndexOutOfBoundsException e) {

            /*Flow IDs on Normal and New Wireless formats are supplied by ns-2 itself and
are integer numbers

             *This Method throws ArrayIndexOutOfBoundsException in case an
Normal/NewWireless Flowid be parameter

             */

            return false;

        }


        return false;

    }


    @Override

    public String toString() {


        if (ip != null) {

            String ipToString = ip.toString();

            return "OldWireless{ " + " Abbreviation= " + Abbreviation + " Time= " + Time + "
NodeId= " + NodeId + " XCoordinate= " + XCoordinate + " YCoordinate= " + YCoordinate
+ " TraceName= " + TraceName + " Reason= " + Reason + " EventIdentifier= " +
EventIdentifier + " PacketType= " + PacketType + " PacketSize= " + PacketSize + "
[TimeToSentDataHex= " + TimeToSentDataHex + " DestinationMacAddressHex= " +
DestinationMacAddressHex + " SourceMacAddressHex= " + SourceMacAddressHex + "
TypeHex= " + TypeHex + "}] " + ipToString;

        }
```

```
        return "OldWireless{ " + " Abbreviation= " + Abbreviation + " Time= " + Time + "
NodeId= " + NodeId + " XCoordinate= " + XCoordinate + " YCoordinate= " + YCoordinate
+ " TraceName= " + TraceName + " Reason= " + Reason + " EventIdentifier= " +
EventIdentifier + " PacketType= " + PacketType + " PacketSize= " + PacketSize + "
[TimeToSentDataHex= " + TimeToSentDataHex + " DestinationMacAddressHex= " +
DestinationMacAddressHex + " SourceMacAddressHex= " + SourceMacAddressHex + "
TypeHex= " + TypeHex + "}] " + "ip= NULL";

    }

}


package Data;


import util.StringUtils;


/**

 *

 * @author Christos-Charalampos Papagiannidis

 */

public class Tcp {


    private int SequenceNumber;

    private int AcknowledgmentNumber;

    private int NumberOfTimesPacketForwarded;

    private int OptimalNumberOfForwards;


    Tcp (String s,Object obj) {

        if(obj instanceof NewWireless) {

            SequenceNumber = Integer.parseInt(StringUtils.NwFlagValue(s, "-Ps"));

            AcknowledgmentNumber = Integer.parseInt(StringUtils.NwFlagValue(s, "-Pa"));

            NumberOfTimesPacketForwarded = Integer.parseInt(StringUtils.NwFlagValue(s, "-
Pf"));

            OptimalNumberOfForwards = Integer.parseInt(StringUtils.NwFlagValue(s, "-Po"));

        }

    }
```

```
    public int getAcknowledgmentNumber() {
        return AcknowledgmentNumber;
    }

    public int getNumberOfTimesPacketForwarded() {
        return NumberOfTimesPacketForwarded;
    }

    public int getOptimalNumberOfForwards() {
        return OptimalNumberOfForwards;
    }

    public int getSequenceNumber() {
        return SequenceNumber;
    }


}

package Global;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Christos-Charalampos Papagiannidis
```

```
 */


public class Debug {


    public static boolean string=false; // string methods return values .esp StringUtils.java

    public static boolean analyze=false; // Data analyzing procedures after TrcReader.java excecuted

    public static boolean ui=false; // User Interface logs

    public static boolean graphs=false; //

    public static boolean settings=false; // Debug help about program's settings or user's settings

    public static boolean error=false;


    public static int PRINT_TO_CONSOLE=0;

    public static int PRINT_TO_FILE=1;


    public static void print(String s,int DESTINATION_CODE){

        if(DESTINATION_CODE==PRINT_TO_CONSOLE)

            System.out.println(s);

        else if(DESTINATION_CODE==PRINT_TO_FILE){

            String strFilePath = Trace.getFilePath()+".Debug";

            try {

                s+=Global.Prefs.getEOL();

                FileOutputStream fos = new FileOutputStream(strFilePath, true);

                fos.write(s.getBytes());

                fos.close();


            } catch (FileNotFoundException ex) {

                Logger.getLogger(Debug.class.getName()).log(Level.SEVERE, null, ex);

            } catch (IOException ex) {

                    Logger.getLogger(Debug.class.getName()).log(Level.SEVERE, null, ex);

                }
```

```
        }
    }
}

package Global;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import ui.Editors.SettingsForm;
import ui.OpenFileDialog;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class GMenu {
    private ArrayList<JMenu>     codedMenus     = new ArrayList();
    private ArrayList<JMenuItem> codedMenuItems = new ArrayList();
    private JMenuItem            JMenuSeperator = new JMenuItem("-&-");

    public GMenu() {
        JMenuItem temp;
        //add first menu.
        codedMenus.add(new JMenu("File",false));
```

```
        temp=new JMenuItem("Open Trace");

        temp.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e){

                try {

                    new OpenFileDialog();

                } catch (Exception ex) {

                    Logger.getLogger(GMenu.class.getName()).log(Level.SEVERE, "GMenu
Line 36:OpenFileDialog() Error", ex);

                }

            }

        });

        codedMenuItems.add(temp);


        temp= new JMenuItem("Exit");

        temp.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {

                System.exit(1);

            }

        });

        codedMenuItems.add(temp);


        //----------------------------SEPERATOR--------------------------------

        codedMenuItems.add(JMenuSeperator);


        //add second menu

        codedMenus.add(new JMenu("Graphs",false));


        temp= new JMenuItem("Filter & Create");

        temp.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {

                JFrame frame= new JFrame();

                frame.setResizable(false);

                frame.setContentPane(new SettingsForm());
```

```
                frame.setSize(600,450);
                frame.setAlwaysOnTop(true);
                frame.setTitle("Network Analyzer Settings");
                frame.setVisible(true);



            }
        });
        codedMenuItems.add(temp);
        //----------------------------SEPERATOR--------------------------------
        codedMenuItems.add(JMenuSeperator);
    }

    public  ArrayList getJMenu() {
        return codedMenus;
    }

    public  JMenuItem getMenuSeperator(){
        return JMenuSeperator;
    }

    public  ArrayList getJMenuItems(){
        return codedMenuItems;
    }

}

package Global;

/**
 *
```

```
 * @author Christos-Charalampos Papagiannidis
 */

public class Prefs {

    private static String eol = System.getProperty("line.separator");
    //--Values "Windows" || "Linux" || "Macintosh" --\\
    private static String OperatingSystem = "Windows";
    private static String ApplicationName = "jTrcezer";
    private static String ApplicationVersion = "1.0";



    public Prefs(){

    }

    public static String getEOL() {
        return eol;
    }

    public static String getOperatingSystem(){
        return OperatingSystem;
    }

    public static String getApplicationName(){
        return ApplicationName;
    }

    public static String getApplicationVersion(){
        return ApplicationVersion;
    }
```

```
}

package Global;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Settings {

    public static String[] AcceptablePacketNames = {};
    public static String[] AcceptablePacketTypes = {};
    public static String[] AcceptablePacketTypen = {};
    public static String[] AcceptableNodes = {};
    public static String[] AcceptableNetworkTraceLevel = {};
    public static double TimeInterval = 0.1;

    public static int DebugDestination = Settings.DebugDestination;

    public static void clearAllSettings(){

    }

     public static boolean isAcceptableNetworkTraceLevel(String s) {
        if (AcceptableNetworkTraceLevel == null) {
            return true;
        }

        for (int i = 0; i < AcceptableNetworkTraceLevel.length; i++) {
            if (s != null && s.equals(AcceptableNetworkTraceLevel[i])) {
                return true;
```

```
        }
    }
    return false;
}

public static boolean isAcceptableNode(String s) {
    if (AcceptableNodes == null) {
        return true;
    }

    for (int i = 0; i < AcceptableNodes.length; i++) {
        if (s != null && s.equals(AcceptableNodes[i])) {
            return true;
        }
    }
    return false;
}

 public static boolean isAcceptableNode(int s) {
    String node = String.valueOf(s);
    if (AcceptableNodes == null) {
        return true;
    }

    for (int i = 0; i < AcceptableNodes.length; i++) {
        if (node != null && node.equals(AcceptableNodes[i])) {
            return true;
        }
    }
    return false;
}
```

```
public static boolean isAcceptablePacketName(String s) {
    if (AcceptablePacketNames == null) {
        return true;
    }

    for (int i = 0; i < AcceptablePacketNames.length; i++) {
        if (s != null && s.equals(AcceptablePacketNames[i])) {
            return true;
        }
    }
    return false;
}

public static boolean isAcceptablePacketType(String s) {
    if (AcceptablePacketTypes == null) {
        return true;
    }

    for (int i = 0; i < AcceptablePacketTypes.length; i++) {
        if (s != null && s.equals(AcceptablePacketTypes[i])) {
            return true;
        }
    }
    return false;
}

public static boolean isAcceptablePacketTypen(String s) {
    if (AcceptablePacketTypen == null) {
        return true;
    }
```

```
    for (int i = 0; i < AcceptablePacketTypen.length; i++) {
        if (s != null && s.equals(AcceptablePacketTypen[i])) {
            return true;
        }
    }
    return false;
}

public static void addAcceptableNode(String node){
    if(node==null)
        AcceptableNodes=null;
    else{
        AcceptableNodes = new String[1];
        AcceptableNodes[0]=node;
    }

}

public static String printSettings(){
    String s=" for "+Trace.getFile().getName()+" ";
    if(AcceptableNodes==null)
        s+="All Network ";
    else
        s+="Node " + AcceptableNodes[0] + " ";
    if(AcceptablePacketNames!=null){
        s+=" PNs:";
        for(int i=0;i<AcceptablePacketNames.length;i++){
            s+=AcceptablePacketNames[i]+":";
        }
    }
```

```
        if(AcceptablePacketTypes!=null){
            s+=" PTs:";
             for(int i=0;i<AcceptablePacketTypes.length;i++){
                s+=AcceptablePacketTypes[i]+":";
            }
        }

        return s;
    }
}

package Global;

import Data.Event;
import Data.NewWireless;
import Data.NormalEvent;
import Data.OldWireless;
import java.io.File;
import java.io.Serializable;
import java.lang.reflect.InvocationTargetException;
import java.util.Vector;
import ui.ProgressBar;
import util.TrcReader;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public final class Trace implements Runnable, Serializable {

    private static File File = null;
```

```
    private static Vector<Event> Data = new Vector();
    ProgressBar progress = null;

    public static Vector<Event> getData() {
        return Data;
    }

    public Trace() {
    }

    public Trace(File f) throws InterruptedException, InvocationTargetException {
        File = f;
        run();
        progress.run();
    }

    public static File getFile() {
        return File;
    }

    public static String getFilePath() {
        return File.getAbsolutePath();
    }

    public void run() {
        progress = new ProgressBar("Reading & Analyzing...");
        if (Debug.analyze) {
            Debug.print("---Started Reading/Analyzing Trace File Data...---",
Settings.DebugDestination);
        }
```

```
        if (!Data.isEmpty()) {
            Data.removeAllElements();
        }

        Vector<String> Temp = new TrcReader(File).getTraceData();
        for (String tempValue : Temp) {
            if (!tempValue.startsWith("s") && !tempValue.startsWith("r")
                    && !tempValue.startsWith("d") && !tempValue.startsWith("f")
                    && !tempValue.startsWith("-") && !tempValue.startsWith("+")) {
                continue;
            } else if (tempValue.contains("-t")) {
                NewWireless temp = new NewWireless(tempValue);
                Data.add(temp);
            } else {
                try {
                    OldWireless temp = new OldWireless(tempValue);
                    Data.add(temp);
                } catch (Exception e) {
                    if (Debug.analyze) {
                        Debug.print("12002 error: OldWireless failed, try NormalEvent
Constructor!!!",Settings.DebugDestination);
                    }
                    NormalEvent temp = new NormalEvent(tempValue);
                    Data.add(temp);
                }
            }
        }

        if (Debug.analyze) {
            Debug.print("---Ended Reading/Analyzing Trace File Data.---
",Settings.DebugDestination);
        }
```

```
    }
}

package charts;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */



import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;

public class AsIsDrop extends JPanel implements Runnable{

    public AsIsDrop(){}

    public void run() {
        final XYSeries data = new XYSeries("As Is Drop Packets");

        double[][] temp = new DataAnalyzer().getDropData();
        double pastTime = 0;
```

```
        double currentTime = temp[0][0];
        double packetsCounter = 0;


        for(int i=0;i<temp.length;i++){
            if(temp[i][0]>currentTime || i == temp.length-1){
                data.add(currentTime, packetsCounter);
                pastTime=currentTime;
                currentTime=temp[i][0];
                packetsCounter=0;


                if(i!=temp.length-1)
                    i--;
            }
            else if(temp[i][0]>pastTime){
                packetsCounter+=temp[i][1];
            }
        }


        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbits",
collection, PlotOrientation.VERTICAL, true, true, true);
        data.toString();


        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("AsIs Drop "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}
```

```
package charts;


import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.DefaultTableXYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class AsIsentToEndDelay extends JPanel implements Runnable{


    public AsIsentToEndDelay() {}


    public void run() {
        DefaultTableXYDataset dataset = new DefaultTableXYDataset();
        XYSeries series = new XYSeries("AsIs EndToEnd Delay", true, false);


        double[][] temp = new DataAnalyzer().getDelayData();
        double sumdelay = 0;
        int countdelays =1;


        for (int i = 0; i < temp.length; i++) {
```

```
            if (temp[i][1] != -1) {
                sumdelay = temp[i][1];
                for(int j=i;j<temp.length;j++) {
                    if(temp[j][0]==temp[i][0]){
                        sumdelay += temp[j][1];
                        countdelays++;
                    }
                }
                if(series.getMaxX() != temp[i][0] )
                    series.add(temp[i][0], sumdelay/countdelays);
                countdelays=1;
            }
        }

        dataset.addSeries(series);

        final XYSeriesCollection collection = new XYSeriesCollection(series);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Seconds",
collection, PlotOrientation.VERTICAL, true, true, true);

        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("AsIs EndToEndDelay "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}

package charts;
```

```
import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import util.DataAnalyzer;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.ArrayUtils;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 *
 */
public class AsIsForwarded extends JPanel implements Runnable{

    public AsIsForwarded() {}

    public void run() {
            final XYSeries data = new XYSeries("Forwarded Data");

        double[][] temp = new DataAnalyzer().getForwardedData();
        double pastTime = temp[0][0];
        double TimeStep = Settings.TimeInterval;
        double currentTime = temp[0][0] + TimeStep;
        double maxTime = ArrayUtils.getMaximumValueOfColumn(temp, 0);
        double bytes = 0;
        int packetCounter = 0;
```

```
        while (currentTime <= maxTime) {
            bytes = 0;
            packetCounter = 0;

            for (int i = 0; i < temp.length; i++) {
                if (pastTime <= temp[i][0] && temp[i][0] < currentTime) {
                    bytes += temp[i][1];
                    packetCounter++;
                } else if(i==temp.length-1) {
                    data.add(currentTime - TimeStep / 2, bytes * 8 /1000 );
                    pastTime = currentTime;
                    currentTime += TimeStep;
                }
            }
        }

        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbits",
collection, PlotOrientation.VERTICAL, true, true, true);

        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("AsIs Forwarded "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}

package charts;
```

```
import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartPanel;
import util.DataAnalyzer;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class AsIsJitter extends JPanel implements Runnable{

    public AsIsJitter(){}

    public void run() {
        ScatterPlotSlow plot = new ScatterPlotSlow();
        double[][] temp = new DataAnalyzer().getJitterData();
        float[][] scatterData = new float[2][temp.length];
        for(int i=0;i<temp.length;i++){
            scatterData[0][i] = (float) temp[i][0];
            scatterData[1][i] = Math.abs((float) temp[i][1]);
        }
        plot.setData(scatterData);

        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(plot.renderAndDisplay()));
        frame.setTitle("AsIsJitter "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
```

```
}

package charts;

import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import util.DataAnalyzer;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.ArrayUtils;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 *
 */
public class AsIsLoad extends JPanel implements Runnable{

    public AsIsLoad() {}

    public void run() {
        final XYSeries data = new XYSeries("AsIs Load Data");

        double[][] temp = new DataAnalyzer().getLoadData();
        double pastTime = temp[0][0];
        double TimeStep = Settings.TimeInterval;
```

```
        double currentTime = temp[0][0] + TimeStep;
        double maxTime = ArrayUtils.getMaximumValueOfColumn(temp, 0);
        double bytes = 0;
        int packetCounter = 0;



        while (currentTime <= maxTime) {
            bytes = 0;
            packetCounter = 0;

            for (int i = 0; i < temp.length; i++) {
                if (pastTime <= temp[i][0] && temp[i][0] < currentTime) {
                    bytes += temp[i][1];
                    packetCounter++;
                } else if(i==temp.length-1) {
                    data.add(currentTime - TimeStep / 2, bytes * 8 /1000 );

                    pastTime = currentTime;
                    currentTime += TimeStep;

                }
            }
        }

        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbits",
collection, PlotOrientation.VERTICAL, true, true, true);

        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        frame.setTitle("Load "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);

    }
}

package charts;

import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import util.DataAnalyzer;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.ArrayUtils;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 *
 */
public class AsIsSent extends JPanel implements Runnable{

    public AsIsSent() {}

    public void run() {
```

```
final XYSeries data = new XYSeries("AsIs Sent Data");

double[][] temp = new DataAnalyzer().getSentData();
double pastTime = temp[0][0];
double TimeStep = Settings.TimeInterval;
double currentTime = temp[0][0] + TimeStep;
double maxTime = ArrayUtils.getMaximumValueOfColumn(temp, 0);
double bytes = 0;
int packetCounter = 0;



while (currentTime <= maxTime) {
    bytes = 0;
    packetCounter = 0;

    for (int i = 0; i < temp.length; i++) {
        if (pastTime <= temp[i][0] && temp[i][0] < currentTime) {
            bytes += temp[i][1];
            packetCounter++;
        } else if(i==temp.length-1) {
            data.add(currentTime - TimeStep / 2, bytes * 8 /1000 );

            pastTime = currentTime;
            currentTime += TimeStep;

        }
    }
}

final XYSeriesCollection collection = new XYSeriesCollection(data);
```

```
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbits",
collection, PlotOrientation.VERTICAL, true, true, true);


        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Sent "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);


    }
}


package charts;


import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import util.DataAnalyzer;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.ArrayUtils;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 *
 */
```

```
public class AsIsThroughput extends JPanel implements Runnable {

   public AsIsThroughput() {}

   public void run() {
      final XYSeries data = new XYSeries("AsIs Throughput Data");

      double[][] temp = new DataAnalyzer().getThroughputData();
      double pastTime = temp[0][0];
      double TimeStep = Settings.TimeInterval;
      double currentTime = temp[0][0] + TimeStep;
      double maxTime = ArrayUtils.getMaximumValueOfColumn(temp, 0);

      double bytes = 0;
      int packetCounter = 0;



      while (currentTime <= maxTime) {
         bytes = 0;
         packetCounter = 0;

         for (int i = 0; i < temp.length; i++) {
            if (pastTime <= temp[i][0] && temp[i][0] < currentTime) {
               bytes += temp[i][1];
               packetCounter++;
            } else if (i == temp.length - 1) {
               data.add(currentTime - TimeStep / 2, bytes * 8 / 1000);
               pastTime = currentTime;
               currentTime += TimeStep;
            }
         }
```

```
        }


        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbits",
collection, PlotOrientation.VERTICAL, true, true, true);


        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Throughput "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}


package charts;


import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
```

```
public class Drop extends JPanel implements Runnable{

   public Drop(){

   }

   public void run() {
      final XYSeries data = new XYSeries("Drop Data");

      double[][] temp = new DataAnalyzer().getDropData();
      double currentTime = temp[0][0];
      double firstTrafficTime = new DataAnalyzer().getFirstTrafficTime();
      double bytes = 0;

      for(int i=0;i<temp.length;i++){
         if(temp[i][0]>currentTime){
            if(currentTime<=firstTrafficTime)
               data.add(currentTime, bytes*8/(1000*(currentTime-temp[0][0])));
            else
               data.add(currentTime,bytes*8/(1000*(currentTime-firstTrafficTime)));

            currentTime=temp[i][0];
            bytes+=temp[i][1];
         }
         else{
            bytes+=temp[i][1];
         }
      }

      final XYSeriesCollection collection = new XYSeriesCollection(data);
```

```
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbps",
collection, PlotOrientation.VERTICAL, true, true, true);


        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Drop "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }


}


package charts;


import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class EndToEndDelay extends JPanel implements Runnable{
```

```
    public EndToEndDelay() {}


    public void run() {
        final XYSeries data = new XYSeries("EndToEnd Delay Chart");


        double[][] temp = new DataAnalyzer().getDelayData();
        double sumDelay = 0;


        int sucPacketCounter = 0;


        for (int i = 0; i < temp.length; i++) {
            if (temp[i][1] != -1) {
                sumDelay += temp[i][1];
                sucPacketCounter++;
            }
            data.add(temp[i][0], sumDelay / sucPacketCounter);
        }


        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Seconds",
collection, PlotOrientation.VERTICAL, true, true, true);


        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("EndToEndDelay "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}


package charts;
```

```
import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Forwarded extends JPanel implements Runnable{

    public Forwarded() {}

    public void run() {
        final XYSeries data = new XYSeries("Forwarded Data");

        double[][] temp = new DataAnalyzer().getForwardedData();
        double currentTime = temp[0][0];
        double firstTrafficTime = new DataAnalyzer().getFirstTrafficTime();
        double bytes = 0;

        for(int i=0;i<temp.length;i++){
            if(temp[i][0]>currentTime){
                if(currentTime<=firstTrafficTime)
```

```
                data.add(currentTime, bytes*8/(1000*(currentTime-temp[0][0])));
            else
                data.add(currentTime,bytes*8/(1000*(currentTime-firstTrafficTime)));

            currentTime=temp[i][0];
            bytes+=temp[i][1];
          }
          else{
            bytes+=temp[i][1];
          }
       }

       final XYSeriesCollection collection = new XYSeriesCollection(data);
       final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbps",
collection, PlotOrientation.VERTICAL, true, true, true);

       JFrame frame = new JFrame();
       frame.setContentPane(new ChartPanel(chart));
       frame.setTitle("Forwarded "+ Settings.printSettings());
       frame.setSize(300, 300);
       frame.setVisible(true);
   }
}

package charts;

import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
```

```
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Jitter extends JPanel implements Runnable{

    public Jitter() {}

    public void run() {
        final XYSeries data = new XYSeries("Jitter Chart");

        double[][] temp = new DataAnalyzer().getJitterData();
        double jitter = 0;
        double currentTime = temp[0][0];

        for(int i=1;i<temp.length;i++){
            if(temp[i][0]>currentTime){
                data.add(currentTime,jitter/i);
                currentTime=temp[i][0];
                jitter+=Math.abs(temp[i][1]);
            }
            else{
                jitter+=Math.abs(temp[i][1]);
            }
        }
```

```
        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Seconds",
collection, PlotOrientation.VERTICAL, true, true, true);

        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Jitter "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}

package charts;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */

import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import util.DataAnalyzer;
```

```
public class Load extends JPanel implements Runnable{

   public Load(){}

   public void run() {
      final XYSeries data = new XYSeries("Load Data");

       double[][] temp = new DataAnalyzer().getLoadData();
       double currentTime = temp[0][0];
       double firstTrafficTime = new DataAnalyzer().getFirstTrafficTime();
       double bytes = 0;

       for(int i=0;i<temp.length;i++){
          if(temp[i][0]>currentTime){
             if(currentTime<=firstTrafficTime+0.1)
                data.add(currentTime, bytes*8/(1000));
             else
                data.add(currentTime,bytes*8/(1000*(currentTime-firstTrafficTime)));

             currentTime=temp[i][0];
             bytes+=temp[i][1];
          }
          else{
             bytes+=temp[i][1];
          }
       }

       final XYSeriesCollection collection = new XYSeriesCollection(data);
       final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbps",
collection, PlotOrientation.VERTICAL, true, true, true);
```

```
        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Load "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}

package charts;

import java.awt.Color;
import java.awt.RenderingHints;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class ScatterPlotSlow extends JPanel {

    private static final long serialVersionUID = -5045238125844119782L;
    XYSeries data;

    public ScatterPlotSlow() {
```

```
    }

    public void setData(float scatterData[][]) {
        data = new XYSeries("AsIs Jitter");
        int size = 0;
        if (scatterData.length > 0) {
            size = scatterData[0].length;
            for (int i = 0; i < size;
                    i++) {
                final double x = scatterData[0][i];
                final double y = scatterData[1][i];
                data.add(x, y);
            }
        }
    }

    public JFreeChart renderAndDisplay() {
        final XYDataset dataSet = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createScatterPlot("AsIs Jitter", "Time", "Jitter",
                dataSet, PlotOrientation.VERTICAL, true, false, false);
        final XYPlot plot = chart.getXYPlot();
        plot.setBackgroundPaint(Color.WHITE);
        XYLineAndShapeRenderer xylineandshaperenderer = (XYLineAndShapeRenderer)
plot.getRenderer();
        xylineandshaperenderer.setSeriesShape(0, new java.awt.Rectangle(-2, -2, 2, 2));
        xylineandshaperenderer.setSeriesPaint(0, Color.RED);
        chart.getRenderingHints().put(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
        return renderChartInPanel(chart);
    }
```

```
    private JFreeChart renderChartInPanel(final JFreeChart chart) {
        final ChartPanel panel = new ChartPanel(chart, true);
        panel.setPreferredSize(new java.awt.Dimension(500, 270));
        panel.setMinimumDrawHeight(10);
        panel.setMaximumDrawHeight(2000);
        panel.setMinimumDrawWidth(20);
        panel.setMaximumDrawWidth(2000);
        setVisible(true);
        return chart;
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package charts;

import Global.Settings;
import java.util.Vector;
import javax.swing.JFrame;
import javax.swing.JPanel;
import util.DataAnalyzer;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

/**
```

```
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Sent extends JPanel implements Runnable {

    public Sent() {}

    public void run() {
        final XYSeries data = new XYSeries("Sent Data");

        double[][] temp = new DataAnalyzer().getSentData();
        double currentTime = temp[0][0];
        double firstTrafficTime = new DataAnalyzer().getFirstTrafficTime();
        double bytes = 0;

        for(int i=0;i<temp.length;i++){
            if(temp[i][0]>currentTime){

                if(currentTime<=firstTrafficTime+0.1)
                    data.add(currentTime, bytes*8/(1000*0.1));
                else
                    data.add(currentTime,bytes*8/(1000*(currentTime-firstTrafficTime)));

                currentTime=temp[i][0];
                bytes+=temp[i][1];
            }
            else{
                bytes+=temp[i][1];
            }
        }
```

```
        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbps",
collection, PlotOrientation.VERTICAL, true, true, true);



        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Sent "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }

}


package charts;

import Global.Settings;
import javax.swing.JFrame;
import javax.swing.JPanel;
import util.DataAnalyzer;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
```

```
public class Throughput extends JPanel implements Runnable {

   public Throughput() {}

   private double calculateThroughputByTime(double[][] temp, double time) {
       double tempBytes = 0;
       for (int i = 0; i < temp.length; i++) {
           if (temp[i][0] <= time) {
               tempBytes += temp[i][1];
           }
       }
       return tempBytes;
   }

   public void run() {
      final XYSeries data = new XYSeries("Throughput Data");

       double[][] temp = new DataAnalyzer().getThroughputData();
       double pastTime = 0;
       double currentTime = temp[0][0];
       double firstTrafficTime = new DataAnalyzer().getFirstTrafficTime();

       double bytes = 0;

       for(int i=0;i<temp.length;i++){
           if(temp[i][0]>currentTime){
               if(currentTime<=firstTrafficTime)
                   data.add(currentTime, bytes*8/(1000*(currentTime-temp[0][0])));
               else
                   data.add(currentTime,bytes*8/(1000*(currentTime-firstTrafficTime)));
```

```
                pastTime=currentTime;
                currentTime=temp[i][0];
                bytes+=temp[i][1];
            }
            else{
                bytes+=temp[i][1];
            }
        }

        final XYSeriesCollection collection = new XYSeriesCollection(data);
        final JFreeChart chart = ChartFactory.createXYLineChart(null, "Time", "Kbps",
collection, PlotOrientation.VERTICAL, true, true, true);

        JFrame frame = new JFrame();
        frame.setContentPane(new ChartPanel(chart));
        frame.setTitle("Throughput "+ Settings.printSettings());
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}

package jtrcezer;

import Global.GMenu;
import Global.Prefs;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import ui.BaseMenu;
import ui.Baseview;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Main {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {

            public void run() {
                JFrame main = new JFrame();
                                        main.setBounds(0, 0, 770, 425);
                main.setLayout(new BorderLayout());
                main.add(new BaseMenu(new GMenu().getJMenu(), new
GMenu().getJMenuItems()), BorderLayout.NORTH);
                      main.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                main.setResizable(false);
                //Create main panel and attach
                JPanel base = new JPanel();
                java.net.URL imageURL = getClass().getResource("/images/jtrcezer.jpg");

                if (imageURL != null) {
                    ImageIcon icon = new ImageIcon(imageURL);
                    base.add(new JLabel(icon));
                }
                Baseview.BasePanel = base;
                main.add(Baseview.BasePanel, BorderLayout.CENTER);
```

```
            main.setTitle(Prefs.getApplicationName() + " " + Prefs.getApplicationVersion());
            main.setVisible(true);
        }
    });
  }
}

/*
 * SettingsForm.java
 *
 * Created on 8 Αυγ 2011, 6:01:50 μμ
 */
package ui.Editors;

import Global.Settings;
import charts.AsIsDrop;
import charts.AsIsEndToEndDelay;
import charts.AsIsForwarded;
import charts.AsIsLoad;
import charts.AsIsSent;
import charts.AsIsThroughput;
import charts.AsIsJitter;
import charts.Drop;
import charts.EndToEndDelay;
import charts.Forwarded;
import charts.Jitter;
import charts.Load;
import charts.Sent;
import charts.Throughput;
import util.DataAnalyzer;
```

```
/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class SettingsForm extends javax.swing.JPanel {

    /** Creates new form SettingsForm */
    public SettingsForm() {
        initComponents();
        putDataToChoiceComponent();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jLabel4 = new javax.swing.JLabel();
        jScrollPane4 = new javax.swing.JScrollPane();
        jList4 = new javax.swing.JList(new DataAnalyzer().getExistingNodes());
        CreateGraphButton = new javax.swing.JButton();
        CancelButton = new javax.swing.JButton();
```

```
jScrollPane3 = new javax.swing.JScrollPane();
jList3 = new javax.swing.JList(new DataAnalyzer().getPacketTypens());
jScrollPane2 = new javax.swing.JScrollPane();
jList2 = new javax.swing.JList(new DataAnalyzer().getPacketTypes());
jScrollPane1 = new javax.swing.JScrollPane();
jList1 = new javax.swing.JList(new DataAnalyzer().getPacketNames());
ThroughputCheckBox = new javax.swing.JCheckBox();
SendPacketsCheckBox = new javax.swing.JCheckBox();
ForwardedPacketsCheckBox = new javax.swing.JCheckBox();
DropPacketsCheckBox = new javax.swing.JCheckBox();
LoadCheckBox = new javax.swing.JCheckBox();
EndToEndDelayCheckBox = new javax.swing.JCheckBox();
JitterCheckBox = new javax.swing.JCheckBox();
TimeAverageRadioButton = new javax.swing.JRadioButton();
AsIsRadioButton = new javax.swing.JRadioButton();
choice1 = new java.awt.Choice();
label1 = new java.awt.Label();
jLabel5 = new javax.swing.JLabel();
jTextField1 = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();

setPreferredSize(new java.awt.Dimension(536, 370));

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel1.setText("Packet Names :");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel2.setText("Packet Types :");

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel3.setText("Packet Typens :");
```

```
jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);

jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel4.setText("Select Nodes :");

jScrollPane4.setViewportView(jList4);

CreateGraphButton.setText("Create Graphs");
CreateGraphButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        CreateGraphButtonActionPerformed(evt);
    }
});

CancelButton.setText("ClearForm");
CancelButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        CancelButtonActionPerformed(evt);
    }
});

jList3.setToolTipText("Used by New Wireless Format Events");
jScrollPane3.setViewportView(jList3);

jList2.setToolTipText("Used by Old/New Wireless Format Events");
jScrollPane2.setViewportView(jList2);

jList1.setToolTipText("Used by Normal Format Events");
jScrollPane1.setViewportView(jList1);
```

```
ThroughputCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11));
ThroughputCheckBox.setText("Throughput");

SendPacketsCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11));
SendPacketsCheckBox.setText("Sent Packets");

ForwardedPacketsCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11));
ForwardedPacketsCheckBox.setText("Forwarded Packets");

DropPacketsCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
DropPacketsCheckBox.setText("Dropped Packets");

LoadCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11));
LoadCheckBox.setText("Load");

EndToEndDelayCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11));
EndToEndDelayCheckBox.setText("EndToEnd Delay");

JitterCheckBox.setFont(new java.awt.Font("Tahoma", 1, 11));
JitterCheckBox.setText("Jitter");

buttonGroup1.add(TimeAverageRadioButton);
TimeAverageRadioButton.setFont(new java.awt.Font("Tahoma", 1, 11));
TimeAverageRadioButton.setForeground(new java.awt.Color(255, 51, 51));
TimeAverageRadioButton.setSelected(true);
TimeAverageRadioButton.setText("Time Average");

buttonGroup1.add(AsIsRadioButton);
AsIsRadioButton.setFont(new java.awt.Font("Tahoma", 1, 11));
AsIsRadioButton.setForeground(new java.awt.Color(255, 51, 51));
AsIsRadioButton.setText("As Is");
```

```
        label1.setFont(new java.awt.Font("Dialog", 1, 12));
        label1.setText("Network Trace Level");

        jLabel5.setFont(new java.awt.Font("Dialog", 1, 12));
        jLabel5.setText("Time Interval");

        jTextField1.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
        jTextField1.setText("0.1");
        jTextField1.setToolTipText("AsIs Throughput-Sent-Forwarded-Load");

        jLabel6.setText("sec");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2)
                    .addComponent(jLabel3))
                .addGap(29, 29, 29)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
108, Short.MAX_VALUE)
                    .addComponent(jScrollPane2, 0, 0, Short.MAX_VALUE)
                    .addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT_SIZE,
109, Short.MAX_VALUE))
```

```
            .addGap(16, 16, 16)

            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 8,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

                .addGroup(layout.createSequentialGroup()

                    .addComponent(jLabel4)

                    .addGap(18, 18, 18)

                    .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 122,
javax.swing.GroupLayout.PREFERRED_SIZE))

                .addGroup(layout.createSequentialGroup()

                    .addGap(62, 62, 62)

                    .addComponent(CancelButton)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(CreateGraphButton))

                .addGroup(layout.createSequentialGroup()


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

                        .addComponent(ThroughputCheckBox)

                        .addComponent(ForwardedPacketsCheckBox,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

                        .addComponent(LoadCheckBox))

                    .addGap(25, 25, 25)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                        .addComponent(DropPacketsCheckBox)

                        .addComponent(SendPacketsCheckBox)
```

```
                        .addComponent(EndToEndDelayCheckBox))))
                .addComponent(JitterCheckBox)
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(TimeAverageRadioButton)
                        .addComponent(jLabel5)
                        .addComponent(label1, javax.swing.GroupLayout.DEFAULT_SIZE, 131,
Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 85, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                            .addComponent(jLabel6)
                            .addGap(27, 27, 27))
                        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                            .addComponent(AsIsRadioButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 85,
Short.MAX_VALUE))
                        .addComponent(choice1, javax.swing.GroupLayout.DEFAULT_SIZE,
138, Short.MAX_VALUE))))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(57, 57, 57)
                        .addComponent(jLabel4))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(44, 44, 44)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(ThroughputCheckBox)
                    .addComponent(SendPacketsCheckBox))
                .addGap(7, 7, 7)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(ForwardedPacketsCheckBox)
                    .addComponent(DropPacketsCheckBox))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(LoadCheckBox))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(30, 30, 30)
                        .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addGap(47, 47, 47)

                        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE))

                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                        .addGroup(layout.createSequentialGroup()

                            .addGap(190, 190, 190)

                            .addComponent(EndToEndDelayCheckBox))

                        .addGroup(layout.createSequentialGroup()

                            .addGap(216, 216, 216)

                            .addComponent(JitterCheckBox)))


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

                            .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                            .addComponent(label1,
javax.swing.GroupLayout.DEFAULT_SIZE, 23, Short.MAX_VALUE)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED))

                        .addGroup(layout.createSequentialGroup()


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                            .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

                            .addComponent(jLabel6))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                                .addComponent(choice1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

                                .addGap(21, 21, 21)))


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                            .addComponent(TimeAverageRadioButton)

                            .addComponent(AsIsRadioButton))

                        .addGap(18, 18, 18)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                            .addComponent(CancelButton)

                            .addComponent(CreateGraphButton)))

                    .addComponent(jSeparator1,
javax.swing.GroupLayout.DEFAULT_SIZE, 381, Short.MAX_VALUE))))

            .addContainerGap())

        .addGroup(layout.createSequentialGroup()

            .addGap(47, 47, 47)

            .addComponent(jLabel1)

            .addGap(116, 116, 116)

            .addComponent(jLabel2)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
100, Short.MAX_VALUE)

            .addComponent(jLabel3)

            .addGap(98, 98, 98))

      );

   }// </editor-fold>


   private void CreateGraphButtonActionPerformed(java.awt.event.ActionEvent evt) {

      configureSettings();
```

```
    String[] SelectedNodes = new String[jList4.getSelectedValues().length];

    for (int i = 0; i < SelectedNodes.length; i++) {
        SelectedNodes[i] = String.valueOf(jList4.getSelectedValues()[i]);
    }

    for (String node : SelectedNodes) {
        if (node.equals("All Nodes")) {
            Settings.addAcceptableNode(null);
            createSelectedChartsbyNode();
        } else {
            Settings.addAcceptableNode(node);
            createSelectedChartsbyNode();
        }

    }
}

private void CancelButtonActionPerformed(java.awt.event.ActionEvent evt) {
    jList1.clearSelection();
    jList2.clearSelection();
    jList3.clearSelection();
    jList4.clearSelection();
    AsIsRadioButton.setSelected(false);
    CreateGraphButton.setSelected(false);
    DropPacketsCheckBox.setSelected(false);
    EndToEndDelayCheckBox.setSelected(false);
    ForwardedPacketsCheckBox.setSelected(false);
    JitterCheckBox.setSelected(false);
    LoadCheckBox.setSelected(false);
    SendPacketsCheckBox.setSelected(false);
```

```
    ThroughputCheckBox.setSelected(false);
    TimeAverageRadioButton.setSelected(false);

}
// Variables declaration - do not modify
private javax.swing.JRadioButton AsIsRadioButton;
private javax.swing.JButton CancelButton;
private javax.swing.JButton CreateGraphButton;
private javax.swing.JCheckBox DropPacketsCheckBox;
private javax.swing.JCheckBox EndToEndDelayCheckBox;
private javax.swing.JCheckBox ForwardedPacketsCheckBox;
private javax.swing.JCheckBox JitterCheckBox;
private javax.swing.JCheckBox LoadCheckBox;
private javax.swing.JCheckBox SendPacketsCheckBox;
private javax.swing.JCheckBox ThroughputCheckBox;
private javax.swing.JRadioButton TimeAverageRadioButton;
private javax.swing.ButtonGroup buttonGroup1;
private java.awt.Choice choice1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JList jList1;
private javax.swing.JList jList2;
private javax.swing.JList jList3;
private javax.swing.JList jList4;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
```

```
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JTextField jTextField1;
    private java.awt.Label label1;
    // End of variables declaration


    private void createSelectedChartsbyNode() {


        if (ThroughputCheckBox.isSelected()) {
            if (TimeAverageRadioButton.isSelected()) {
                Throughput throughput = new Throughput();
                throughput.run();
            } else if (AsIsRadioButton.isSelected()) {
                AsIsThroughput asisthroughput = new AsIsThroughput();
                asisthroughput.run();
            }
        }
        if (ForwardedPacketsCheckBox.isSelected()) {
            if (TimeAverageRadioButton.isSelected()) {
                Forwarded forwarded = new Forwarded();
                forwarded.run();
            } else if (AsIsRadioButton.isSelected()) {
                AsIsForwarded forwarded = new AsIsForwarded();
                forwarded.run();
            }
        }
        if (LoadCheckBox.isSelected()) {
            if (TimeAverageRadioButton.isSelected()) {
                Load load = new Load();
                load.run();
            } else if (AsIsRadioButton.isSelected()) {
```

```
            AsIsLoad asisload = new AsIsLoad();
            asisload.run();
        }
    }
    if (SendPacketsCheckBox.isSelected()) {
        if (TimeAverageRadioButton.isSelected()) {
            Sent sent = new Sent();
            sent.run();
        } else if (AsIsRadioButton.isSelected()) {
            AsIsSent asissend = new AsIsSent();
            asissend.run();
        }
    }
    if (DropPacketsCheckBox.isSelected()) {
        if (TimeAverageRadioButton.isSelected()) {
            Drop droppped = new Drop();
            droppped.run();
        } else if (AsIsRadioButton.isSelected()) {
            AsIsDrop asisdrop = new AsIsDrop();
            asisdrop.run();
        }
    }
    if (EndToEndDelayCheckBox.isSelected()) {
        if (TimeAverageRadioButton.isSelected()) {
            EndToEndDelay endtoenddelay = new EndToEndDelay();
            endtoenddelay.run();
        } else if (AsIsRadioButton.isSelected()) {
            AsIsEndToEndDelay asisendtoenddelay = new AsIsEndToEndDelay();
            asisendtoenddelay.run();
        }
    }
```

```
        if (JitterCheckBox.isSelected()) {
            if (TimeAverageRadioButton.isSelected()) {
                Jitter jitter = new Jitter();
                jitter.run();
            } else if (AsIsRadioButton.isSelected()) {
                AsIsJitter asisjitter = new AsIsJitter();
                asisjitter.run();
            }
        }
    }

    private void configureSettings() {

        if (!jList1.isSelectionEmpty()) {
            Settings.AcceptablePacketNames = new String[jList1.getSelectedValues().length];
            for (int i = 0; i < jList1.getSelectedValues().length; i++) {
                Settings.AcceptablePacketNames[i] =
String.valueOf(jList1.getSelectedValues()[i]);
            }
        } else {
            Settings.AcceptablePacketNames = null;
        }

        if (!jList2.isSelectionEmpty()) {
            Settings.AcceptablePacketTypes = new String[jList2.getSelectedValues().length];
            for (int i = 0; i < jList2.getSelectedValues().length; i++) {
                Settings.AcceptablePacketTypes[i] =
String.valueOf(jList2.getSelectedValues()[i]);
            }

        } else {
            Settings.AcceptablePacketTypes = null;
```

```
        }



        if (!jList3.isSelectionEmpty()) {
            Settings.AcceptablePacketTypen = new String[jList3.getSelectedValues().length];
            for (int i = 0; i < jList3.getSelectedValues().length; i++) {
                Settings.AcceptablePacketTypen[i] =
String.valueOf(jList3.getSelectedValues()[i]);
            }


        } else {
            Settings.AcceptablePacketTypen = null;
        }


        if (choice1.getSelectedIndex() != -1) {
            Settings.AcceptableNetworkTraceLevel = new String[1];
            Settings.AcceptableNetworkTraceLevel[0] = choice1.getSelectedItem();
        } else {
            Settings.AcceptableNetworkTraceLevel = null;
        }


        Settings.TimeInterval = Double.parseDouble(jTextField1.getText());
    }


    private void putDataToChoiceComponent() {


        for (String networkTraceLevel : new DataAnalyzer().getAllNetworkTraceLevels()) {
            choice1.add(networkTraceLevel);
        }
    }
}
```

```
/*
 * About.java
 *
 * Created on 6 Σεπ 2011, 5:45:21 μμ
 */

package ui;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class About extends javax.swing.JFrame {

    /** Creates new form About */
    public About() {
        initComponents();
        setTitle("About");
        setAlwaysOnTop(true);
        setVisible(true);
        setDefaultCloseOperation(this.DISPOSE_ON_CLOSE);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
```

```
        jScrollPane1 = new javax.swing.JScrollPane();

        jTextPane1 = new javax.swing.JTextPane();


        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);


        jTextPane1.setEditable(false);

        jTextPane1.setText(" jTrcezer \n\nThis application is result of Information Technology Department's thesis \nat Alexander Technological Institute of Thessaloniki.\n\nDeveloped by : Christos - Charalampos Papagiannidis\nSupervised by : Thomas Lagkas\n\njTrcezer is licensed for use under version 3 of the GNU General Public License.\n\nRelease Date : 6th September 2011\nWebsite :\n\n\n");

        jScrollPane1.setViewportView(jTextPane1);


        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

        getContentPane().setLayout(layout);

        layout.setHorizontalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(layout.createSequentialGroup()

                .addContainerGap()

                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 380, Short.MAX_VALUE)

                .addContainerGap())

        );

        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(layout.createSequentialGroup()

                .addContainerGap()

                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 278, Short.MAX_VALUE)

                .addContainerGap())

        );
```

```
        pack();
    }// </editor-fold>

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new About().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextPane jTextPane1;
    // End of variables declaration

}

package ui;

import Global.GMenu;
import java.awt.Component;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.event.MenuEvent;
```

```
import javax.swing.event.MenuListener;


/**
 *
 * @author Christos-Charalampos Papagiannidis
 */


public class BaseMenu extends JMenuBar{
   public BaseMenu(){}


   public BaseMenu(ArrayList<JMenu> menus) {
      for(int i=0;i<menus.size();i++) {
         this.add(menus.get(i));
      }
   }


   public BaseMenu(ArrayList<JMenu> menus,ArrayList<JMenuItem> menuItems) {
      Iterator itrMenu = menus.iterator();
      Iterator itr =menuItems.iterator();
      for(int i = 0 ; i < menus.size() ; i++) {
        add(menus.get(i));
        while(itr.hasNext()) {
           Object tempItem =itr.next();
           if(!((JMenuItem)tempItem).getText().equals(new
GMenu().getMenuSeperator().getText()))
              menus.get(i).add((JMenuItem) tempItem);
           else{
              break;
           }
        }
      }
```

```
    JMenu about = new JMenu("About");
    about.addMenuListener(new MenuListener() {

        public void menuSelected(MenuEvent e) {
            new About();
        }

        public void menuDeselected(MenuEvent e) {
          //Do nothing
        }

        public void menuCanceled(MenuEvent e) {
           //Do nothing
        }
    });
     add(about);
}



@Override
public Component add(Component comp) {
    return super.add(comp);
}



@Override
public JMenu add(JMenu c) {
    return super.add(c);
}
```

```
    public void addNewMenu(JMenu newmenu) {
        add(newmenu);
    }

    public void addNewMenuItem(JMenu targetMenu, JMenuItem newMenuItem) {
        targetMenu.add(newMenuItem);
    }

    public void addSeperator(JMenu targetMenu) {
        targetMenu.addSeparator();
    }

}

package ui;

import java.awt.Color;
import javax.swing.JPanel;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class Baseview {

   public static JPanel BasePanel = new JPanel();

    public Baseview() {
        init();
    }
```

```
    public static void setBaseview(JPanel newPanel){
        BasePanel = newPanel;
        BasePanel.doLayout();
        BasePanel.updateUI();
    }

    private void init() {
        BasePanel.setBackground(Color.red);
    }
}

package ui;

import Global.Trace;
import java.io.File;
import javax.swing.JFileChooser;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */

public class OpenFileDialog {

    private static JFileChooser js;

    public OpenFileDialog() throws Exception {
        init();
    }
```

```
private synchronized void init() throws Exception {
    js = new JFileChooser();
    js.setCurrentDirectory(new File("C:\\Users\\Chrison\\Desktop\\ns_stuff\\ns_stuff"));
    HandleAction(js.showOpenDialog(js));
}

private void HandleAction(int act) throws Exception {
    switch (act) {
        case JFileChooser.APPROVE_OPTION:
            approvePressed();
            break;
        case JFileChooser.CANCEL_OPTION:
            cancelPressed();
            break;
        case JFileChooser.ERROR_OPTION:
            errorPressed();
            break;
    }

}

private void approvePressed() throws Exception {
    new Trace(js.getSelectedFile());
}

private void cancelPressed() {
    System.out.println("File canceled");
}

private void errorPressed() {
    System.out.println("There was an unexpected error");
```

```
    }

    public synchronized File getChosenFile() {
        if (js.getClass().equals(JFileChooser.class)) {
            return js.getSelectedFile();
        } else {
            return null;
        }
    }

    public OpenFileDialog getInstance() {
        return this;
    }
}

package ui;

import java.awt.Font;
import javax.swing.BoxLayout;
import javax.swing.JLabel;
import javax.swing.JProgressBar;
import javax.swing.JWindow;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class ProgressBar extends JWindow implements Runnable{

    public ProgressBar(String message) {
        this.setSize(230, 50);
```

```
        this.setAlwaysOnTop(true);
        this.setLocation(WIDTH, WIDTH);
        this.setLayout(new BoxLayout(this.getContentPane(),BoxLayout.Y_AXIS));

        JLabel Message = new JLabel(message);
        Message.setFont(new Font("Serif", Font.BOLD, 20));



        JProgressBar progress = new JProgressBar();
        progress.setSize(200, 50);
        progress.setIndeterminate(true);

        this.getContentPane().add(Message);
        this.getContentPane().add(progress);

        this.setVisible(true);

    }

    public void run() {
        this.setVisible(false);
        System.out.println("Progress Bar disposed");
    }

}

package util;

/**
 *
 * @author Christos-Charalampos Papagiannidis
```

```
 */
public class ArrayUtils {


    public static boolean contains(double[][] array, int value , int searchColumn){
        for( int i=0;i<array.length;i++ ){
            if(array[i][searchColumn] == value)
                return true;
        }
        return false;
    }


    public static double searchOneColumnAndReturn(double[][] array, double value, int
searchColumn, int returnColumn){
        for( int i=0;i<array.length;i++ ){
            if(array[i][searchColumn] == value)
                return array[i][returnColumn];
        }


        return 0;


    }


     public static double searchTwoColumnAndReturn(double[][] array, double
value1,double value2, int searchColumn1,int searchColumn2, int returnColumn){
        for( int i=0;i<array.length;i++ ){
            if(array[i][searchColumn1] == value1 && array[i][searchColumn2] == value2 )
                return array[i][returnColumn];
        }


        return -1;


    }
```

```
    public static double getMinimumValueOfColumn(double[][] array,int columnNumber){
        double min = array[0][columnNumber];
        for (int i=1;i<array.length;i++) {
            if(min>array[i][columnNumber]){
                min = array[i][columnNumber];
            }
        }

        return min;
    }

    public static double getMaximumValueOfColumn(double[][] array,int columnNumber){
        double max = array[0][columnNumber];
        for (int i=1;i<array.length;i++) {
            if(max<array[i][columnNumber]){
                max = array[i][columnNumber];
            }
        }

        return max;
    }
}

package util;

import Data.Event;
import Data.NewWireless;
import Data.NormalEvent;
import Data.OldWireless;
import Global.Debug;
```

```
import Global.Settings;
import Global.Trace;
import java.util.Vector;

/**
 *
 * @author Christos-Charalampos Papagiannidis
 */
public class DataAnalyzer {

    private Vector<Event> Data = null;

    public DataAnalyzer() {
        Data = Trace.getData();
    }

    public double[][] getSentData() {
        double[][] temp = new double[Data.size()][2];
        if (Debug.analyze) {
            Debug.print("Sent temp array length: " + temp.length, Settings.DebugDestination);
        }

        //Filter Trace Data to get only the Events actually needed.
        int j = 0;
        for (int i = 0; i < Data.size(); i++) {
            if (Debug.analyze) {
                Debug.print(Data.get(i).toString(), Settings.DebugDestination);
            }

            if (Data.get(i) instanceof NormalEvent) {
                temp[j][0] = ((NormalEvent) Data.get(i)).getTime();
```

```
            if (((NormalEvent) Data.get(i)).isDequeue()
                    && !((NormalEvent) Data.get(i)).isForward()
                    && Settings.isAcceptableNode(((NormalEvent)
Data.get(i)).getSourceNode())
                    && Settings.isAcceptablePacketName(((NormalEvent)
Data.get(i)).getPacketName())) {
                temp[j][1] = ((NormalEvent) Data.get(i)).getPacketSize();
            } else {
                temp[j][1] = 0;
            }

            if (Debug.analyze) {
                Debug.print("A Normal Sent packet Detected", Settings.DebugDestination);
            }

            j++;

        } else if (Data.get(i) instanceof NewWireless && ((NewWireless)
Data.get(i)).getIp() != null) {
            temp[j][0] = ((NewWireless) Data.get(i)).getTime();

            if (((NewWireless) Data.get(i)).isSent()
                    && Settings.isAcceptableNode(((NewWireless) Data.get(i)).getNodeId())
                    && Settings.isAcceptableNetworkTraceLevel(((NewWireless)
Data.get(i)).getNetworkTraceLevel())
                    && Settings.isAcceptablePacketType(((NewWireless)
Data.get(i)).getPacketType())
                    && Settings.isAcceptablePacketTypen(((NewWireless)
Data.get(i)).getPacketTypen())) {
                temp[j][1] = ((NewWireless) Data.get(i)).getIp().getPacketSize();
            } else {
                temp[j][1] = 0;
```

```
            }


            if (Debug.analyze) {

                Debug.print("A NewWireless Sent packet Detected",
Settings.DebugDestination);

            }


            if (Debug.analyze) {

                Debug.print("\n" + ((NewWireless) Data.get(i)).toString() + "\n",
Settings.DebugDestination);

                Debug.print(temp[j][0] + ":" + temp[j][1], Settings.DebugDestination);

                Debug.print("============================SENT
DATA==============================", Settings.DebugDestination);

            }


            j++;


        } else if (Data.get(i) instanceof OldWireless && ((OldWireless) Data.get(i)).getIp()
!= null) {

            temp[j][0] = ((OldWireless) Data.get(i)).getTime();


            if (((OldWireless) Data.get(i)).isSent()

                    && Settings.isAcceptableNode(((OldWireless) Data.get(i)).getNodeId())

                    && Settings.isAcceptableNetworkTraceLevel(((OldWireless)
Data.get(i)).getTraceName())

                    && (Settings.isAcceptablePacketType(((OldWireless)
Data.get(i)).getPacketType()))) {

                temp[j][1] = ((OldWireless) Data.get(i)).getPacketSize();

            } else {

                temp[j][1] = 0;

            }


            if (Debug.analyze) {
```

```
                Debug.print("An OldWireless Sent packet Detected",
Settings.DebugDestination);
            }


            j++;
        }
        /*
         * Add code for any new Trace Format that jTrcezer will support in the future.
         * Filter the traces in order to keep only the data is needed to calculate Throughput
         */


    }
    return temp;
}


public double[][] getForwardedData() {
    double[][] temp = new double[Data.size()][2];
    if (Debug.analyze) {
        Debug.print("Forwarded temp array length: " + temp.length,
Settings.DebugDestination);
    }


    //Filter Trace Data to get only the Events actually needed.
    int j = 0;
    for (int i = 0; i < Data.size(); i++) {
        if (Debug.analyze) {
            Debug.print(Data.get(i).toString(), Settings.DebugDestination);
        }


        if (Data.get(i) instanceof NormalEvent) {
            temp[j][0] = ((NormalEvent) Data.get(i)).getTime();
```

```
            if (((NormalEvent) Data.get(i)).isDequeue()

                    && ((NormalEvent) Data.get(i)).isForward()

                    && Settings.isAcceptableNode(((NormalEvent)
Data.get(i)).getSourceNode())

                  //&&
!((NormalEvent)Data.get(i)).getSourceAddress().equals(String.valueOf(((NormalEvent)
Data.get(i)).getSourceNode())) &&

                    && Settings.isAcceptablePacketName(((NormalEvent)
Data.get(i)).getPacketName())) {

                temp[j][1] = ((NormalEvent) Data.get(i)).getPacketSize();

            } else {

                temp[j][1] = 0;

            }



            if (Debug.analyze) {

                Debug.print("A Normal Forwarded packet Detected",
Settings.DebugDestination);

            }



            j++;



        } else if (Data.get(i) instanceof NewWireless && ((NewWireless)
Data.get(i)).getIp() != null) {

            temp[j][0] = ((NewWireless) Data.get(i)).getTime();



            if ((((NewWireless) Data.get(i)).isForward())

                    && Settings.isAcceptableNode(((NewWireless) Data.get(i)).getNodeId())

                    && Settings.isAcceptableNetworkTraceLevel(((NewWireless)
Data.get(i)).getNetworkTraceLevel())

                    && Settings.isAcceptablePacketType(((NewWireless)
Data.get(i)).getPacketType())

                    && Settings.isAcceptablePacketTypen(((NewWireless)
Data.get(i)).getPacketTypen())) {

                temp[j][1] = ((NewWireless) Data.get(i)).getIp().getPacketSize();

            } else {
```

```
                temp[j][1] = 0;
            }

            if (Debug.analyze) {
                Debug.print("Forwarded New Wireless", Settings.DebugDestination);
            }

            j++;

        } else if (Data.get(i) instanceof OldWireless) {
            temp[j][0] = ((OldWireless) Data.get(i)).getTime();

            if (((OldWireless) Data.get(i)).isForward()
                    && Settings.isAcceptableNode(((OldWireless) Data.get(i)).getNodeId())
                    && Settings.isAcceptableNetworkTraceLevel(((OldWireless)
Data.get(i)).getTraceName())
                    && Settings.isAcceptablePacketType(((OldWireless)
Data.get(i)).getPacketType())) {

                temp[j][1] = ((OldWireless) Data.get(i)).getPacketSize();

            } else {
                temp[j][1] = 0;
            }

            if (Debug.analyze) {
                Debug.print("Forwarded normal", Settings.DebugDestination);
            }
            j++;
        }

        /*
```

```
            * Add code for any new Trace Format that jTrcezer will support in the future.
            * Filter the traces in order to keep only the data is needed to calculate Throughput
            */


        }


        return temp;
    }


    public double getFirstTrafficTime() {
        int i = 0;
        while (i < Data.size()) {
            if (Data.get(i) instanceof NormalEvent) {
                if (((NormalEvent) Data.get(i)).isEnqueue()
                        && Settings.isAcceptableNode(((NormalEvent)
Data.get(i)).getSourceNode())
                        && Settings.isAcceptablePacketName(((NormalEvent)
Data.get(i)).getPacketName())) {
                    return ((NormalEvent) Data.get(i)).getTime();
                }
            } else if (Data.get(i) instanceof NewWireless) {
                if (((NewWireless) Data.get(i)).isSent()
                        && (((NewWireless) Data.get(i)).getIp() != null
                        && Settings.isAcceptablePacketType(((NewWireless)
Data.get(i)).getPacketType())
                        && Settings.isAcceptablePacketTypen(((NewWireless)
Data.get(i)).getPacketTypen()))) {
                    return ((NewWireless) Data.get(i)).getTime();
                }
            } else if (Data.get(i) instanceof OldWireless) {
                if (((OldWireless) Data.get(i)).isSent()
                        && Settings.isAcceptableNode(((OldWireless) Data.get(i)).getNodeId())
```

```
                && (Settings.isAcceptablePacketType(((OldWireless)
Data.get(i)).getPacketType()))) {
                return ((OldWireless) Data.get(i)).getTime();
            }
        }
        i++;
    }

    return 0;
}

public double[][] getThroughputData() {
    double[][] temp = new double[Data.size()][2];

    //Filter Trace Data to get only the Events actually needed.
    int j = 0;
    for (int i = 0; i < Data.size(); i++) {

        if (Data.get(i) instanceof NormalEvent) {
            temp[j][0] = ((NormalEvent) Data.get(i)).getTime();

            if (((NormalEvent) Data.get(i)).isReceive()
                && Settings.isAcceptableNode(((NormalEvent)
Data.get(i)).getDestinationNode())
                && ((NormalEvent) Data.get(i)).getDestinationNode() ==
Integer.parseInt(((NormalEvent) Data.get(i)).getDestinationAddress())
                && Settings.isAcceptablePacketName(((NormalEvent)
Data.get(i)).getPacketName())) {
                temp[j][1] = ((NormalEvent) Data.get(i)).getPacketSize();
            } else {
                temp[j][1] = 0;
            }
```

```
            j++;

        } else if (Data.get(i) instanceof NewWireless && ((NewWireless)
Data.get(i)).getIp() != null ) {

            temp[j][0] = ((NewWireless) Data.get(i)).getTime();



            if (((NewWireless) Data.get(i)).isReceive()

                && ((NewWireless) Data.get(i)).getNodeId() == ((NewWireless)
Data.get(i)).getIp().getDestinationIpAddress()

                && Settings.isAcceptableNode(((NewWireless) Data.get(i)).getNodeId())

                && Settings.isAcceptablePacketType(((NewWireless)
Data.get(i)).getPacketType())

                && Settings.isAcceptablePacketTypen(((NewWireless)
Data.get(i)).getPacketTypen())

                && Settings.isAcceptableNetworkTraceLevel(((NewWireless)
Data.get(i)).getNetworkTraceLevel())) {

                temp[j][1] = ((NewWireless) Data.get(i)).getIp().getPacketSize();

            } else {

                temp[j][1] = 0;

            }



            if (Debug.analyze) {

                Debug.print("\n" + ((NewWireless) Data.get(i)).toString() + "\n",
Settings.DebugDestination);

                Debug.print(temp[j][0] + ":" + temp[j][1], Settings.DebugDestination);

                Debug.print("=========================THROUGHPUT
DATA==================================", Settings.DebugDestination);

            }



            j++;



        } else if (Data.get(i) instanceof OldWireless) {

            temp[j][0] = ((OldWireless) Data.get(i)).getTime();
```

```
            if (((OldWireless) Data.get(i)).isReceive() && ((OldWireless) Data.get(i)).getIp()
!= null) {

                if (Settings.isAcceptableNode(((OldWireless) Data.get(i)).getNodeId())

                        && ((OldWireless) Data.get(i)).getNodeId() == ((OldWireless)
Data.get(i)).getIp().getDestinationIpAddress()

                        && Settings.isAcceptableNetworkTraceLevel(((OldWireless)
Data.get(i)).getTraceName())

                        && (Settings.isAcceptablePacketType(((OldWireless)
Data.get(i)).getPacketType()))) {

                    temp[j][1] = ((OldWireless) Data.get(i)).getPacketSize();

                } else {

                    temp[j][1] = 0;

                }

                j++;

            }

        }


        /*

         * Add code for any new Trace Format that jTrcezer will support in the future.

         * Filter the Events in order to keep only the data is needed to calculate
Throughput

         */

    }

    return temp;


}


public double[][] getLoadData() {

    double[][] temp = new double[Data.size()][2];


    //Filter Trace Data to get only the Events actually needed.

    int j = 0;

    for (int i = 0; i < Data.size(); i++) {
```

```
        if (Data.get(i) instanceof NormalEvent) {
            temp[j][0] = ((NormalEvent) Data.get(i)).getTime();

            if (((NormalEvent) Data.get(i)).isEnqueue()
                    && !((NormalEvent) Data.get(i)).isForward()
                    && Settings.isAcceptableNode(((NormalEvent)
Data.get(i)).getSourceNode())
                    && Settings.isAcceptablePacketName(((NormalEvent)
Data.get(i)).getPacketName())) {
                temp[j][1] = ((NormalEvent) Data.get(i)).getPacketSize();
            } else {
                temp[j][1] = 0;
            }
            j++;

        } else if (Data.get(i) instanceof NewWireless && ((NewWireless)
Data.get(i)).getIp() != null) {
            temp[j][0] = ((NewWireless) Data.get(i)).getTime();

            if (((NewWireless) Data.get(i)).isSent()
                    && Settings.isAcceptableNode(((NewWireless) Data.get(i)).getNodeId())
                    && Settings.isAcceptablePacketType(((NewWireless)
Data.get(i)).getPacketType())
                    && Settings.isAcceptableNetworkTraceLevel(((NewWireless)
Data.get(i)).getNetworkTraceLevel())
                    && Settings.isAcceptablePacketTypen(((NewWireless)
Data.get(i)).getPacketTypen())) {
                temp[j][1] = ((NewWireless) Data.get(i)).getIp().getPacketSize();
            } else {
                temp[j][1] = 0;
            }
```

```
            if (Debug.analyze) {
                Debug.print("\n" + ((NewWireless) Data.get(i)).toString() + "\n",
Settings.DebugDestination);
                Debug.print(temp[j][0] + ":" + temp[j][1], Settings.DebugDestination);
                Debug.print("=========================LOAD
DATA==================================", Settings.DebugDestination);
            }

            j++;

        } else if (Data.get(i) instanceof OldWireless) {
            temp[j][0] = ((OldWireless) Data.get(i)).getTime();

            if (((OldWireless) Data.get(i)).isSent()
                    && Settings.isAcceptableNode(((OldWireless) Data.get(i)).getNodeId())
                    && Settings.isAcceptableNetworkTraceLevel(((OldWireless)
Data.get(i)).getTraceName())
                    && (Settings.isAcceptablePacketType(((OldWireless)
Data.get(i)).getPacketType()))) {
                temp[j][1] = ((OldWireless) Data.get(i)).getPacketSize();
            } else {
                temp[j][1] = 0;
            }
            j++;
        }

        /*
         * Add code for any new Trace Format that jTrcezer will support in the future.
         * Filter the traces in order to keep only the data is needed to calculate Load
         */
    }
    //trim table to its real size.
    return temp;
```

```
    }


    public double[][] getDropData() {
        double[][] temp = new double[Data.size()][2];


        //Filter Trace Data to get only the Events actually needed.
        int j = 0;
        for (int i = 0; i < Data.size(); i++) {


            if (Data.get(i) instanceof NormalEvent) {
                temp[j][0] = ((NormalEvent) Data.get(i)).getTime();


                if (Debug.analyze) {
                    Debug.print("\n " + ((NormalEvent) Data.get(i)).getAbbreviation() + " || " +
((NormalEvent) Data.get(i)).isDrop() + " || " + ((NormalEvent)
Data.get(i)).getPacketName() + " || " + ((NormalEvent)
Data.get(i)).getPacketName().equals("cbr"), Settings.DebugDestination);
                    Debug.print(" || result = " + (((NormalEvent) Data.get(i)).isDrop() &&
Settings.isAcceptablePacketName(((NormalEvent) Data.get(i)).getPacketName())),
Settings.DebugDestination);
                }


                if (((NormalEvent) Data.get(i)).isDrop()
                        && Settings.isAcceptableNode(((NormalEvent)
Data.get(i)).getDestinationNode())
                        && Settings.isAcceptablePacketName(((NormalEvent)
Data.get(i)).getPacketName())) {
                    temp[j][1] = ((NormalEvent) Data.get(i)).getPacketSize();
                } else {
                    temp[j][1] = 0;
                }


                j++;
```

```
        } else if (Data.get(i) instanceof NewWireless && ((NewWireless)
Data.get(i)).getIp() != null ) {

            temp[j][0] = ((NewWireless) Data.get(i)).getTime();



            if (((NewWireless) Data.get(i)).isDrop()

                && Settings.isAcceptableNode(((NewWireless) Data.get(i)).getNodeId())

                && Settings.isAcceptableNetworkTraceLevel(((NewWireless)
Data.get(i)).getNetworkTraceLevel())

                && Settings.isAcceptablePacketType(((NewWireless)
Data.get(i)).getPacketType())

                && Settings.isAcceptablePacketTypen(((NewWireless)
Data.get(i)).getPacketTypen())) {

                temp[j][1] = ((NewWireless) Data.get(i)).getIp().getPacketSize();

            } else {

                temp[j][1] = 0;

            }



            if (Debug.analyze) {

                Debug.print("\n" + ((NewWireless) Data.get(i)).toString() + "\n",
Settings.DebugDestination);

                Debug.print(temp[j][0] + ":" + temp[j][1], Settings.DebugDestination);

                Debug.print("=========================DROP
DATA==================================", Settings.DebugDestination);

            }



            j++;



        } else if (Data.get(i) instanceof OldWireless) {

            temp[j][0] = ((OldWireless) Data.get(i)).getTime();



            if (((OldWireless) Data.get(i)).isDrop()

                && Settings.isAcceptableNode(((OldWireless) Data.get(i)).getNodeId())

                && Settings.isAcceptableNetworkTraceLevel(((OldWireless)
Data.get(i)).getTraceName())
```

```
                    && (Settings.isAcceptablePacketType(((OldWireless)
Data.get(i)).getPacketType()))) {
                temp[j][1] = ((OldWireless) Data.get(i)).getPacketSize();
            } else {
                temp[j][1] = 0;
            }
            j++;
        }

        /*
         * Add code for any new Trace Format that jTrcezer will support in the future.
         * Filter the traces in order to keep only the data is needed to calculate Throughput
         */
    }
    return temp;
}

public Vector<String> getExistingNodes() {
    Vector<String> temp = new Vector();
    temp.add("All Nodes");
    for (Event event : Data) {
        String node = null;
        String node1 = null;
        if (event instanceof NormalEvent) {
            node = String.valueOf(((NormalEvent) event).getSourceNode());
            node1 = String.valueOf(((NormalEvent) event).getDestinationNode());
        } else if (event instanceof NewWireless) {
            node = String.valueOf(((NewWireless) event).getNodeId());
        } else if (event instanceof OldWireless) {
            node = String.valueOf(((OldWireless) event).getNodeId());
        }
```

```
            /*
             * Add code to Support Other Formats
             */

            if (!temp.contains(node)) {
                temp.add(node);
            }
            if (!temp.contains(node1) && node1 != null) {
                temp.add(node1);
            }
        }
        temp.trimToSize();

        if (Debug.analyze) {
            Debug.print("-------NODE IDs--------", Settings.DebugDestination);
            StringUtils.printVector(temp);
        }

        return temp;
    }

    public double[][] getDelayData() {
        double[][] sentArray = new double[Data.size()][3]; //{Time,Unique Packet
ID,Destination Address}
        double[][] recvArray = new double[Data.size()][3];
        double[][] delays = null;

        int SentPacketCounter = 0;

        for (int i = 0; i < Data.size(); i++) {
```

```
        if (Data.get(i) instanceof NormalEvent) {
            NormalEvent NormalTemp = (NormalEvent) Data.get(i);
            if (NormalTemp.isDequeue()
                    && Settings.isAcceptableNode(NormalTemp.getSourceNode())
                    && Double.parseDouble(NormalTemp.getDestinationAddress()) >= 0
                    && Settings.isAcceptablePacketName(NormalTemp.getPacketName())
                    && NormalTemp.getSourceNode() ==
Integer.parseInt(NormalTemp.getSourceAddress())
                    && !ArrayUtils.contains(sentArray, NormalTemp.getUniquePacketId(), 1)) {
                sentArray[i][0] = NormalTemp.getTime();
                sentArray[i][1] = NormalTemp.getUniquePacketId();
                sentArray[i][2] = Double.parseDouble(NormalTemp.getDestinationAddress());
                SentPacketCounter++;
                //System.out.println(i+ " Found sent event at" + sentArray[i][0]);
            } else if (NormalTemp.isReceive()
                        &&
Settings.isAcceptablePacketName(NormalTemp.getPacketName())
                        &&
NormalTemp.getDestinationAddress().equals(String.valueOf(NormalTemp.getDestination
Node()))) {
                recvArray[i][0] = NormalTemp.getTime();
                recvArray[i][1] = NormalTemp.getUniquePacketId();
                recvArray[i][2] = NormalTemp.getDestinationNode();
                //System.out.println("Found received event at" + recvArray[i][0]);

            }
        } else if (Data.get(i) instanceof NewWireless) {
            NewWireless NewTemp = (NewWireless) Data.get(i);

            if (NewTemp.isSent()
                    && Settings.isAcceptableNode(NewTemp.getNodeId())
```

```
                    &&
Settings.isAcceptableNetworkTraceLevel(NewTemp.getNetworkTraceLevel())
                    && Settings.isAcceptablePacketType(NewTemp.getPacketType())
                    && Settings.isAcceptablePacketTypen(NewTemp.getPacketTypen())) {
                sentArray[i][0] = NewTemp.getTime();
                sentArray[i][1] = NewTemp.getIp().getUniqueId();
                sentArray[i][2] = NewTemp.getIp().getDestinationIpAddress();
                SentPacketCounter++;
            } else if (NewTemp.isReceive()
                    &&
Settings.isAcceptableNetworkTraceLevel(NewTemp.getNetworkTraceLevel())
                    && Settings.isAcceptablePacketType(NewTemp.getPacketType())
                    && Settings.isAcceptablePacketTypen(NewTemp.getPacketTypen())) {
                recvArray[i][0] = NewTemp.getTime();
                recvArray[i][1] = NewTemp.getIp().getUniqueId();
                recvArray[i][2] = NewTemp.getNodeId();
            }

        } else if (Data.get(i) instanceof OldWireless) {
            OldWireless OldTemp = (OldWireless) Data.get(i);

            if (OldTemp.isSent()
                    && Settings.isAcceptableNode(OldTemp.getNodeId())
                    && Settings.isAcceptableNetworkTraceLevel(OldTemp.getTraceName())
                    && Settings.isAcceptablePacketType(OldTemp.getPacketType())) {
                sentArray[i][0] = OldTemp.getTime();
                sentArray[i][1] = OldTemp.getEventIdentifier();
                sentArray[i][2] = OldTemp.getIp().getDestinationIpAddress();
                SentPacketCounter++;
            } else if (OldTemp.isReceive()
                    && Settings.isAcceptableNetworkTraceLevel(OldTemp.getTraceName())
                    && Settings.isAcceptablePacketType(OldTemp.getPacketType())) {
```

```
                recvArray[i][0] = OldTemp.getTime();

                recvArray[i][1] = OldTemp.getEventIdentifier();

                recvArray[i][2] = OldTemp.getNodeId();

            }

        }

    }

    if (Debug.analyze) {


        Debug.print("===DataAnalyzer.getDelayData()=== \n\n\n\n Sent Array",
Settings.DebugDestination);


        for (int i = 0; i < sentArray.length; i++) {

            Debug.print(sentArray[i][0] + " " + sentArray[i][1] + " " + sentArray[i][2],
Settings.DebugDestination);

        }


        Debug.print("\n\n\n\n Receive Array", Settings.DebugDestination);


        for (int i = 0; i < sentArray.length; i++) {

            Debug.print(recvArray[i][0] + " " + recvArray[i][1] + " " + recvArray[i][2],
Settings.DebugDestination);

        }

    }


    delays = new double[SentPacketCounter][3];


    int TempCounter = 0;

    for (int i = 0; i < sentArray.length; i++) {

        if (sentArray[i][0] != 0) {

            delays[TempCounter][0] = sentArray[i][0];

            double tempReceiveTime = ArrayUtils.searchTwoColumnAndReturn(recvArray,
sentArray[i][1], sentArray[i][2], 1, 2, 0);
```

```
            if (tempReceiveTime != -1) {
                delays[TempCounter][1] = tempReceiveTime - sentArray[i][0];
                delays[TempCounter][2] = sentArray[i][1];
            } else {
                delays[TempCounter][1] = -1;
            }

            if (Debug.analyze) {
                Debug.print(delays[TempCounter][0] + " " + delays[TempCounter][1] + " " +
delays[TempCounter][2], Settings.DebugDestination);
            }

            TempCounter++;
        }
    }

    return delays;
  }

  public double[][] getJitterData(){
    Vector<String> Flows = getAllFlowIDs();
    double[][] AllFlowsJitters = new double[Data.size()][3];
    int j=0;

    for(String Flowid: Flows){
      double[][] temp= getJitterDataByFlowID(Flowid);

      for(int i=0;i<temp.length;i++){
        AllFlowsJitters[j][0]=temp[i][0];
        AllFlowsJitters[j][1]=temp[i][1];
        AllFlowsJitters[j][2]=temp[i][2];
```

```
            j++;
          }
        }
        return AllFlowsJitters;
    }

    public double[][] getJitterDataByFlowID(String Flow){
        double[][] sentArray = new double[Data.size()][3]; //{Time,Unique Packet
ID,Destination Address}
        double[][] recvArray = new double[Data.size()][3];
        double[][] delays = null;

        int SentPacketCounter = 0;

        for (int i = 0; i < Data.size(); i++) {

          if (Data.get(i) instanceof NormalEvent) {
            NormalEvent NormalTemp = (NormalEvent) Data.get(i);
            if (NormalTemp.isDequeue()
                && Settings.isAcceptablePacketName(NormalTemp.getPacketName())
                && NormalTemp.getSourceNode() ==
Double.parseDouble(NormalTemp.getSourceAddress())
                && !ArrayUtils.contains(sentArray, NormalTemp.getUniquePacketId(), 1)
                && String.valueOf(NormalTemp.getFlowId()).equals(Flow)) {
              sentArray[i][0] = NormalTemp.getTime();
              sentArray[i][1] = NormalTemp.getUniquePacketId();
              sentArray[i][2] = Double.parseDouble(NormalTemp.getDestinationAddress());
              SentPacketCounter++;
              //System.out.println(i+ " Found sent event at" + sentArray[i][0]);
            } else if ((NormalTemp.isReceive())
                && Settings.isAcceptablePacketName(NormalTemp.getPacketName())
```

```
                    &&
NormalTemp.getDestinationAddress().equals(String.valueOf(NormalTemp.getDestination
Node()))
                    && String.valueOf(NormalTemp.getFlowId()).equals(Flow)) {
                recvArray[i][0] = NormalTemp.getTime();
                recvArray[i][1] = NormalTemp.getUniquePacketId();
                recvArray[i][2] = NormalTemp.getDestinationNode();
                //System.out.println("Found received event at" + recvArray[i][0]);


            }
        } else if (Data.get(i) instanceof NewWireless && ((NewWireless)Data.get(i)).getIp()
!= null ) {
            NewWireless NewTemp = (NewWireless) Data.get(i);


            if (NewTemp.isSent()
                    &&
Settings.isAcceptableNetworkTraceLevel(NewTemp.getNetworkTraceLevel())
                    && Settings.isAcceptablePacketType(NewTemp.getPacketType())
                    && Settings.isAcceptablePacketTypen(NewTemp.getPacketTypen())
                    && NewTemp.getIp().getFlowId() == Integer.parseInt(Flow)) {
                sentArray[i][0] = NewTemp.getTime();
                sentArray[i][1] = NewTemp.getIp().getUniqueId();
                sentArray[i][2] = NewTemp.getIp().getDestinationIpAddress();
                SentPacketCounter++;
            } else if (((NewWireless) Data.get(i)).isReceive()
                    &&
Settings.isAcceptableNetworkTraceLevel(NewTemp.getNetworkTraceLevel())
                    && Settings.isAcceptablePacketType(NewTemp.getPacketType())
                    && Settings.isAcceptablePacketTypen(NewTemp.getPacketTypen())
                    && NewTemp.getIp().getFlowId() == Integer.parseInt(Flow)) {
                recvArray[i][0] = NewTemp.getTime();
                recvArray[i][1] = NewTemp.getIp().getUniqueId();
                recvArray[i][2] = NewTemp.getNodeId();
```

```
            }

        } else if (Data.get(i) instanceof OldWireless) {
            OldWireless OldTemp = (OldWireless) Data.get(i);

            if (OldTemp.isSent()
                    && Settings.isAcceptableNetworkTraceLevel(OldTemp.getTraceName())
                    && OldTemp.equalsFlow(Flow)
                    && Settings.isAcceptablePacketType(OldTemp.getPacketType())) {
                sentArray[i][0] = OldTemp.getTime();
                sentArray[i][1] = OldTemp.getEventIdentifier();
                sentArray[i][2] = OldTemp.getIp().getDestinationIpAddress();
                SentPacketCounter++;
            } else if (OldTemp.isReceive()
                    && Settings.isAcceptableNetworkTraceLevel(OldTemp.getTraceName())
                    && OldTemp.equalsFlow(Flow)
                    && Settings.isAcceptablePacketType(OldTemp.getPacketType())) {
                recvArray[i][0] = OldTemp.getTime();
                recvArray[i][1] = OldTemp.getEventIdentifier();
                recvArray[i][2] = OldTemp.getNodeId();
            }


        }
    }

    delays = new double[SentPacketCounter][3];

    int TempCounter = 0;
    for (int i = 0; i < sentArray.length; i++) {
        if (sentArray[i][0] != 0) {
```

```
            delays[TempCounter][0] = sentArray[i][0];
            double tempReceiveTime = ArrayUtils.searchTwoColumnAndReturn(recvArray,
sentArray[i][1], sentArray[i][2], 1, 2, 0);


            if (tempReceiveTime != -1) {
               delays[TempCounter][1] = tempReceiveTime - sentArray[i][0];
               delays[TempCounter][2] = sentArray[i][1];
            } else {
               delays[TempCounter][1] = -1;
            }


            if (Debug.analyze) {
               Debug.print(delays[TempCounter][0] + " " + delays[TempCounter][1] + " " +
delays[TempCounter][2], Settings.DebugDestination);
            }


            TempCounter++;
         }
      }


      //----------------   Count Flow Jitter   ---------------------------
      //delays [sentTime,delay,UniquePacketId]


      double[][] jitter = new double[delays.length][3];


      //System.out.println("==== JITTER ====");
      for(int i=1;i<delays.length;i++){
         if(delays[i][1]!=-1 && delays[i-1][1]!=-1) {
            jitter[i-1][0] = delays[i][0];
            jitter[i-1][1] = delays[i][1]-delays[i-1][1];
            jitter[i-1][2] = delays[i][2];
            //System.out.println(jitter[i-1][0] + " | " + jitter[i-1][1] + " | " + jitter[i-1][2] );
```

```
            }
            else{
                jitter[i-1][0] = delays[i][0];
                jitter[i-1][1] = 0.0;
                jitter[i-1][2] = 0;
            }
        }
        return jitter;
    }

    public Vector<String> getAllFlowIDs(){
        Vector<String> FlowIDs = new Vector();
        for (Event event : Data) {
            String FlowID = null;
            if (event instanceof NormalEvent) {
                if(((NormalEvent)event).isDequeue()
                        && !((NormalEvent)event).isForward()
                        && Settings.isAcceptableNode(((NormalEvent)event).getSourceNode())
                        &&
Settings.isAcceptablePacketName(((NormalEvent)event).getPacketName()))
                    FlowID = String.valueOf(((NormalEvent)event).getFlowId());
            } else if (event instanceof NewWireless) {
                if(((NewWireless)event).getIp() != null
                        && ((NewWireless)event).isSent()
                        && Settings.isAcceptableNode(((NewWireless)event).getNodeId())
                        &&
Settings.isAcceptablePacketType(((NewWireless)event).getPacketType())
                        &&
Settings.isAcceptablePacketTypen(((NewWireless)event).getPacketTypen()))
                    FlowID = String.valueOf(((NewWireless)event).getIp().getFlowId());
            } else if (event instanceof OldWireless) {
                if(((OldWireless)event).getIp() != null
```

```
                    && ((OldWireless)event).isSent()
                    && Settings.isAcceptableNode(((OldWireless)event).getNodeId())
                    &&
Settings.isAcceptablePacketType(((OldWireless)event).getPacketType()) )
                FlowID = ((OldWireless)event).getIp().getSourceIpAddress() + ":" +
((OldWireless)event).getIp().getSourcePortNumber() + ":" +
((OldWireless)event).getIp().getDestinationIpAddress() + ":"
+((OldWireless)event).getIp().getDestinationPortNumber() + ":" +
((OldWireless)event).getPacketType();
            }

            /*
             * Add code to Support Other Formats which have PacketName attribute
             */

            if (!FlowIDs.contains(FlowID) && FlowID != null) {
                FlowIDs.add(FlowID);
            }
        }
        FlowIDs.trimToSize();

        if (Debug.analyze) {
            Debug.print("-------Flow IDs--------", Settings.DebugDestination);
            StringUtils.printVector(FlowIDs);
        }

        return FlowIDs;
    }

    public Vector<String> getPacketNames() {

        Vector<String> packetNames = new Vector();
        for (Event event : Data) {
```

```
        String PckName = null;
        if (event instanceof NormalEvent) {
            PckName = String.valueOf(((NormalEvent) event).getPacketName());
        }

        /*
         * Add code to Support Other Formats which have PacketName attribute
         */

        if (!packetNames.contains(PckName) && PckName != null) {
            packetNames.add(PckName);
        }
    }
    packetNames.trimToSize();

    if (Debug.analyze) {
        Debug.print("-------Packet Names--------", Settings.DebugDestination);
        StringUtils.printVector(packetNames);
    }

    return packetNames;
}

public Vector<String> getPacketTypes() {

    Vector<String> packetTypes = new Vector();
    for (Event event : Data) {
        String PckType = null;
        if (event instanceof NormalEvent) {
        } else if (event instanceof NewWireless && ((NewWireless) event).getIp()!= null) {
            PckType = ((NewWireless) event).getIp().getPacketType();
```

```
        } else if (event instanceof OldWireless) {
            PckType = ((OldWireless) event).getPacketType();
        }

        /*
         * Add code to Support Other Formats which have PacketType attribute
         */

        if (!packetTypes.contains(PckType) && PckType != null) {
            packetTypes.add(PckType);
        }
    }
    packetTypes.trimToSize();

    if (Debug.analyze) {
        Debug.print("-------Packet Names--------", Settings.DebugDestination);
        StringUtils.printVector(packetTypes);
    }

    return packetTypes;
}

public Vector<String> getPacketTypens() {

    Vector<String> packetTypes = new Vector();
    for (Event event : Data) {
        String PckType = null;
        if (event instanceof NewWireless) {
            PckType = ((NewWireless) event).getPacketTypen();
        }
```

```
        /*
         * Add code to Support Other Formats which have PacketTypen attribute
         */

        if (!packetTypes.contains(PckType) && PckType != null) {
            packetTypes.add(PckType);
        }
    }
    packetTypes.trimToSize();

    if (Debug.analyze) {
        Debug.print("-------Packet Names--------", Settings.DebugDestination);
        StringUtils.printVector(packetTypes);
    }

    return packetTypes;
}



 public Vector<String> getAllNetworkTraceLevels() {

    Vector<String> NetworkTrcLevels = new Vector();
    for (Event event : Data) {
        String traceLevel = null;
        if (event instanceof NormalEvent) {
            //Normal Format does not contain Network Trace Level informations
            traceLevel=null;
        } else if (event instanceof NewWireless) {
            traceLevel = ((NewWireless) event).getNetworkTraceLevel();
        } else if (event instanceof OldWireless) {
            traceLevel = ((OldWireless) event).getTraceName();
```

```
            }

            /*
             * Add code to Support Other Formats which have NetworkTraceLevel attribute
             */

            //Avoid Dublicate entries of Network Trace Levels
            if (!NetworkTrcLevels.contains(traceLevel) && traceLevel != null) {
                NetworkTrcLevels.add(traceLevel);
            }
        }

        NetworkTrcLevels.trimToSize();

        if (Debug.analyze) {
            Debug.print("-------Packet Names--------", Settings.DebugDestination);
            StringUtils.printVector(NetworkTrcLevels);
        }

        return NetworkTrcLevels;
    }
}



package util;

import Global.Debug;
import java.util.Vector;

/**
 *
```

```
 * @author Christos-Charalampos Papagiannidis
 */
public class StringUtils {


    public static boolean NwFlagExists(String s, String flag) {
        if (s.contains(flag)) {
            return true;
        }
        return false;
    }


    public static String NwFlagValue(String s, String flag) {
        String value = null;


        if (NwFlagExists(s, flag)) {
            int flaglength = flag.length();
            int flagPosition = s.indexOf(flag);
            int spacePositionAfterFlagValue = s.indexOf(" ", flagPosition + flaglength + 1);


            if (Debug.string) {
                System.out.println("flagPosition:" + flagPosition +
"\nspacePositionAfterFlagValue:" + spacePositionAfterFlagValue + "\ns.length:" +
s.length());
            }


            if (spacePositionAfterFlagValue != -1) {
                value = s.substring(flagPosition + flaglength + 1, spacePositionAfterFlagValue);
            } else {
                value = s.substring(flagPosition + flaglength + 1, s.length());
            }
            if (Debug.string) {
                System.out.println(value);
```

```
        }

    }

    return value;
}

public static String getAbbreviation(String s) {
    if (Debug.string) {
        System.out.println("Abbreviation value taken from file: " + s.substring(0, 1));
    }
    return s.substring(0, 1);
}

public static void printVector(Vector v) {
    for (int i = 0; i < v.size(); i++) {
        System.out.println(v.get(i));
    }
}

public static boolean containString(String s, String searchString) {
    String substr = null;
    for (int i = 0; i < s.length(); i++) {
        substr = s.substring(i);
        if (substr.startsWith(searchString)) {
            System.out.println(substr + "\n" + searchString + "\n returned true");
            return true;
        }
    }
    System.out.println(substr + "\n" + searchString + "\n returned false");
    return false;
```

```
    }
}

package util;

import Global.Trace;
import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Vector;

/**
 * This program reads a text file line by line and returns Vector<String> . It uses
 * FileOutputStream to read the file.
 *@author Christos-Charalampos Papagiannidis
 */
public class TrcReader  {

    private File traceFile = Trace.getFile();

    public TrcReader(File traceFile) {
        this.traceFile = traceFile;
    }
    /**
     * Reads Traces.class file and adds every line to a Vector<String>
     * @see Trace
     * @return
     */
```

```
public synchronized Vector<String> getTraceData() {
    return addFileToVector();
}

private synchronized Vector<String> addFileToVector() {
    Vector<String> temp = new Vector(10);

    FileInputStream fis = null;
    BufferedInputStream bis = null;
    DataInputStream dis = null;

    try {
        fis = new FileInputStream(traceFile);

        // Here BufferedInputStream is added for fast reading.
        bis = new BufferedInputStream(fis);
        dis = new DataInputStream(bis);

        // dis.available() returns 0 if the file does not have more lines.
        while (dis.available() != 0) {

            // this statement reads the line from the file and add it to
            // temp<String> vector
            temp.add(dis.readLine());
        }

        // dispose all the resources after using them.
        fis.close();
        bis.close();
        dis.close();
        temp.trimToSize();
```

```
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return temp;
    }
}
```

9 Οδηγός Χρήσης Λογισμικού

9.1 Εισαγωγή

Στην ενότητα αυτή υπάρχουν όλες οι πληροφορίες για την εγκατάσταση και χρήση της εφαρμογής jTrcezer, όπως επίσης και τυχόν προβλήματα και τρόπους για την επίλυσή τους.

9.2 Εγκατάσταση

Για την εγκατάσταση της εφαρμογής σε έναν υπολογιστή δεν απαιτείται κάποια πολύπλοκη διαδικασία. Η εφαρμογή είναι υλοποιημένη στη γλώσσα προγραμματισμού java γιαυτό είναι απαραίτητη η εγκατάσταση της Java πριν την προσπάθεια χρήσης της.

Αν δεν έχετε εγκατεστημένη την java στον υπολογιστή σας, μπορείτε να κατεβάσετε την τελευταία έκδοση από την διεύθυνση, http://www.java.com/en/download/ .

Η εφαρμογή είναι ένα jar αρχείο συνοδευόμενο με ένα φάκελο lib, οποίος θα πρέπει πάντα να βρίσκεται στον ίδιο φάκελο-directory με το jar της εφαρμογής, που σημαίνει ότι η αποθήκευση των παραπάνω σε οποιοδήποτε φάκελο του υπολογιστή είναι η μόνη διαδικασία εγκατάστασης που απαιτείται.

9.3 Εκτέλεση Εφαρμογής

Εάν κατεβάσατε τον πηγαίο κώδικα της εφαρμογής και θέλετε να κάνετε compile, θα πρέπει να γνωρίζετε ότι για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το IDE Netbeans 6.9.1.

Βρείτε το αρχείο jTrcezer.jar στο φάκελο που το αποθηκεύσατε ή στο φάκελο που εξάγατε το αρχείο κατά την διαδικασία του compile. Στη συνέχεια εκτελέστε το αρχείο jar και η εφαρμογή θα ξεκινήσει. Εάν εκτελόντας το jTrcezer.jar δεν εκκινεί η εφαρμογή ίσως να μην έχετε περάσει την java στο PATH των windows. Για το λόγο αυτό παρέχεται και ένα bat αρχείο το οποίο θα εκκινήσει σε κάθε περίπτωση την εφαρμογή.

9.4 Χρήση Εφαρμογής

Το γραφικό περιβάλλον της εφαρμογής σχεδιάστηκε με βασικά κριτήρια την απλότητα και την λειτουργικότητα. Το κύριο παράθυρο της εφαρμογής όπως φαίνεται και στην εικόνα 26, αποτελείται από ένα μενού στο πάνω μέρος του παραθύρου και οι επιλογές που διαθέτει εκτελούν τις παρακάτω λειτουργίες:

- ✓ Άνοιγμα αρχείων Trace
- ✓ Άνοιγμα παραθύρου φιλτραρίσματος και εμφάνισης γραφημάτων

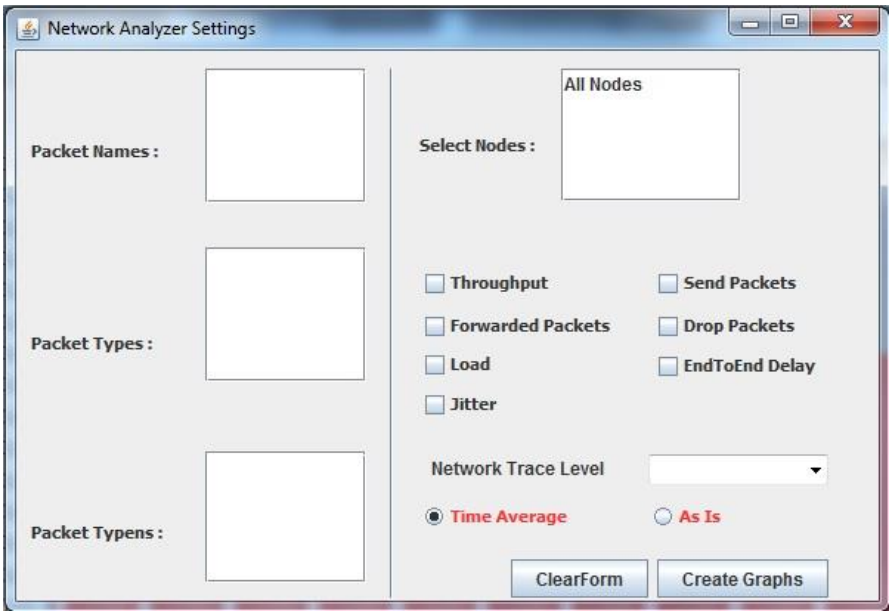
- ✓ Έξοδος από την εφαρμογή

Εικονα 26 : αρχικο παραθυρο της εφαρμογης jtrcezer

Πριν την χρήση οποιασδήποτε λειτουργίας του προγράμματος θα πρέπει να φορτωθεί κάποιο αρχείο Trace από το Open File Dialog που βρίσκεται στο File->Open Trace.

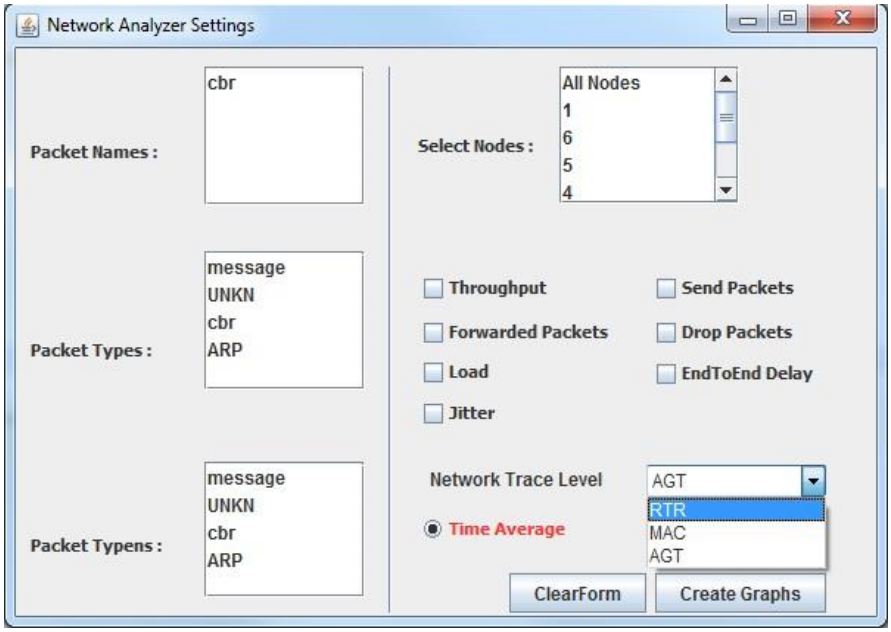
Στην περίπτωση που το αρχείο είναι αρκετά μεγάλο η εφαρμογή μπορεί να μην είναι διαθέσιμη για μερικά δευτερόλεπτα έως ότου διαβαστεί και αναλυθεί και η τελευταία γραμμή του Trace αρχείου που φορτώσαμε.

Για την εμφάνιση των γραφημάτων στο Graphs->Filter And Create υπάρχουν συσσωρευμένα όλα τα εργαλεία και οι ρυθμίσεις που πρέπει ή επιθυμεί να κάνεις ο χρήστης σχετικά με τα γραφήματα που ενδιαφέρεται να εμφανίσει. Αν το παράθυρο αυτό πριν φορτωθεί κάποιο Trace αρχείο τότε θα φαίνεται όπως στην εικόνα 27.



ΕΙΚΟΝΑ 27 : ΠΑΡΑΘΥΡΟ ΕΠΙΛΟΓΗΣ ΦΙΛΤΡΩΝ ΠΡΙΝ ΤΗΝ ΦΟΡΤΩΣΗ ΑΡΧΕΙΟΥ TRACE

Αντίθετα αν έχει φορτωθεί κάποιο αρχείο Trace θα πρέπει τα components που περιέχει το παράθυρο αυτό να περιλαμβάνουν προτεινόμενες ρυθμίσεις ανάλογα με τις πληροφορίες που εντοπίστηκαν κατά την διαδικασία ανάγνωσης του Trace αρχείου, όπως δείχνει η εικόνα 28.



ΕΙΚΟΝΑ 28 : ΠΑΡΑΘΥΡΟ ΕΠΙΛΟΓΗΣ ΦΙΛΤΡΩΝ ΜΕΤΑ ΤΗΝ ΦΟΡΤΩΣΗ ΑΡΧΕΙΟΥ TRACE

Η λογική αντιμετώπισης αυτών των επιλογών διαφέρει από τύπο σε τύπο. Για τις επιλογές Packet Names,Packet Types και Packet Typens ότι είναι επιλεγμένο λαμβάνονται μόνο υπόψη κατά τους υπολογισμούς των γραφημάτων. Αν δεν υπάρχει κανένα επιλεγμένο σε μια λίστα τότε θεωρείτε πως δεν εξαιρείται τίποτα για τον υπολογισμό των γραφημάτων άρα σημαίνει ότι ο χρήστης τα επέλεξε όλα.

Αντίθετα η επιλογή Select Nodes δεν αποτελεί φίλτρο εξαίρεσης των μη επιλεγμένων. Το πεδίο αυτό δηλώνει τα Nodes για τα οποία θα υπολογισθούν ξεχωριστά γραφήματα. Η επιλογή “All Nodes” υπάρχει πάντα ως επιλογή και δηλώνει ότι θα εξαχθούν γραφήματα που θα ανταποκρίνονται στην κατάσταση ολόκληρου του δικτύου προσομοίωσης.

Τα CheckBoxes που ακολουθούν προσφέρουν την δυνατότητα στον χρήστη να επιλέξει μόνο τις μετρήσεις που επιθυμεί να εμφανίσει. Όπως αναφέρθηκε στην προηγούμενη ενότητα κάθε μέτρηση αξιολόγησης του δικτύου έχει δύο τύπους γραφημάτων, Time Average και AsIs κάτι που μπορεί να επιλέξει ο χρήστης στα radioButtons που εμφανίζονται στο κάτω δεξιά μέρος του παραθύρου με κόκκινα έντονα γράμματα. Ορίζεται ως προεπιλεγμένο το Time Average για κάθε περίπτωση που δεν επιλέξει το αντίθετο ο χρήστης.

Τέλος το Drop Down List που ρυθμίζει το Network Trace Level χρησιμοποιείται και έχει τιμές μόνο όταν το Trace αρχείο που φορτώθηκε περιλαμβάνει κάποια διακριτά γεγονότα τύπου Old Wireless ή New Wireless.