ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εξομοιωτές Δικτύων Υπολογιστών

Μήλας Σωτήριος
Α.Μ:0555

Παπαδόπουλος Κωνσταντίνος
Α.Μ:0420

ΕΠΙΒΛΕΠΩΝ: Βασίλειος Τριανταφύλλου, Καθηγητής

ΑΝΤΙΡΡΙΟ 2016

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Αντίρριο, ……2016

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1._____

2._____

3._____

Περιεχόμενα

Ευρετήριο Πινάκων

Ευρετήριο Εικόνων

Περίληψη

Το Διαδίκτυο και τα ιδιωτικά δίκτυα ευρείας περιοχής αντιπροσωπεύουν ορισμένες από τις πτυχές που συνδέουν το χρήστη με τις εφαρμογές που παρέχουν. Καθώς οι χρήστες ολοένα απαιτούν ταχύτερη απόκριση και πιο σύνθετα δεδομένα από τις εφαρμογές τους, τα δίκτυα που μεταφέρουν αυτά τα δεδομένα βρίσκονται σε μεγάλη πίεση να ανταποκριθούν σε τέτοιες προσδοκίες. Προκειμένου να δοκιμάσουν εκτενώς τις εφαρμογές πριν τις προβάλουν στους χρήστες, είναι απαραίτητη η χρησιμοποίηση των εξομοιωτών δικτύων. Οι εξομοιωτές παρέχουν τη δυνατότητα δημιουργίας αξιόπιστων και επαναλαμβανόμενων δοκιμών πριν από την ανάπτυξη της εφαρμογής. Σε αυτή την εργασία εξετάστηκαν αναλυτικά οι εξομοιωτές GNS 3.0, Boson Netsim 8.0 και NS-3. Μέσω των παραδειγμάτων που υλοποιήθηκαν εξετάστηκε ο τρόπος λειτουργίας τους. Τέλος, αναλύθηκε το Πρωτόκολλο Κυλιόμενου Παραθύρου(SWP) εν μέσω δημιουργίας κατάλληλου αλγόριθμου που υλοποιήθηκε στον NS-3. Ο στόχος ήταν να εξεταστεί πως το μέγεθος του παραθύρου επηρεάζει την απόδοση του αλγόριθμου.

Λέξεις κλειδιά: GNS3,Boson NetSim,NS-3, Εξομοιωτές Δικτύων, SWP

Abstract

The Internet and private wide area networks represent some of the aspects that connect the user to their applications. As users increasingly demand faster response and more complex data from their applications, the networks carrying this data are under greater pressure to meet these expectations. In order to truly test out applications before rolling them out to the users, network emulators must be used. Emulators provide the ability to create reliable and repeatable test configurations before the development of application. In this project we analyzed the emulators GNS 3.0,Boson NetSim 8.0 and NS-3. With examples that implemented we examined their operating methods. Finally, was analyzed the Sliding Window Protocol(SWP) by creating suitable algorithm that implemented in NS-3.The aim was to examine how the window size affects the algorithm performance.

Keywords: GNS3,Boson NetSim,NS-3, network emulators, SWP

Κεφάλαιο 1

1.1 Εισαγωγή

Η προσομοίωση αποτελεί μια από τις πιο σημαντικές και ισχυρές μεθόδους, που χρησιμοποιούνται στην έρευνα για τον σχεδιασμό και την παρακολούθηση της λειτουργίας των δικτύων υπολογιστών. Σύμφωνα με τον R.E.Shannon (1975) η προσομοίωση ορίζεται ως η διαδικασία σχεδιασμού του μοντέλου κάποιου πραγματικού δικτύου και πραγματοποίησης πειραμάτων με το μοντέλο αυτό που αποσκοπούν στην κατανόηση της συμπεριφοράς του δικτύου ή/και στην αξιολόγηση εναλλακτικών στρατηγικών για τη λειτουργία του δικτύου.

Δεδομένου του παραπάνω ορισμού, θα μπορούσαμε να χαρακτηρίσουμε την προσομοίωση ως μια εφαρμοσμένη πειραματική μεθοδολογία, με την οποία μπορούμε:

- Να περιγράψουμε την συμπεριφορά ενός δικτύου υπολογιστών.
- Να χρησιμοποιήσουμε το μοντέλο προκειμένου να προβλέψουμε μελλοντικές συμπεριφορές, δηλαδή αποτελέσματα από την αλλαγή του δικτύου ή του τρόπου λειτουργίας του.

Εμβαθύνοντας λίγο περισσότερο στην έννοια της μοντελοποίησης, θα μπορούσαμε να πούμε πως ένα μοντέλο αναπαριστά την δομή και την λειτουργικότητα ενός πραγματικού δικτύου υπολογιστών και είναι παρόμοιο αλλά απλούστερο από το δίκτυο που αντιπροσωπεύει. Ένα καλό μοντέλο ουσιαστικά είναι αυτό που ενσωματώνει τα θετικά στοιχεία του ρεαλισμού και ταυτόχρονα της απλότητας. Γενικά, ένα μοντέλο που προορίζεται για την χρήση σε μελέτες προσομοίωσης είναι στην πράξη ένα μαθηματικό μοντέλο που αναπτύχθηκε με τη βοήθεια λογισμικού προσομοίωσης.

Τα μαθηματικά αυτά μοντέλα, μπορούν να ταξινομηθούν ως ντετερμινιστικά (οι μεταβλητές εισόδου και εξόδου παίρνουν σταθερές τιμές), ως στοχαστικά (τουλάχιστον μια από τις μεταβλητές εισόδου ή εξόδου έχει τη μορφή πιθανότητας), ως στατικά (ο χρόνος δεν λαμβάνεται υπόψη) ή ως δυναμικά (οι κυμαινόμενες με τον χρόνο αλληλεπιδράσεις μεταξύ μεταβλητών λαμβάνεται υπόψη). Τυπικά, τα μοντέλα προσομοίωσης είναι στοχαστικά και δυναμικά.

Ωστόσο μια κρίσιμη ερώτηση αφορά το πότε πρέπει να γίνεται χρήση προσομοίωσης αντί του πειραματισμού με το πραγματικό σύστημα. Οι γενικές περιπτώσεις στις οποίες συνήθως χρησιμοποιείται μοντελοποίηση και ανάλυση βάση προσομοίωσης είναι οι ακόλουθες:

- Όταν είναι αδύνατο ή έχει υπερβολικό κόστος η παρατήρηση συγκεκριμένων διαδικασιών στον πραγματικό κόσμο.
- Όταν υπάρχουν προβλήματα για τα οποία υπάρχει μεν μαθηματικό μοντέλο, ωστόσο είναι αδύνατο ή υπερβολικά πολύπλοκο να βρεθούν λύσεις με αναλυτικές μεθόδους.
- Όταν είναι αδύνατη ή έχει υπερβολικό κόστος η επικύρωση της ορθότητας του μαθηματικού μοντέλου που περιγράφει το πραγματικό δίκτυο επειδή π.χ. δεν υπάρχουν επαρκή δεδομένα.

Η προσομοίωση χρησιμοποιείται επίσης πριν εφαρμοστούν αλλαγές σε ένα υπάρχον δίκτυο ή πριν κατασκευαστεί ένα νέο δίκτυο, προκειμένου να μειωθούν οι πιθανότητες αποτυχίας, να εξαλειφθούν απρόβλεπτα σημεία συμφόρησης, να προληφθεί η υπερκατανάλωση ή υποκατανάλωση πόρων και να βελτιστοποιηθεί η συνολική απόδοση του δικτύου.

Επιπλέον η προσομοίωση υπερτερεί των αναλυτικών μεθόδων για την ανάλυση δικτύων για τους εξής λόγους:

- Μπορούμε να εξετάσουμε νέους σχεδιασμούς του δικτύου χωρίς να αφιερώνουμε πόρους στην υλοποίηση τους.
- Η προσομοίωση επιτρέπει την εξέταση υποθέσεων σχετικά με το πώς ή το γιατί συγκεκριμένα φαινόμενα συμβαίνουν σε ένα δίκτυο.
- Η προσομοίωση επιτρέπει τον πλήρη έλεγχο του χρόνου. Έτσι είναι εφικτό η καταγραφεί μέσα σε μερικά δευτερόλεπτα η συμπεριφορά ενός δικτύου που λειτουργεί για μήνες ή χρόνια. Εναλλακτικά είναι δυνατή η επιβράδυνση των φαινομένων προκειμένου να μελετηθούν.

- Η προσομοίωση επιτρέπει την διεξαγωγή συμπερασμάτων σχετικά με το πώς λειτουργεί στην πραγματικότητα το μοντελοποιημένο δίκτυο και ποιες μεταβλητές είναι οι πιο σημαντικές για την απόδοσή του.
- Η προσομοίωση έχει την δύναμη να επιτρέπει τον πειραματισμό με νέες και άγνωστες καταστάσεις ώστε να απαντά σε υποθετικά ερωτήματα.

Τα βήματα που εμπλέκονται στην διαδικασία ανάπτυξης ενός μοντέλου προσομοίωσης, στον σχεδιασμό του πειράματος και στην διενέργεια της ανάλυσης των αποτελεσμάτων της προσομοίωσης είναι τα εξής:

- **Βήμα 1:** Ορισμός του προβλήματος.
- **Βήμα 2:** Μορφοποίηση του προβλήματος.
- **Βήμα 3:** Συλλογή και επεξεργασία δεδομένων για το πραγματικό σύστημα.
- **Βήμα 4:** Μορφοποίηση και ανάπτυξη του μοντέλου του συστήματος.
- **Βήμα 5:** Επικύρωση του μοντέλου.
- **Βήμα 6:** Τεκμηρίωση μοντέλου για μελλοντική χρήση.
- **Βήμα 7:** Επιλογή του κατάλληλου σχεδίου πειραμάτων.
- **Βήμα 8:** Επαλήθευση των συνθηκών των πειραμάτων για όλες τις εκτελέσεις της προσομοίωσης.
- **Βήμα 9:** Εκτέλεση προσομοίωσης.
- **Βήμα 10:** Ερμηνεία και παρουσίαση αποτελεσμάτων.
- **Βήμα 11:** Πρόταση περαιτέρω ενεργειών.

Αυτή είναι μια λογική ταξινόμηση των βημάτων σε μια μελέτη προσομοίωσης. Ωστόσο, ίσως αποδειχτεί αναγκαίο να γίνουν επαναλήψεις σε διάφορα σημεία προτού επιτευχθούν οι στόχοι της προσομοίωσης. Ίσως να μην χρειαστούν όλα τα παραπάνω βήματα ή μπορεί μερικά από αυτά να μην είναι εφικτά. Από την άλλη πλευρά ίσως

απαιτηθούν και επιπλέον βήματα για την ολοκλήρωση μιας μελέτης. Κλείνοντας αυτήν την εισαγωγή στην προσομοίωση, θα πρέπει να τονίσουμε ότι οι περισσότερες μελέτες προσομοίωσης σήμερα γίνονται χρησιμοποιώντας πακέτα λογισμικού προσομοίωσης, αντί να κάνουν χρήση μοντέλων που αναπτύχθηκαν με γλώσσες προγραμματισμού γενικού σκοπού.

Υπάρχουν εκατοντάδες προϊόντα προσομοίωσης στην αγορά, και το ερώτημα που τίθεται είναι πως διαλέγει κανείς το καλύτερο δεδομένου του προβλήματος που θέλει να μελετήσει. Κάποιες από τις παραμέτρους που συχνά εξετάζονται κατά την επιλογή είναι: η προσφερόμενη ευελιξία στην μοντελοποίηση, η ευκολία στη χρήση, η δυνατότητα επαναχρησιμοποίησης τμημάτων κώδικα, η διεπαφή χρήστη, οι απαιτήσεις σε λογισμικό και υλισμικό, οι στατιστικές δυνατότητες, η δυνατότητες γραφικής απεικόνισης των αποτελεσμάτων, η υποστήριξη πελατών και η τεκμηρίωση.

Μερικοί δημοφιλείς προσομοιωτές είναι οι εξής: OPNET, NS-2, NS-3, NetSim, OMNET++, REAL, J-SIM και QualNet. Στη συνέχεια της πτυχιακής θα γίνει αναφορά των προαναφερθέντων προσομοιωτών και ειδικά για τον NS2, NS3, Netsim, και GNS3 θα υλοποιηθούν παραδείγματα.

ΠΙΝΑΚΑΣ 1

Γλώσσα προγραμματισμού προσομοιωτών

Προσομοιωτής Δικτύου	**Γλώσσα προγραμματισμού**
NS2	C++, Otcl
NS3	C++, Python
OPNET	C (C++):
NetSim	Java

OMNet++	C++
REAL	C
J-Sim	Java, Tcl
Qualnet	C++

1.2 Διαφορές μεταξύ Εξομοιωτή και Προσομοιωτή

Στον τομέα της έρευνας τηλεπικοινωνιών και δικτύων, η πιο χρήσιμη τεχνική που χρησιμοποιείται είναι η ***προσομοίωση***, η οποία μπορεί να αξιολογήσει τη συνολική συμπεριφορά ενός δικτύου καθώς και να υπολογίσει την αλληλεπίδραση μεταξύ των διαφορετικών στοιχείων που περιλαμβάνει ένα δίκτυο, όπως δρομολογητές(routers), πακέτα, φυσικές συνδέσεις χρησιμοποιώντας διαφορετικά μαθηματικά μοντέλα. Μέσω των πειραμάτων προσομοίωσης μπορούμε να τροποποιήσουμε όλες τις ιδιότητες στο περιβάλλον που εργαζόμαστε προκειμένου να αξιολογήσουμε πως συμπεριφέρεται το δίκτυο κάτω από διαφορετικές παραμέτρους και συνδυασμούς. Ένα αξιοσημείωτο χαρακτηριστικό είναι ότι ένα πρόγραμμα προσομοίωσης μπορεί να χρησιμοποιηθεί μαζί με διαφορετικές εφαρμογές και υπηρεσίες(point-to point, end-to-end).

Από την άλλη πλευρά, με τον όρο ***εξομοίωση*** εννοούμε ότι προσομοιώνουμε ένα δίκτυο το οποίο βρίσκεται υπό σχεδιασμό για να αξιολογήσουμε την απόδοσή του ή να προβλέψουμε πιθανές αλλαγές και βελτιστοποιήσεις. Το κύριο σημείο είναι ότι η δουλειά του εξομοιωτή δικτύου είναι να εξομοιώνει ένα δίκτυο το οποίο συνδέει end-hosts, αλλά όχι οι end-hosts μεταξύ τους. Ο NS-2 είναι ένα εργαλείο προσομοίωσης δικτύου, ο οποίος συμπεριλαμβάνεται στα εργαλεία εξομοίωσης δικτύου με περιορισμένες λειτουργίες εξομοίωσης, σε αντίθεση με ένα τυπικό εξομοιωτή ο οποίος χρησιμοποιεί λειτουργίες Linux, όπως ο WANsim.

1.3 Κατηγορίες Προσομοιωτών

Οι προσομοιωτές δικτύων κατηγοριοποιούνται ως εξής: σε απλούς (simple) και σύνθετους (complex) καθώς επίσης σε ανοικτού κώδικα (open source) και εμπορικούς (commercial).

1.3.1 Ανοικτού κώδικα και Εμπορικοί

Το κύριο πλεονέκτημα των προσομοιωτών ανοικτού κώδικα είναι ότι παρέχουν στους χρήστες τον πηγαίο τους κώδικα ελεύθερα. Όλα είναι ανοιχτά και ο καθένας μπορεί να συνεισφέρει και να βρει σφάλματα σε αυτό. Δεν υπάρχουν περιορισμοί όσον αφορά την διεπαφή τους και έτσι είναι ανοιχτοί για μελλοντικές βελτιώσεις. Εξαιτίας της ευελιξίας τους, πρόσφατες ανακαλύψεις νέων τεχνολογιών υλοποιούνται με πιο γρήγορους ρυθμούς σε σύγκριση με τους εμπορικούς.

Από την άλλη πλευρά, οι εμπορικοί εξομοιωτές δεν παρέχουν τον πηγαίο τους κώδικα στους χρήστες ελεύθερα. Για να αποκτήσουν την εξουσιοδοτημένη έκδοση θα πρέπει να πληρώσουν για να αποκτήσουν το λογισμικό. Ένα παράδειγμα εμπορικού εξομοιωτή είναι ο εξομοιωτής OPNET. Το κύριο πλεονέκτημα ενός εξομοιωτή τέτοιου τύπου είναι ότι διαχειρίζεται από το εξειδικευμένο προσωπικό της εταιρίας που το κατασκευάζει, που έχει ως συνέπεια να διαθέτει ολοκληρωμένες και ενημερωμένες τεκμηριώσεις. Αντιθέτως, ο εξομοιωτής ανοικτού κώδικα μειονεκτεί σε αυτό το κομμάτι, λόγω του ότι τα άτομα που εργάζονται για την τεκμηρίωσή του δεν είναι αρκετά εξειδικευμένα. Αυτό προκαλεί σοβαρά προβλήματα όταν διαφορετικές εκδόσεις φέρουν νέα πράγματα με έλλειψη κατάλληλης τεκμηρίωσης.

1.3.2 Απλοί και Σύνθετοι

Υπάρχουν αρκετοί προσομοιωτές δικτύων διαθέσιμοι στην αγορά και μπορούμε να τους διακρίνουμε σε απλούς και σύνθετους. Οποιοσδήποτε προσομοιωτής δικτύου θα πρέπει να δίνει τη δυνατότητα στους χρήστες να μπορούν να αναπαραστήσουν μία δικτυακή τοπολογία, ορίζοντας διαφορετικά σενάρια, καθορίζοντας τους κόμβους στο δίκτυο, την σύνδεση μεταξύ των κόμβων καθώς και την μεταξύ τους κίνηση. Οι χρήστες μπορούν να καθορίσουν οποιαδήποτε παράμετρο για κάθε πρωτόκολλο που χρησιμοποιείται για να επεξεργαστεί τη δικτυακή κίνηση σε πολύπλοκα συστήματα. Όσον αφορά τις εφαρμογές γραφικών, μερικές από αυτές παρέχουν ένα text-based περιβάλλον, το οποίο έχει περιορισμούς στην οπτικοποίηση της διεπαφής, αλλά επιτρέπει την παραμετροποίηση σε

πιο προηγμένες μορφές. Άλλες παρέχουν ένα προγραμματιστικό πλαίσιο, παραμετροποιήσιμο δημιουργώντας μια εφαρμογή η οποία δοκιμάζει το περιβάλλον του δικτύου.

ΠΙΝΑΚΑΣ 2

Προσομοιωτές Δικτύων

Κατηγορία	**Προσομοιωτής Δικτύου**
Εμπορικοί	OPNET, QualNet
Ανοικτού Κώδικα	NS2, NS3, OMNET++, SSFNet, J-Sim

1.4 Ανασκόπηση εξομοιωτών

1.4.1 NS 2 (Network Simulator 2): Ο Network Simulator-2 (NS-2) είναι ένας αντικειμενοστρεφής προσομοιωτής δικτύων, γραμμένος σε C++ και OTcl (αντικειμενοστρεφής έκδοση της γλώσσας TCL). Στον NS-2 είναι ενσωματωμένα τα πιο γνωστά δικτυακά πρωτόκολλα, όπως π.χ τα πρωτόκολλα TCP και UDP, πρωτόκολλα εφαρμογής (FTP, Telnet, Web, CBR), προσομοιώσεις διαχείρισης ουρών δρομολογητών (DropTail, RED), δρομολόγησης κτλ.

Οι προγραμματιστές του NS-2 προκειμένου να ισορροπήσουν ανάμεσα στην υψηλού επιπέδου υλοποίηση και στην ευκολία προγραμματισμού χώρισαν τον προσομοιωτή σε δύο τμήματα. Το ένα τμήμα (κυρίως για την διασύνδεση με τον χρήστη) είναι γραμμένο στη γλώσσα OTcl ώστε να μην χρειάζεται ο απλός χρήστης να γνωρίζει την δυσκολότερη αλλά πανίσχυρη C++ προκειμένου να τον χρησιμοποιήσει. Ο υπόλοιπος κώδικας του NS-2 είναι γραμμένος στην γλώσσα C++ ώστε το αποτέλεσμα να είναι υψηλού επιπέδου και να είναι αποδοτικός σε σχέση με την ταχύτητα.

Δηλαδή, ένα δικτυακό πρόβλημα μπορεί να εκφραστεί μόνο με την Otcl, μόνο με την C++ ή και με συνδυασμό των δύο. Συνήθως η παραμετροποίηση του προβλήματος γίνεται με την OTcl για μεγαλύτερη ταχύτητα και ευκολία, ενώ το υπόλοιπο πρόβλημα χρησιμοποιεί κώδικα C++.

Τα αντικείμενα της C++ έχουν αναπαρασταθεί και σαν αντικείμενα της OTcl, έτσι ώστε να μπορεί να εκφράσει κάποιος ένα πρόβλημα και με τους δύο τρόπους.

Λίγα λόγια για την OTcl

Η OTcl αποτελεί την αντικειμενοστραφή έκδοση της TCL. Είναι μια αρκετά εύκολη στη χρήση γλώσσα προγραμματισμού. Με την βοήθειά της, συνήθως παραμετροποιούμε και περιγράφουμε ένα σύστημα, την λειτουργία του οποίου προσομοιώνουμε με την βοήθεια του NS-2.

Με την δήλωση set θέτουμε μια τιμή σε μία μεταβλητή ή δημιουργούμε το στιγμιαίο δείγμα (instance) ενός αντικειμένου :

```
Set a 43

# Writing a procedure called "test"

proc test {} {

set a 43

set b 27

set c [expr $a + $b]

set d [expr [expr $a - $b] * $c]

for {set k 0} {$k < 10} {incr k} {

if {$k < 5} {

puts "k < 5, pow = [expr pow($d, $k)]"

} else {

puts "k >= 5, mod = [expr $d % $k]"

} } }

# Calling the "test" procedure created above test
```

Βασικός κώδικας ενός OTcl Script του NS-2

** Στο κεφάλαιο 4 θα γίνει εκτενή αναφορά του προσομοιωτή Ns-2 με παραδείγματα και αποτελέσματα αυτών.*

1.4.2 NS 3 (Network Simulator 3) : Ο ns-3 είναι διακριτών γεγονότων προσομοιωτής δικτύων για συστήματα διαδικτύου στον οποίο ο πυρήνας και τα μοντέλα της προσομοίωσης έχουν υλοποιηθεί σε C++ και χρησιμοποιείται κυρίως για εκπαιδευτικούς κι ερευνητικούς σκοπούς. Είναι ελεύθερο λογισμικό σύμφωνα με την άδεια GNU GPLv2. Είναι χτισμένος ως μία βιβλιοθήκη που μπορεί να συνδεθεί στατικά ή δυναμικά με ένα C++ πρόγραμμα που ορίζει την τοπολογία της προσομοίωσης και ξεκινάει τον προσομοιωτή. Επίσης ο ns-3 εξάγει σχεδόν όλα του τα APIs στην Python επιτρέποντας έτσι στα Python προγράμματα να εισάγουν μια "ns-3" ενότητα με το ίδιο τρόπο που η ns-3 βιβλιοθήκη συνδέεται με εκτελέσιμα στη C++. Δεν αποτελεί επέκταση του ns-2 και δεν υποστηρίζει το API του ns-2 παρόλο που είναι επίσης γραμμένος σε C++.

Στόχος του έργου ns-3 είναι να αναπτυχθεί ένα ανοικτό περιβάλλον προσομοίωσης που να ανταποκρίνεται στις ανάγκες της σύγχρονής έρευνας και να ενθαρρύνει τη συμβολή της επιστημονικής κοινότητας. Επίσης, στοχεύει στην δημιουργία πλήρους τεκμηρίωσης (documentation) που να είναι εύκολη στη χρήση και τον εντοπισμό σφαλμάτων για τις ρυθμίσεις των προσομοιώσεων καθώς επίσης, και της συλλογής κι ανάλυσης των αποτελεσμάτων .

Υποστηρίζει την έρευνα σε IP και μη IP – based δίκτυα κι ένα πραγματικού χρόνου χρονοπρογραμματιστή για αλληλεπίδραση με πραγματικά συστήματα.

** Στο κεφάλαιο 4 θα γίνει εκτενή αναφορά του προσομοιωτή Ns-3 με παραδείγματα και αποτελέσματα αυτών.*

1.4.3 OPNET (Optimized Network Engineering Tools) : Το OPNET (Optimized Network Engineering Tools) είναι ένα εμπορικό εργαλείο για την προσομοίωση και την ανάλυση των δικτύων επικοινωνιών, των κατανεμημένων συστημάτων, των υπολογιστικών συστημάτων και εφαρμογών υπολογιστών. Επιτρέπει τον σχεδιασμό και τη μελέτη δικτύων επικοινωνιών, συσκευών, πρωτοκόλλων και εφαρμογών. Υποστηρίζει τόσο προσομοίωση διακριτών γεγονότων όσο και αναλυτική προσομοίωση.

Πρόκειται για ένα ιδιαίτερα ισχυρό πρόγραμμα το οποίο όμως δεν είναι ανοιχτού κώδικα και επομένως πρέπει κάποιος να το αγοράσει. Παρόλα αυτά δίνεται η δυνατότητα σε ερευνητές – ερευνητικές ομάδες να παραδώσουν κώδικα μοντέλων που επεκτείνουν τις δυνατότητες του προσομοιωτή και αυτά να προσφερθούν με την σειρά τους μαζί με το πακέτο.

Υποστηρίζει έρευνα πάνω σε πάρα πολλά πεδία όπως:

• Έλεγχο λειτουργίας πρωτοκόλλων και εμπλουτισμό τους (ενσύρματα, ασύρματα).

• Σχεδιασμός νέων πρωτοκόλλων.

• Σχεδιασμός νέων τρόπων διαχείρισης της ενεργειακής κατανάλωσης στα δίκτυα.

• Εμπλουτισμός των δικτύων κορμού (Core networks).

• Ανάλυση σχεδίασης δικτύων.

Διαθέτει πολύ ισχυρό πλαίσιο συντήρησης:

• Επαγγελματική υποστήριξη.

• Οργανωμένη διάθεση αναβαθμίσεων.

• Αλληλεπίδραση με ειδικούς του χώρου μέσω forums.

• Εκπαίδευση (Webinars, White papers, Lab manuals κλπ).

Υπάρχει η δυνατότητα σύνδεσης της προσομοίωσης με πραγματικά συστήματα (System-in the-loop) και μπορεί να συλλέξει ίχνη πραγματικών εφαρμογών για ανάλυση.

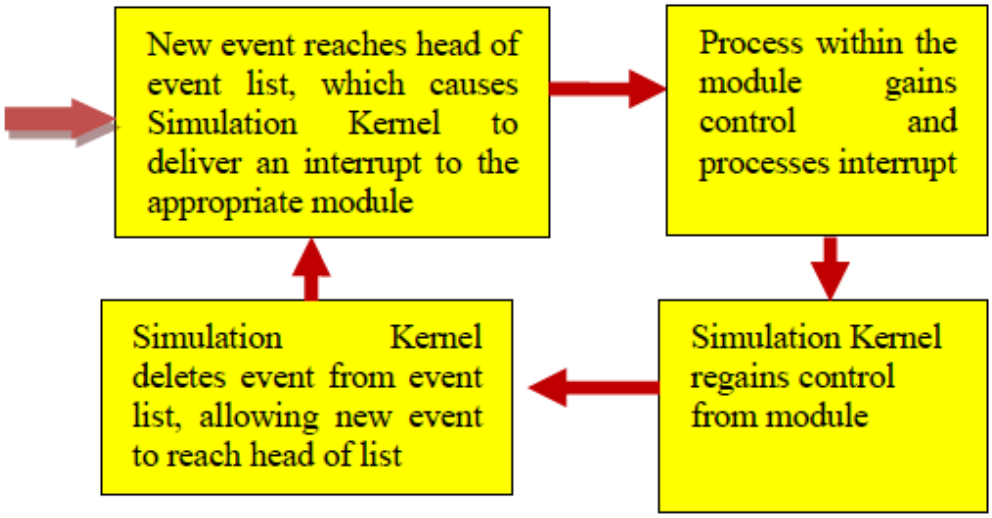
Το OPNET έχει τα ακόλουθα χαρακτηριστικά:

- Domain Specific, Hierarchical Models –το OPNET είναι ειδικά σχεδιασμένο για την ανάλυση και ανάπτυξη των δικτύων επικοινωνιών, και παρέχει εκτενείς λεπτομέρειες που δεν είναι διαθέσιμες σε απλούστερα resource-based πακέτα προσομοίωσης.

- Τα μοντέλα του δικτύου (υλικού και λογισμικού) είναι ιεραρχικά δομημένα, επιτρέποντας επαναχρησιμοποίηση των ανεπτυγμένων μοντέλων σε διαφορετικές προσομοιώσεις. Γραφικές λεπτομέρειες των μοντέλων εισάγονται με ειδικούς editors οι οποίοι παρέχουν ένα αποδοτικό μέσο για τον σχεδιασμό.

- Αυτόματη προσομοίωση-Το OPNET μειώνει την απαιτούμενη προσπάθεια προσομοίωσης, παρέχοντας έναν αποδοτικό πυρήνα (kernel), βιβλιοθήκες και μεταφραστές που δημιουργούν την εκτελέσιμη προσομοίωση. Επίσης μειώνεται η διαδικασία υλοποίησης εκτενέστερου λογισμικού.

- Ευελιξία και λεπτομερής μοντελοποίηση. Τα μοντέλα των πρωτοκόλλων και των αλγορίθμων χρησιμοποιούν μια υβριδική προσέγγιση γλώσσας που ονομάζεται Proto-C και επιτρέπει στους χρήστες να ενσωματώνουν τον κώδικα γλώσσας C μέσα σε γραφικά προσδιορισμένα διαγράμματα πεπερασμένων καταστάσεων (finite state machines).



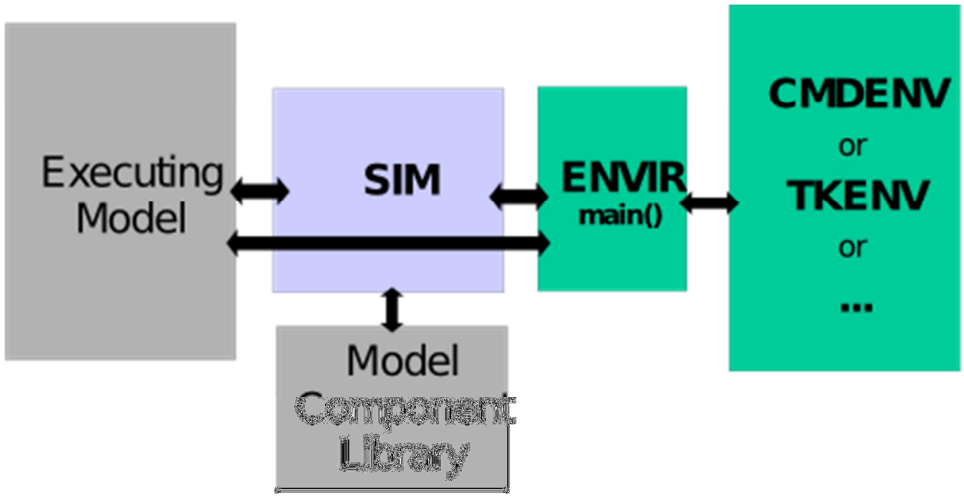
Εικόνα 1: Αρχιτεκτονική του OPNET

1.4.4 OMNET++ (Optical Micro-Networks Plus Plus): Το OMNeT++ είναι ένας αντικειμενοστραφής προσομοιωτής διακριτών γεγονότων (Discrete Event Simulator – DES). Διαθέτει μια «γενική» αρχιτεκτονική έτσι μπορεί να χρησιμοποιηθεί σε διάφορους τομείς όπως:

- Μοντελοποίηση ασύρματων και ενσύρματων δικτύων επικοινωνιών
- Μοντελοποίηση πρωτοκόλλων
- Μοντελοποίηση δικτύων ουρών

- Μοντελοποίηση μικροεπεξεργαστών και άλλων συστημάτων hardware

Γενικά μπορεί να χρησιμοποιηθεί για την προσομοίωση οποιουδήποτε συστήματος για το οποίο είναι κατάλληλη η προσέγγιση των διακριτών γεγονότων και το οποίο μπορεί να αντιστοιχηθεί σε οντότητες που επικοινωνούν μεταξύ τους ανταλλάσοντας μηνύματα.



Εικόνα 2: Αρχιτεκτονική του OMNET++

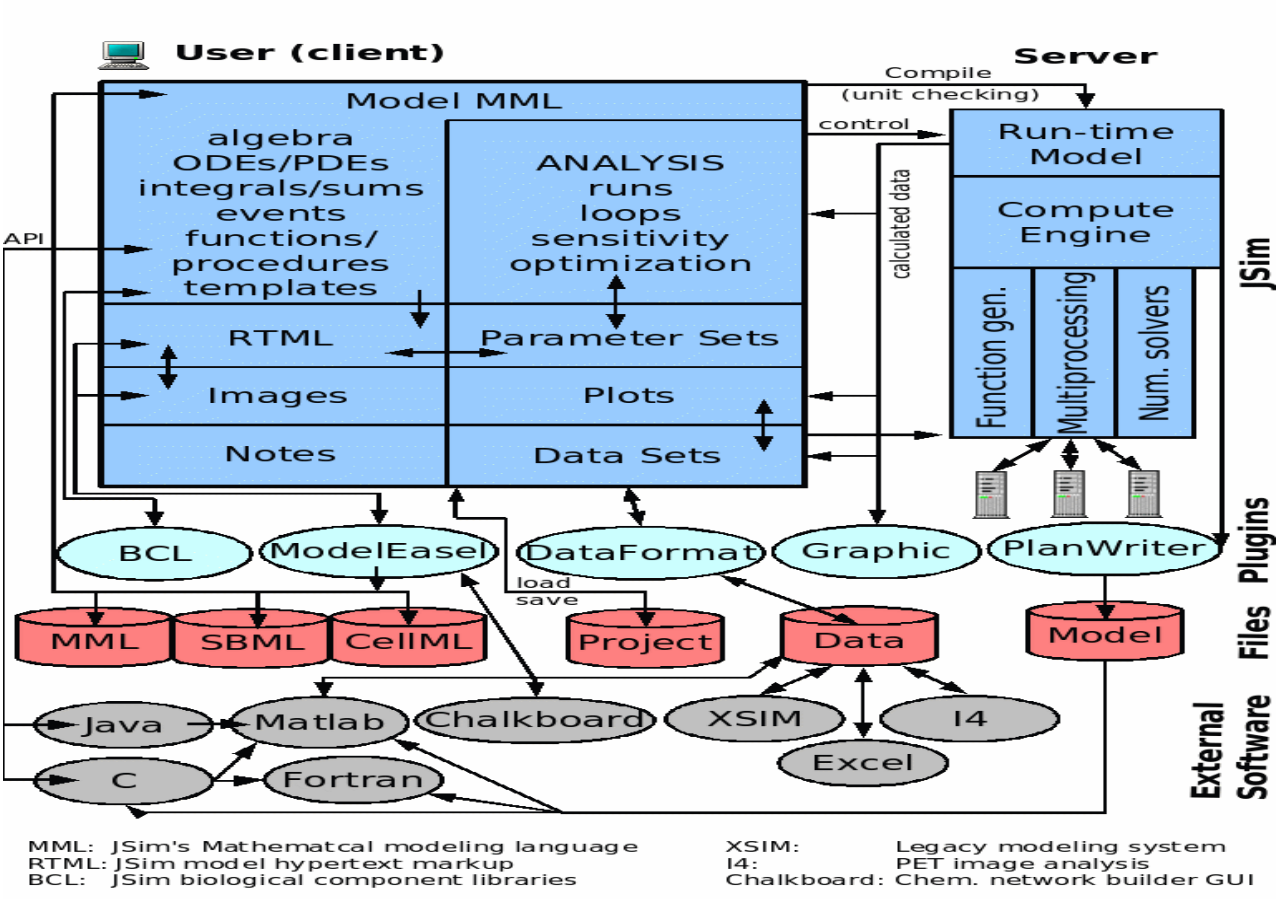
1.4.5 JSIM (Java-based simulation) : Το J-Sim αποτελεί ένα περιβάλλον προσομοίωσης που βασίζεται κυρίως σε αυτόνομα components. Για την δικτυακή προσομοίωση ορίζεται ένα δίκτυο μεταγωγής πακέτων. Το μοντέλο αυτό ορίζει μια γενική δομή για ένα κόμβο, είτε αυτός είναι ένας τελικός host είτε είναι δρομολογητής, καθώς και τα γενικά συστατικά του δικτύου, τα οποία μπορούν να χρησιμοποιηθούν σαν βασικές κλάσεις προκειμένου να υλοποιηθούν πρωτόκολλα πολλαπλών επιπέδων. Επιπρόσθετα, το μοντέλο αυτό είναι ικανό να εξυπηρετήσει πολλαπλές δικτυακές αρχιτεκτονικές, όπως είναι η IETF, η δικτυακή αρχιτεκτονική των κινητών ασύρματων δικτύων καθώς και η δικτυακή αρχιτεκτονική των WDM-based οπτικών δικτύων.

Τα πιο σημαντικά χαρακτηριστικά του J-Sim είναι τα ακόλουθα:

- Ένα χαλαρά συνδεδεμένο προγραμματιστικό μοντέλο βασισμένο σε components.
- Μια προσομοίωση πραγματικού χρόνου που βασίζεται κατά κύριο λόγο σε διεργασίες.

- Ένα αποτελεσματικό σύνολο από Internet Integrated, Differentiated και Best Effort Services πρωτόκολλα.

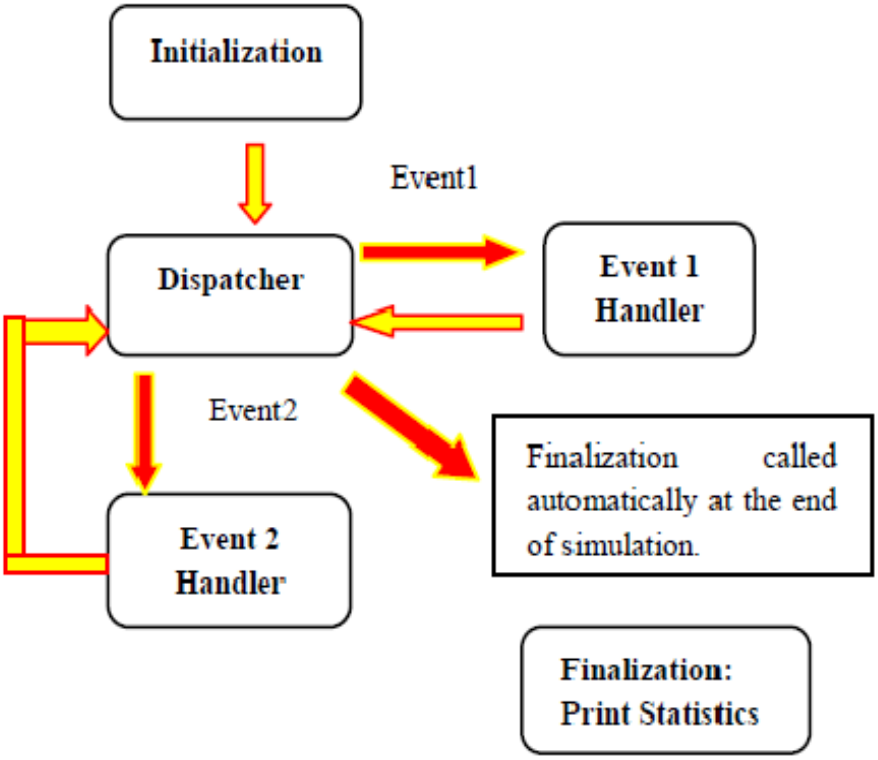
Πρέπει να σημειωθεί πως ο δικτυακός αυτός προσομοιωτής έχει υλοποιηθεί εξολοκλήρου σε Java παρά το γεγονός ότι η τρέχουσα έκδοση του επαυξήθηκε με μια Java υλοποίηση του TCL διερμηνέα που αποκαλείται Jacl.



Εικόνα 3: Αρχιτεκτονική του JSIM

1.4.6 QUALNET: Το QualNet είναι ένα εργαλείο μοντελοποίησης για ασύρματα και ενσύρματα δίκτυα. Η μηχανή προσομοίωσης του QualNet είναι εξαιρετικά κλιμακώσιμη, με ικανότητες προσομοίωσης με υψηλή πιστότητα μοντέλων δικτύων αποτελούμενα από δεκάδες χιλιάδες κόμβων. Το QualNet κάνει πολύ καλή χρήση των διαθέσιμων υπολογιστικών πόρων και γι'αυτό μπορεί να αναπαράγει μοντέλα δικτύων μεγάλης

κλίμακας σε μικρό χρόνο. Επίσης παράγει λεπτομερή αποτελέσματα για ανάλυση σε βάθος. Τέλος υποστηρίζει και την κατά δεσμίδες εκτέλεση προσομοιώσεων, η οποία εξοικονομεί ακόμα περισσότερο χρόνο εκτέλεσης των πειραμάτων.



Εικόνα 4: Αρχιτεκτονική του QUALNET

ΠΙΝΑΚΑΣ 3

Διαθεσιμότητα λογισμικού

Προσομοιωτής Δικτύου	**Διαθεσιμότητα (Ιστοσελίδα)**

NS2	Ελεύθερο για χρήση http://www.isi.edu/nsnam/ns/ns-build.html
NS3	Ελεύθερο για χρήση http://www.nsnam.org/ns-3-13/download/
OPNET	Εμπορικός προσομοιωτής δικτύου http://www.opnet.com/university_program/itguru_academic_edition/
NetSim	Εμπορικός προσομοιωτής δικτύου για φοιτητική χρήση http://www.ssfnet.org/download/license.html
OMNet++	Ελεύθερο για ακαδημαϊκή χρήση και μη κερδοσκοπικού χαρακτήρα http://www.omnetpp.org/component/docman/cat_view/17-downloads/1-omnet-releases
REAL	Ελεύθερο για χρήση http://www.cs.cornell.edu/skeshav/real/overview.html
J-Sim	Ελεύθερο για χρήση https://sites.google.com/site/jsimofficial/downloads
Qualnet	Εμπορικός προσομοιωτής δικτύου http://www.it.iitb.ac.in/~qualnet/

Κεφάλαιο 2

2.1 Ο εξομοιωτής GNS3

Ο GNS3 είναι ένας σύγχρονος εξομοιωτής(emulator) δικτύων που προσομοιώνει σύνθετα δίκτυα, όσον το δυνατόν πιο κοντά στο τρόπο που λειτουργούν τα κανονικά δίκτυα. Όσοι ασχολούνται με τον τομέα της πληροφορικής είναι λίγο πολύ εξοικειωμένοι με το VMWare και το Virtual PC, τα οποία χρησιμοποιούνται για να εξομοιώσουν λειτουργικά συστήματα σε ένα εικονικό περιβάλλον. Αυτά τα προγράμματα επιτρέπουν να τρέξεις λειτουργικά συστήματα όπως Windows XP Professional ή Ubuntu Linux σε ένα εικονικό περιβάλλον στον υπολογιστή. Ο GNS3 επιτρέπει ίδιου τύπου εξομοίωση χρησιμοποιώντας Cisco IOS.

Είναι ένα λογισμικό ανοικτού κώδικα και λειτουργεί σε διάφορα λειτουργικά συστήματα όπως Microsoft Windows, Linux, MacOS. Παρέχει ολοκληρωμένες και ακριβείς προσομοιώσεις χρησιμοποιώντας τους ακόλουθους εξομοιωτές για να τρέξει τα ίδια λειτουργικά συστήματα όπως σε αληθινά δίκτυα:

- Dynamips, εξομοιωτής Cisco IOS
- VirtualBox, τρέχει λειτουργικά συστήματα με χρήση desktop και server, όπως και το Juniper JunOS.
- QEMU, εξομοιωτής μηχανής ανοικτού κώδικα που τρέχει Cisco ASA, PIX και IPS.

Η εξομοίωση είναι δυνατή για έναν μεγάλο αριθμό δρομολογητών (routers) και PIX firewalls. Η μέγιστη διεκπεραιωτική ικανότητα (Throughput) που επιτυγχάνεται σε ένα εικονικό περιβάλλον εξομοίωσης είναι 1000 packets/sec. Ο χρήστης ουσιαστικά μπορεί να τρέξει ένα πραγματικό Cisco IOS, βλέποντας ακριβώς τι παράγει το IOS και έχει τη δυνατότητα πρόσβασης σε οποιαδήποτε εντολή και παράμετρο υποστηρίζεται από αυτό.

Αξίζει να αναφερθεί ότι παρέχεται η δυνατότητα προσθήκης ενός Wireshark capture, που σημαίνει ότι ο χρήστης μπορεί να αποθηκεύσει ένα στιγμιότυπο ενός συνδέσμου και στη συνέχεια να το στείλει στο πρόγραμμα Wireshark για ανάλυση πακέτων.

Το GNS3 είναι ένα εξαιρετικό εργαλείο για μηχανικούς και διαχειριστές δικτύου, όπως και για σπουδαστές Cisco CCNA, CCNP και CCIE. Το γεγονός ότι είναι ένα ελεύθερο

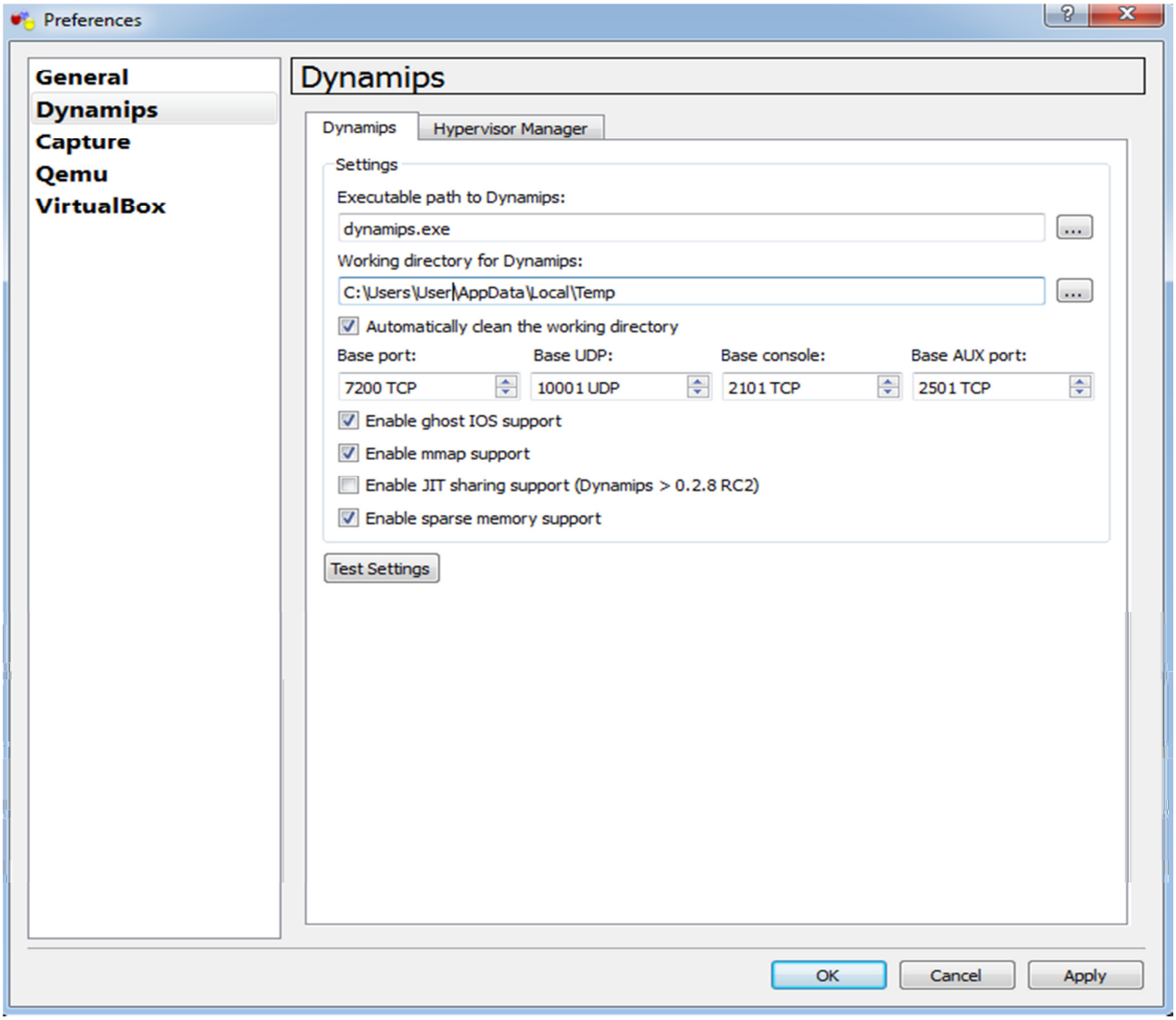
λογισμικό(open source), όπως και το φιλικό του interface και η ευκολία στην εκμάθηση και την χρήση του αποτελούν τα μεγαλύτερα πλεονεκτήματά του.

Παρακάτω παρουσιάζονται δύο διαφορετικά σενάρια, τα οποία υλοποιήθηκαν στον εξομοιωτή. Στο πρώτο εξετάζεται η λειτουργία του πρωτοκόλλου μετάφρασης διευθύνσεων NAT και στο δεύτερο η λειτουργία του πρωτοκόλλου δρομολόγησης EIGRP.

2.2 Απαιτούμενα

Χρησιμοποιήσαμε την έκδοση GNS3-0.8.6-all-in-one η οποία είναι διαθέσιμη στο σύνδεσμο: http://sourceforge.net/projects/gns-3/files/GNS3/0.8.6/GNS3-0.8.6-all-in-one.exe/download. Η συγκεκριμένη έκδοση περιέχει τους εξομοιωτές Dynamips και Qemu καθώς και τη κονσόλα διαχείρισης Putty. Απαιτείται να ενσωματωθεί στον GNS3 η εικόνα του Cisco router, δηλαδή το Cisco IOS image, η οποία αποτελεί μια ψηφιακή προσομοίωση λειτουργίας ενός μοντέλου Cisco router. Η εικόνα είναι ένα συμπιεσμένο αρχείο μορφής BIN(σε δυαδική μορφή). Η αποσυμπίεσή του γίνεται αυτόματα μέσω του GNS3. Το αρχείο που προκύπτει από την αποσυμπίεση είναι αρχείο τύπου image και αποτελεί το μοντέλο του router που θα χρησιμοποιηθεί στα σενάρια που θα υλοποιήσουμε. Επιλέξαμε την c7000 πλατφόρμα και το μοντέλο 7200.

2.3 Ρυθμίζοντας το GNS3

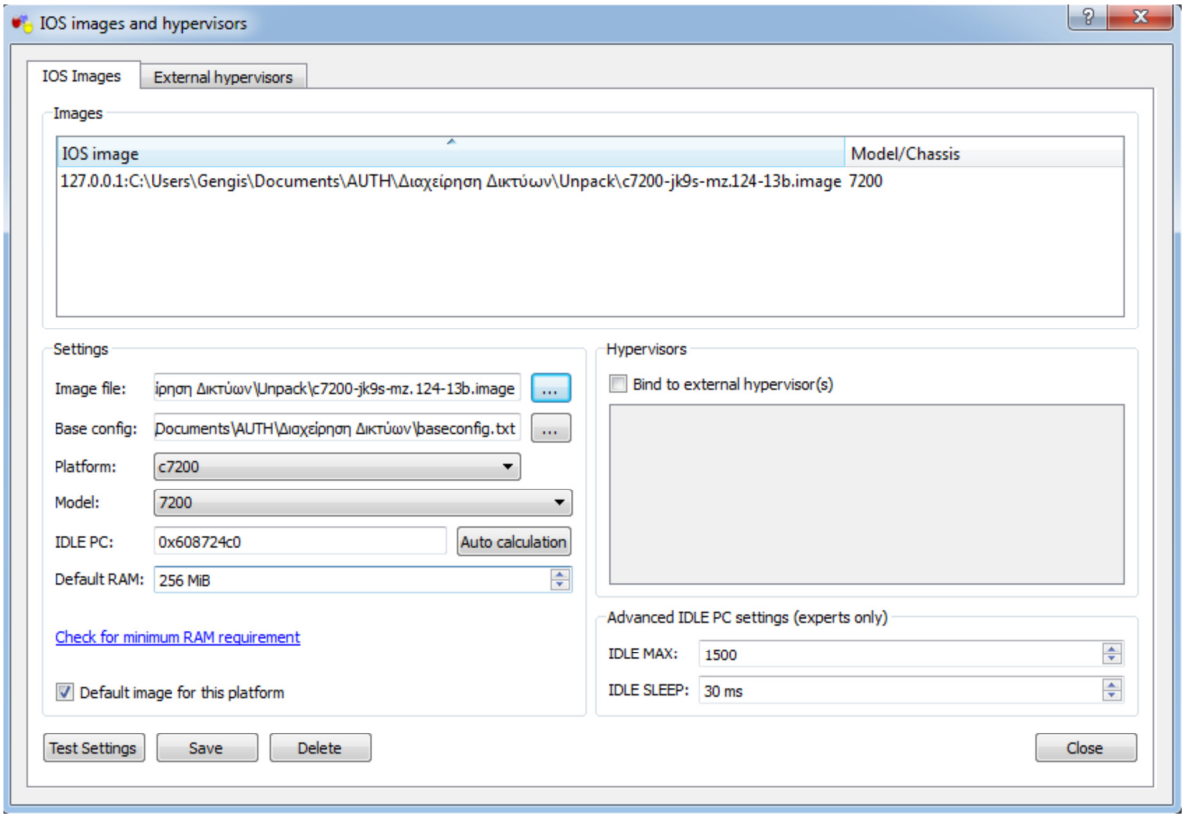


Πριν ξεκινήσουμε να δημιουργούμε το δίκτυό μας, πρέπει να κάνουμε τις απαραίτητες ρυθμίσεις στο πρόγραμμα. Αρχικά, στο Edit->Preferences->General επιλέγουμε ένα directory αποθήκευσης των project το οποίο είναι εύκολο να βρεθεί στο μέλλον. Ακριβώς πιο κάτω, θέτουμε το directory στο οποίο βρίσκεται το IOS image μας. Στη συνέχεια, επιλέγουμε στα πλάγια την κατηγορία Dynamips, και επαληθεύουμε με το Test Settings την ορθή λειτουργία του εξομοιωτή. Αν δεν υπάρξει σφάλμα, εμφανίζεται μήνυμα σωστής εκκίνησης.

Στο Edit->IOS Images and Hypervisors θα ενσωματώσουμε την IOS εικόνα στο project μας. Στην επιλογή Image file επιλέγουμε το αρχείο της IOS εικόνας. Σε περίπτωση που

αυτό δεν είναι αποσυμπιεσμένο, το GNS3 ρωτάει τον χρήστη αν θέλει να αποσυμπιεστεί αυτόματα το αρχείο. Αυτό ισχύει μόνο για τις καινούριες εκδόσεις του GNS3 και οι παλιότερες απαιτούν αποσυμπίεση μέσω του unpacker που αναφέρθηκε προηγουμένως. Το unpacked αρχείο που προκύπτει είναι αρχείο τύπου image και αποτελεί το μοντέλο του router που θα χρησιμοποιήσουμε στο project μας. Ανάλογα με το image που έχουμε, μπορούμε να ρυθμίσουμε τη πλατφόρμα και το μοντέλο του router που θα χρησιμοποιήσουμε. Για την εργασία επιλέγουμε την c7200 πλατφόρμα και μοντέλο 7200.

Ένα σημαντικό χαρακτηριστικό του GNS3 είναι το IDLEPC. Χωρίς αυτό η χρήση της CPU του υπολογιστή θα ήταν μόνιμα στο 100% όσο το router είναι ανοιχτό. Πατώντας την επιλογή auto calculation το GNS3 μαντεύει μια τιμή IDLEPC, η οποία συμβολίζει τη διεργασία του IOS που καταναλώνει την περισσότερη υπολογιστική ισχύ και την σκοτώνει. Μια τελευταία ρύθμιση που απαιτείται είναι η Default RAM. Η προεπιλεγμένη ρύθμιση των 256MB είναι κατάλληλη για κάθε IOS**.** Η επιλογή Check for minimum RAM requirement οδηγεί στο http://tools.cisco.com/, όπου τυπώνοντας το ακριβές όνομα του IOS (c7200-jk9s-mz.124-13b.bin στην περίπτωσή μας) γίνεται ανάλυση της έκδοσης, της απαιτούμενης μνήμης και των χαρακτηριστικών του IOS. Πατώντας save, αποθηκεύονται οι επιλογές μας και εμφανίζεται το IOS Image στο πλαίσιο Images.



2.4 Υλοποίηση Σεναρίων

2.4.1 Σενάριο 1ο: Στατικό NAT (Static NAT)

Το πρωτόκολλο μετάφρασης διευθύνσεων δικτύου σχεδιάστηκε για απλοποίηση και διατήρηση των IP διευθύνσεων αφού αυτό που κάνει είναι να επιτρέπει σε ιδιωτικά δίκτυα που χρησιμοποιούν μη εγγεγραμμένες IP διευθύνσεις να έχουν σύνδεση με το Internet. Το σύστημα NAT λειτουργεί σε κάποιον δρομολογητή, ο οποίος συνδέει συνήθως δύο δίκτυα και μεταφράζει τις ιδιωτικές (μη μοναδικές στον παγκόσμιο ιστό) διευθύνσεις του εσωτερικού δικτύου σε νόμιμες διευθύνσεις προτού τα πακέτα προωθηθούν σε άλλο δίκτυο. Σαν μέρος αυτής της λειτουργίας το NAT μπορεί να ρυθμιστεί να κάνει γνωστή μόνο μία διεύθυνση στον έξω κόσμο για ολόκληρο το δίκτυο που συνδέει με αυτόν. Αυτό το χαρακτηριστικό παρέχει επιπλέον ασφάλεια αφού κρύβει ολόκληρο το εσωτερικό δίκτυο από το κόσμο πίσω από μία διεύθυνση.

Ο Πίνακας Μετάφρασης Διευθύνσεων που διατηρεί το NAT αποτελείται από καταχωρήσεις δύο τύπων: *μετάφραση εσωτερικών διευθύνσεων* και *μετάφραση*

εξωτερικών διευθύνσεων. Κάθε καταχώρηση αποτελείται από δύο μέρη: την τοπική διεύθυνση και τη παγκόσμια διεύθυνση.

Το στοιχείο της τοπικής διεύθυνσης ενός τύπου καταχώρησης εσωτερικής μετάφρασης είναι μία διεύθυνση παρμένη από το πεδίο των εσωτερικών τοπικών διευθύνσεων. Αναφερόμαστε σε μια τέτοια διεύθυνση ως γνωστόν με το χαρακτηρισμό "εσωτερική τοπική διεύθυνση" (IL). Το στοιχείο της παγκόσμιας διεύθυνσης μιας τέτοιας καταχώρησης είναι μια διεύθυνση που έχει εξαχθεί από το πεδίο των εσωτερικών παγκοσμίων διευθύνσεων που έχει δοθεί στο NAT και αναφερόμαστε σε μια τέτοια διεύθυνση με το όνομα "εσωτερική παγκόσμια διεύθυνση" (IG).

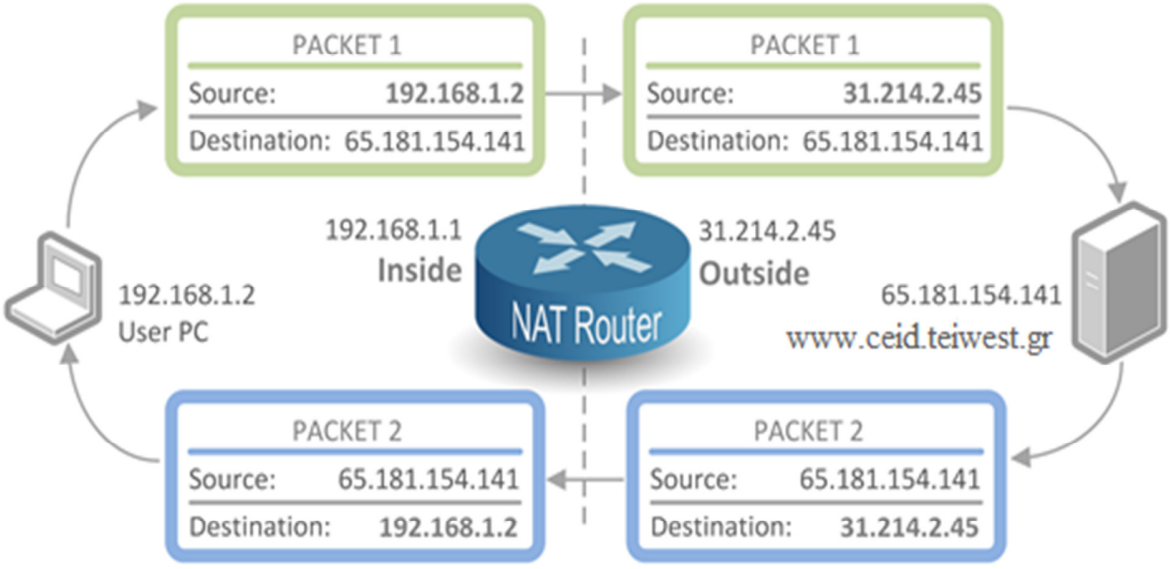
Το στοιχείο της τοπικής διεύθυνσης ενός εξωτερικού τύπου μετάφρασης διεύθυνσης είναι μία διεύθυνση που έχει εξαχθεί από το εξωτερικό τοπικό πεδίο διευθύνσεων που υπάρχει στο NAT. Αναφερόμαστε σε μια τέτοια διεύθυνση με το χαρακτηρισμό "εξωτερική τοπική διεύθυνση" (OL). Τέλος το στοιχείο της παγκόσμιας διεύθυνσης μιας τέτοιας καταχώρησης είναι μια διεύθυνση ενός υπολογιστή έξω από την επιχείρηση. .Αναφερόμαστε σε μια τέτοια διεύθυνση με το χαρακτηρισμό "εξωτερική παγκόσμια διεύθυνση" (OG).

Περιληπτικά:

- *Εσωτερική Τοπική (IL)*—Η IP διεύθυνση που δίνεται σε υπολογιστή εντός της επιχείρησης. Αυτή η διεύθυνση μπορεί να είναι παγκοσμίως μοναδική, να έχει παρθεί από το ιδιωτικό πεδίο διευθύνσεων ή να έχει ανατεθεί επίσημα σε κάποια άλλη επιχείρηση.
- *Εσωτερική Παγκόσμια (IG)*—Η IP διεύθυνση ενός υπολογιστή εντός της επιχείρησης όπως εμφανίζεται στο internet. Αυτές οι διευθύνσεις έχουν εξαχθεί από ένα παγκοσμίως μοναδικό πεδίο διευθύνσεων που τυπικά παρέχεται από τον ISP.
- *Εξωτερική Τοπική (EL)*—Η IP διεύθυνση ενός υπολογιστή εκτός της επιχείρησης όπως φαίνεται μέσα στην επιχείρηση. Αυτές οι διευθύνσεις παίρνονται εάν είναι επιθυμητό από το πεδίο διευθύνσεων που ορίζεται στο RFC 1918 και αναφέρεται σε ιδιωτικά πεδία διευθύνσεων.

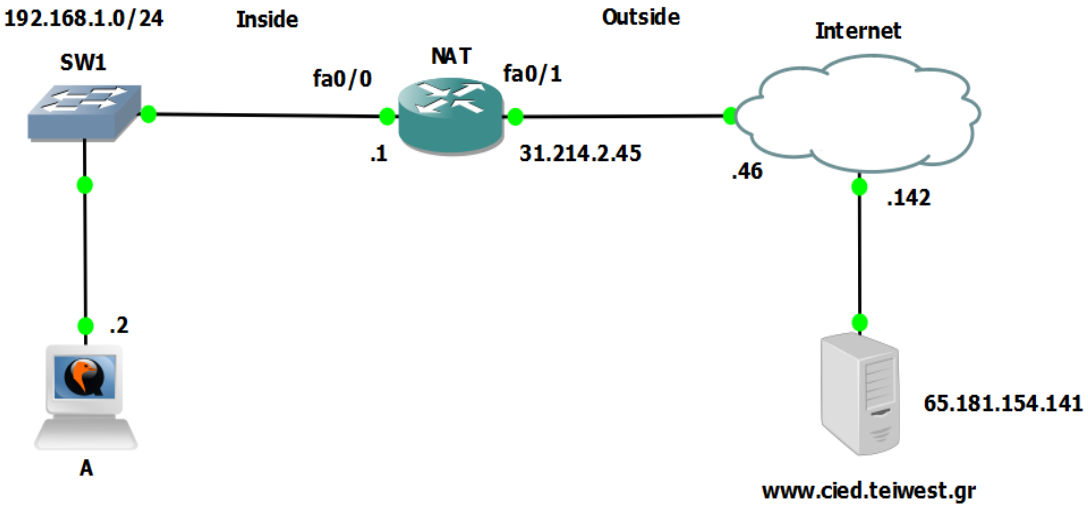
- *Εξωτερική Παγκόσμια (EG)*—Η IP διεύθυνση ενός υπολογιστή εκτός της επιχείρησης όπως φαίνεται στο internet. Αυτές οι διευθύνσεις παίρνονται από ένα παγκοσμίως μοναδικό πεδίο διευθύνσεων.

Σε αυτό το σενάριο εξετάζουμε τη λειτουργία Static NAT(στατική μετάφραση διεύθυνσης δικτύου). Η παρακάτω εικόνα δείχνει αναλυτικά τα βήματα εκτέλεσης της λειτουργίας NAT από τον router.



Ο χρήστης PC(192.168.1.2) δημιουργεί ένα πακέτο και το στέλνει στο www.ceid.teiwest.gr (65.181.154.141). Ο router NAT παραλαμβάνει το πακέτο στο interface εισόδου. Αλλάζει την IP διεύθυνση πηγής στη κεφαλίδα του πακέτου από 192.168.1.2 σε 31.214.2.45 πριν τη στείλει από το interface εξόδου. Το πακέτο με IP διεύθυνση πηγής 31.214.2.45 παραλαμβάνεται από τον server www.ceid.teiwet.gr. Ο server δεν γνωρίζει ότι η IP διεύθυνση πηγής έχει αλλάξει από το router NAT και πιστεύει ότι το πακέτο έρχεται από την 31.214.2.45. Ο server δημιουργεί ένα πακέτο με την δική του IP διεύθυνση(65.181.154.141) σαν πηγή και την 31.214.2.45 σαν διεύθυνση προορισμού και το στέλνει πίσω. Ο router αλλάζει την IP διεύθυνση προορισμού στη κεφαλίδα του πακέτου από 31.214.2.45 σε 192.168.1.2 και το στέλνει στο interface εισόδου του. Ο χρήστης PC τελικά παραλαμβάνει το πακέτο.

Παρακάτω παρουσιάζεται η τοπολογία του δικτύου που υλοποιήσαμε στον GNS3. Χρησιμοποιήσαμε MicroCore Linux hosts τα οποία εξομοιώνονται με το Qemu για τον υπολογιστή Α και τον server(www.ceid.teiwest.gr) .



Εικόνα 5: Σενάριο Static Nat στον GNS3

Αρχικά ρυθμίζουμε τις IP διευθύνσεις στους δρομολογητές NAT και Internet(εσωτερικά και εξωτερικά interface) καθώς επίσης και στον υπολογιστή Α και στον server(www.ceid.teiwest.gr). Παρακάτω παρουσιάζονται οι εντολές που γράφτηκαν στη κονσόλα των δύο router.

#NAT	**#Internet**
NAT(config)#int fa0/0	Internet(config)#int fa0/0
NAT(config-if)#ip address 192.168.1.1 255.255.255.0	Internet(config-if)#ip address 31.214.2.46 255.255.255.252
NAT(config-if)#ip nat inside	Internet(config-if)#no shutdown

NAT(config-if)#no shutdown NAT(config-if)#exit NAT(config)#int fa0/1 NAT(config-if)#ip address 31.214.2.45 255.255.255.252 NAT(config-if)#ip nat outside NAT(config-if)#no shutdown NAT(config)#int l0 NAT(config-if)#ip address 192.168.1.2 255.255.255.0 NAT(config)# ip route 0.0.0.0 0.0.0.0 65.181.154.142 NAT(config)#access-list 1 permit 192.168.1.0 0.0.0.255 NAT(config)#ip nat inside source static 192.168.2.2 31.214.2.45 NAT(config)#ip route 65.181.154.140 255.255.255.252 31.214.2.46	Internet(config-if)#exit Internet(config)#int fa0/1 Internet(config-if)#ip address 65.181.154.142 255.255.255.252 Internet(config-if)#exit Internet(config)#int l1 Internet(config-if)#ip address 65.181.154.141 255.255.255.252

Με την εντολή **ip nat inside source static 192.168.2.2 31.214.2.45** εγκαθίσταται η αντιστοίχιση μεταξύ της εσωτερικής τοπικής διεύθυνσης 192.168.1.2 και της εσωτερικής παγκόσμιας διεύθυνσης 31.214.2.45. Για να δούμε το αποτέλεσμα δίνουμε την εντολή **sh ip nat translations** στο NAT και βλέπουμε το παρακάτω:

```
NAT# show ip nat translations
```

```
Pro Inside global      Inside local       Outside local      Outside global
--- 31.214.2.45        192.168.1.2        ---                ---
```

Αν θέλουμε να δούμε περισσότερες πληροφορίες δίνουμε την εντολή **sh ip nat statistics.**

```
NAT# show ip nat statistics
Total active translations: 1 (1 static, 0 dynamic; 0 extended)
Outside interfaces:
  FastEthernet0/1
Inside interfaces:
  FastEthernet0/0
Hits: 0  Misses: 0
CEF Translated packets: 0, CEF Punted packets: 0
Expired translations: 0
Dynamic mappings:
Appl doors: 0
Normaldoors: 0
QueuedPackets: 0
```

Μπορούμε να παράγουμε μερική κίνηση στέλνοντας πέντε ICMP πακέτα από τον host A στο server www.ceid.teiwest.gr: **ping –c 5 65.181.154.141** και έχουμε:

```
PING 65.181.154.141 (65.181.154.141): 56 data bytes
64 bytes from 65.181.154.141: seq=0 ttl=62 time=41.323 ms
64 bytes from 65.181.154.141: seq=1 ttl=62 time=44.565 ms
64 bytes from 65.181.154.141: seq=2 ttl=62 time=35.984 ms
64 bytes from 65.181.154.141: seq=3 ttl=62 time=37.608 ms
64 bytes from 65.181.154.141: seq=4 ttl=62 time=55.699 ms

--- 65.181.154.141 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 35.984/43.035/55.699 ms
```

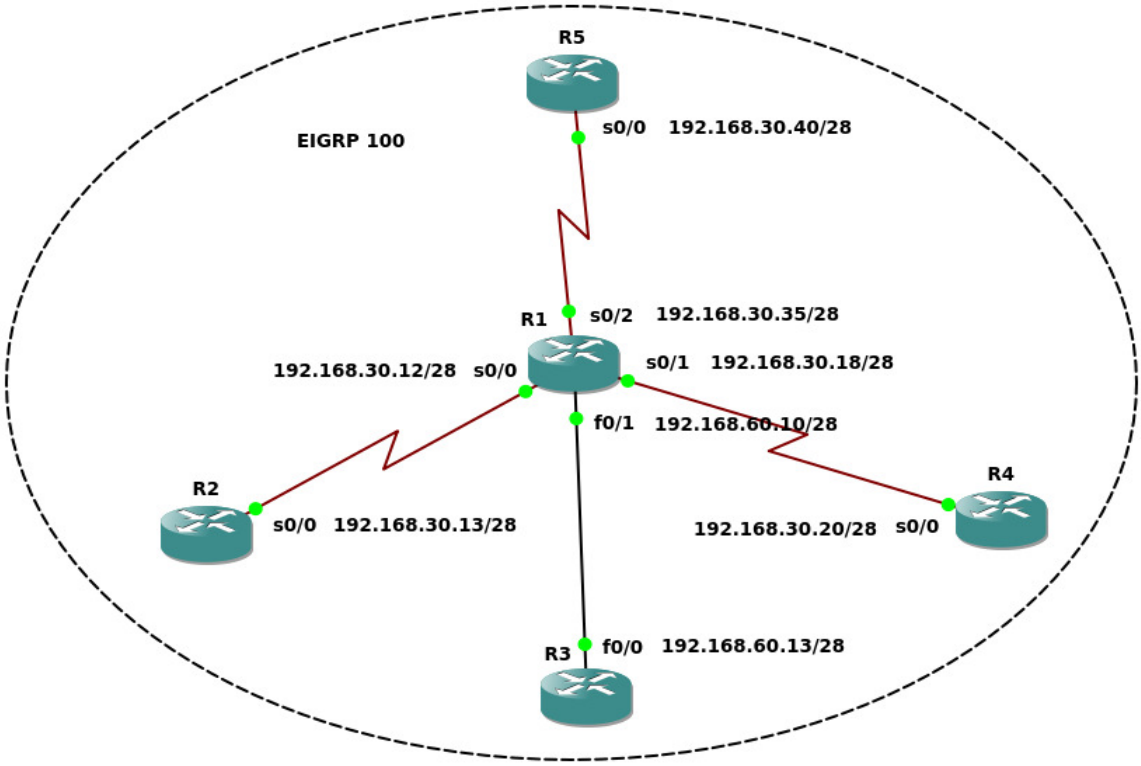
Αναμένουμε ότι ο router NAT θα έχει εκτελέσει κάποιες μεταφράσεις:

```
R1#show ip nat translations
Pro   Inside global      Inside local       Outside local         Outside global

icmp  31.214.2.45:38150  192.168.1.2:38150  65.181.154.141:38150  65.181.154.141:38150

---   31.214.2.45        192.168.1.2        ---                   ---
```

Στην έξοδο τώρα παρατηρούμε δύο γραμμές, με την πρώτη να αναφέρεται στην δικιά μας ρύθμιση στατικής NAT. Η δεύτερη γραμμή αναφέρεται σε μια πραγματική μετάφραση NAT των ICMP pings από τη διεύθυνση 192.168.1.2 στην 65.181.154.141 καθώς και των ping απάντησης από την αντίθετη κατεύθυνση.

2.4.2 Σενάριο 2ο: Πρωτόκολλο Δρομολόγησης EIGRP

Στο παρακάτω σενάριο θα υλοποιήσουμε το πρωτόκολλο δρομολόγησης EIGRP πάνω σε ένα μικρό δίκτυο 5 δρομολογητών, η τοπολογία στην οποία θα δουλέψουμε είναι η παρακάτω:



Εικόνα 6: Σενάριο EIGRP στον GNS3

Το σενάριο αποτελείται από 5 δρομολογητές τους οποίους πρέπει να ρυθμίσουμε με τον EIGRP, αυτό θα επιτευχθεί όταν θα μπορέσουν οι δρομολογητές να επικοινωνήσουν μεταξύ τους μέσω της εντολής ping και οι πίνακες δρομολόγησης αυτών τρέξουν το συγκεκριμένο πρωτόκολλο (το σύμβολο που αντιπροσωπεύει τον EIGRP είναι το "D").

Το IOS που θα χρησιμοποιηθεί είναι: **c7200-jk9s-mz.124-13b.bin**(με ελάχιστη μνήμη RAM 64 MB) φυσικά για το συγκεκριμένο σενάριο μπορούν να χρησιμοποιηθούν και άλλα IOS που κυκλοφορούν ελεύθερα στο διαδίκτυο.

Ρυθμίσεις δρομολογητών:

Όλοι οι δρομολογητές πρέπει να έχουν το ίδιο αριθμό αυτόνομου συστήματος (Autonomus System - AS) για να αναγνωρίζονται μεταξύ τους, στο συγκεκριμένο σενάριο το επιλεγμένο AS είναι το 100.

Το δίκτυο το οποίο έχουμε δημιουργήσει είναι **discontiguous network** λόγω των ip 192.158.30.0 και 192.168.60.0 σε δύο διαφορετικούς δρομολογητές και για αυτό θα χρησιμοποιήσουμε την εντολή no auto-summary σε κάθε δρομολογητή.

Το EIGRP (αρκτικόλεξο της φράσης Enhanced Interior Gateway Routing Protocol) είναι ένα πρωτόκολλο δρομολόγησης δικτύων υπολογιστών, αναπτυγμένο από την εταιρεία Cisco Systems, και βασισμένο στο παλιότερο πρωτόκολλο IGRP (με το οποίο είναι προς τα πίσω συμβατό). Ανήκει στην κατηγορία των πρωτοκόλλων Διανύσματος απόστασης (Distance Vector), και είναι βελτιστοποιημένο αφ' ενός προς την ελαχιστοποίηση της αστάθειας που συμβαίνει όταν αλλάξει η τοπολογία ενός δικτύου, και αφ' ετέρου προς την βέλτιστη αξιοποίηση του εύρους ζώνης και της επεξεργαστικής ισχύος του δρομολογητή. Οι περισσότερες από αυτές τις δυνατότητες αποτελούν μέρους του αλγόριθμου DUAL, ο οποίος αναπτύχθηκε από το ινστιτούτο SRI International. Ο αλγόριθμος DUAL εγγυάται την αποτροπή βρόχων στη δρομολόγηση, και την ταχεία εύρεση εναλλακτικών δρομολογίων, τηρώντας αναπληρωματικά δρομολόγια για κάθε δίκτυο.

Το πρωτόκολλο EIGRP ρυθμίζει την ανταλλαγή δρομολογίων, αφού πρώτα επιτευχθεί η άμεση γειτονία μεταξύ δύο τουλάχιστον router. Αυτή την γειτνίαση την αναπτύσσει

μέσω των μηνυμάτων του Hello protocol, τα οποία ανταλλάζονται ανά τακτά διαστήματα που ορίζονται από τους χρόνους ανταλλαγής αυτών των πακέτων. Οι τεχνικές και οι πληροφορίες του IGRP ισχύουν κατά τη χρήση του EIGRP, αλλά η χρήση του EIGRP είναι πιο αποτελεσματική.

Τα δίκτυα υπολογιστών που διασυνδέονται με το πρωτόκολλο EIGRP είναι πιο αναπτυγμένα δομικά.

Θα ακολουθήσουμε 2 βασικά βήματα για να υλοποιήσουμε το σενάριο, αρχικά στο βήμα 1 θα ρυθμίσουμε κάποιες συγκεκριμένες IP σε καθένα δρομολογητή χωριστά και στο βήμα 2 θα υλοποιήσουμε το πρωτόκολλο δρομολόγησης EIGRP.

Βήμα 1: Ρυθμίζουμε σε όλους τους δρομολογητές της IP διεύθυνσης.

Το πρώτο βήμα 1 απαιτεί την ρύθμιση των δρομολογητών το καθένα ξεχωριστά ώστε στην συνέχεια να μπορούμε να επιβεβαιώσουμε ότι οι διεπαφές είναι σε λειτουργία και υπάρχει επικοινωνία των απευθείας συνδεδεμένων δρομολογητών.

Για να επιβεβαιώσουμε ότι όλες οι διεπαφές είναι ενεργοποιημένες πρέπει να πληκτρολογήσουμε την εντολή **“ show ip interface brief ”**, και τα αποτελέσματα που θα δούμε πρέπει να είναι τα παρακάτω:

```
R1#sh ip int brief
Interface              IP-Address      OK? Method Status                Protocol
Serial0/0              192.168.30.12   YES manual up                    up
Serial0/1              192.168.30.18   YES manual up                    up
Serial0/2              192.168.30.35   YES manual up                    up
Serial0/3              unassigned      YES unset administratively down down
FastEthernet1/0        192.168.60.10   YES manual                       up
```

Σημείωση : Όλα τα status και όλα τα πρωτόκολλα των συνδεδεμένων διεπαφών πρέπει να είναι up.

Μπορούμε να δούμε όλους τους γείτονες του R1 πληκτρολογώντας την εντολή " show cdp neighbors" στον δρομολογητή R1, τα αποτελέσματα είναι εμφανή παρακάτω:

```
R1#sh cdp neigh

Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge

                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local IntrfceHoldtmeCapability  Platform  Port ID

R2               Ser 0/0           121        R S I      3640      Ser 0/0

R3               Fas 1/0           118        R S I      3640      Fas 0/0

R4               Ser 0/1           162        R S I      3640      Ser 0/0

R5               Ser 0/2           142        R S I      3640      Ser 0/0
```

Σε αυτό το σημείο μπορούμε να ελέγξουμε την σύνδεση ανάμεσα σε απευθείας συνδεδεμένες διεπαφές. Για παράδειγμα ας ελέγξουμε την σύνδεση μεταξύ του δρομολογητή RI και R2 (192.168.30.13). Το ping από το s0/0 του δρομολογητή R1 στο s0/0 του δρομολογητή R2(192.168.30.13) θα είναι επιτυχές, για το λόγο του αληθές :

```
R1#ping 192.168.30.13

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.30.13, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 60/70/100 ms
```

Φυσικά το ping μεταξύ μακρινών διεπαφών θα είναι ανεπιτυχής. Για παράδειγμα δεν υπάρχει επικοινωνία μεταξύ του δρομολογητή R2 και R4. Το ping από το s0/0 του R2

στο s0/0 του R4 θα είναι αποτυχημένη, αυτό γίνετε επειδή δεν έχουμε ενεργοποιήσει κάποιο πρωτόκολλο δρομολόγησης , για το λόγο του αληθές :

```
R2#ping 192.168.30.20

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.30.20, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)
```

<u>Βήμα 2</u>: Ενεργοποιούμε τον EIGRP σε όλους τους δρομολογητές.

Αφότου έχουμε ενεργοποιήσει στους δρομολογητές τον EIGRP είναι λογικό να υπάρχει επικοινωνία από τον δρομολογητή R2 στο R4. Κάνοντας ping από το s0/0 του R2 στο s0/0 του R4 θα δούμε ότι η επικοινωνία επιτυγχάνεται κανονικά:

```
R2#ping 192.168.30.20

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 192.168.30.20, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 64/68/88 ms
```

Ελέγχοντας τους πίνακες δρομολόγησης των δρομολογητών R2, R3, R4, R5 θα διαπιστώσουμε πως λειτουργούν σωστά. Για παράδειγμα, χρησιμοποιώντας την εντολή **‘‘ show ip route’’** στον δρομολογητή R5 θα δούμε τα εξής:

```
R5#sh ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
   192.168.30.0/28 is subnetted, 3 subnets
C       192.168.30.32 is directly connected, Serial0/0
D       192.168.30.16 [90/2681856] via 192.168.30.35, 00:06:31, Serial0/0
D       192.168.30.0 [90/2681856] via 192.168.30.35, 00:06:31, Serial0/0
     192.168.60.0/28 is subnetted, 1 subnets
D       192.168.60.0 [90/2172416] via 192.168.30.35, 00:06:31, Serial0/0
```

Σημείωση : Με την εντολή **" show ip route "** μπορούμε να διαπιστώσουμε αυτό που αναφέραμε νωρίτερα δηλαδή ότι το γράμμα "D" συμβολίζει ότι η επικοινωνία γίνεται μέσω του EIGRP.

Μπορούμε επίσης να ελέγξουμε τούς γείτονες του κάθε δρομολογητή με την εντολή **" show ip eigrp neighbors ".** Παρακάτω ένα παράδειγμα πάνω στον δρομολογητή R1.

```
R1#shipeigrpneighbors
IP-EIGRPneighborsforprocess100
H   Address                 Interface       Hold Uptime   SRTT   RTO  QSeq
                                            (sec)         (ms)       CntNum
3   192.168.30.40           Se0/2             10 00:08:05   56   336  0  5
2   192.168.30.20           Se0/1             13 00:08:53   60   360  0  5
1   192.168.60.13           Fa1/0             11 00:09:42   97   582  0  5
```

```
0   192.168.30.13           Se0/0              9 00:10:38   65   390  0  5
```

Για να δούμε την τοπολογία ενός δρομολογητή , χρησιμοποιούμε την εντολή **“ show ip eigrp topology “.** Παρακάτω η έξοδος του δρομολογητή R4.

```
R4#showipeigrptopology
IP-EIGRP Topology Table for AS(100)/ID(192.168.30.20)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
 r - reply Status, s - sia Status

P 192.168.60.0/28, 1 successors, FD is 2172416
via 192.168.30.18 (2172416/28160), Serial0/0
P 192.168.30.32/28, 1 successors, FD is 2681856
via 192.168.30.18 (2681856/2169856), Serial0/0
P 192.168.30.16/28, 1 successors, FD is 2169856
via Connected, Serial0/0
P 192.168.30.0/28, 1 successors, FD is 2681856
via 192.168.30.18 (2681856/2169856), Serial0/0
```

Κεφάλαιο 3

3.1 Ο εξομοιωτής Boson NetSim

3.1.1 Γενικά

Το Boson NetSim είναι ένα πρόγραμμα που προσομοιώνει υλικό και λογισμικό συστημάτων δικτύωσης Cisco και είναι σχεδιασμένο προκειμένου να βοηθάει το χρήστη

στην εκμάθηση της δομής εντολών του Cisco IOS. Συγκαταλέγεται στους εμπορικούς εξομοιωτές και βασίζεται στην πλατφόρμα των Windows.

Το NetSim χρησιμοποιεί τον Router Simulator και τεχνολογίες λογισμικού EROUTER, σε συνδυασμό με το εργαλείο Virtual Packet Technology προκειμένου να παράγει πακέτα. Αυτά τα πακέτα δρομολογούνται μέσω του προσομοιωμένου δικτύου επιτρέποντας στο NetSim να δημιουργήσει έναν κατάλληλο εικονικό πίνακα δρομολόγησης και να προσομοιώσει ένα πραγματικό δίκτυο.

Κάποια προηγμένα χαρακτηριστικά τα οποία περιλαμβάνει είναι τα παρακάτω:

- Προσομοίωση δικτυακής κίνησης με την τεχνολογία Virtual Packet Technology.
- Παρέχει τη δυνατότητα παραμετροποίησης των ρυθμίσεων των δικτυακών συσκευών: μέσω της κονσόλας τους(console mode) ή κάνοντας telnet στην δικτυακή συσκευή(telnet mode).
- Υποστηρίζει μέχρι και 200 συσκευές σε μία δικτυακή τοπολογία.
- Επιτρέπει στους εκπαιδευτές να δημιουργήσουν και να συμπεριλάβουν δικά τους σενάρια, προσφέροντας την δυνατότητα αξιολόγησης(grading) ορθής λειτουργίας τους.
- Περιλαμβάνει σενάρια(labs) τα οποία υποστηρίζουν SDM(Security Device Manager) προσομοίωση.
- Εκτός από τις συσκευές Cisco περιλαμβάνει και άλλου τύπου συσκευές, όπως TFTP Server, TACACS+ και Packet Generator.

Είναι διαθέσιμος σε τρεις εκδόσεις: το Netsim για CCENT, το NetSim για CCNA και το NetSim για CCNP. Τα καθένα από τα παραπάνω υποστηρίζει τις τεχνολογίες και τις δεξιότητες που χρειάζεται κάποιος για να αποκτήσει τις αντίστοιχες πιστοποιήσεις. Στο παρακάτω πίνακα παρουσιάζονται συγκεντρωτικά τα χαρακτηριστικά των τριών εκδόσεων του NetSim.

	NetSim for CCENT 8.0	NetSim for CCNA 8.0	NetSim for CCNP 8.0
Labs	44	94	163
Lab grading function	✔	✔	✔
Υποστηριζόμενες συσκευές	44	44	47
BGP and IS-IS			✔
Configuring QoS			✔
Crossover/Straight-Through Cables	✔	✔	✔
Custom Queuing and Compression			✔
EIGRP	✔	✔	✔
EIGRP Authentication, Debugging, Optimization, Path Selection, Summarization and Wildcard Mask			✔
HSRP and VRRP			✔
IGRP	✔	✔	✔
IPSec			✔
IPv6 Addressing		✔	✔
ISDN (BRI/PRI)	✔	✔	✔
Legacy DDR	✔	✔	✔
Low Latency Queuing			✔
Multicasting Configuration			✔
Network Baseline Discovery	✔	✔	✔
OSPF (Single-Area/Multi-Area)		✔	✔
OSPF Authentication		✔	✔
OSPF Route Summarization			✔
OSPF Stub and Totally Stubby Areas			✔
OSPFv3 using IPv6 Addresses			✔
PAP			✔
Per-VLAN Spanning Tree (PVST)			✔
Point-to-Point/Point-to-Multipoint Serial	✔	✔	✔
Policy Routing			✔
PortFast			✔
PPP and CHAP		✔	✔
RIPv1 and RIPv2	✔	✔	✔
RIPng v3 for IPv6		✔	✔
Route Redistribution			✔

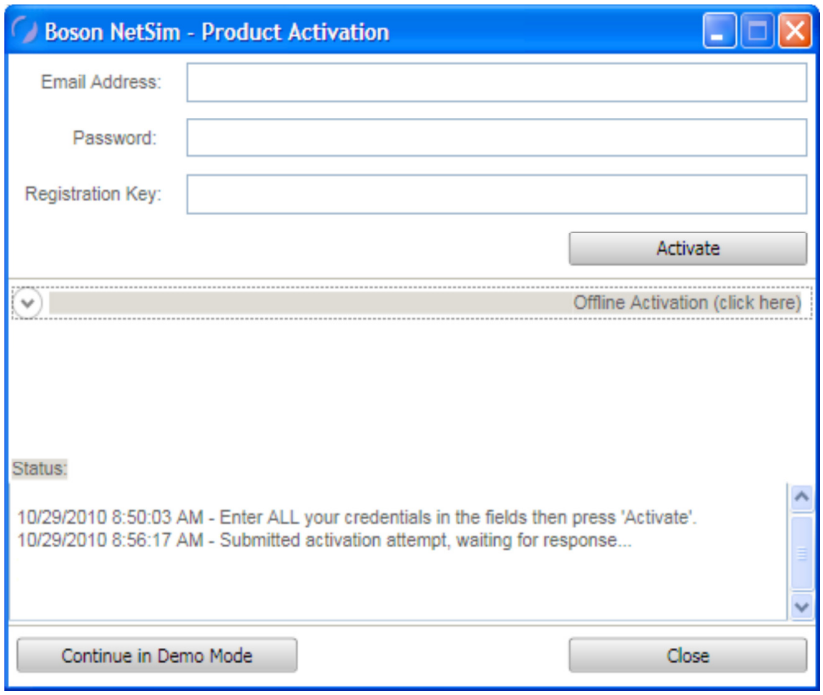
Routing on Distribution Switches			✔
SDM Simulation			✔
Site-to-Site VPN using Preshared Keys			✔
Slot-Based Devices with Ability to Change Modules	✔	✔	✔
Spanning Tree Protocol (STP) Configuration			✔
Troubleshooting Network Layer Problems	✔	✔	✔
Troubleshooting Physical and Data Link Layer Problems			✔
Troubleshooting Physical, Data Link and Network Layer Problems		✔	✔
Troubleshooting Problems at all Logical Layers		✔	✔
Troubleshooting Transport and Application Layer Problems		✔	✔
UplinkFast			✔
VLAN Access Control Lists			✔
VLANs, VTP, Trunking and Spanning Tree Protocol (STP)		✔	✔
VLSM	✔	✔	✔

Πίνακας 4: Χαρακτηριστικά του NetSim για CCENT,CCNA,CCNP

3.1.2 Ρυθμίζοντας το Boson NetSim

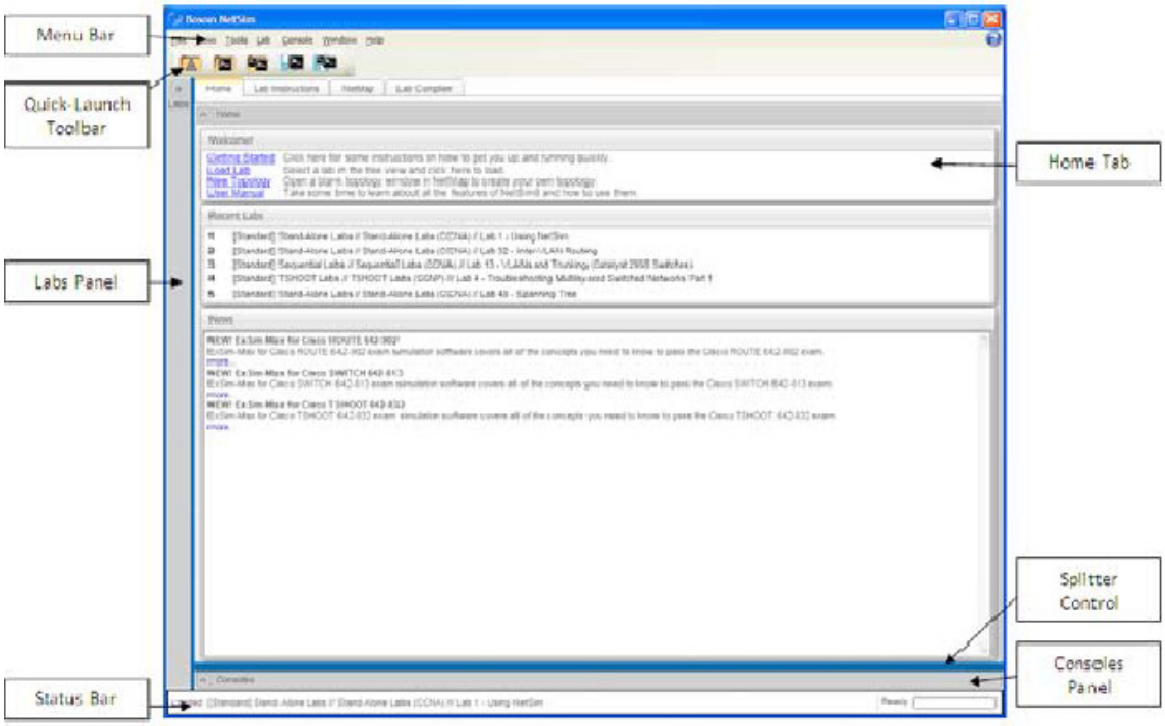
Ο χρήστης θα πρέπει να διαθέτει έναν λογαριασμό στη Boson προκειμένου να κατεβάσει τη δοκιμαστική έκδοση(demo). Για να δημιουργήσουμε ελεύθερα ένα λογαριασμό επισκεπτόμαστε τον σύνδεσμο: https://www.boson.com/Account/Default.aspx και δίνουμε ένα έγκυρο email.

Αφού ολοκληρωθεί η λήψη του εξομοιωτή μας εμφανίζει το παρακάτω παράθυρο διαλόγου που μας αναφέρει την ενεργοποίηση του προϊόντος. Λόγω του ότι δεν έχουμε αγοράζει κάποια άδεια(Licence) ώστε να είναι διαθέσιμη η ολοκληρωμένη έκδοση θα επιλέξουμε το πεδίο **Continue in Demo Mode.**



Εικόνα 7: Ενεργοποίηση του προγράμματος εξομοίωσης

Στη συνέχεια μας εμφανίζεται ο κεντρικός πίνακας του εξομοιωτή. Υπάρχουν αρκετά συστατικά μέρη σε αυτόν τον πίνακα και ποικίλες λειτουργίες και χαρακτηριστικά.



Εικόνα 8: Κεντρικό μενού του εξομοιωτή Boson

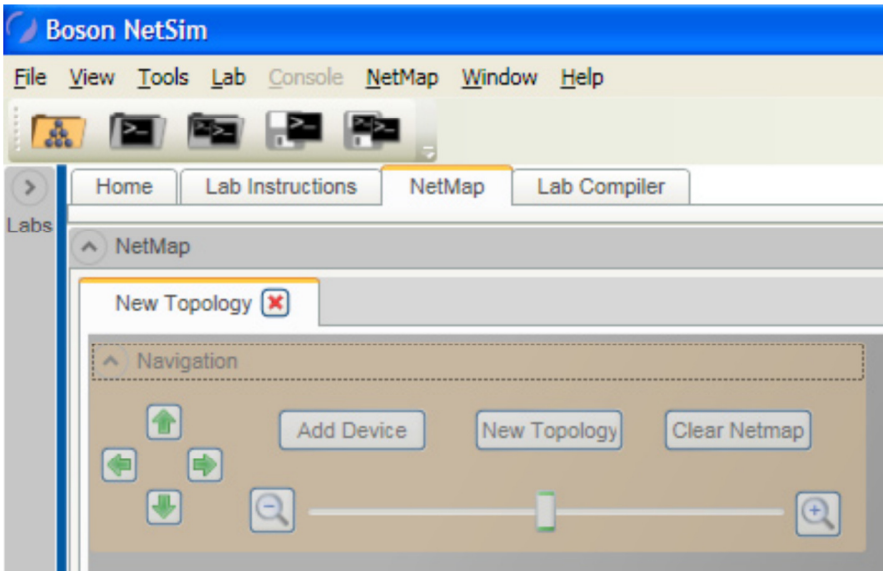
Το NetSim σε όλες τις εκδόσεις του συμπεριλαμβανομένου και της δοκιμαστικής του περιέχει έτοιμα σενάρια. Υπάρχουν πάνω από 150 έτοιμα Labs στο NetSim 8.0 και αυτά κατηγοριοποιούνται ως εξής: Stand-Alone Labs, Sequence Labs, Scenario Labs, Route Labs, Switch Labs, TSHOOT Labs και Demo Labs.

Μπορούμε να ξεκινήσουμε ένα σενάριο πηγαίνοντας στο πεδίο **Labs Panel** και στη συνέχεια μέσω του πεδίου **Standard Lab Packs** επιλέγουμε από τα διαθέσιμα που υπάρχουν. Για κάθε σενάριο υπάρχουν οδηγίες υλοποίησής του. Επομένως όταν φορτώσουμε το σενάριο το δεύτερο πράγμα που κάνουμε είναι να επιλέξουμε το πεδίο **Lab Instructions** στον κεντρικό πίνακα. Ακολούθως, στον πίνακα **Consoles panel** επιλέγουμε τη συσκευή (router,switch,pc) που χρειάζεται να ρυθμίσουμε προκειμένου να ολοκληρωθεί η υλοποίηση του σεναρίου. Τέλος, αφού ολοκληρώσουμε το σενάριο επιλέγουμε από το πεδίο **Lab** στο αρχικό μενού την τιμή **Grade Lab** για να σιγουρευτούμε ότι ολοκληρώθηκε ορθά.

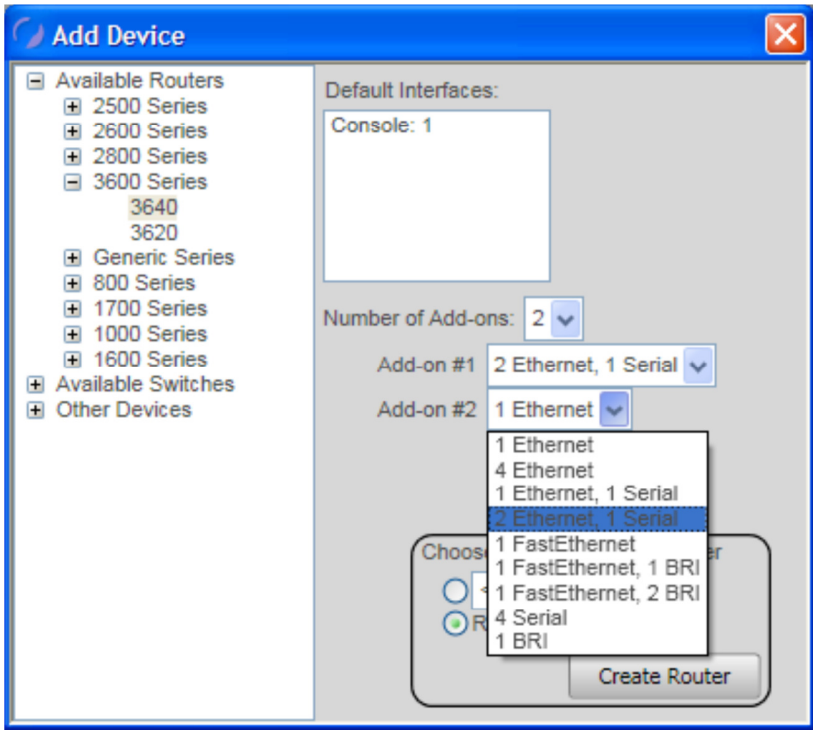
3.1.3 Δημιουργία τοπολογίας

Όπως κάθε πρόγραμμα εξομοίωσης που υπάρχει στην αγορά, έτσι και το Boson παρέχει τη δυνατότητα δημιουργίας δικών μας σεναρίων και τοπολογιών δικτύου με σκοπό την προσομοίωση αυτών και την εξαγωγή αποτελεσμάτων. Παραθέτουμε τον τρόπο με τον οποίο δημιουργείται μια τοπολογία δικτύου στον εξομοιωτή:

Στο κεντρικό μενού επιλέγουμε το πεδίο **NetMap** και στη συνέχεια το πεδίο **Navigation** το οποίο προβάλλει τις ιδιότητες που χρειαζόμαστε για να προσθέσουμε συσκευές στην τοπολογία.



Μέσω του πεδίου **Add Device** μας δίνεται η δυνατότητα να επιλέξουμε μια σειρά από διαθέσιμες συσκευές(routers,switches,PCs,IP phone).



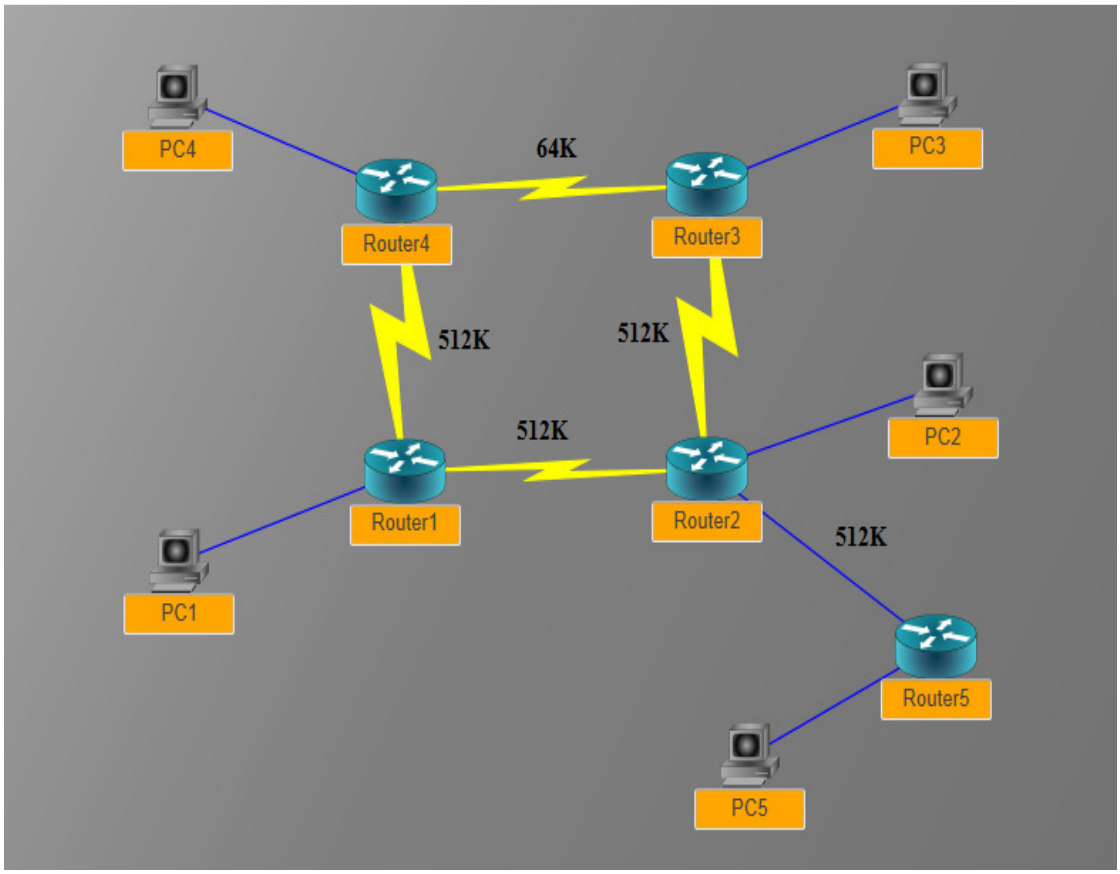
Επιλέξαμε τυχαία το μοντέλο δρομολογητή 3640. Αφού ολοκληρώσουμε το στάδιο της σύνθεσης των συσκευών στη τοπολογία μας θα πρέπει να τις συνδέσουμε μεταξύ τους. Το Boson υποστηρίζει τρεις τύπους συνδέσεων μεταξύ των δικτυακών συσκευών και υπολογιστών του δικτύου:

1. Ethernet
    - Ethernet-10Mbps
    - Fast Ethernet-100Mbps
    - Gigabit Ethernet-1000Mbps
2. Serial
    - T1-1.544Mbps
    - E1-2.048Mbps
    - DS3-44.736Mbps
3. ISDN-Basic Rate Interface(BRI)
    - ISDN B-64Kbps
    - ISDN 2B+D-144Kbps

Αφού ολοκληρώσουμε την σχεδίαση της τοπολογίας επιλέγουμε την τιμή **Apply Selected Topology** από το πεδίο NetMap στο κεντρικό μενού προκειμένου να αποκτήσουμε πρόσβαση στις ρυθμίσεις των συσκευών μέσω της κονσόλας(console mode).

3.2 Υλοποίηση Σεναρίου

Η έκδοση του εξομοιωτή που χρησιμοποιούμαι είναι η δοκιμαστική(demo). Αυτό σημαίνει ότι υπάρχουν περιορισμοί στη δυνατότητα χρήσης των εντολών ρύθμισης των συσκευών(π.χ.routers). Θα επιλέξουμε να υλοποιήσουμε ένα διαθέσιμο σενάριο που παρέχει ο εξομοιωτής από το οποίο θα προκύψουν αποτελέσματα. Η τοπολογία του δικτύου που θα χρησιμοποιήσουμε για την υλοποίηση του σεναρίου είναι η παρακάτω:



Εικόνα 9: Τοπολογία δικτύου

Το δίκτυο αυτού του σεναρίου αποτελείται από πέντε routers τα οποία επικοινωνούν μεταξύ τους με σύνδεση σημείο-προς σημείο σε ένα δίκτυο ευρείας περιοχής(WAN).

Κάθε router έχει ένα τοπικό δίκτυο συνδεδεμένο στη FastEthernet διεπαφή του. Σε κάθε τοπικό δίκτυο συνδέεται ένα υπολογιστής.

Στο παραπάνω δίκτυο έχει ήδη ρυθμιστεί το πρωτόκολλο EIGRP καθώς και οι IP διευθύνσεις στις διεπαφές τους. Σκοπός μας είναι να εντοπίσουμε και να διορθώσουμε τα λάθη στις ρυθμίσεις. Στον παρακάτω πίνακα παρουσιάζονται οι IP διευθύνεις των routers και των PC hosts.

Device	**Interface**	**IP Address**	**Subnet Mask**	**Default Gateway**
Router1	Serial0/0 Serial0/1 FastEthernet0/0	192.168.100.2 192.168.100.9 192.168.2.1	255.255.255.252 255.255.255.252 255.255.255.0	
Router2	Serial0/0 Serial0/1 FastEthernet0/0 FastEthernet0/1	192.168.100.10 192.168.100.14 192.168.3.1 192.168.100.17	255.255.255.252 255.255.255.252 255.255.255.0 255.255.255.252	
Router3	Serial0/0 Serial0/1 FastEthernet0/0	192.168.100.6 192.168.100.13 192.168.4.1	255.255.255.252 255.255.255.252 255.255.255.0	
Router4	Serial0/0 Serial0/1 FastEthernet0/0	192.168.100.1 192.168.100.5 192.168.1.1	255.255.255.252 255.255.255.252 255.255.255.0	
Router5	FastEthernet0/0 FastEthernet0/1	192.168.5.1 192.168.100.18	255.255.255.0 255.255.255.252	
PC1		192.168.2.2	255.255.255.0	192.168.2.1
PC2		192.168.3.2	255.255.255.0	192.168.3.1

PC3		192.168.4.2	255.255.255.0	192.168.4.1
PC4		192.168.1.2	255.255.255.0	192.168.1.1
PC5		192.168.5.2	255.255.255.0	192.168.5.1

Πίνακας 5:IP διευθύνσεις συσκευών

Αφού ολοκληρώσουμε τις απαραίτητες ρυθμίσεις στους routers συνδεόμαστε στον PC5 και εκτελούμε την εντολή **ping** προς τον PC1,PC2,PC3,PC4. Παρατηρούμε ότι δεν είναι όλα τα pings επιτυχημένα. Το ping στον PC1(192.168.2.2) απέτυχε. Ο PC1 ανήκει στο δίκτυο 192.168.2.0/24. Ο Router 5 λειτουργεί σαν προεπιλεγμένη πύλη του PC5. Επομένως θα πρέπει να εξετάσουμε τον πίνακα δρομολόγησης στον Router 5 προκειμένου να επιλύσουμε τα προβλήματα συνδεσιμότητας που παρατηρήσαμε. Συνδεόμαστε στον Router 5 και δίνουμε την εντολή **show ip route** και έχουμε:

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

C  192.168.5.0 is directly connected, FastEthernet0/0
  192.168.100.0/24 is variably subnetted, 6 subnets
C  192.168.100.16/30 is directly connected, FastEthernet0/1
D  192.168.100.0/24 is a summary, 00:01:38, Null0
D  192.168.100.8/30 [90/5514496] via 192.168.100.17, 00:01:28, FastEthernet0/1
D  192.168.100.4/30 [90/6026496] via 192.168.100.17, 00:01:28, FastEthernet0/1
D  192.168.100.12/30 [90/5514496] via 192.168.100.17, 00:01:28, FastEthernet0/1
D  192.168.100.0/30 [90/6026496] via 192.168.100.17, 00:01:28, FastEthernet0/1
D  192.168.3.0 [90/5005056] via 192.168.100.17, 00:01:28, FastEthernet0/1
D  192.168.1.0 [90/6029056] via 192.168.100.17, 00:01:28, FastEthernet0/1
D  192.168.4.0 [90/5517056] via 192.168.100.17, 00:01:23, FastEthernet0/1
```

Παρατηρούμε ότι το δρομολόγιο στο δίκτυο 192.168.2.0/24 δεν υπάρχει. Το EIGRP είναι πρωτόκολλο δυναμικής δρομολόγησης. Οι routers ενημερώνουν αυτόματα τους άλλους routers στο δίκτυο για τα απευθείας συνδεδεμένα δίκτυά τους. Για να διορθώσουμε το παραπάνω πρόβλημα συνδεόμαστε στον Router 1 και δίνουμε τις παρακάτω ενολές:

Router1(config)#router eigrp 100

Router1(config)#network 192.168.2.0

Η εντολή **network 192.168.2.0** ρυθμίζει το EIGRP να ανακοινώσει πληροφορίες για το Router1 LAN στους άλλους routers του δικτύου. Εκτελώντας πάλι την εντολή **show ip route** στον Router5 έχουμε:

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is not set

C   192.168.5.0 is directly connected, FastEthernet0/0
 192.168.100.0/24 is variably subnetted, 6 subnets
C   192.168.100.16/30 is directly connected, FastEthernet0/1
D   192.168.100.0/24 is a summary, 00:04:38, Null0
D   192.168.100.8/30 [90/5514496] via 192.168.100.17, 00:03:28, FastEthernet0/1
D   192.168.100.4/30 [90/6026496] via 192.168.100.17, 00:03:28, FastEthernet0/1
D   192.168.100.12/30 [90/5514496] via 192.168.100.17, 00:03:28, FastEthernet0/1
D   192.168.100.0/30 [90/6026496] via 192.168.100.17, 00:03:28, FastEthernet0/1
D   192.168.3.0 [90/5005056] via 192.168.100.17, 00:03:28, FastEthernet0/1
D   192.168.1.0 [90/6029056] via 192.168.100.17, 00:03:28, FastEthernet0/1
D   192.168.4.0 [90/5517056] via 192.168.100.17, 00:03:23, FastEthernet0/1
D   192.168.2.0 [90/5517056] via 192.168.100.17, 00:00:48, FastEthernet0/1
```

Παρατηρούμε ότι το δρομολόγιο στο δίκτυο 192.168.2.0/24 είναι ορατό στον πινάκα δρομολόγησης του Router 5.

Στη συνέχεια εξετάζουμε κάποια στοιχεία σχετικά με την απόδοση του δικτύου. Θα στείλουμε κίνηση από τον PC5 στον PC4 και θα χρησιμοποιήσουμε την εντολή **tracert** η οποία αποτυπώνει το μονοπάτι που ακολουθεί το πακέτο προκειμένου να φτάσει στον προορισμό του.

```
C:>tracert 192.168.1.2

"Type escape sequence to abort."
Tracing the route to 192.168.1.2

 1 192.168.5.1 0 msec 16 msec 0 msec
 2 192.168.100.17 20 msec 16 msec 16 msec
 3 192.168.100.13 20 msec 16 msec 16 msec
 4 192.168.100.5 20 msec 16 msec 16 msec
 5 192.168.1.2 20 msec 16 msec *
```

Το πακέτο ταξιδεύει από τον PC5 στον Router5(192.168.5.1). Στη συνέχεια κατευθύνεται στον Router2(192.168.100.17), μετά στον Router3(192.168.100.13) και

στο τέλος στον Router4(192.168.100.5) , ο οποίος μεταφέρει το πακέτο στον προορισμό του, στον PC4(192.168.1.2). Το μονοπάτι που ακολούθησε η κίνηση έχει αρνητική επίδραση στην απόδοση του δικτύου(WAN) επειδή η ζεύξη Router4-Router3 είναι υπεύθυνη να μεταφέρει τη κίνηση με ρυθμό 64Kbps σε αντίθεση με τις άλλες ζεύξεις που διαμορφώνονται στα 512Kbps.

Γενικά το πρωτόκολλο EIGRP προτιμεί ζεύξεις με υψηλότερο εύρος ζώνης(bandwidth) όταν αποφασίζει το μονοπάτι που θα ακολουθήσει. Εξετάζουμε προσεκτικά τις ρυθμίσεις σε κάθε Router για να διαπιστώσουμε αν υπάρχει η ρύθμιση για το εύρος ζώνης. Παρατηρούμε ότι στον Router2 και Router3, δεν υπάρχει η ανάλογη ρύθμιση. Επομένως προβαίνουμε στις ακόλουθες ρυθμίσεις:

Router3(config)#interface serial 0/0

Router3(config-if)#bandwidth 64

Router2(config)#interface serial 0/1

Router2(config-if)# bandwidth 512

Στην περίπτωση που δεν υπάρχει η ρύθμιση για το εύρος ζώνης το EIGRP χρησιμοποιεί το default,το οποίο είναι 1544Kbps. Σε αυτό το σημείο εκτελούμε ξανά την εντολή **tracert** για να δούμε πιο θα είναι το νέο μονοπάτι(από τον PC5 στον PC4).

```
C:>tracert 192.168.1.2

"Type escape sequence to abort."
Tracing the route to 192.168.1.2

 1 192.168.5.1 0 msec 16 msec 0 msec
 2 192.168.100.17 20 msec 16 msec 16 msec
 3 192.168.100.9 20 msec 16 msec 16 msec
 4 192.168.100.1 20 msec 16 msec 16 msec
 5 192.168.1.2 20 msec 16 msec *
```

Παρατηρούμε ότι η κίνηση του πακέτου ακολουθεί διαφορετικό μονοπάτι. Πηγαίνει από τον Router 2 στον Router1 και στο τέλος στον Router 4, αγνοώντας την μικρότερη ζεύξη

των 64Kbps. Είναι σαφές ότι αυτό το μονοπάτι βελτιώνει την απόδοση του δικτύου(WAN) επειδή χρησιμοποιεί μόνο τις 512Kbps ζεύξεις.

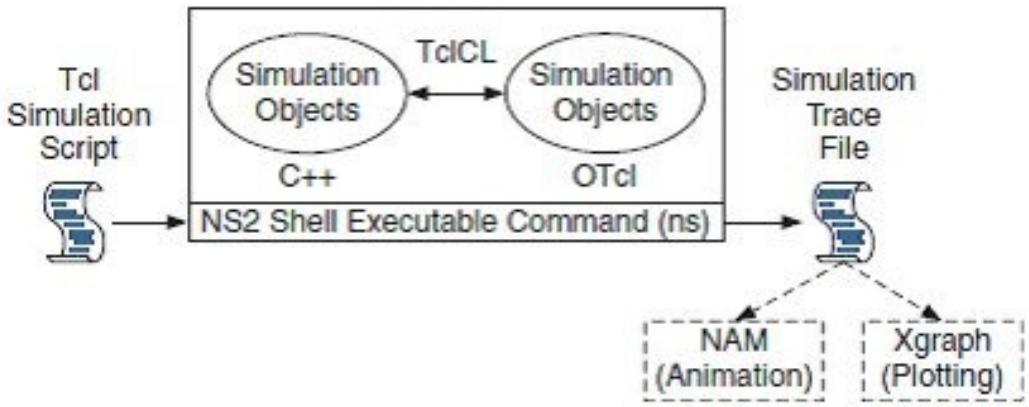
Κεφάλαιο 4

4.1 Προσομοιωτές Ns-2 και Ns-3

4.1.1 Προσομοιωτής Ns-2

Το περισσότερα διαδεδομένο εργαλείο προσομοίωσης είναι κατά πολλούς το Network Simulator 2 ή αλλιώς NS-2. Το NS-2 είναι ένα open source εργαλείο προσομοίωσης που τρέχει κυρίως σε περιβάλλον Linux. Είναι προσομοιωτής διακριτών γεγονότων που στοχεύει στην έρευνα δικτύωσης. Παρέχει ουσιαστική υποστήριξη για την προσομοίωση της δρομολόγησης, των multicast πρωτοκόλλων και πρωτοκόλλων IP, όπως τα UDP, TCP, RTP και SRM πάνω από συνδεμένα με καλώδιο αλλά και ασύρματα (τοπικά και δορυφορικά)δίκτυα. Το NS-2 έχει πολλά πλεονεκτήματα που το κατατάσσουν ως ένα χρήσιμο εργαλείο για την υποστήριξη πολλαπλών πρωτοκόλλων και τη δυνατότητα της γραφικής λεπτομερειακής απεικόνισης της δικτυακής κίνησης. Επιπλέον υποστηρίζει αρκετούς αλγόριθμους που αφορούν τη δρομολόγηση και τις ουρές αναμονής.

Το NS άρχισε ως παραλλαγή του REAL network simulator το 1989 και έχει εξελιχθεί σημαντικά τα τελευταία χρόνια. Το 1995 η ανάπτυξη του NS υποστηρίχθηκε από την DAPRA μέσω του VINT project στις LBL, Xerox PARC, UCB, και USC/ISI. Αυτή την περίοδο η ανάπτυξη του NS υποστηρίζεται μέσω της DAPRA με την SAMAN και μέσω της NSF με την CONSER, και οι δύο σε συνεργασία με άλλους ερευνητές συμπεριλαμβανομένου της ACIRI. Το NS λαμβάνει συνεχώς σημαντική βοήθεια και από άλλους ερευνητές, συμπεριλαμβανομένου του ασύρματου κώδικα από τα UCB Daedelus και CMU Monarch projects και από την Sun Microsystems.



Εικόνα 10: Αρχιτεκτονική του ns2

4.1.2 Εγκαθιστώντας το Ns-2

Το NS-2 είναι διαθέσιμο σε αρκετές πλατφόρμες όπως είναι οι : FreeBSD, Linux, SunOS και Solaris. Επίσης μπορεί να εγκατασταθεί και να τρέξει σε περιβάλλον Windows. Τα απλά σενάρια θα πρέπει λογικά να τρέχουν σε οποιοδήποτε απλό μηχάνημα, παρόλα αυτά όμως τα πολύ μεγάλα σενάρια απαιτούν και επωφελούνται από την ύπαρξη μεγάλης κεντρικής μνήμης από το υπολογιστικό σύστημα. Επιπλέον το NS-2 χρειάζεται τα ακόλουθα πακέτα να είναι εγκατεστημένα ώστε να μπορεί να τρέξει : Tcl release 8.3.2, Tk release 8.3.2, OTcl release 1.0a7 και TclCL release 1.0b11.

Το NS-2 είναι αρκετά μεγάλο σαν πρόγραμμα και απαιτεί περίπου 250MB χώρο στο δίσκο ώστε να εγκατασταθεί. Η εγκατάσταση του NS-2 σε ξεχωριστά κομμάτια μπορεί να εξοικονομήσει κάποιο χώρο στο δίσκο. Υπάρχουν δύο τρόποι για την εγκατάσταση του NS-2 : from all the pieces ή all at once. Αν κάποιος θέλει να το δοκιμάσει και να το εγκαταστήσει γρήγορα θα πρέπει να επιλέξει τη δεύτερη περίπτωση, ενώ αν κάποιος θέλει αναπτύξει το πρόγραμμα στο επίπεδο της C ή να γλιτώσει χρόνο στο κατέβασμα του προγράμματος και χώρο στο δίσκο θα πρέπει να δοκιμάσει την πρώτη περίπτωση.

Το NS-2 δεν είναι ένα ολοκληρωμένο προϊόν, αλλά το αποτέλεσμα μιας συνεχούς προσπάθειας έρευνας και ανάπτυξης. Ειδικότερα, λάθη στο λογισμικό του προγράμματος ακόμα ανακαλύπτονται και διορθώνονται. Για να βοηθηθεί ο χρήστης σε αυτό το πρόβλημα καταβάλλεται μεγάλη προσπάθεια με την επέκταση και την αυτοματοποίηση των προγραμμάτων ελέγχου της αξιοπιστίας των παραγόμενων αποτελεσμάτων.

Οι χρήστες είναι αρμόδιοι στο να επαληθεύσουν ότι οι προσομοιώσεις τους δεν ακυρώνονται επειδή το πρότυπο που εφαρμόζεται στον προσομοιωτή δεν είναι το πρότυπο που ανέμεναν. Το τρέχον εγχειρίδιο του NS-2 θα πρέπει να τους βοηθήσει σε αυτήν την διαδικασία.

4.1.3 Περιγραφή και δυνατότητες του Ns-2

Ο NS-2 είναι ένας αντικειμενοστραφής προσομοιωτής (Object Oriented Simulator), ο οποίος χρησιμοποιεί στο προσκήνιο την γλώσσα προγραμματισμού OTcl (Object Tool Command Language) και στο παρασκήνιο την C++. Ο προσομοιωτής διαθέτει μια ιεραρχία τάξεων σε C++ (που αναφέρεται ως compiled hierarchy) και μια παρόμοια ιεραρχία στην OTcl (που αναφέρεται ως interpreted hierarchy, μιας και η Tcl διαθέτει interpreter και όχι compiler).

Από την πλευρά του χρήστη υπάρχει μια αντιστοιχία 1-1 ανάμεσα στις τάξεις που ανήκουν στην compiled hierarchy (C++) και σε εκείνες που ανήκουν στην interpreted hierarchy(OTcl) . Η ρίζα σε αυτή την ιεραρχία είναι η τάξη Tcl Object. Οι χρήστες μπορούν να δημιουργούν νέα αντικείμενα προσομοίωσης μέσα από τον OTcl interpreter. Η αρχικοποίηση αυτών των αντικειμένων γίνεται από τον OTcl interpreter, κατά την οποία αντιστοιχίζονται και με το ανάλογο αντικείμενο που υπάρχει στην class hierarchy (C++).

Η φιλοσοφία που κρύβεται πίσω από τη χρήση δύο γλωσσών προγραμματισμού και δύο παρόμοιων ιεραρχιών τάξεων, είναι ότι κατά τη διάρκεια μιας προσομοίωσης υπάρχουν δύο διαφορετικά πράγματα που πρέπει να εκτελεστούν από τον προσομοιωτή. Από τη μια πλευρά, η λεπτομερής προσομοίωση πρωτοκόλλων και δικτύων απαιτεί μια γλώσσα προγραμματισμού, η οποία να μπορεί να διαχειρίζεται με αποδοτικό τρόπο bytes, επικεφαλίδες πακέτων (packet headers), καθώς και να τρέχει αλγόριθμους, οι οποίοι να διαχειρίζονται μεγάλα σύνολα δεδομένων. Για αυτές τις ενέργειες, ο χρόνος εκτέλεσης (run-time) αποτελεί το σημαντικότερο στοιχείο, ενώ ο χρόνος διόρθωσης & επαναπροσδιορισμού της προσομοίωσης (turn-around time), ο οποίος περιλαμβάνει τη διαδικασία εύρεσης και διόρθωσης λαθών στον κώδικα και επανάληψη της μεταγλώττισης και της εκτέλεσης, είναι λιγότερο σημαντικός.

Από την άλλη πλευρά, ένα μεγάλο κομμάτι της έρευνας σε δίκτυα υπολογιστών

περιλαμβάνει τις διαρκείς δοκιμές και εκτελέσεις της προσομοίωσης μεταβάλλοντας ίσως κατά πολύ λίγο ορισμένες παραμέτρους (όπως π.χ. την ταχύτητα των ζεύξεων, τον ρυθμό λαθών στο κανάλι, τον αριθμό των κόμβων, κλπ), έτσι ώστε να εξερευνηθεί ένας όσο το δυνατόν μεγαλύτερος αριθμός «σεναρίων» σχετικά με την εξεταζόμενη δικτυακή τοπολογία. Για τις ενέργειες αυτές, ο χρόνος κατά τον οποίο μεταβάλλουμε τις παραμέτρους ρύθμισης της προσομοίωσης είναι πολύ σημαντικός και η interpreted γλώσσα προγραμματισμού OTcl αποτελεί μια πολύ καλή λύση, μιας και η ρύθμιση μιας δικτυακής τοπολογίας γίνεται στην αρχή και μετά παραμένει η ίδια σε όλη τη διάρκεια της προσομοίωσης.

Αν και θα μπορούσαμε να υλοποιήσουμε τα πάντα σε όποια γλώσσα προγραμματισμού θέλουμε από τις δύο (αν και σε C++ είναι λίγο δυσκολότερο μιας και στο προσκήνιο χρησιμοποιείται ο OTcl interpreter), καλό είναι την OTcl να τη χρησιμοποιούμε κυρίως για να ρυθμίζουμε τη δικτυακή τοπολογία μας και τις διάφορες παραμέτρους της, ενώ τη C++ για να υλοποιούμε πολύπλοκους και απαιτητικούς αλγόριθμους δρομολόγησης, ελέγχου λαθών, πρωτοκόλλων εφαρμογών, δικτύου, κλπ. Ωστόσο, αξίζει να σημειωθεί ότι ο NS-2 παρέχει τη δυνατότητα να «σπάμε» την υλοποίηση ενός αντικειμένου ανάμεσα στις 2 γλώσσες, με μερικές από τις ιδιότητες και μεθόδους να βρίσκονται στο OTcl αντικείμενο (ίσως αυτές της ρύθμισης της προσομοίωσης) και τις περισσότερο πολύπλοκες στο αντίστοιχο C++ αντικείμενο που ανήκει στην compiled hierarchy. Στη συνέχεια, μπορούμε (μέσα από κατάλληλους μηχανισμούς που παρέχει ο NS-2), να καλούμε μια μέθοδο του αντικειμένου που είναι γραμμένη σε C++, μέσα από το OTcl αντικείμενο, ελέγχοντας και εκτελώντας την προσομοίωσή μας κατευθείαν από την OTcl , η οποία άλλωστε χρησιμοποιείται και στο προσκήνιο από τον NS-2. Ο NS-2 είναι ένας Object-Oriented Simulator και κατά συνέπεια τα πάντα γίνονται με τη χρήση αντικειμένων. Ο NS-2 διαθέτει ένα ευρύ σύνολο αντικειμένων για τη δημιουργία και τη διαχείριση των προσομοιώσεών μας. Το κάθε αντικείμενο διαθέτει ορισμένες ιδιότητες (χαρακτηριστικά, τα οποία αντιπροσωπεύονται ως μεταβλητές που είναι ενσωματωμένες στο αντικείμενο) και μεθόδους (δυνατότητες εκτέλεσης ορισμένων εργασιών από το αντικείμενο, οι οποίες αντιπροσωπεύονται ως διαδικασίες που είναι ενσωματωμένες κι αυτές στο αντικείμενο).

Κάθε προσομοίωση που δημιουργούμε στον NS-2, εξαρτάται από ένα κεντρικό

αντικείμενο, το αντικείμενο ελέγχου της προσομοίωσης Simulator. Με τις ιδιότητες και τις μεθόδους του αντικειμένου αυτού μπορούμε να ελέγχουμε την προσομοίωσή μας, ρυθμίζοντας την τοπολογία της, τα πρωτόκολλα που θα «τρέχουν» οι κόμβοι του δικτύου, τις συνδέσεις ανάμεσα στους κόμβους, το πρωτόκολλο δρομολόγησης που θα χρησιμοποιεί η προσομοίωση, τη δυναμικότητα του δικτύου (απενεργοποίηση & ενεργοποίηση συνδέσεων και κόμβων), κλπ. Το αντικείμενο Simulator δημιουργείται στην αρχή και το χρησιμοποιούμε σε όλη τη διάρκεια της προσομοίωσης.

Αφού δημιουργήσουμε το αντικείμενο ελέγχου της προσομοίωσης, σειρά έχει η δημιουργία της δικτυακής μας τοπολογίας, η οποία ξεκινά με τη δημιουργία των κόμβων. Ένας κόμβος μπορεί να αντιπροσωπεύει τον υπολογιστή ενός χρήστη, έναν δρομολογητή, ή ακόμη και ένα ολόκληρο εργαστήριο υπολογιστών, ανάλογα ίσως και με το ρυθμό με τον οποίο μεταδίδει δεδομένα. Επίσης, μπορεί να αντιπροσωπεύει και έναν ασύρματο κόμβο (κινητό ή ακίνητο),ή έναν δορυφόρο, κλπ, ανάλογα πάντα με το τι θέλουμε να προσομοιώσουμε. Αφού δημιουργήσουμε τους κόμβους και τις συνδέσεις τους, θα πρέπει να δημιουργήσουμε και κάποια κυκλοφορία δεδομένων ανάμεσά τους. Αυτό απαιτεί δύο πράγματα. Καταρχήν θα πρέπει να εξοπλίσουμε τους κόμβους μας με τα κατάλληλα πρωτόκολλα μετάδοσης δεδομένων, όπως είναι το TCP, το UDP, το SCTP, κλπ. Ωστόσο, επειδή αυτά τα πρωτόκολλα δεν είναι ικανά να αποστέλλουν και να λαμβάνουν δεδομένα από μόνα τους, χρειάζεται να προσδιορίσουμε και κάποιες εφαρμογές που να τα χρησιμοποιούν (π.χ. μια εφαρμογή FTP).Όπως προαναφέραμε, θα πρέπει να δημιουργήσουμε και τις κατάλληλες εφαρμογές που θα χρησιμοποιούν τα πρωτόκολλα αυτά για την αποστολή και λήψη των δεδομένων τους.

Υπάρχουν διαφορετικές εφαρμογές που χρησιμοποιούν το πρωτόκολλο TCP και άλλες που χρησιμοποιούν το πρωτόκολλο UDP. Ένα παράδειγμα εφαρμογής για το TCP πρωτόκολλο είναι η μεταφορά αρχείων FTP Application, ενώ παραδείγματα εφαρμογών για το UDP πρωτόκολλο είναι : μία εφαρμογή που να αποστέλλει δεδομένα με σταθερό ρυθμό (Constant Bit Rate – CBR Application) και μία που αποστέλλει δεδομένα σε τυχαίες χρονικές στιγμές(exponential application).

Με τον NS-2 έχουμε τη δυνατότητα να εισάγουμε μοντέλα δυναμικής διαχείρισης της δικτυακής τοπολογίας μας. Τα μοντέλα αυτά έχουν να κάνουν με την ενεργοποίηση και

την απενεργοποίηση των ζεύξεων μεταξύ των κόμβων του δικτύου, έτσι ώστε να προσομοιωθούν περιπτώσεις όπου μια ζεύξη ή ένας κόμβος είναι ελαττωματικός και προκαλεί κατά συνέπεια απώλεια δεδομένων.

Επιπλέον με τον NS-2 έχουμε τη δυνατότητα να παρακολουθούμε μία ή περισσότερες ουρές της δικτυακής μας τοπολογίας. Αυτό γίνεται με τη χρήση των αντικειμένων τύπου Monitor,τα οποία μπορούν να συσχετίζονται με τις ουρές της δικτυακής μας τοπολογίας και να καταγράφουν την κίνηση σε αυτές μέσα στις ιδιότητές τους. Στο τέλος της προσομοίωσης (ή όποτε άλλοτε θέλουμε) μπορούμε απλά να ανακτήσουμε τις τιμές αυτών των ιδιοτήτων και είτε να τις αποθηκεύσουμε σε αρχεία, ή να τις εμφανίσουμε στην οθόνη.

Ο προσομοιωτής NS-2 συνοδεύεται από ένα βοηθητικό εργαλείο οπτικής απεικόνισης της εκτέλεσης της προσομοίωσής μας, τον Network Animator (NAM). Όταν απεικονίζουμε την προσομοίωσή του NS-2 σε NAM μπορούμε να χρωματίσουμε με διαφορετικό χρώμα τις ροές των Agents πρωτοκόλλου, έτσι ώστε να μπορούμε να τις ξεχωρίσουμε και να παρακολουθούμε με μεγαλύτερη ευκρίνεια τη ροή των αποτελεσμάτων.

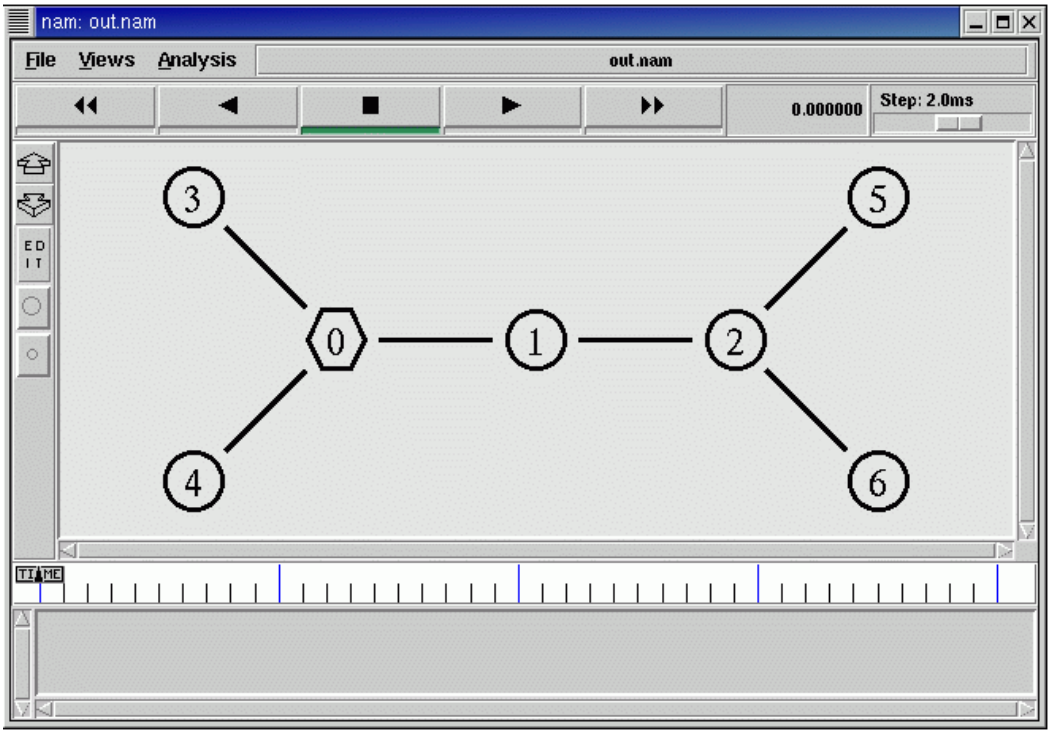
4.1.4 Network animator(NAM)

Το NAM είναι ένα πρόγραμμα απεικόνισης βασισμένο στην γλώσσα προγραμματισμού Tcl/Tk για την παρακολούθηση αρχείων δικτυακών προσομοιώσεων που έχουν δημιουργηθεί από εργαλεία προσομοίωσης όπως είναι το NS-2. Υποστηρίζει την σχεδίαση τοπολογιών, την απεικόνιση του επιπέδου των πακέτων και πολλά εργαλεία παρακολούθησης των δεδομένων. Το NAM ξεκίνησε στην LBL και εξελίχθηκε σημαντικά τα τελευταία χρόνια. Έτσι λοιπόν αν κανείς προτιμά ένα γραφικό περιβάλλον στο οποίο θα μπορεί να υλοποιεί τις προσομοιώσεις που δημιουργεί μέσω του NS-2 το NAM υποστηρίζει ένα drag and drop περιβάλλον χρήστη.

Σε αυτό μπορούν να τοποθετηθούν οι δικτυακοί κόμβοι, να ενωθούν μεταξύ τους και στη συνέχεια να οριστούν οι παράγοντες που επιθυμεί ο χρήστης καθώς και η αντίστοιχη εφαρμογή ή η γεννήτρια κίνησης.

4.1.5 Παράδειγμα λειτουργίας Ns-2

Για να μας γίνει ποιο κατανοητό πως λειτουργεί το NS-2 θα παρουσιαστεί ένα παράδειγμα με την παρακάτω τοπολογία προσομοίωσης.



Εικόνα 11: Τοπολογία προσομοίωσης

Στο παράδειγμά μας υπάρχουν 2 παράλληλες TCP συνεδρίες που μοιράζονται έναν ενιαίο σύνδεσμο συμφόρησης (bottleneck) ανάμεσα στον κόμβο 1 και στον κόμβο 2(χωρητικότητας 700kb).

```
# Δημιουργία της τοπολογίας
$nsduplex-link $s(0) $n(0) 1Mb 5msDropTail
$nsduplex-link $s(1) $n(0) 1Mb 5msDropTail

$nsduplex-link $n(0) $n(1) 1Mb 20msRED/myRIO
$nsduplex-link $n(1) $n(2) 700Kb 25msRED/myRIO

$nsduplex-link $n(2) $r(0) 1Mb 5msDropTail
$nsduplex-link $n(2) $r(1) 1Mb 5msDropTail
```

Μία από τις συνεδρίες μεταδίδει πακέτα από τον κόμβο 3 στον κόμβο 5, στον οποίο

στέλνει 1000 πακέτα. Ως εκ τούτου, όλες οι ροές σε αυτή τη συνεδρία είναι μεγάλης απόστασης σύμφωνα με αυτά που έχουμε ορίσει. Αντιθέτως, στην συνεδρία από τον κόμβο 4 στον κόμβο 6, μόνο 4 πακέτα μεταδίδονται σε κάθε ροή. Το σενάριο προσομοίωσης παρουσιάζεται παρακάτω:

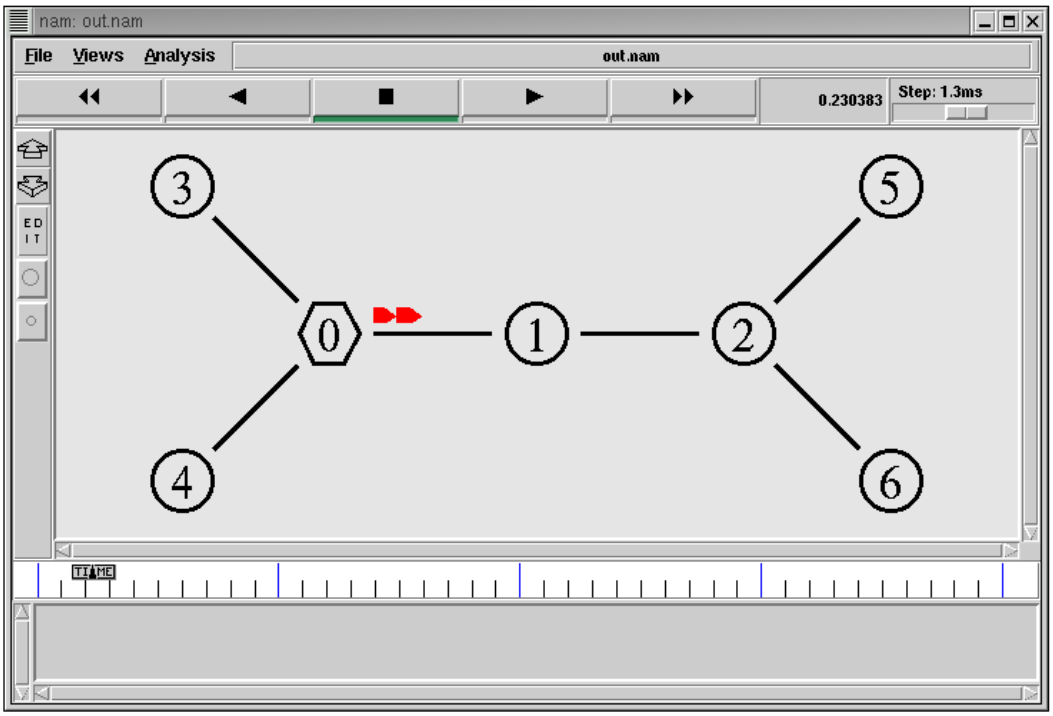
```
# Δημιουργίασυνεδριών
proc build-fore-tcp { idx size intvstime } {
        global ns ftcpfsink

        set ftcp($idx) [new Agent/TCP/Newreno]
        set fsink($idx) [new Agent/TCPSink]

        $ns at $stime "start-conn 1 $idx $intv $size"
}
proc start-conn { firsttimeidxintv size } {
        global ns ftcpfsink s r

        set now [$ns now]

        if { $firsttime == 0 } {
                $ns detach-agent $s([expr $idx%2]) $ftcp($idx)
                $ns detach-agent $r([expr $idx%2]) $fsink($idx)
                $ftcp($idx) reset
                $fsink($idx) reset
        }
        $ns attach-agent $s([expr $idx%2]) $ftcp($idx)
        $ns attach-agent $r([expr $idx%2]) $fsink($idx)
        $ns connect $ftcp($idx) $fsink($idx)
        $ftcp($idx) set fid_ 0

        $ftcp($idx) proc done {} "close-conn $idx $intv $size"
        $ftcp($idx) advanceby $size
}
proc close-conn { idxintv size } {
        global ns

        set now [$ns now]
        $ns at [expr $now + $intv] "start-conn 0 $idx $intv $size"
        puts "at $now + $intv start next"
}
set forel_intv 1
set fores_intv 0.05
set ssize 4
set lsize 1000
build-fore-tcp 1 $ssize 1 0.1
build-fore-tcp 0 $lsize $forel_intv 0.5

for {set i 0} {$i< 5} { incri} {

        build-fore-tcp [expr 2*$i+3] $ssize $fores_intv [expr
1.2+$i*0.1]
}
```

Ο κόμβος 0 είναι ο "size-aware classifier", το οποίο καταμετράει τα εισερχόμενα πακέτα από την κάθε ροή των 2 συνδέσμων. Μόλις ο αριθμός υπερβαίνει έναν ορισμένο threshold (στην περίπτωση μας είναι 5), τα πακέτα από την αντίστοιχη ροή προσδιορίζονται ως πακέτα μεγάλης ροής. Στην προσομοίωση μας, τα πακέτα μεγάλης ροής προσδιορίζονται με μπλε χρώμα. Όλα τα άλλα πακέτα, δηλαδή τα πακέτα από ροές με μέγεθος μικρότερο από 5, και τα πρώτα 5 πακέτα από μια μεγάλη ροή, προσδιορίζονται όλα ως σύντομα πακέτα ροής και προσδιορίζονται με κόκκινο χρώμα στην προσομοίωση.

Για να θέσουμε ένα "size-aware classifier" με threshold 5 στον κόμβο 0 πράττουμε ως εξής:

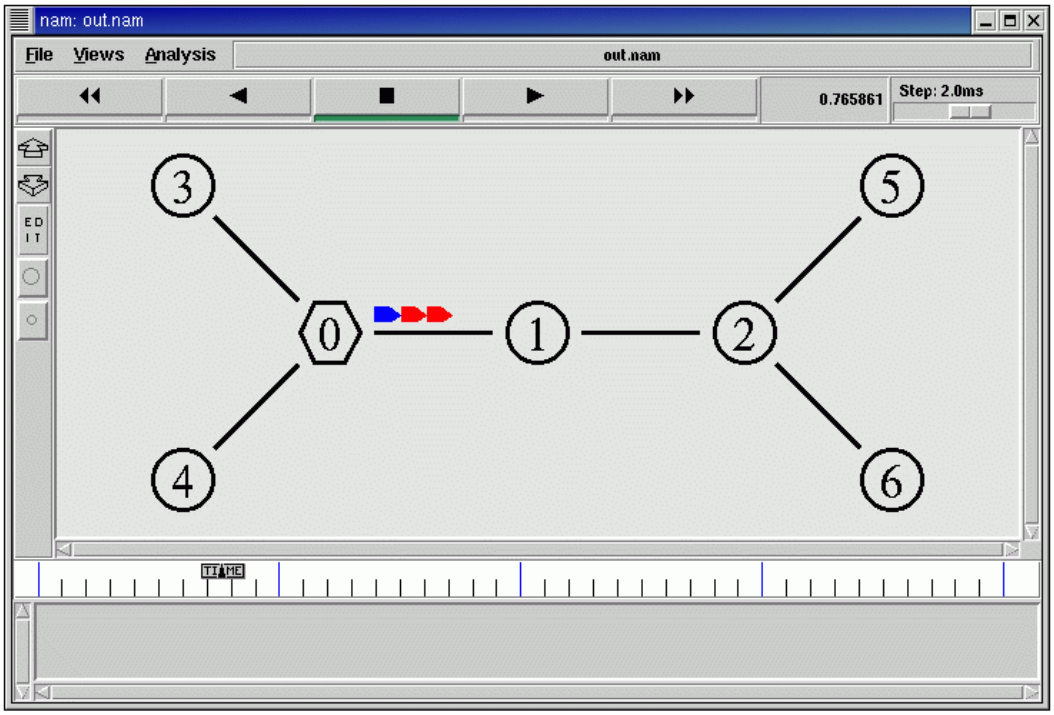
```
# Φόρτωσητης size-aware classifier στονκόμβο 0
set cls [new Classifier/Hash/SizeAware 128]
$cls set default_ -1
$cls set flowlen_thr_ 5
$cls set refresh_intv_ 2
$cls set dynamic_update_ 0
set n(0) [node_with_classifier $cls]
```

Ως αποτέλεσμα, όλα τα πακέτα μεταξύ του κόμβου 4 και του κόμβου 6 προσδιορίζονται με κόκκινο:



Εικόνα 12: Πακέτα σύντομης ροής

Για ροές με μέγεθος μεγαλύτερο από 5, τα πρώτα 5 πακέτα προσδιορίζονται με κόκκινο χρώμα, και τα υπόλοιπα πακέτα προσδιορίζονται με μπλε:

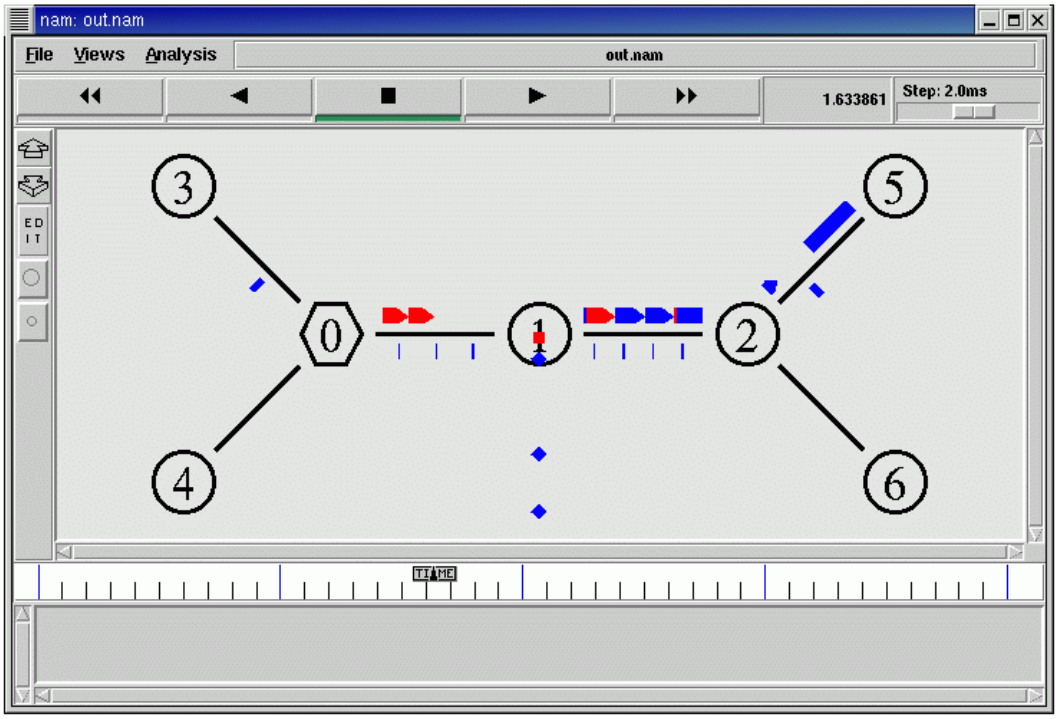


Εικόνα 13:Πακέτα μεγάλης ροής

Στον σύνδεσμό συμφόρησης (bottleneck) μεταξύ του κόμβου 1 και του κόμβου 2, χρησιμοποιείται ένα διαφοροποιημένο σενάριο απόρριψης πακέτων. Αυτό υλοποιείται με τον εξής κώδικα :

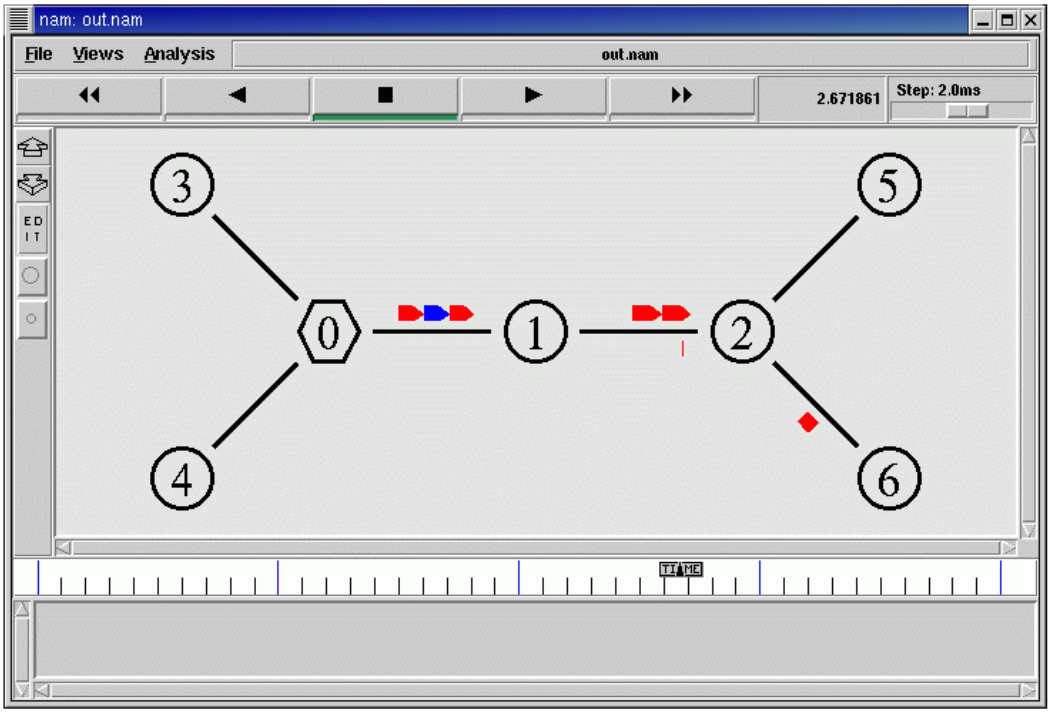
```
$ns duplex-link $n(1) $n(2) 700Kb 25ms RED/myRIO
$ns duplex-link-op $n(1) $n(2) orient right

Queue/RED/myRIO set gentle_ true
Queue/RED/myRIO set thresh_ 1
Queue/RED/myRIO set maxthresh_ 15
Queue/RED/myRIO set weight_ 10
Queue/RED/myRIO set setbit_ false

set redq [[$ns link $n(1) $n(2)] queue]
$redq set q_weight_ [expr 1.0/2]
$redq set linterm_ [expr 4.0]
$ns queue-limit $n(1) $n(0) 30
```

Όταν συμβαίνει συμφόρηση, τα πακέτα χαμηλής προτεραιότητας μειώνονται με ταχύτερο ρυθμό (κατά μέσο όρο 10 φορές γρηγορότερα) από ότι τα πακέτα υψηλής προτεραιότητας. Αυτό μπορούμε να το διαπιστώσουμε στην Εικόνα 14:



Εικόνα 14: Τα μπλε πακέτα απορρίπτονται σε μεγαλύτερο ποσοστό από τα κόκκινα

Σαν συμπέρασμα μπορούμε να δηλώσουμε, ότι οι ροές υψηλής προτεραιότητας, ή σύντομες ροές, μεταδίδονται με ρυθμό υψηλότερο από της ροές χαμηλής προτεραιότητας, ή μεγάλες ροές, έτσι οι ροές TCP ανταποκρίνονται με απόρριψη πακέτων. Αυτή είναι η βασική λειτουργία του size-aware. Στην Εικόνα 6 φαίνονται τα πακέτα που παράγονται από τις δύο συνεδρίες, αργότερα παρατηρούμε ότι ορισμένα πακέτα απορρίπτονται:



Εικόνα 15**:** Τα κόκκινα πακέτα παράγονται γρηγορότερα από τα μπλε

4.2 Προσομοιωτής Ns-3

Ο ns-3 είναι ένας δικτυακός προσομοιωτής διακριτών συμβάντων, που ως πρωταρχικό του στόχο έχει την έρευνα και την εκπαιδευτική χρήση. Είναι δωρεάν λογισμικό κάτω από την άδεια GNU GPL v2 license και είναι ανοικτός προς το κοινό για έρευνα, ανάπτυξη και χρήση του.

Ο στόχος του ns-3 project είναι να αναπτύξει ένα ιδανικό ανοικτό περιβάλλον προσομοίωσης για έρευνα πάνω στα δίκτυα που θα προσφέρει όλα εκείνα τα οποία χρειάζεται για να γίνει μια σύγχρονή προσομοίωση πάνω στα δίκτυα και θα ενθαρρύνει την κοινότητα να συνεισφέρει, να εκφράσει μια σφαιρική άποψη πάνω στο λογισμικό και να το εγκρίνει.

Το ns-3 project είναι δεσμευμένο να αναπτύσσει ισχυρό πυρήνα προσομοίωσης ο οποίος ταυτόχρονα θα είναι και πολύ καλά τεκμηριωμένος, εύκολος στη χρήση και στην εκσφαλμάτωση και θα ικανοποιεί τις απαιτήσεις ολόκληρης της προσομοίωσης, από τις ρυθμίσεις ως την συλλογή αποτελεσμάτων για ανάλυση.

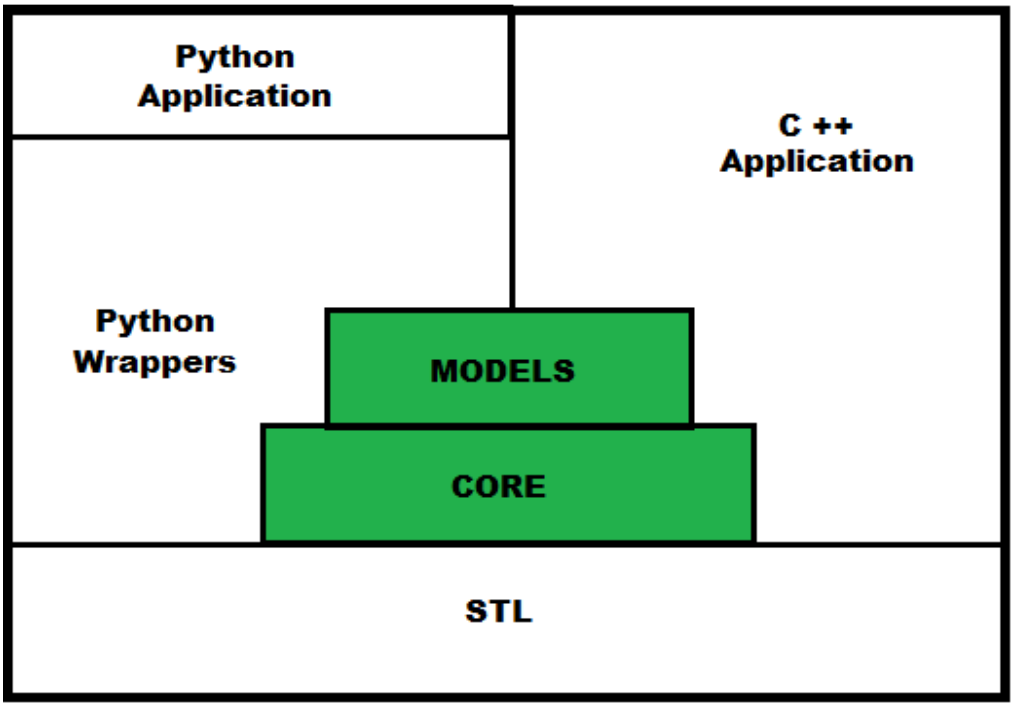
Επίσης η υποδομή του λογισμικού ενθαρρύνει την ανάπτυξη μοντέλων προσομοίωσης

τα οποία θα είναι τόσο ρεαλιστικά ώστε να επιτρέπουν στον ns-3 να χρησιμοποιηθεί σαν εξομοιωτή πραγματικού χρόνου, συνδεδεμένος με τον πραγματικό κόσμο, το οποίο επιτρέπει πολλές από τις πραγματικές υλοποιήσεις πρωτοκόλλων που υπάρχουν στην πράξη να ξαναχρησιμοποιηθούν μέσα από τον ns-3.

Ο πυρήνας προσομοίωσης του ns-3 υποστηρίζει έρευνα πάνω και σε IP και σε non- IP δίκτυα, Ωστόσο η πλειοψηφία των χρηστών επικεντρώνεται στις wireless/IP προσομοιώσεις όπου περιλαμβάνουν μοντέλα για Wi-Fi, WiMAX και διάφορα στατικά η δυναμικά πρωτόκολλα δρομολόγησης όπως το OLSR και το AODV για IP εφαρμογές.

Επίσης ο ns-3 υποστηρίζει έναν προγραμματιστή πραγματικού χρόνου το οποίο διευκολύνει έναν αριθμό από "simulation-in-theloop" περιπτώσεων που χρησιμοποιούνται για επικοινωνία με πραγματικά συστήματα. Για παράδειγμα, οι χρήστες μπορούν να στέλνουν και να παίρνουν πακέτα που έχουν βγει από τον ns-3 σε πραγματικές δικτυακές συσκευές, με τον ns-3 χρησιμοποιείται σαν μια διεπαφή για να προσθέσει λειτουργίες σε σύνδεση μεταξύ εικονικών μηχανών.

Τέλος δίνει ιδιαίτερη έμφαση στην επαναχρησιμοποίηση πραγματικών εφαρμογών και κώδικα kernel.



Εικόνα 16: Αρχιτεκτονική του ns-3

4.2.1 Διαδικασία εγκατάστασης του NS-3

Αρχικά, γίνεται λήψη από την κεντρική σελίδα του ns-3 της τελευταίας σταθερής έκδοσης. Υπάρχει ένα πακέτο, που ονομάζεται ns-3-allinone, για τη διευκόλυνση της εγκατάστασης. Τις περισσότερες φορές γίνεται λήψη του development tree ns-3-dev. Με αυτό τον τρόπο απαιτείται η χρήση του Mercurial που είναι ένα δωρεάν, κατανεμημένου ελέγχου εργαλείο συντονισμένης κατασκευής με την εντολή build.py. Έτσι ολοκληρώνεται η διαδικασία εγκατάστασης κι ο χρήστης μπορεί να τρέξει τα παραδείγματα που υπάρχουν ή να δημιουργήσει νέα προσθέτοντας τα σωστά αρχεία στον φάκελο scratch.

4.2.2 Προτερήματα Ns-3 σε σύγκριση με άλλους εξομοιωτές

1. **Έμφαση στη C++ και Python:** Οι περισσότεροι προσομοιωτές χρησιμοποιούν μία ειδική γλώσσα μοντελοποίησης για τη περιγραφή των μοντέλων και των ροών του προγράμματος. Με την χρήση των C++ και Python, οι χρήστες επωφελούνται από όλα όσα υποστηρίζει η κάθε γλώσσα.
2. **Γεγονότα και συνδέσεις με γνώμονα την επανάκληση:** Τα γεγονότα

προσομοίωσης είναι απλά κλήσεις συναρτήσεων – λειτουργιών που είναι προγραμματισμένες να εκτελούνται σε ένα προκαθορισμένο χρονικό διάστημα της προσομοίωσης. Κάθε συνάρτηση μπορεί να φτιαχτεί σε ένα γεγονός και να προγραμματιστεί με τη χρήση της συνάρτησης επανάκτησης. Αυτό έρχεται σε αντίθεση με συναρτήσεις που συγκεντρώνουν την επεξεργασία γεγονότων σε κάθε αντικείμενο προσομοίωσης. Οι επανακλήσεις χρησιμοποιούνται σε μεγάλο βαθμό για να μειώσουν τις εξαρτήσεις του χρόνου μεταγλώττισης μεταξύ των αντικειμένων της προσομοίωσης.

3. **Ευέλικτος πυρήνας με βοηθητικά επίπεδα:** Ο ns-3 διαθέτει ένα ισχυρό, χαμηλού επιπέδου API που επιτρέπει στους χρήστες να ρυθμίσουν τα πράγματα με διαφορετικούς τρόπους. Στην κορυφή των επιπέδων είναι ένα σύνολο "βοηθός" APIs που παρέχει ευκολότερες στη χρήση συναρτήσεις.

4. **Έμφαση στην εξομοίωση:** Ο σχεδιασμός προσομοίωσης είναι προσανατολισμένος σε περιπτώσεις χρήσης που επιτρέπουν στον προσομοιωτή να αλληλεπιδράσει με τον πραγματικό κόσμο. Τα ns-3 αντικείμενα αποθηκεύονται εσωτερικά ως packet byte buffers , έτοιμα να αποσταλούν σε ένα πραγματικό περιβάλλον δικτύου.

5. **Εναρμόνιση με το πραγματικό περιβάλλον:** Οι ns-3 κόμβοι είναι σχεδιασμένοι μετά την αρχιτεκτονική δικτύωσης των Linux και οι βασικές διεπαφές και τα αντικείμενα εναρμονίζονται με αυτά σε ένα υπολογιστή με Linux. Αυτό διευκολύνει την επαναχρησιμοποίηση του κώδικα, κάνει πιο ρεαλιστικά τα μοντέλα και κάνει τη ροή ελέγχου του προσομοιωτή ευκολότερη στη σύγκριση με πραγματικά συστήματα.

6. **Διαχείριση παραμετροποιήσεων:** Ο ns-3 διαθέτει ένα ενσωματωμένο σύστημα για τη διαχείριση προεπιλεγμένων κι ανά περιπτώσεις τιμές για τις παραμέτρους της προσομοίωσης. Όλες οι παραμετροποιημένες προεπιλεγμένες τιμές για τις παραμέτρους διαχειρίζονται από αυτό το σύστημα, το οποίο έχει ενισχυθεί με την επεξεργασία ορισμάτων από τη γραμμή εντολών, την τεκμηρίωση Doxygen, ένα XML-based και προαιρετικά ένα GTK-based υποσύστημα παραμετροποίησης.

7. **Έλλειψη ενός Ενσωματωμένου Περιβάλλοντος Ανάπτυξης (Integrated Development Environment – IDE)**: Ο ns-3 δε διαθέτει ένα IDE για τη ρύθμιση, τη διόρθωση, την εκτέλεση και την απεικόνιση των προσομοιώσεων σε ένα ενιαίο παράθυρο εφαρμογής όπως σε άλλους προσομοιωτές.

Ο ns-3 είναι ανοικτού κώδικα έργο που βασίζεται στη συμβολή των χρηστών και των ερευνητών.

Μερικοί από τους τρόπους που μπορεί κάποιος να συμβάλλει είναι οι παρακάτω:

- Λίστες συζητήσεων,
- Αναφορά σφαλμάτων,
- Παροχή εγχειριδίων εκμάθησης,
- Συμβολή κώδικα και
- Συντήρηση του προσομοιωτή

Πολλοί χρήστες κάνουν προσομοιώσεις βασιζόμενοι στα υπάρχοντα παραδείγματα του ns-3. Όταν κάποιο παράδειγμα δεν καλύπτει τις ανάγκες του μοντέλου, τότε ο χρήστης πρέπει να κάνει τροποποιήσεις σε υπάρχον μοντέλο ή να δημιουργήσει ένα νέο από την αρχή. Πολλοί χρήστες έρχονται σε επαφή με την ερευνητική κοινότητα του ns-3 και πολλές φορές συμβάλλουν στην ανάπτυξη του.

4.2.3 Βασικές έννοιες

1. Κόμβος (Node): Είναι η βασική υπολογιστική συσκευή του ns-3 και παρουσιάζεται στη C++ από την κλάση Node, η οποία παρέχει διαχείριση των υπολογιστικών συσκευών στις προσομοιώσεις. Ο χρήστης μπορεί να προσθέσει εφαρμογές, περιφερειακές κάρτες και πρωτόκολλα για να ενισχύσει τη λειτουργικότητά τους.

2.Εφαρμογή (Application): Δημιουργεί τη δραστηριότητα που πρόκειται να προσομοιωθεί και παρουσιάζεται στη C++ από την κλάση Application.

3. Κανάλι (Channel): Είναι το κανάλι επικοινωνίας στο οποίο μπορεί να συνδεθεί ένας κόμβος και παρουσιάζεται στη C++ από την κλάση Channel, που παρέχει μεθόδους για

τη διαχείριση αντικειμένων του δικτύου επικοινωνίας και τη σύνδεση των κόμβων με αυτά.

4.Συσκευή Δικτύου (NetDevice): Εγκαθίστανται σε ένα Κόμβο προκειμένου να μπορέσει να επικοινωνήσει με άλλους Κόμβους μέσω του Καναλιού. Παρουσιάζεται στη C++ από την κλάση NetDevice που παρέχει μεθόδους για τη διαχείριση των συνδέσεων μεταξύ Κόμβων και Καναλιών. Ένας Κόμβος μπορεί να συνδεθεί με περισσότερα του ενός Καναλιού μέσω πολλαπλών συσκευών δικτύου.

5.Βοηθοί Τοπολογίας (Topology Helpers): Συνδυάζουν πολλές διαφορετικές εργασίες για τη διευκόλυνση του χρήστη. Κανονίζουν τις συνδέσεις μεταξύ Κόμβων, Καναλιών και Συσκευών Δικτύου αναλαμβάνοντας συγκεκριμένες εργασίες, όπως η διευθυνσιοδότηση και πολλά άλλα.

4.2.4 Καταγραφή

Ο ns-3 παρέχει μία πολυεπίπεδη προσέγγιση για την καταγραφή μηνυμάτων. Η καταγραφή μηνυμάτων μπορεί να απενεργοποιηθεί τελείως, να ενεργοποιηθεί μερικώς η καθολικά παρέχοντας τη δυνατότητα επιλογής του επιπέδου καταγραφής. Υπάρχουν επτά επίπεδα καταγραφής μηνυμάτων με αυξανόμενη λεπτομέρεια. Η καταγραφή μηνυμάτων είναι ιδιαίτερα σημαντική και βοηθάει στον εντοπισμό σφαλμάτων σε πληροφορίες, προειδοποιήσεις, μηνύματα λάθους και την άμεση άντληση πληροφοριών από το σενάριο ή το μοντέλο που προσομοιώνεται.

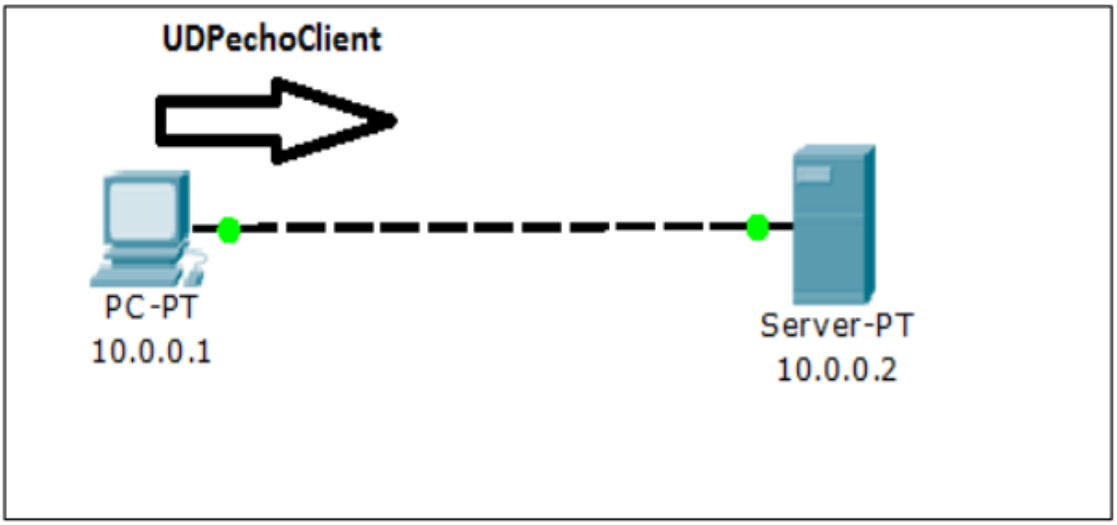
4.2.5 Συστήματα εντοπισμού

Είναι πολύ σημαντική η ύπαρξη Συστήματος Εντοπισμού σε έναν προσομοιωτή λόγω της δυνατότητας για περαιτέρω μελέτη των αποτελεσμάτων της προσομοίωσης. Το Σύστημα Εντοπισμού που διαθέτει ο ns-3 επιτρέπει στους χρήστες να δημιουργήσουν ένα τυπικό σύστημα εντοπισμού που να μπορούν να ρυθμίσουν ποια είναι τα αντικείμενα που θα παράγουν τον εντοπισμό. Οι ενδιάμεσοι χρήστες έχουν την δυνατότητα να επεκτείνουν το σύστημα εντοπισμού ώστε να διαμορφώσουν τη μορφή εξόδου που παράγεται και να εισάγουν νέες πηγές εντοπισμού χωρίς να διαμορφώνουν τον πυρήνα του προσομοιωτή. Μόνο για τους προχωρημένους χρηστές υπάρχει η δυνατότητα διαμόρφωσης του πυρήνα του προσομοιωτή.

Το Σύστημα Εντοπισμού του ns-3 έχει ανεξάρτητο Εντοπισμό για τις πηγές κι ανεξάρτητο Εντοπισμό για τους αποδέκτες, αλλά έχει ενιαίο μηχανισμό για τη σύνδεση μεταξύ τους. Οι πηγές (trace sources) είναι οντότητες που μπορούν να σηματοδοτήσουν τα γεγονότα που συμβαίνουν σε μια προσομοίωση, ενώ οι αποδέκτες (trace sinks) είναι αυτοί που αποδέχονται τα γεγονότα και τα δεδομένα που τους παρέχουν οι πηγές. Ο ns-3 υποστηρίζει τον εντοπισμό ASCII κάνοντας χρήση του AsciiTraceHelper για τη δημιουργία ASCII αρχείων. Τέλος, υποστηρίζει τη δημιουργία .pcap (packet capture) αρχείων εντοπισμού. Το Wireshark χρησιμοποιείται για την ανάγνωση κι εμφάνιση αυτής της μορφής αρχείων.

4.2.6 Παράδειγμα λειτουργίας Ns-3

Ο ns-3 είναι κατασκευασμένος πάνω σε C++ και λειτουργεί μέσω ενός συστήματος βιβλιοθηκών που αποτελεί τον πυρήνα του, τις οποίες εμείς καλούμε κατάλληλα ώστε να δημιουργηθεί η κατάλληλη προσομοίωση. Σε αυτό το σημείο θα παρουσιαστεί ένα παράδειγμα ώστε να γίνει αντιληπτή η λειτουργία του προσομοιωτή ns-3.



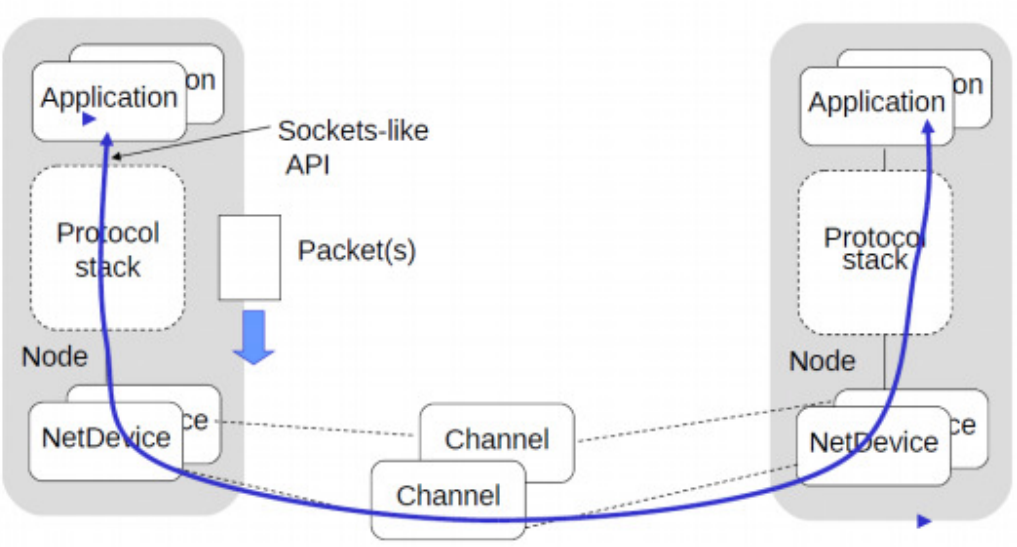
Εικόνα 17: Απεικόνιση δομής παραδείγματος ns-3

Για το παράδειγμα αυτό θα υποθέσουμε ότι έχουμε έναν υπολογιστή ο οποίος θέλει να στείλει ένα πακέτο udpecho από την εφαρμογή του UdpEchoClient στον server στην πόρτα 9 του, όπου βρίσκεται η εφαρμογή UdpEchoServer. Μόλις ο server λάβει το μήνυμα θα απαντήσει πίσω στον υπολογιστή το ίδιο πακέτο. Ο τρόπος με τον οποίο γίνεται αυτό είναι ο παρακάτω:

```
#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"


using namespace ns3;


NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int

main (intargc, char *argv[])

  {

   LogComponentEnable ("UdpEchoClientApplication",
   LOG_LEVEL_INFO);

   LogComponentEnable ("UdpEchoServerApplication",
   LOG_LEVEL_INFO);



NodeContainer nodes;

nodes.Create (2);



PointToPointHelperpointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue
("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer devices;

devices = pointToPoint.Install (nodes);
```

```
InternetStackHelper stack;
stack.Install (nodes);
 Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);

UdpEchoServerHelperechoServer (9);

ApplicationContainerserverApps = echoServer.Install (nodes.Get
(1));

serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));


UdpEchoClientHelperechoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainerclientApps = echoClient.Install (nodes.Get
(0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));


Simulator::Run ();
Simulator::Destroy ();

return 0;
```

Έχουμε λοιπόν το πρόγραμμα το οποίο θα εκτελεστεί ώστε να προσομοιωθεί το παράδειγμα της Εικόνας 17**.** Το πρόγραμμα πρέπει να καλύπτει τις ανάγκες τις Εικόνας 18 ώστε να έχουν οριστεί όλες οι οντότητες τις οποίες θα χρησιμοποιήσει.



Εικόνα 18: Διάγραμμα λειτουργίας οντοτήτων ns-3

Τι ακριβώς όμως κάνει αυτό; Για να γίνει καλύτερη η κατανόηση και στη συνέχεια, σε αυτό το σημείο θα γίνει επεξήγηση των κύριων εντολών που χρησιμοποιήθηκαν για να δούμε πως χτίζεται μια προσομοίωση βήμα-βήμα.

Ξεκινώντας το πρόγραμμα ορίζουμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε. Αυτές είναι γραμμένες σε C++ όπως είπαμε προηγουμένως και εμείς το μόνο που έχουμε να κάνουμε είναι να χρησιμοποιήσουμε τις μεθόδους τους.

```
#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"
```

Πρώτο βήμα , αφού έχουμε φορτώσει τις βιβλιοθήκες μας είναι να δημιουργήσουμε τους κόμβους τους οποίους θα χρειαστούμε. Στην περίπτωση μας είναι 2, ο client και ο server.

```
NodeContainernodes;

nodes.Create (2);
```

Στη συνέχεια αυτό πού έχουμε να κάνουμε είναι να δημιουργήσουμε τον τύπο του καναλιού επικοινωνίας που θα έχουν ο client με τον server και να τον ορίσουμε στους 2 αυτούς κόμβους. Η σύνδεση θα είναι point-to-point με ταχύτητα 5Mbps και η καθυστέρηση του καναλιού θα είναι 2ms.

```
PointToPointHelperpointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue
("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer devices;

devices = pointToPoint.Install (nodes);
```

Έπειτα ανεβαίνουμε άλλο ένα επίπεδο πάνω και δημιουργούμε ένα IP stack το οποίο ορίζει ότι η προσομοίωση μας αναφέρεται στο IP δίκτυο. Επίσης βάζουμε ότι οι διευθύνσεις οι οποίες θα μοιραστούν στους κόμβους θα είναι από το δίκτυο 10.0.0.0/24

```
InternetStackHelper stack;

stack.Install (nodes);

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Στο επίπεδο εφαρμογής φτιάχνουμε πρώτα την εφαρμογή του UdpeEchoServer η οποία θα τρέχει στην πόρτα 9, την εγκαθιστούμε στον κόμβο 2, και ορίζουμε να ξεκινάει στο πρώτο δευτερόλεπτο τις προσομοίωσης και να τελειώνει στο δέκατο.

```
UdpEchoServerHelperechoServer (9);

ApplicationContainerserverApps = echoServer.Install (nodes.Get
(1));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));
```

Το ίδιο πράττουμε και για την εφαρμογή του UdpEchoClient στην οποία ορίζουμε να στείλει ένα πακέτο (Maxpackets), με μέγεθος 1024 bytes (PacketSize) και να αρχίσει στο δεύτερο δευτερόλεπτο της προσομοίωσης και να τελειώσει στο δέκατο. Την εγκαθιστούμε στον πρώτο κόμβο και τέλος ορίζουμε με το Simulator::Run την αρχή της προσομοίωσης και με το Simulator::Destroy ενημερώνουμε το λειτουργικό (μετά το δέκατο δευτερόλεπτο που τελειώνει) ότι η προσομοίωση τελείωσε.

```
UdpEchoClientHelperechoClient (interfaces.GetAddress (1), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainerclientApps = echoClient.Install (nodes.Get
(0));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Simulator::Run ();

Simulator::Destroy ();

return 0;
```

Εκτελώντας το παραπάνω παίρνουμε τα ακόλουθα αποτελέσματα:

```
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
```

Βλέπουμε ότι όντως στο δεύτερο δευτερόλεπτο τις προσομοίωσης ξεκίνησε η αποστολή του πακέτου των 1024 bytes από τον client στο server. Στη συνέχεια ο server το λαμβάνει και απαντάει απευθείας πίσω με το ίδιο μέγεθος bytes και τέλος δέχεται το πακέτο ο client. Ο χρόνος καθορίζεται από την ταχύτητα της σύνδεσης, (5Mbps→625KBps) και την καθυστέρηση που έχει το μέσο που είναι 2 ms.

Κεφάλαιο 5

5.1 Σύγκριση Των Εξομοιωτών GNS3, Boson NetSim, Ns-3

Παρακάτω γίνεται μια αναφορά στις διαφορές των τριών εξομοιωτών (GNS3,Boson NetSim,NS-3) αναλύοντας τα πλεονεκτήματα και τα μειονεκτήματά τους καθώς και ποια σενάρια υλοποιεί καλύτερα ο καθένας.

Όσον αφορά τον GNS3, είναι ελεύθερο λογισμικό ανοικτού κώδικα και τρέχει σε λειτουργικά συστήματα Windows, Linux, MacOsx. Υποστηρίζει όλα τα χαρακτηριστικά των συσκευών και λειτουργικών συστημάτων που εξομοιώνει. Αξίζει να σημειωθεί ότι εξομοιώνει ένα μεγάλο πλήθος συσκευών, όπως Cisco ASA, PIX Firewalls, Cisco IPs, Juniper Routers ή hosts, με αποτέλεσμα να είναι χρήσιμο σε όλα τα επίπεδα μελέτης και έρευνας. Επιπλέον υποστηρίζει σύνδεση με το VirtualBox(δωρεάν λογισμικό εικονικής μηχανής). Τέλος διαθέτει την ικανότητα να αλληλεπιδρά με ένα πραγματικό δίκτυο. Κάποια από τα μειονεκτήματά του είναι ότι οι χρήστες πρέπει να διαθέτουν έγκυρα αρχεία λειτουργικών συστημάτων όλων των συσκευών που εξομοιώνονται καθώς επίσης απαιτείται να εγκατασταθεί επιπλέον λογισμικό όπως Dynamips, Qemu, Putty και Winpcap για την πλήρη λειτουργία του.

Συμπερασματικά ο εξομοιωτής GNS3 λόγω της πολυπλοκότητάς του απευθύνεται σε χρήστες με εξειδικευμένες γνώσεις σε θέματα δικτύων. Είναι το κατάλληλο εργαλείο για την απόκτηση όλων των πιστοποιήσεων της εταιρίας Cisco(CCNA,CCNP,CCIE).

Από την άλλη πλευρά το Boson NetSim δεν είναι ελεύθερο λογισμικό και απαιτείται αγορά άδειας(licence) για την απόκτηση πλήρους έκδοσης. Διατίθεται σε τρεις εκδόσεις: CCENT,CCNA,CCNP. Η καθεμία έκδοση περιλαμβάνει έτοιμες εργαστηριακές ασκήσεις(Labs) με οδηγίες εκπόνησης των σεναρίων. Βασίζεται στο περιβάλλον λειτουργικού συστήματος Windows. Στα πλεονεκτήματά του συγκαταλέγεται το μεγάλος εύρος συσκευών Cisco που προσομοιώνει. Επίσης δεν απαιτεί υψηλό επίπεδο γνώσης σε θέματα δικτύων. Τέλος απευθύνεται καθαρά σε χρήστες που ενδιαφέρονται να αποκτήσουν πιστοποιήσεις της εταιρίας Cisco(CCNA,CCNP).

Τέλος ο προσομοιωτής ns-3 είναι διακριτών γεγονότων και έχει αναπτυχθεί στο να υλοποιεί έναν συμπαγή πυρήνα προσομοίωσης που είναι καλά τεκμηριωμένος, εύκολος στον εντοπισμό σφαλμάτων καθώς και την επίλυσή τους, και που ικανοποιεί όλες τις απαραίτητες ροές εργασιών της προσομοίωσης που είναι αναγκαίες, ξεκινώντας από την διαμόρφωση προσομοίωσης , στην συλλογή ιχνών μέχρι και την ανάλυση.

Επιπλέον, η υποδομή του προσομοιωτή ns-3 ενθαρρύνει την ανάπτυξη προτύπων προσομοίωσης που είναι αρκετά ρεαλιστικά ώστε να επιτρέψουν τον ns-3 να χρησιμοποιηθεί ως εξομοιωτής δικτύων σε πραγματικό χρόνο, που διασυνδέεται με τον πραγματικό κόσμο και που επιτρέπει σε πολλές υπάρχουσες πραγματικές εφαρμογές πρωτοκόλλων να επαναχρησιμοποιηθούν μέσα στον ns-3.

Ο πυρήνας του ns-3 υποστηρίζει την έρευνα πάνω σε IP και μη-IP δίκτυα. Ωστόσο ο ns-3 είναι πιο κατάλληλος σε προσομοιώσεις πάνω σε wireless/IP που βασίζονται σε πρότυπα όπως WI-FI, WiMAX, ή LTE για τα στρώματα 1 και 2 και ένα πλήθος στατικών ή δυναμικών πρωτοκόλλων δρομολόγησης όπως OLSR και AODV για τις IP εφαρμογές.

Επίσης ο ns-3 υποστηρίζει έναν χρονοπρογραμματιστή πραγματικού χρόνου που διευκολύνει τον αριθμό των "simulation-in-the-loop" περιπτώσεων χρήσης για την αλληλεπίδραση με πραγματικά συστήματα. Για παράδειγμα, οι χρήστες μπορούν να εκπέμψουν και να λάβουν τα πακέτα που έχει παράγει ο ns-3 στις πραγματικές συσκευές δικτύων, και ο ns-3 μπορεί να χρησιμεύσει ως ένα πλαίσιο διασύνδεσης για να προσθέσει τα αποτελέσματα των συνδέσεων μεταξύ των εικονικών μηχανών.

Μια άλλη έμφαση του προσομοιωτή είναι η επαναχρησιμοποίηση των πραγματικών εφαρμογών και του κώδικα kernel.

Κάθε τρεις μήνες, διατίθεται μια νέα σταθερή έκδοση ns-3 με τα νέα πρότυπα που αναπτύσσονται, τεκμηριώνονται, επικυρώνονται, και διατηρούνται από τους εθελοντές ερευνητές. Ο ns-3 ενθαρρύνει την ανοικτή επικύρωση αυτών των προτύπων που αναπτύχθηκαν από ερευνητές, από τρίτους ώστε να εξασφαλιστεί ότι τα πρότυπα που δημιουργήθηκαν είναι όσο το δυνατόν υψηλότερης ποιότητας.

Παρακάτω είναι μερικά χαρακτηριστικά γνωρίσματα του ns-3 σε αντίθεση με άλλα εργαλεία:

Ο ns-3 έχει σχεδιαστεί ως ένα σύνολο από βιβλιοθήκες που μπορούν να συνδυάζονται μεταξύ τους αλλά και με άλλες εξωτερικές βιβλιοθήκες. Όσον αφορά το γραφικό περιβάλλον ο ns-3 χρησιμοποιεί κάτι πιο σπονδυλωτό δηλαδή δεν έχει ενσωματωμένο γραφικό περιβάλλον σε σύγκριση με άλλους προσομοιωτές που χρησιμοποιούν έναν

ενιαίο εσωτερικό animator. Υπάρχει μεγάλο εύρος εξωτερικών animator και λογισμικό ανάλυσης δεδομένων όπως επίσης και εργαλεία απεικόνισης αποτελεσμάτων που μπορεί να χρησιμοποιήσει ο ns-3. Ωστόσο, ο χρήστης πρέπει να είναι εξοικειωμένος με εργαλεία ανάπτυξης όπως η C++ και η Python.

Ο ns-3 χρησιμοποιείται κυρίως σε συστήματα Linux, ωστόσο υπάρχει υποστήριξη για FreeBSD, Cygwin(για Windows).

5.2 Πρωτόκολλο Κυλιόμενου Παραθύρου

Στα πρωτόκολλα κυλιόμενου παραθύρου(sliding window protocols), κάθε αποστελλόμενο πακέτο περιέχει έναν αύξοντα αριθμό που μεταβάλλεται από το 0 μέχρι κάποια μέγιστη τιμή. Η μέγιστη τιμή είναι συνήθως 2^n-1, οπότε ο αύξων αριθμός χωρά άνετα σε πεδίο των n bit. Η ουσία όλων των πρωτοκόλλων κυλιόμενου παραθύρου είναι ότι σε κάθε χρονική στιγμή ο πομπός διατηρεί ένα σύνολο αυξόντων αριθμών που αντιστοιχούν σε πακέτα που μπορεί να στείλει. Αυτά τα πακέτα λέγεται ότι πέφτουν μέσα στο παράθυρο αποστολής. Παρομοίως ο δέκτης διατηρεί επίσης ένα παράθυρο λήψης, που αντιστοιχεί στο σύνολο των πακέτων που του επιτρέπεται να δεχθεί. Το παράθυρο του πομπού και το παράθυρο του δέκτη, δεν είναι ανάγκη να έχουν το ίδιο κατώτερο και ανώτερο όριο ή ακόμη να έχουν το ίδιο μέγεθος. Σε μερικά πρωτόκολλα, έχουν σταθερό μέγεθος, αλλά σε άλλα μπορεί να μεγαλώνουν ή να μικραίνουν καθώς στέλνονται ή λαμβάνονται πακέτα.

Οι αύξοντες αριθμοί μέσα στο παράθυρο αποστολής του πομπού αντιπροσωπεύουν πακέτα που στάλθηκαν, αλλά δεν έχουν επιβεβαιωθεί ακόμη. Στον πομπό, κάθε φορά που φθάνει ένα νέο πακέτο προς αποστολή, δίδεται σε αυτό ο επόμενος μεγαλύτερος αύξων αριθμός και το άνω όριο του παραθύρου αυξάνεται κατά ένα. Όταν φθάνει στον πομπό μια επιβεβαίωση, το κάτω όριο του παραθύρου αυξάνεται κατά ένα. Κατ' αυτόν τον τρόπο, το παράθυρο αποστολής διατηρεί διαρκώς τη λίστα των πακέτων που εστάλησαν και δεν επιβεβαιώθηκαν ακόμα.

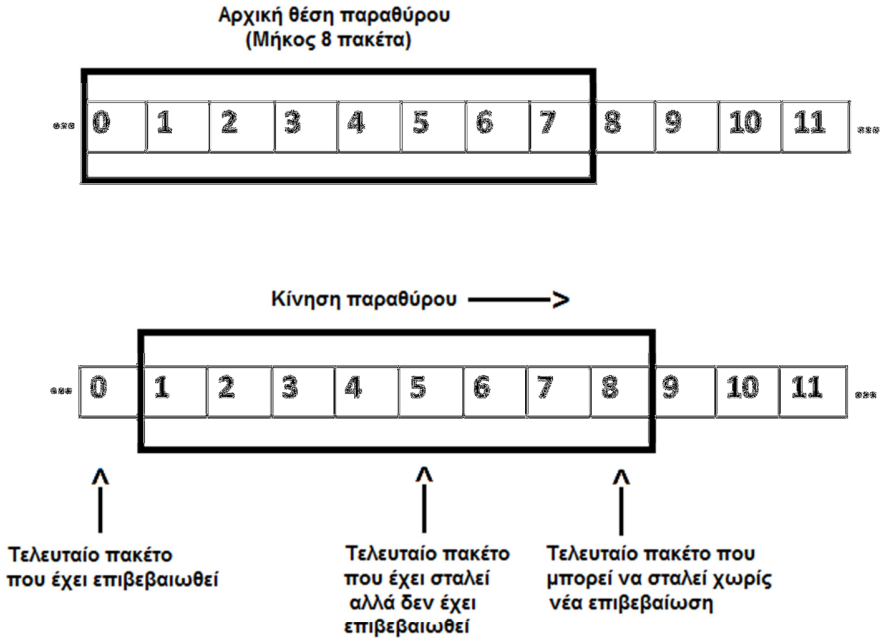
Επειδή τα πακέτα που βρίσκονται κάποια στιγμή στο παράθυρο αποστολής μπορεί τελικά να χαθούν ή να καταστραφούν κατά τη μετάδοσή τους, ο πομπός πρέπει να κρατάει όλα αυτά τα πακέτα στη μνήμη του για πιθανή αναμετάδοση. Έτσι, εάν το

μέγιστο μέγεθος παραθύρου είναι n, ο πομπός χρειάζεται n χώρους προσωρινής αποθήκευσης για να κρατά τα πακέτα που δεν έχουν ακόμη επιβεβαιωθεί. Εάν το παράθυρο φτάσει στο μέγιστο μέγεθός του, ο πομπός πρέπει να σταματήσει την αποστολή μέχρι να ελευθερωθεί μία θέση προσωρινής αποθήκευσης.

Το παράθυρο λήψης του δέκτη αντιστοιχεί στον αριθμό των πακέτων που μπορεί να λάβει ο δέκτης. Κάθε λαμβανόμενο πακέτο που πέφτει έξω από το παράθυρο απορρίπτεται χωρίς σχόλια. Όταν λαμβάνεται ένα πακέτο του οποίου ο αύξων αριθμός είναι ίσος με το κάτω όριο του παραθύρου λήψης, περνά στο ανώτερο στρώμα, παράγεται μια επιβεβαίωσή του και το παράθυρο ολισθαίνει κατά μία θέση. Το παράθυρο του δέκτη παραμένει πάντοτε στο αρχικό του μέγεθος, αντίθετα προς το παράθυρο του πομπού. Σημειώστε ότι μέγεθος παραθύρου λήψης ίσο με 1 σημαίνει ότι ο δέκτης δέχεται μόνο τα πακέτα που έρχονται με τη σωστή σειρά. Αυτό όμως δεν ισχύει στα μεγαλύτερα παράθυρα. Αντίθετα, τα δεδομένα διοχετεύονται στο ανώτερο στρώμα πάντοτε με τη σωστή σειρά, ανεξάρτητα από το μέγεθος του παραθύρου του κατωτέρου στρώματος.

Με την τεχνική του κυλιόμενου παραθύρου χρησιμοποιείται αποδοτικότερα το εύρος ζώνης του δικτύου, αφού επιτρέπεται στον πομπό να στείλει περισσότερα πακέτα πριν αρχίσει να περιμένει για επιβεβαίωση. Το παράδειγμα της Εικόνας 19 βοηθά να κατανοήσουμε καλύτερα την τεχνική του παραθύρου ολίσθησης.

Έστω ότι ο πομπός θέλει να στείλει την ακολουθία των πακέτων που φαίνονται στο σχήμα. Το πρωτόκολλο τοποθετεί ένα μικρό παράθυρο στην αρχή της σειράς των πακέτων και μεταδίδει όλα τα πακέτα που βρίσκονται μέσα στο παράθυρο, χωρίς να περιμένει επιβεβαίωση.



Εικόνα 19: Μηχανισμός Sliding Window

Ένα πακέτο που στάλθηκε και δεν έχει επιβεβαιωθεί λέγεται *unacknowledged* (ανεπιβεβαίωτο). Ο αριθμός των πακέτων που μπορεί να είναι ανεπιβεβαίωτα κάθε χρονική στιγμή περιορίζεται από το εύρος του παραθύρου που περιλαμβάνει τόσα πακέτα όσα χωράνε και στην πραγματική σύνδεση. Αν για παράδειγμα το μήκος του παραθύρου είναι 8 ο πομπός μπορεί να στείλει 8 πακέτα πριν αρχίσει να περιμένει για επιβεβαιώσεις.

Όπως φαίνεται στην Εικόνα 19, μόλις ο πομπός λάβει μία επιβεβαίωση ack για το πρώτο πακέτο, το παράθυρο ολισθαίνει κατά μία θέση και μπορεί πλέον να στείλει το επόμενο πακέτο που πλέον καλύπτεται από το παράθυρο. Στη συνέχεια το παράθυρο εξακολουθεί να ολισθαίνει συνεχώς καθώς φτάνουν από το δέκτη μηνύματα επιβεβαίωσης. Κάθε φορά ο πομπός στέλνει όσα πακέτα βρίσκονται μέσα στα όρια του παραθύρου. Η απόδοση του πρωτοκόλλου παραθύρου ολίσθησης εξαρτάται από το μέγεθός του.

5.2.1 Πρωτόκολλο TCP και κυλιόμενο παράθυρο

Το κυλιόμενο παράθυρο χρησιμοποιείται στα περισσότερα connection oriented πρωτόκολλα δικτύου, ανάμεσα σε άλλα στο PPP και στο TCP. Η άμεση σχέση του TCP πρωτοκόλλου με τον sliding window μηχανισμό φαίνεται και από την διαδικασία ελέγχου ροής που χρησιμοποιεί το πρωτόκολλο TCP.

Ο έλεγχος ροής απαιτεί την επιβεβαίωση λήψης (acknowledgment) κάθε πακέτου από τον απόμακρο host πριν να σταλεί το επόμενο. Οι αλγόριθμοι για το sliding window, που χρησιμοποιούνται από το TCP, επιτρέπουν σε πολλαπλά πακέτα δεδομένων να μεταφέρονται ταυτόχρονα για να χρησιμοποιείται αποδοτικότερα το εύρος ζώνης(bandwidth) ενός δικτύου.

Για παράδειγμα, εαν ένας υπολογιστής Α στείλει 4 byte με αριθμό ακολουθίας (sequence number) 100 - συνεπώς, τα 4 bytes έχουν αριθμό ακολουθίας 100, 101, 102 και 103 - τότε ο παραλήπτης πρέπει να απαντήσει με επιβεβαίωση (acknowledgement) που φέρει sequence number 104. Αυτό πρόκειται να είναι το επόμενο byte που περιμένει στο επόμενο πακέτο. Εαν για κάποιο λόγο, τα τελευταία δύο bytes περιέχουν σφάλματα τότε η τιμή της επιβεβαίωσης θα είναι 102, εφόσον τα bytes με αριθμό 100 και 101 έχουν φτάσει με επιτυχία.

5.3 Υλοποίηση πρωτοκόλλου «ολισθαίνοντος παραθύρου»

Στα πλαίσια της εργασίας εξετάστηκε η λειτουργία του πρωτοκόλλου ολισθαίνοντος παραθύρου στον προσομοιωτή NS-3. Η προσπάθεια μας επικεντρώθηκε στην εύρεση των στοιχείων εκείνων τα οποία παρέχει ο προσομοιωτής αυτός τα οποία θα επέτρεπαν, κατά το δυνατόν πιο πειστικά και πιο συγγενικά, την εξομοίωση, τη μεγαλύτερη δυνατή προσέγγιση αυτής της λειτουργίας. Κατά της διάρκειας της προκαταρκτικής έρευνας για υπαρκτές υλοποιήσεις του πρωτοκόλλου αυτού, περιήλθε σε γνώση μας μια συγκεκριμένη υλοποίηση που έγινε για την προηγούμενη έκδοση του ns(ns-2) εκ μέρους του *Ινστιτούτου Επιστημών Πληροφορίας του Πανεπιστημίου της Νότιας Καλιφόρνια*. Μετά και την επισκόπηση αυτού του κώδικα, θεωρήθηκε σκόπιμο, ιδιαίτερα για λόγους πληρότητας και βεβαιότητας της ορθότητας της προγραμματιστικής προσέγγισης, να

ακολουθηθεί η λογική αυτής της υλοποίησης. Ο κώδικας αυτός είναι δημόσια προσβάσιμος και μπορεί να προσπελαστεί καθ'ολοκληρία σε αυτή την ιστοσελίδα[1].

Βάσει των παραπάνω, αποκρυσταλλώθηκαν μια σειρά από σημεία τα οποία έπρεπε να προσαρμοστούν στο ns-3 με βάση και τις ιδιαίτερες δυνατότητες και παραμέτρους του πάντοτε, έτσι ώστε να πραγματοποιηθεί η προσομοίωση του πρωτοκόλλου του ολισθαίνοντος παραθύρου. Έτσι, οι απαιτήσεις σε ότι αφορά την υλοποίηση κωδικοποιούνται στα παρακάτω:

α)ύπαρξη ενός εναρκτήριου κόμβου αποστολέα σε TCP(TCP sender)
β)ύπαρξη ενός καταληκτικού κόμβου παραλήπτη σε TCP(TCP receiver)
γ) ύπαρξη δύο κόμβων που παρεμβάλλονται ανάμεσα στον εναρκτήριο και τον τελικό κόμβο
δ)χρησιμοποίηση αμφίδρομων/διπλής κατεύθυνσης(duplex)καναλιών/ζεύξεωνν(links) σημείο-προς-σημείο
ε)χρήση ουρών για τις προαναφερόμενες ζεύξεις τύπου drop-tail
στ)η ιδιότητα *«queue-limit»* που αναφέρεται στην ουρά του επιπέδου ISO/OSI 2(ζεύξης δεδομένων) αναπαριστά τη λειτουργία του ολισθαίνοντος παραθύρου

5.3.1 Επιλεγμένες παρατηρήσεις πάνω στην υλοποίηση

Το σενάριο που υλοποιείται στον κώδικα, ο οποίος είναι αναρτημένος στο Παράρτημα Α, ουσιαστικά είναι η αποστολή πακέτων δεδομένων από έναν κόμβο, τον οποίο ονομάζουμε Α σε αυτή την υλοποίηση, προς ένα καταληκτικό κόμβο, τον οποίο ονομάζουμε Β, διαμέσου δύο κόμβων που παρεμβάλλονται με βάση την υλοποίηση που χρησίμευσε ως πρότυπο της δικής μας.

Το ίσως περισσότερο καίριο στοιχείο της υλοποίησης είναι η παράμετρος που αντιστοιχίζεται στην ιδιότητα «queue-limit», η οποία είναι η ιδιότητα(Attribute) *"MaxPackets"*. Η ιδιότητα αυτή προσομοιώνει την ουρά του πρωτοκόλλου του ολισθαίνοντος παραθύρου και μπορεί να ρυθμιστεί μέσα από τη μέθοδο SetQueue, εάν αυτή εφαρμοστεί στο κατάλληλο στιγμιότυπο της κλάσης *PointToPointHelper*. Σε κάθε ένα από τους δύο ακραίους κόμβους, τον Α και τον Β, πραγματοποιείται

[1]http://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_HYUNAH/D-Research/sliding-vs-ns2.html

εγκατάσταση ενός socket τύπου *«TCP-Socket-Base»,* προκειμένου να διεξαχθεί η μεταφορά των πακέτων με τις παραμέτρους που επιθυμεί ο χρήστης(datarate, mtu, delay κ.λπ.).Συγκεκριμένα, η δημιουργία αυτών των sockets επιτυγχάνεται με την αλλαγή του *default* του socket που δημιουργεί η κλάση BulkSendHelper όπως και η κλάση PacketSinkHelper. Το *TCP-Socket-Base* είναι το πλέον παραμετροποιήσιμο είδος socket, σχετιζόμενο άλλωστε και με την υλοποίηση του ολισθαίνοντος παραθύρου, σύμφωνα με τα αρχεία *«τεκμηρίωσης»*(tutorial/documentation) του ns-3.

Ένα τελευταίο σημείο που χρειάζεται ιδιαίτερη προσοχή είναι η διαφοροποίηση των εννοιών *"SlidingWindow"*και *"CongestionWindow"*. Το Sliding Window(ολισθαίνον παράθυρο) είναι έννοια που περιλαμβάνεται στο $2^{ο}$ επίπεδο(επίπεδο ζεύξης δεδομένων) του μοντέλου ISO/OSI, ενώ το *«παράθυρο συμφόρησης»*(congestion window) περιλαμβάνεται στο $4^{ο}$ επίπεδο(επίπεδο «εφαρμογής»-TCP). Συγκεκριμένα, η ιδιότητα *InitialCwnd* που παραμετροποιεί την αρχή του παραθύρου συμφόρησης έχει να κάνει με τον αλγόριθμο *Slow-Start* του TCP, οπότε δεν υπάρχει αλλαγή του στον κώδικα και παραμένει στην αρχική του τιμή, δηλαδή το 1. Επισημαίνεται ότι οι μονάδες μέτρησής του είναι τα segments του TCP(65535 bytes).
Στον κώδικα δεν χρειάστηκε να υπάρξει πρόβλεψη για παραμετροποίηση ως προς το *InitialCwnd*, αν και οι συνθήκες συμφόρησης που ήταν επιθυμητό να δημιουργηθούν αφορούσαν το επίπεδο ISO/OSI2(datalink layer) και όχι το 4(application layer).

5.3.2 Κύριες κλάσεις του NS-3 που χρησιμοποιήθηκαν

α)Η κλάση **"Node"** είναι μια χρήσιμη, όπως και ευέλικτη κλάση, η οποία καθιστά λιγότερο απαραίτητη όπως και συχνή μέσα στον κώδικα την κλάση NodeContainer. Η κλάση αυτή σε κάθε στιγμιότυπό της αποδίδει ένα δείκτη(pointer) για ένα κόμβο δικτύου. Ο κόμβος αυτός πολύ εύκολα μπορεί να αξιοποιηθεί μέσα από ορισμένες μεθόδους που υπάρχουν σε άλλες κλάσεις και ειδικά την μέθοδο *"Install"*. Στα πλαίσια του κώδικα, οι εν λόγω δείκτες δημιουργούνται με την εντολή CreateObject, όπως στην παρακάτω γραμμή:

```
Ptr<Node> A = CreateObject<Node> ();
```

β)Η κλάση ***NetDeviceContainer*** είναι μια κλάση η οποία προκύπτει συνήθως μέσα στον κώδικα του ns-3 από την εγκατάσταση των κόμβων στο στιγμιότυπο του PointToPointHelper το οποίο αντιστοιχεί στη ζεύξη μεταξύ δύο διαδοχικών κόμβων. Θα λέγαμε ότι η κλάση αυτή είναι ένας υποδοχέας ενός συνόλου συσκευών δικτύου, οι οποίες αντιστοιχίζονται σε ισάριθμους σταθμούς εργασίας εκ των οποίων καθένας βρίσκεται σε κάθε κόμβο του δικτύου μας. Όλα αυτά υλοποιούνται στην παρακάτω εντολή, καθώς και σε όλες τις όμοιές της:

devAI=AI.Install(A,I);

γ) Η ***PointToPointHelper*** είναι από τις σημαντικότερες κλάσεις και επιπλέον από τις πιο εύχρηστες των προγραμμάτων του ns-3. Χρησιμοποιείται ώστε να τίθενται διάφορα στοιχεία που αφορούν τις p2p ζεύξεις του προσομοιωμένου δικτύου μας. Στο πρόγραμμα προσομοίωσης του ολισθαίνοντος παραθύρου έχουμε συγκεκριμένα τρεις βασικούς σκοπούς για τους οποίους αξιοποιείται η κλάση αυτή. Ειδικότερα, ανατίθενται τιμές σε μια σειρά από διαφορετικά στοιχεία των τριών υποζεύξεων οι οποίες συνιστούν τη μία ζεύξη σημείο-προς-σημείο(από τον A στον B). Τα στοιχεία αυτά είναι τα παρακάτω:

i) Η συσκευή δικτύου(NetDevice). Μέσω των συσκευών δικτύου τίθενται ιδιότητες όπως η *«DataRate»*(ρυθμός αποστολής δεδομένων), ως εξής:

AI.SetDeviceAttribute ("DataRate", StringValue ("2Mbps"));

ii) Το κανάλι δικτύου(Channel). Ιδιότητα αυτού του στοιχείου είναι η καθυστέρηση στην μετάδοση πακέτων(*«delay»*), όπως στην ακόλουθη εντολή του κώδικα του πρωτοκόλλου:

AI.SetChannelAttribute ("Delay", StringValue ("2ms"));

iii) Η ουρά(Queue). Για τους σκοπούς αυτής της προσομοίωσης, είναι το σπουδαιότερο στοιχείο, εφόσον από αυτό το στοιχείο τίθεται και η αποδεκτή ουρά πακέτων σε ότι αφορά τον αριθμό τους. Για να διεκπεραιωθεί αυτό, απαιτείται η μεταβολή της ουράς σε τύπου *"drop-tail queue"* και παραμετροποίηση του κώδικα μέσω παροχής δυνατότητας

μεταβολής της ιδιότητας «*MaxPackets*». Κάτι τέτοιο υλοποιείται στην ακόλουθη γραμμή του κώδικα:

```
AI.SetQueue("ns3::DropTailQueue", "MaxPackets", UintegerValue(MaxPac));
```

δ) Ο **InternetStackHelper**[2]είναι το αντίστοιχο του PointToPointHelper για το Διαδίκτυο. Ουσιαστικά εγκαθιστά και εμπεδώνει τη στοίβα πρωτοκόλλων Δικτύου, δηλαδή του επιπέδου ISO/OSI 3, στους κόμβους οι οποίοι εγκαθίστανται σε αυτόν. Δείγμα εντολής αυτών που χρησιμοποιήθηκαν στον κώδικα της προσομοίωσης είναι ο παρακάτω:

```
stack.Install(A);
```

ε) Η **Ipv4AddressHelper** είναι μία βοηθητική ακόμη κλάση η οποία είναι κατάλληλη για την εγκατάσταση και καθορισμό των IP διευθύνσεων οι οποίες κατόπιν χρησιμοποιούνται με το **Ipv4InterfaceContainer.** Η τελευταία κλάση είναι, επιπλέον, το αποτέλεσμα της εφαρμογής της μεθόδου Assign πάνω σε ένα στιγμιότυπο της Ipv4AddressHelper. Κατ'αυτόν τον τρόπο, ανατίθενται οι επιθυμητές IP διευθύνσεις, με την εισαγωγή της βάσης του υποδικτύου καθώς και της μάσκας υποδικτύου σε σημειογραφία όπως αυτή που υποδεικνύεται από την παρακάτω ενδεικτική εντολή του κώδικα:

```
Ipv4InterfaceContainerpairAI =address.Assign(devAI);
```

στ) Η ***PacketSinkHelper*** [3] είναι μια ακόμη βοηθητική κλάση η οποία αφορά τη δημιουργία ενός socket το οποίο θα δέχεται και θα «καταβροχθίζει» τα πακέτα τα οποία φτάνουν σε αυτό. Για τη δημιουργία του, χρειάζεται μια διεύθυνση η οποία θα περιλαμβάνει και μια θύρα, καθώς και μια τιμή που θα αφορά τον τύπο του socket, ο οποίος είναι παραμετροποιήσιμος μέσα από την αλλαγή του default(διαμέσου του «*Config::SetDefault*»). Μια τέτοια εντολή είναι η ακόλουθη:

```
PacketSinkHelpersinkA ("ns3::TcpSocketFactory", B_Sink_Addr);
```

[2]Ns-3 Tutorial, σελ. 29

[3] https://www.nsnam.org/doxygen/classns3_1_1_packet_sink.html#details

ζ) Ο ***BulkSendHelper***[4] είναι το συμπληρωματικό στοιχείο του PacketSinkHelper σε επίπεδο εφαρμογής, δηλαδή του επιπέδου ISO/OSI 4. Και πάλι, σε ότι αφορά τα ορίσματα, υπάρχει ένα στοιχείο που αφορά την κατασκευή ενός socket, καθώς και μιας διεύθυνσης, η οποία ενσωματώνει και μια θύρα, η οποία, όπως ακριβώς και στον PacketSinkHelper, είναι απαραίτητη για τη δημιουργία μιας εφαρμογής αποστολής πακέτων. Ειδικότερα, όπως αναφέρει και η τεκμηρίωση του ns-3, μια τέτοια εφαρμογή, αποστέλλει το μεγαλύτερο αριθμό πακέτων ο οποίος είναι σε κάθε στιγμή εφικτός. Η επιλογή ενός BulkSendHelper έχει να κάνει με την επιθυμητή δημιουργία συνθηκών συμφόρησης στο προσομοιωμένο δίκτυο, εφόσον κάτι τέτοιο θα συμβάλει αισθητά στη δημιουργία τους. Η σχετική εντολή είναι η ακόλουθη:

```
BulkSendHelpersourceAhelper ("ns3::TcpSocketFactory", sinkAddr);
```

5.3.3 Ανάλυση ειδικότερων συνθηκών σεναρίου υλοποίησης και αποτελεσμάτων

Το σενάριο το οποίο θα αναλύσουμε έχει να κάνει με την ύπαρξη συνθηκών απόλυτης συμφόρησης μέσα στο δίκτυο, καθώς και την απουσία αυτών.
Για την δημιουργία τέτοιων συνθηκών, εισήχθη αρχικά μέγεθος παραθύρου ίσο με 2, το οποίο και αντιστοιχεί απόλυτα σε τέτοιες συνθήκες και το οποίο σταδιακά αυξάνεται σε 4,5,8,16,24.

Ταυτόχρονα, σε όλους αυτούς τους πειραματισμούς, παρέμειναν σταθερά τα:

α)**datarate=2 Mbps**,

β)**delay=2 ms** και

γ)**mtu**(Maximum Transmission Unit της ζεύξης-καναλιού) ίσον με **500 bytes.**

Ο στόχος ήταν να υπάρξει μεγάλος αριθμός ip-fragments λόγω της κατάτμησης των ip πακέτων(διαδικασία ***ip-fragmentation***) στο επίπεδο ζεύξης δεδομένων, ώστε να «χωρέσουν» στο MTU της ζεύξης.
Αυτό που αναμένεται και θεωρητικά, δηλαδή να υπάρχει αρχικά μεγάλη απώλεια

[4] https://www.nsnam.org/doxygen/classns3_1_1_bulk_send_helper.html#ad2ce134f59fa593314538b8731ae6977

πακέτων και σταδιακά ολοένα και μικρότερη, κάτι που είναι μέτρο της συμφόρησης, επιβεβαιώνεται και πειραματικά μέσα από το ns-3. Έτσι, λαμβάνουμε τα αποτελέσματα τα οποία συνοψίζονται στον ακόλουθο πίνακα:

Window Size	Dropped Packets
2	11
4	6
5	2
8	3
16	4
24	0

Πίνακας 6:Αποτελέσματα απώλειας πακέτων σε σχέση με το μέγεθος παραθύρου

Πράγματι το δίκτυο για μέγεθος παραθύρου ίσο με 2 αντιμετωπίζει συνθήκες συμφόρησης, ενώ από την αύξηση του μεγέθους στο 4 παρατηρείται ήδη μια μείωση της απώλειας κατά 45,05% της αρχικής. Ανάλογα αποτελέσματα λαμβάνονται και για την μετρική του throughput(διεκπεραιωτική ικανότητα), η οποία αυξάνεται σταδιακά, όπως αυξάνεται και το μέγεθος του παραθύρου. Το σύνολο των αποτελεσμάτων(γραφικές παραστάσεις) που προέκυψαν με τη χρήση του εργαλείου Wireshark δίνονται στο Παράρτημα Β.

Window Size 2: Για το μέγεθος παραθύρου 2 παρατηρούμε ότι έχουμε το χαμηλότερο throughput. Μετά από τα πρώτα δευτερόλεπτα της προσομοίωσης, αυτό φαίνεται ότι κυμαίνεται κοντά στα 10-12 πακέτα ανά δευτερόλεπτο. Επισημαίνουμε από αυτό το σημείο ότι αυτό το οποίο περιμένουμε είναι μια συνεχή αύξηση του throughput καθώς το μέγεθος του παραθύρου αυξάνεται. Αυτή η αύξηση αναμένεται θεωρητικά να αγγίξει κάποιο σημείο όπου υπάρχει κορεσμός της λεγόμενης «μποτιλιαρισμένης ζεύξης»(bottleneck link). Επίσης ο αριθμός των πακέτων που απορρίπτονται είναι με μεγάλη διαφορά, όπως επισημάνθηκε παραπάνω, ο μεγαλύτερος από όλους τους παρατηρούμενους(11). Αυτό δεν οφείλεται σε τίποτα άλλο πέραν της μεγάλης παραγωγής πακέτων, και ειδικά πακέτων που οφείλονται σε κατακερματισμό, τα οποία όμως δεν μπορούν να τύχουν επεξεργασίας στην ουρά που σχηματίζεται λόγω του

μικρού μεγέθους αυτής και έτσι γίνεται αυτό που στα αρχεία tracing ονομάζεται «drop», δηλαδή απόρριψη των πακέτων.

Window Size 4: Εδώ παρατηρείται μια εντυπωσιακή αύξηση του throughput, καθώς και των μεταφερόμενων μέσα στην «μποτιλιαρισμένη ζεύξη» πακέτων ανά δευτερόλεπτο. Η αύξηση αυτή είναι της τάξης του 300 έως 900%. Επίσης οι απώλειες σε πακέτα ελαττώνονται κατά περίπου 45,5%. Αυτό οφείλεται στο ότι το μεγαλύτερο παράθυρο της ζεύξης έχει τη δυνατότητα «ενθυλάκωσης»(enqueue) διπλάσιου αριθμού πακέτων που αναμένουν τη μεταφορά τους.

Window Size 5: Και με αυτήν την μεταβολή παρατηρούμε ότι το μέσο throughput φτάνει τα 800000 bits/sec. Πρόκειται για μια αύξηση από το προηγούμενο μέγεθος παραθύρου μεγαλύτερη του 400%. Αυτή η αύξηση εντάσσεται στη γνωστή από την προαναφερόμενη θεωρία σταδιακή μεγέθυνση του throughput μέχρι ένα πιθανό σημείο κορεσμού της μποτιλιαρισμένης ζεύξης. Επίσης τα απορριφθέντα πακέτα είναι πλέον μόλις 2, κάτι που οφείλεται στο ότι δεν δημιουργούνται μεγαλύτερες ουρές πακέτων μέσα στην ζεύξη.

Window Size 8: κι εδώ παρατηρούμε μια σημαντική αύξηση του throughput, το οποίο πλέον, όπως φαίνεται από το σχετικό γράφημα, προσεγγίζει το 1 Mbps. Σημειώνεται ότι κι εδώ τα πακέτα που απορρίπτονται είναι πολύ λίγα στον αριθμό, μόλις 3. Η ανεπαίσθητη αύξηση από το προηγούμενο μέγεθος παραθύρου που δοκιμάσαμε(5) οφείλεται πολύ πιθανόν στην εγκατάσταση της εφαρμογής *BulkSend* στον TCP αποστολέα, κάτι που συνεπάγεται την αποστολή το δυνατόν μεγαλύτερης ποσότητας δεδομένων βάσει των υπαρχουσών κάθε φορά παραμέτρων.

Window Size 16: Με αυτήν την αλλαγή παρατηρούμε μια τελευταία αύξηση του throughput, το οποίο πλέον προσεγγίζει το 1.2 Mbps. Φαίνεται πλέον ότι βρισκόμαστε κοντά στον κορεσμό της bottleneck ζεύξης. Τα απορριφθέντα πακέτα και πάλι αυξάνονται κατά ένα, κάτι που και πάλι θα πρέπει να αποδοθεί, εκτός από πιθανή τυχαιότητα, στην ιδιότητα της εφαρμογής BulkSend του κόμβου Α, να αποστέλλει το δυνατόν περισσότερα πακέτα.

Window Size 24; Εδώ πλέον είναι σαφές ότι έχουμε υπερβεί το σημείο κορεσμού της μποτιλιαρισμένης ζεύξης βάσει των παραμέτρων που δόθηκαν, εφόσον το throughput παρουσιάζει μια εικόνα πολύ παρεμφερή με αυτή του προηγούμενου μεγέθους παραθύρου. Τέλος, παρατηρητέο είναι ότι τα απορριφθέντα πακέτα πλέον είναι ίσα με μηδέν. Αυτό είναι απολύτως εύλογο, εφόσον πλέον έχουμε ένα κατά 50% μεγαλύτερο μέγεθος παραθύρου, με το throughput, λόγω του επελθόντος κορεσμού της ζεύξης, να μένει σχεδόν ίδιο. Έτσι, τα 4 πακέτα που απωλέσθηκαν με το προηγούμενο μέγεθος εδώ εξυπηρετούνται, δηλαδή ενθυλακώνονται(enqueue) και εξάγονται(dequeue) από την ουρά κανονικά, δηλαδή μεταφέρονται χωρίς κώλυμα.

Βιβλιογραφία

[1] S. Siraj, A. Kumar Gupta, Rinku-Badgujar, "Network Simulation Tools Survey", International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 4, June 2012

[2] A. Mahmood, F. Saleem, A. Latif, M. Ahmad, "Key Features and Optimum Performance of Network Simulators: A Brief Study", International Journal of Computer Trends and Technology (IJCTT) Vol. 4, Issue 9, September 2013

[3] James F.Kurose, Keith W.Ross, "Δικτύωση Υπολογιστών: Προσέγγιση από Πάνω προς τα Κάτω", 4η Έκδοση, σελ 225-240, 2008

[4] Μαρία Ζογκού, "Δρομολόγηση με κριτήριο την διαθέσιμη ενέργεια για την παροχή Ποιότητας Υπηρεσίας σε IEEE 802.11s Ασύρματα Πλεγματικά Δίκτυα'', Μεταπτυχιακή Διατριβή, Πανεπιστήμιο Πειραιώς-Τμήμα Πληροφορικής, σελ 57-60, 2014

[5] Σιβάκης Νικόλαος, "Free and Open-source Tools for Network Simulation", εργασία στα Δίκτυα, Πανεπιστήμιο Μακεδονίας-Τμήμα MIS, σελ 3-8

Διευθύνσεις Internet

[6] GNS3 official website, https://www.gns3.com/

[7] Static NAT, http://resources.intenseschool.com/gns3-labs-for-ccna-basic-nat-network-address-translation/

[8] NAT, https://en.wikipedia.org/wiki/Network_address_translation

[9] EIGRP, https://en.wikipedia.org/wiki/Enhanced_Interior_Gateway_Routing_Protocol

[10] Boson NetSim official website, http://www.boson.com/

[11] NS3 official website, https://www.nsnam.org/documentation/

[12] https://en.wikipedia.org/wiki/Sliding_window_protocol

[13] http://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_HYUNAH/D-Research/sliding-vs-ns2.html

[14] https://www.nsnam.org/doxygen/classns3_1_1_packet_sink.html#details

[15]https://www.nsnam.org/doxygen/classns3_1_1_bulk_send_helper.html#ad2ce134f59fa593314538b8731ae6977

[16] NS2 complete tutorial, http://www.isi.edu/nsnam/ns/tutorial/

[17] Wireshark tutorial, https://cs.gmu.edu/~astavrou/courses/ISA_674_F12/Wireshark-Tutorial.pdf

ΠΑΡΑΡΤΗΜΑ Α

Παρακάτω παρατίθενται ο κώδικας υλοποίησης του πρωτοκόλλου ολισθαίνοντος παραθύρου στον εξομοιωτή ns-3:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/tcp-socket-base.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/drop-tail-queue.h"
#include "ns3/net-device-container.h"
#include "ns3/net-device.h"
#include "ns3/bulk-send-application.h"
#include "ns3/bulk-send-helper.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/tcp-socket.h"
#include "ns3/drop-tail-queue.h"
#include "ns3/pcap-file.h"

using namespace ns3;
//NS_LOG_COMPONENT_DEFINE ("Sliding_Window_Protocol_Simulation");
int main (int argc, char *argv[])
{
//LogComponentEnable ("Sliding_Window_Protocol_Simulation",
LOG_LEVEL_INFO);
 //LogComponentEnable ("Sliding-W-Receiver-Application", LOG_LEVEL_INFO);
  Ptr<Node> A = CreateObject<Node> ();
  Ptr<Node> I = CreateObject<Node> ();
  Ptr<Node> J = CreateObject<Node> ()
```

```
  Ptr<Node> B = CreateObject<Node> ();
  NodeContainer nodes;
  nodes.Add(A);
  nodes.Add(I);
  nodes.Add(J);
  nodes.Add(B);
//dimiourgoume tous komvous(sender, receiver, dyo endiamesoi)
//dimiourgoume deiktes-pointers gia arhiko, teliko kai endiamesous komvous
//PointToPointNetDevice P2PdevAI, P2PdevIJ, P2PdevJB;
NetDeviceContainer devAI, devIJ, devJB, devTotal;
PointToPointHelper AI, IJ, JB, HTotal;


//Maximum Packets Of the p2p Queue(they are dropped, if more)
uint64_t WinSiz = 2; //edw prepei na ginei h parametropihsh,diladi edw orizetai to
megethos ouras tou parathirou // (gia ton algorithmo tou olisthainontos parathirou)


uint64_t MaxPac = WinSiz;
//CreateSocket Sock1(TcpSocketBase, A);
 AI.SetDeviceAttribute ("DataRate", StringValue ("2Mbps"));
 IJ.SetDeviceAttribute ("DataRate", StringValue ("2Mbps"));
 JB.SetDeviceAttribute ("DataRate", StringValue ("2Mbps"));
 AI.SetChannelAttribute ("Delay", StringValue ("2ms"));
 IJ.SetChannelAttribute ("Delay", StringValue ("2ms"));
 JB.SetChannelAttribute ("Delay", StringValue ("2ms"));
 AI.SetDeviceAttribute ("Mtu", UintegerValue (500));
 IJ.SetDeviceAttribute ("Mtu", UintegerValue (500));
 JB.SetDeviceAttribute ("Mtu", UintegerValue (500));
```

```
 //Config::SetDefault ("ns3::NetDevice::Mtu", UintegerValue (100));
 AI.SetQueue("ns3::DropTailQueue", "MaxPackets", UintegerValue(MaxPac));
 IJ.SetQueue("ns3::DropTailQueue", "MaxPackets", UintegerValue(MaxPac));
 JB.SetQueue("ns3::DropTailQueue", "MaxPackets", UintegerValue(MaxPac));

devAI=AI.Install(A,I);
devIJ=IJ.Install(I,J);
devJB=JB.Install(J,B);

InternetStackHelper stack;
stack.Install(A);
stack.Install(I);
stack.Install(J);
stack.Install(B);


Ipv4AddressHelper address;
address.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer Interfaces;
Ipv4InterfaceContainer pairAI =address.Assign(devAI);
address.SetBase ("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer pairIJ =address.Assign(devIJ);
address.SetBase ("192.168.3.0", "255.255.255.0");
Ipv4InterfaceContainer pairJB =address.Assign(devJB);
//address.SetBase ("192.168.1.0", "255.255.255.0");
//Ipv4InterfaceContainer totNod = address.Assign(devTotal);
Interfaces.Add(pairAI);
Interfaces.Add(pairIJ);
Interfaces.Add(pairJB);
```

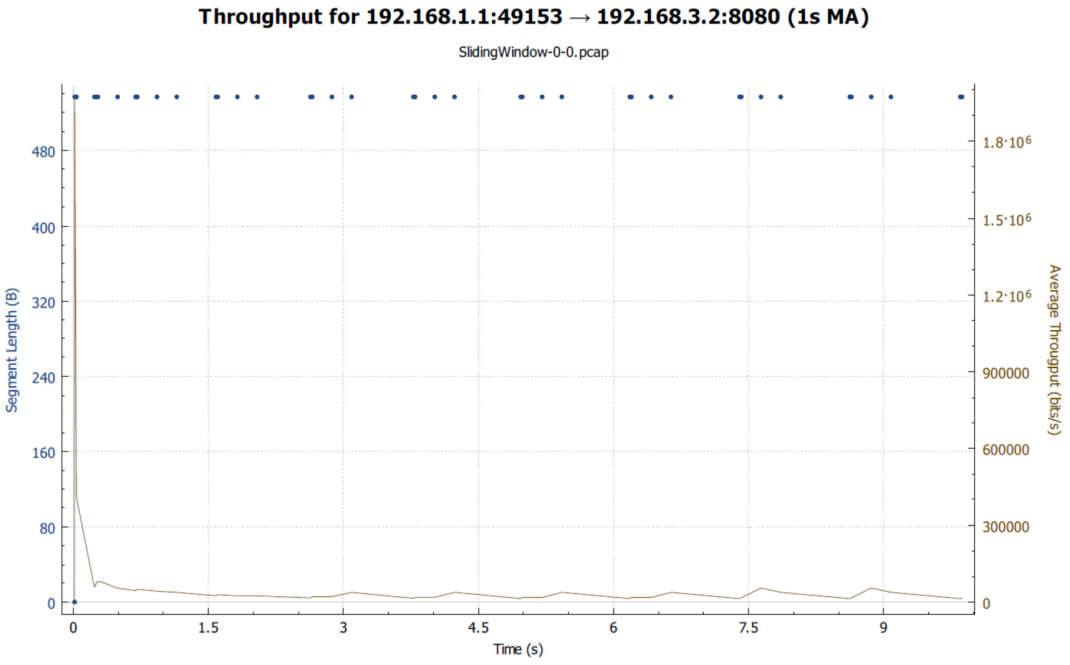
```
//Interfaces.Add(totNod);
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();


Config::SetDefault("ns3::TcpL4Protocol::SocketType",
StringValue("ns3::TcpSocketBase"));
// Config::SetDefault("ns3::TcpSocketBase::SetInitialCwnd", UintegerValue (2)); //edw
fainetai to congestion window tou TCP gia ton//algorithmo SlowStart
uint16_t Bport = 8080;
Address B_Sink_Addr(InetSocketAddress (Ipv4Address("192.168.3.2"), Bport));
PacketSinkHelper sinkA ("ns3::TcpSocketFactory", B_Sink_Addr);
ApplicationContainer sinkAppA = sinkA.Install(B);
sinkAppA.Start (Seconds (0));
sinkAppA.Stop(Seconds(10));


Address sinkAddr(InetSocketAddress(Ipv4Address("192.168.3.2"), Bport));


BulkSendHelper sourceAhelper ("ns3::TcpSocketFactory", sinkAddr);
sourceAhelper.SetAttribute("Remote", AddressValue(InetSocketAddress (Ipv4Address
("192.168.3.2"), Bport)));
//sourceAhelper.SetAttribute ("InitialCwnd", UintegerValue(3));
sourceAhelper.SetAttribute ("MaxBytes", UintegerValue(30000));
sourceAhelper.SetAttribute ("SendSize", UintegerValue (3000));
ApplicationContainer sourceAppsA = sourceAhelper.Install (A);
sourceAppsA.Start (Seconds (0));
sourceAppsA.Stop (Seconds (10));

//FILE * fileNameRoot1, fileNameRoot2, fileNameRoot3;
//AsciiTraceHelper ascii;
```

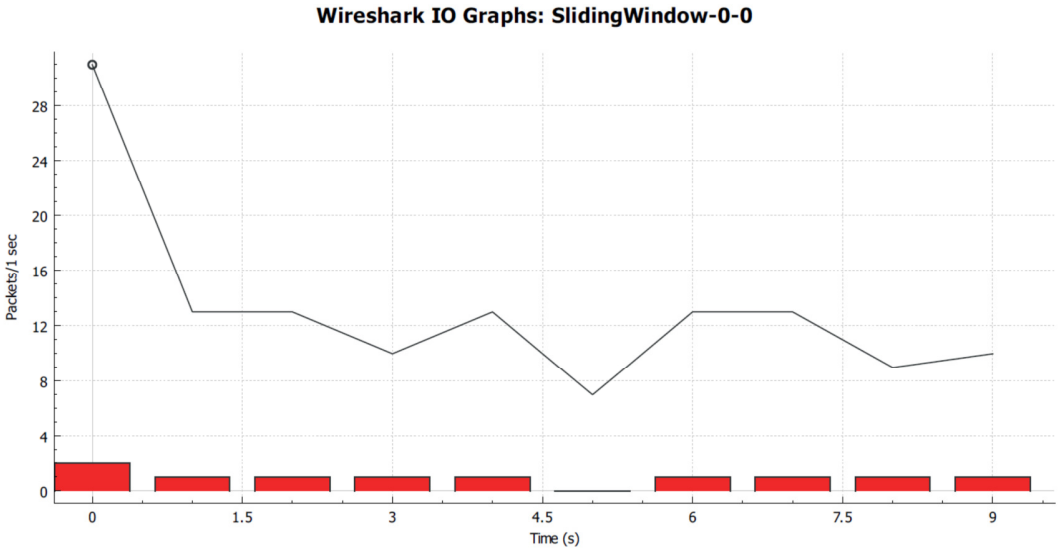
```
//std::string tfname1 = fileNameRoot1 + ".tr";
//std::string tfname2 = fileNameRoot2 + ".tr";
//std::string tfname3 = fileNameRoot3 + ".tr";
//AI.EnableAsciiAll (ascii.CreateFileStream (tfname1));
//IJ.EnableAsciiAll (ascii.CreateFileStream (tfname2));
//JB.EnableAsciiAll (ascii.CreateFileStream (tfname3));
//Simulator::Schedule(Seconds(0.01),&TraceCwnd);
AI.EnablePcapAll ("SlidingWindow");
IJ.EnablePcapAll ("SlidingWindow");
JB.EnablePcapAll ("SlidingWindow");
AsciiTraceHelper ascii;
AI.EnableAsciiAll (ascii.CreateFileStream ("SlidWindA.tr"));
IJ.EnableAsciiAll (ascii.CreateFileStream ("SlidWindB.tr"));
JB.EnableAsciiAll (ascii.CreateFileStream ("SlidWindC.tr"));

Simulator::Stop (Seconds (10.0));
Simulator::Run ();
Simulator::Destroy ();
return 0;
```

ΠΑΡΑΡΤΗΜΑ Β

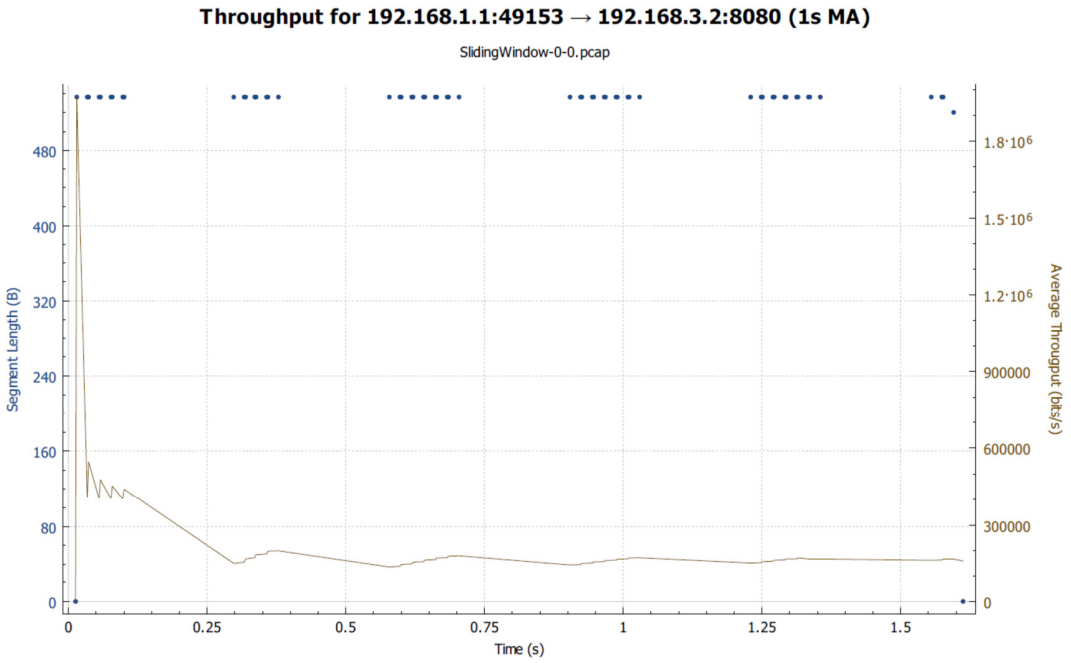
Παρακάτω παρατίθενται τα αποτελέσματα(μετρική throughput, tcp errors) που προέκυψαν από την υλοποίηση του πρωτοκόλλου ολισθαίνοντος παραθύρου ανάλογα με το μέγεθος του παραθύρου:

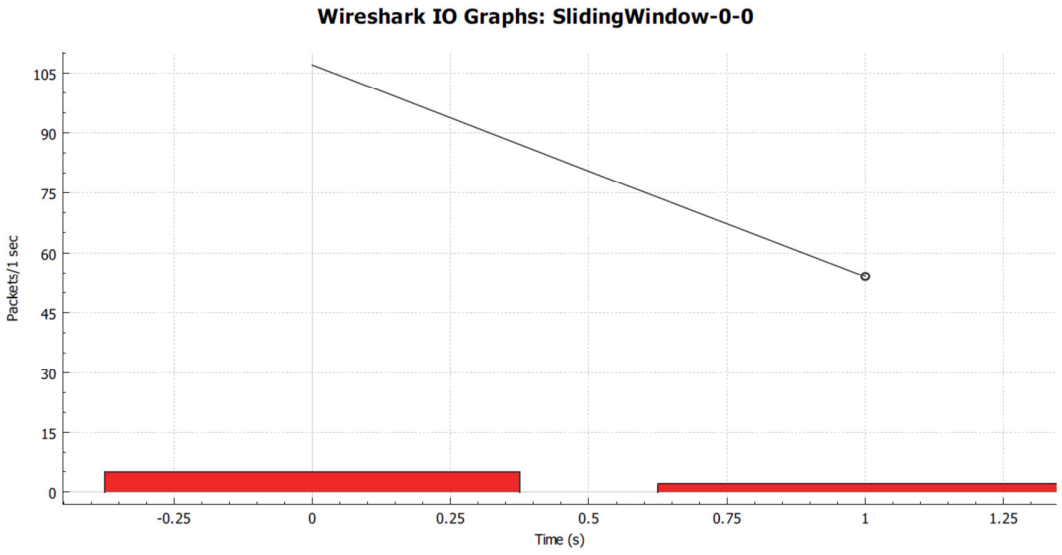
<u>Window Size 2</u>:



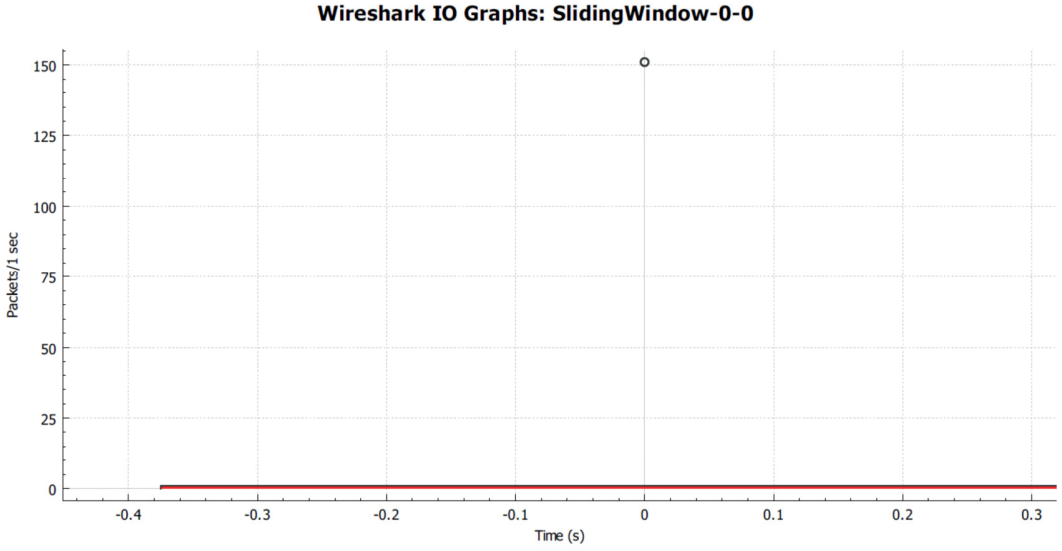


Window Size 4:

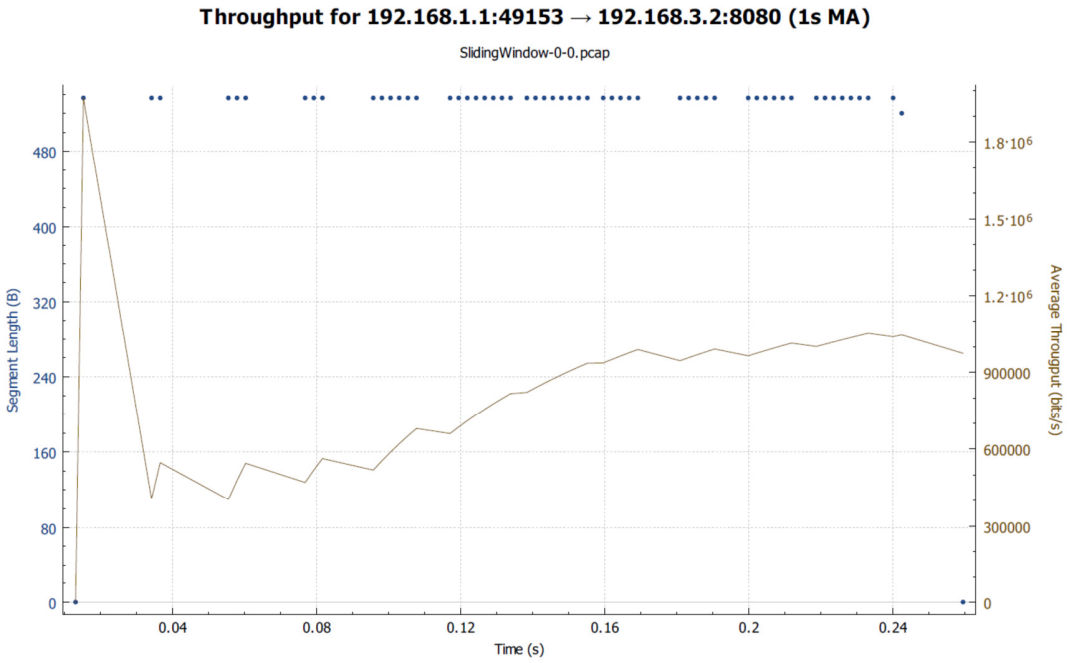


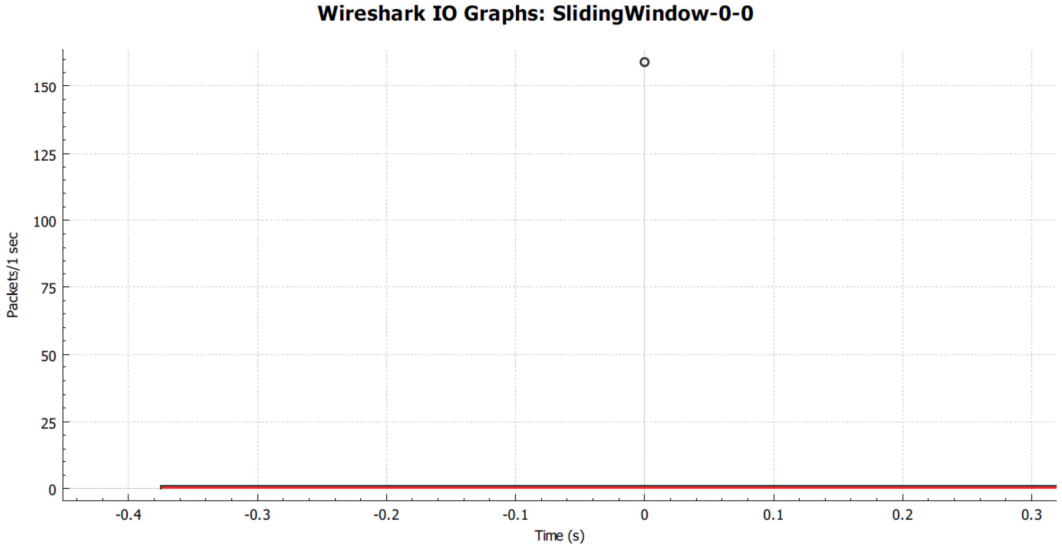


Window Size 5:

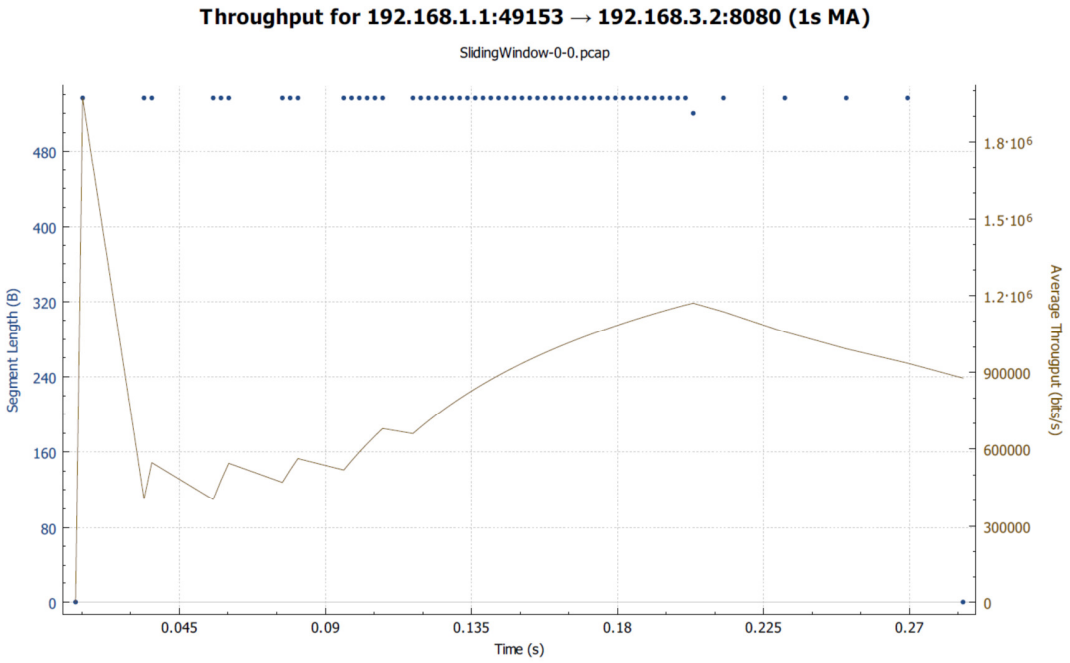


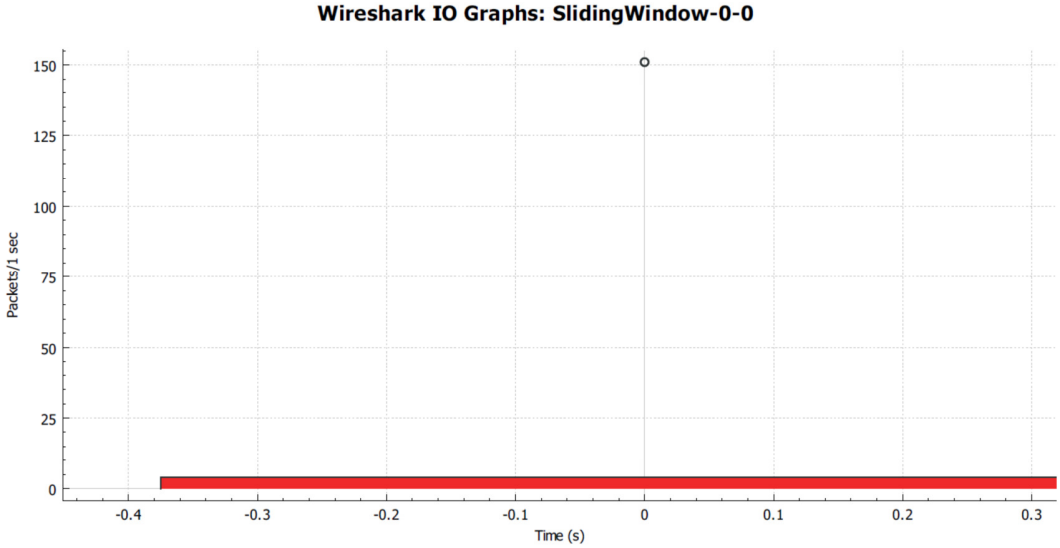
Window Size 8:





Window Size 16:





Window Size 24: