ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ

ΘΕΣΣΑΛΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΚΜΑΘΗΣΗ ΠΡΟΣΟΜΟΙΩΤΗ NS/3 ΓΙΑ ΤΑ ΑΣΥΡΜΑΤΑ ΤΟΠΙΚΑ ΔΙΚΤΥΑ

ΣΥΜΦΩΝΑ ΜΕ ΤΟ ΠΡΩΤΟΚΟΛΛΟ IEEE 802.11 b.

Μαριάδου Βικτώρια **Α.Μ. 03/2272**

5/3/2012

Περίληψη

Η ασύρματη σύνδεση με ένα δίκτυο και η ταυτόχρονη επικοινωνία με απομακρυσμένους κόμβους χωρίς προβλήματα είναι άμεση ανάγκη για τα σημερινά δίκτυα.

Καθώς η τεχνολογία των δικτύων αναπτύσσεται με ταχύτατους ρυθμούς, είναι επιτακτική η ανάγκη κατασκευής υψηλού επιπέδου λογισμικού για την υποστήριξή τους.

Στην εργασία αυτή αρχικός μας σκοπός είναι να μελετήσουμε και να κατανοήσουμε τη λειτουργία των ασύρματων τοπικών δικτύων καθώς και πολλά χαρακτηριστικά τους.

Έπειτα αναλύουμε μία σειρά εργαλείων λογισμοκού που έχουν αναπτυχθεί για να χρησιμοποιηθούν στη δικτυακή προσομοίωση και αναφέρουμε τις παραμέτρους, τρόπο χρήσης, πλεονεκτήματα και μειονεκτήματά τους.

Στη συνέχεια δίνεται ιδιαίτερη έμφαση στον προσομοιωτή Network Simulator(NS-3), το εργαλείο με το οποίο έχει γίνει η προσομοίωση και υλοποίηση του δικτύου που εξετάζουμε στο τελευταίο κομμάτι της πτυχιακής εργασίας.

Abstract

The wireless connection to a network and the simultaneous communication with remote nodes without problems constitute an immediate necessity for today's networks.

As network technology develops rapidly, there is an urgent need to build high quality software for their support.

In this paper our primary purpose is to study and understand the operation of wireless LANs as well as many of their features.

Subsequently, we will analyze a series of software tools developed to be used in network simulation and as well as reporting their parameters, usage, advantages and disadvantages.

In the last part of this dissertation, special emphasis will be placed on the Network Simulator NS-3, our tool for the modeling and implementation of the network under consideration.

Περιεχόμενα

Ευρετήριο εικόνων

Κεφάλαιο 1: Εισαγωγή

Τα περισσότερα τοπικά δίκτυα σε εταιρίες, πανεπιστήμια, καφετερίες παρέχουν ασύρματη σύνδεση με το διαδίκτυο. Πρωταρχική ανάγκη για τους κινητούς αυτούς κόμβους είναι η εξασφάλιση της συνεχούς επικοινωνίας με άλλους κόμβους.

Η ανάπτυξη και επικράτηση νέων συστημάτων και βελτιωμένων σχεδιασμών ανέκαθεν βασιζόταν και θα συνεχίσει να βασίζεται στην ικανότητα παροχής εκτιμήσεων της απόδοσης τους μέσω αναλυτικών μεθόδων ή προσομοίωσης.

Έτσι και στα ασύρματα δίκτυα η μοντελοποίηση (modeling) και η προσομοίωση μπορούν να χρησιμοποιηθούν για την αξιολόγηση σχεδιασμών που απευθύνονται σε αυτά.

Μέχρι σήμερα, η μαθηματική μοντελοποίηση και ανάλυση έχει προσφέρει σημαντικές γνώσεις για την κατανόηση των δυνατοτήτων και των προβλημάτων των ασύρματων δικτύων.

Παρόλα αυτά, οι αναλυτικές μέθοδοι πολλές φορές δεν είναι αρκετά γενικές ή, στο άλλο άκρο, αρκετά λεπτομερείς για την αξιολόγηση και σύγκριση των διαφόρων ασύρματων συστημάτων και των υπηρεσιών τους.

Κατά συνέπεια σε κάποιες περιπτώσεις η χρήση προσομοίωσης είναι ένας τρόπος για τους επιστήμονες να ελέγξουν τις επιδόσεις των διαφόρων λύσεων, ώστε να λάβουν κρίσιμες σχεδιαστικές αποφάσεις.

Η βασιζόμενη σε υπολογιστές, διακριτών γεγονότων προσομοίωση (discrete-event simulation) είναι μια από τις πιο ευέλικτες μεθόδους αξιολόγησης των επιδόσεων πολύπλοκων συστημάτων.

Ο στόχος μιας βασιζόμενης σε προσομοίωση μελέτης είναι η κατασκευή ενός προσομοιωτή που μιμείται την συμπεριφορά και τις μεταβάσεις μεταξύ των διαφόρων καταστάσεων του πραγματικού δικτύου.

Στο επόμενο κεφάλαιο θα αναλύσουμε το πρότυπο IEEE 802.11 που υποστηρίζει ένα ασύρματο δίκτυο καθώς και βασικά χαρακτηριστικά του. Σκοπός μας είναι να κατανοήσουμε τον τρόπο λειτουργίας των ασύρματων δικτύων.

Στη συνέχεια, θα κάνουμε μια εισαγωγή στην δικτυακή προσομοίωση. Θα ορίσουμε την έννοια της δικτυακής προσομοίωσης και θα περιγράψουμε διάφορα εργαλεία που χρησιμοποιούνται στην προσομοίωση δικτύων. Μέσω αυτής της διαδικασίας, θα εξοικειωθούμε με το αντικείμενο της εργασίας και θα αναδείξουμε την χρησιμότητα της προσομοίωσης.

Στο τέταρτο κεφάλαιο, θα μελετήσουμε τα ενδότερα του δικτυακού προσομοιωτή Network Simulator (NS). Θα αναφερθούμε σε προηγούμενες εκδόσεις του και θα μιλήσουμε πιο αναλυτικά για τα ιδιαίτερα χαρακτηριστικά και την λειτουργία της τρέχουσας έκδοσης που είναι η έκδοση 3.

Στο πέμπτο κεφάλαιο, θα περιγράψουμε την υλοποίηση ενός δικτύου που σχεδιάσαμε και προσομοιώσαμε στον NS – 3.

Τέλος, θα συνοψίσουμε την μελέτη μας πάνω στην δικτυακή προσομοίωση και στην υλοποίηση μας.

Κεφάλαιο 2: Ασύρματα Τοπικά Δίκτυα

Εισαγωγή

Λίγο πολύ όλοι έχουμε χρησιμοποιήσει έναν υπολογιστή για να πλοηγηθούμε στο διαδίκτυο(Internet). Ο υπολογιστής αυτός ανήκει σε ένα δίκτυο. Το δίκτυο αυτό μπορεί να είναι κατασκευασμένο είτε μέσω της ενσύρματης τεχνολογίας είτε της ασύρματης.

Σε ένα ενσύρματο δίκτυο(Local Area Network) οι συσκευές συνδέονται χρησιμοποιώντας συνεστραμμένα ζεύγη καλωδίων ή ομοαξονικά καλώδια και προσαρμογείς δικτύου.

Η χρήση των ασύρματων τοπικών δικτύων(Wireless Local Area Network) τα τελευταία χρόνια άρχισε να αυξάνεται με πολύ γρήγορους ρυθμούς, λόγω του χαμηλού κόστους υλοποίησής τους καθώς και της αυξημένης ανάπτυξης προτύπων. Μία σημαντική διαφορά συγκριτικά με τα ενσύρματα δίκτυα είναι πως δεν χρησιμοποιείται κάποιο καλώδιο ως μέσο μετάδοσης αλλά ηλεκτρομαγνητικά κύματα τα οποία μεταφέρουν τα δεδομένα. Οι ασύρματες συσκευές επικοινωνίας συνδέονται μεταξύ τους μέσω μίας συσκευής, η οποία ονομάζεται Access Point. Η συσκευή αυτή συνήθως συνδέεται με ένα ενσύρματο δίκτυο και είναι υπεύθυνη για τη λήψη, προσωρινή αποθήκευση και μετάδοση των δεδομένων.

Πρωτόκολλο Ασύρματης Δικτύωσης IEEE 802.11

Το πρωτόκολλο που χρησιμοποιείται για τα ασύρματα τοπικά δίκτυα είναι το IEEE 802.11(Institute of Electrical and Electronics Engineers). Γενικά το πρωτόκολλο αυτό ορίζει:

A. Το φυσικό στρώμα(Physical Layer).

   Το φυσικό επίπεδο(PHY) είναι το πρώτο και χαμηλότερο επίπεδο από το σύνολο των επτά επιπέδων του μοντέλου αναφοράς του OSI(Open System Interconnection). Παρέχει τα μέσα του υλικού και καθορίζει μεθόδους και διαδικασίες για την αποστολή και λήψη των δεδομένων μεταξύ των κόμβων. Στον πομπό, οι πληροφορίες που προέρχονται από τα υψηλότερα στρώματα μαζί με τις κεφαλίδες του PHY που προστίθενται, σχηματίζονται σε πακέτα έτοιμα για εκπομπή. Στην πλευρά του δέκτη, το φυσικό επίπεδο είναι υπεύθυνο για το σωστό διαχωρισμό των κεφαλίδων από τις πληροφορίες των ανώτερων στρωμάτων. Λόγω των διαφόρων διαθέσιμων τεχνολογιών υλικού με διαφορετικά χαρακτηριστικά, το PHY επίπεδο αποτελεί ένα από τα πιο περίπλοκα στρώματα του μοντέλου.

B. Το υποστρώμα MAC(Medium Access Control).

   Το υπόστρωμα MAC του στρώματος ζεύξης δεδομένων(Data Link Layer), γνωστό και ως υποεπίπεδο ελέγχου πρόσβασης του μέσου, είναι το δεύτερο χαμηλότερο επίπεδο του OSI. Είναι υπεύθυνο για την αξιόπιστη και αποτελεσματική πρόσβαση στο μέσο όλων των σταθμών του δικτύου. Οι βασικές προδιαγραφές του πρωτοκόλλου IEEE 802.11 στο MAC καθορίζουν δύο τρόπους πρόσβασης στο μέσο.

(a) DCF(Distributed Coordination Function) κατανεμημένη διαδικασία πρόσβασης στο μέσο.

(b) PCF(Point Coordination Function) κεντρική διαδικασία πρόσβασης στο μέσο.

Το 802.11 πρωτόκολλο περιλαμβάνει ακόμα δύο φυσικά στρώμματα. Το FHSS(Frequency Hopping Spread Spectrum) φυσικό επίπεδο και το DSSS(Direct Sequence Spread Spectrum) φυσικό επίπεδο.

Για λόγους λειτουργικότητας αλλά και τεχνολογικής εξέλιξης το πρωτόκολλο αυτό έχει επεκταθεί σε IEEE 802.11 a/ c/ e/ f/ g/ h/ i/ j/ n/ s κλπ. Στην συγκεκριμένη πτυχιακή θα ασχοληθούμε με το IEEE 802.11 b πρωτόκολλο.

Το πρωτόκολλο 802.11b είναι μία επέκταση του 802.11, το οποίο αναφέρεται συχνά ως Wi-Fi (Wireless Fidelity). Επικυρώθηκε από την επιτροπή του IEEE τον Ιούλιο του 1999. Αποτελείται από κάποιες συγκεκριμένες προδιαγραφές που το καθορίζουν. Για την μετάδοση των δεδομένων το πρωτόκολλο χρησιμοποιεί τη ζώνη συχντοτήτων από 2.4 έως 2.497 GHz. Σε ανοιχτό χώρο η εμβέλεια του πρωτοκόλλου φτάνει τα 300m. Σε περίπτωση που παρεμβάλλονται εμπόδια(τοίχοι, πόρτες, δέντρα) ανάμεσα στις Wi-Fi συσκευές, η εμβέλεια μειώνεται. Η μέθοδος κωδικοποίησης που έχει επιλεγεί είναι η Τεχνική Ευρέως Φάσματος Άμεσης Ακολουθίας(DSSS), γεγονός που του προσφέρει υψηλή διαμεταγωγή πακέτων καθώς η ταχύτητα φτάνει τα 11Mbps.

Επικοινωνία Κόμβων

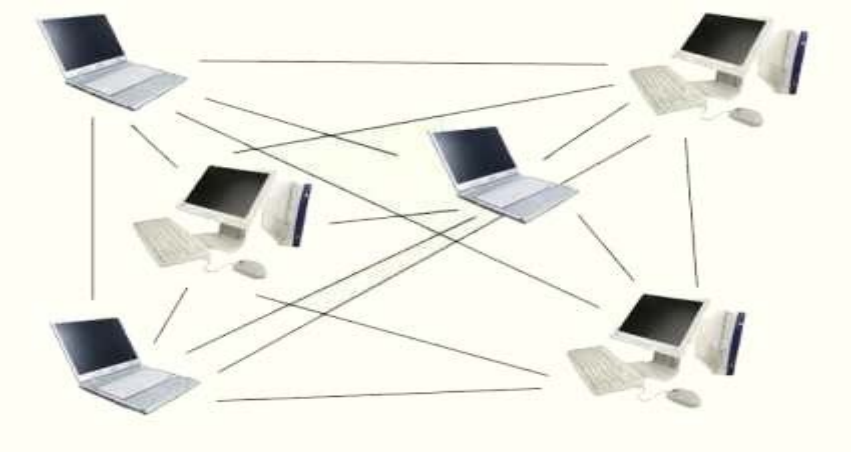
Σε ένα δίκτυο που χρησιμοποιεί το 802.11 πρωτόκολλο, οι κόμβοι μπορούν να επικοινωνούν μεταξύ τους με 3 διαφορετικούς τρόπους.

- Τον IBSS(Independent Basic Service Set) ή αλλιώς ad- hoc,
- Τον BSS(Basic Service Set) ή αλλιώς infrastructure και
- Τον ESS(Extended Service Set).

Τα δίκτυα αυτά χρησιμοποιούν ένα βασικό σύνολο υπηρεσιών BSS(Basic Service Set) το οποίο παρέχει μία περιοχή κάλυψης έτσι ώστε όλοι οι σταθμοί του BSS να παραμένουν πλήρως συνδεδεμένοι. Ένας σταθμός μπορεί να κινηθεί εντός του BSS, αλλά δεν μπορεί να επικοινωνήσει άμεσα με άλλους σταθμούς εάν φύγει από αυτό.

Ανεξάρτητα δίκτυα (IBSS)

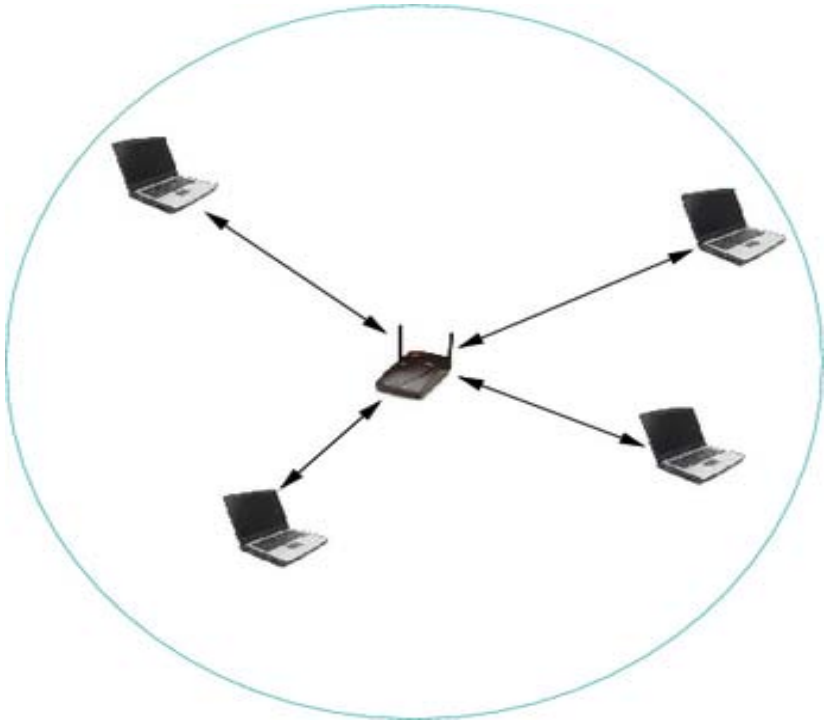
Ένα ad- hoc ή αλλιώς αυτόνομο δίκτυο συνήθως είναι ένα μικρό δίκτυο, 5 έως 50-80 σταθμοί, όπου όλοι επικοινωνούν μεταξύ τους άμεσα. Κάθε σταθμός θεωρείται ομότιμος(peer) και δεν είναι απαραίτητο κάποιος σταθμός να λειτουργεί ως κεντρικός υπολογιστής. Το IBSS δίκτυο αποτελείται από ασύρματους σταθμούς οι οποίοι επικοινωνούν μεταξύ τους άμεσα εντός μίας συγκεκριμένης απόστασης, χωρίς να είναι συνδεδεμένοι σε κάποιο ασύρματο δίκτυο ή κεντρικό σημείο.



Εικόνα 1: IBSS Δίκτυο

Δίκτυα Υποδομής (BSS)

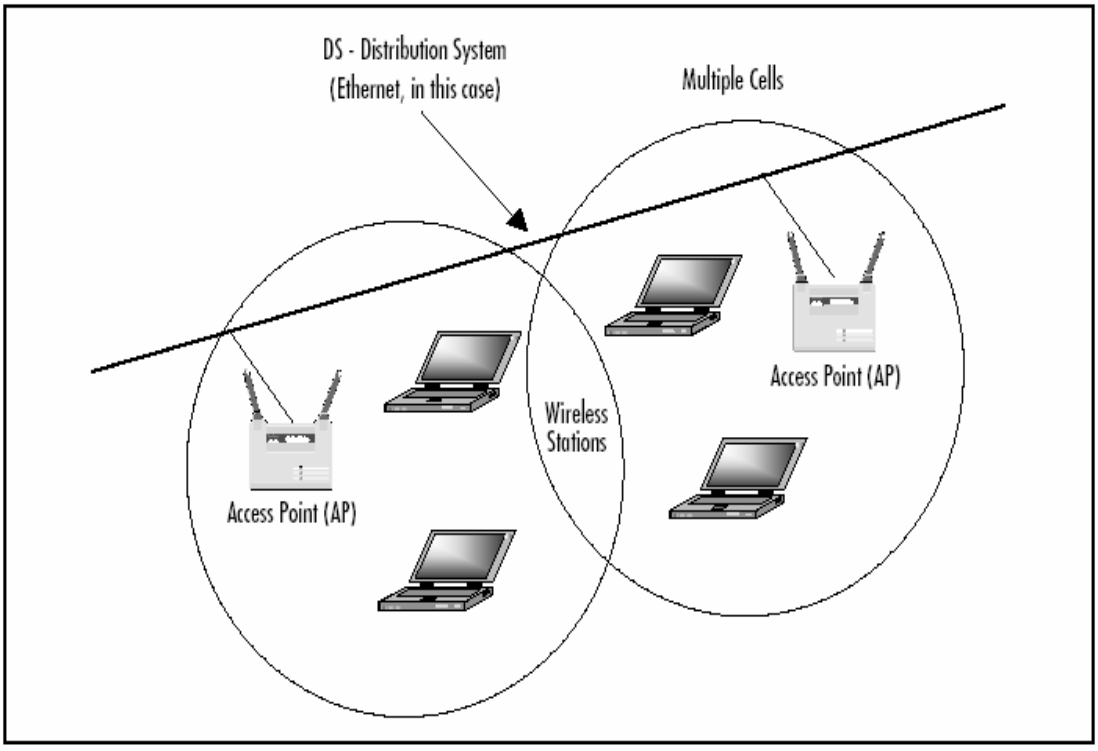
Σε αυτό το δίκτυο όλες οι συσκευές επικοινωνούν μέσω ενός κεντρικού διανομέα, που ονομάζεται Base Station ή αλλιώς Access Point, και δεν είναι απαραίτητο όλοι οι κόμβοι να μπορούν να βλέπουν όλους τους υπόλοιπους. Απαραίτητο όμως είναι να μπορούν όλοι οι κόμβοι να επικοινωνήσουν με το AP. Το τελευταίο αποτελείται από ένα συγκεκριμένο όνομα, το SSID, το οποίο το κάνει να ξεχωρίζει από άλλα AP που πιθανόν να υπάρχουν στον ίδιο χώρο. Για να συνδεθούμε με το AP τις περισσότερες φορές χρειάζεται να γνωρίζουμε το SSID του. Η διαφορά των σταθμών με το AP είναι στη χρήση κεραιών. Το AP χρησιμοποιεί πολυκατευθυντική κεραία, καθώς εκπέμπει κυκλικά το σήμα του και με αυτόν τον τρόπο έχουμε τη μέγιστη κάλυψη του χώρου, ενώ οι σταθμοί χρησιμοποιούν κατευθυντικές κεραίες για να πετύχουν σύνδεση με μακρινά APs.



Εικόνα 2: BSS Δίκτυο

Εκτεταμένη περιοχή υπηρεσιών(ESS)

Τέλος, ένα ESS δίκτυο αποτελείται από επικαλυπτόμενα BSS δίκτυα, όπου το κάθε ένα από αυτά διαθέτει ένα AP, τα οποία με τη σειρά τους συνδέονται μεταξύ τους μέσω ενός Συστήματος Διανομής(Distribution System DS). Το ΣΔ τις περισσότερες φορές είναι ένα Ethernet ενσύρματο τοπικό δίκτυο. Με την τοπολογία αυτή επιτυγχάνουμε τη δημιουργία μεγαλύτερων ασύρματων δικτύων που καλύπτουν μεγάλες περιοχές. Στην ουσία τα APs λειτουργούν σαν γέφυρες μεταξύ ασύρματου και ενσύρματου δικτύου και συνδέουν νοητά τους σταθμούς για ανταλλαγή πληροφοριών.



Εικόνα 3: ESS Δίκτυο

Εύρος Ζώνης

Το πρωτόκολλο IEEE 802.11b όρίζει την ταχύτητα σύνδεσης στα 11mbps, αν και λόγω των σφαλμάτων μετάδοσης των ασύρματων συνδέσεων υπάρχει η πιθανότητα αλλοίωσης των πακέτων. Επίσης όλες οι Wi-Fi συσκευές λειτουργούν σε half- duplex mode, καθώς έχουν έναν ραδιοφωνικό πομποδέκτη και ο τελευταίος μπορεί ή να ακούει το δίκτυο ή να στέλνει σε αυτό, αλλά όχι και τα δύο ταυτόχρονα.

Χρόνοι Αναμονής

Το φυσικό στρώμα ανάλογα με τη μέθοδο διαμόρφωσης ορίζει μία σταθερή χρονική διάρκεια που ονομάζεται χρονοθυρίδα(slot time). Για την FSSS μέθοδο ορίζεται στα 50 μs ενώ για την DSSS στα 20 μs. Χρονοθυρίδα ορίζεται ως η χρονική διάρκεια που χρειάζεται ένας σταθμός για να καταλάβει ότι ένας άλλος σταθμός έχει εκπέμψει.

Υπάρχουν διάφοροι χρόνοι αναμονής που χρησιμοποιούνται από τους αλγορίθμους DCF και PCF για τον έλεγχο πρόσβασης στο μέσο. Κάθε σταθμός όταν θελήσει να εκπέμψει δεδομένα πρέπει να περιμένει ένα χρονικό διάστημα και με το πέρασμα αυτού του διαστήματος να ελέγξει εάν το μέσο είναι κατειλλημένο από άλλη μετάδοση. Εάν δεν είναι τότε μπορεί να εκπέμψει.

Το 802.11 ορίζει τέσσερις χρόνους αναμονής.

1) short interframe space(SIFS)

   Το φυσικό στρώμα του 802.11b ορίζει τη σταθερή χρονική διάρκεια SIFS στα 10μs. Είναι ο ελάχιστος χρόνος αναμομής και χρησιμοποιείται για πλαίσια ελέγχου RTS/CTS και πλαίσια επιβεβαίωσης ACK.

2) priority interframe space(PIFS)

   Ο χρόνος αυτός υπολογίζεται ως ο χρόνος SIFS αυξημένος κατά ένα slot time. Χρησιμοποιείται από την PCF διαδικασία ή το AP με σκοπό την προτεραιότητα πρόσβασης στο μέσο σε σχέση με τους σταθμούς που χρησιμοποιούν την DCF διαδικασία.

3) distributed interframe space(DIFS)

   Υπολογίζεται ως ο χρόνος SIFS αυξημένος κατά δύο slot time. Χρησιμοποιείται από την DCF διαδικασία και είναι ο ελάχιστος χρόνος που αναμένουν οι σταθμοί ώστε να ξεκινήσουν να διεκδικούν το μέσον.

4) extended interframe space(EIFS)

   Όταν συμβαίνει ένα λάθος στην εκπομπή ενός πλαισίου χρησιμοποιείται αυτός ο χρόνος και υπολογίζεται ως EIFS=SIFS+ slot time+ διάρκεια ACK.

Μέθοδος Πρόσβασης στο μέσο(Access Method)

Όπως αναφέραμε σε προηγούμενη ενότητα το υπόστρωμα MAC του πρωτοκόλλου IEEE 802.11 υποστηρίζει δύο μεθόδους πρόσβασης στο μέσο, την DCF και την PCF. Όμως ο βασικός μηχανισμός που χρησιμοποιείται και υποστηρίζεται από όλα τα APs είναι ο DCF.

Όπως συμβαίνει και σε ένα Ethernet δίκτυο, έτσι και στα ασύρματα δίκτυα του IEEE 802.11, χρησιμοποιείται η μέθοδος πολλαπλής πρόσβασης με ανίχνευση φέροντος(CSMA -> Carrier Sense Multiple Access) για την προσπέλαση του ασύρματου μέσου. Η διαφορά όμως είναι πως χρησιμοποιείται η τεχνική αποφυγής συγκρούσεων(CA -> Collision Avoidance) αντί αυτής της ανίχνευσης συγκρούσεων(CD -> Cillision Detection) και αυτό λόγω του διαφορετικού μέσου που χρησιμοποιούν τα δύο αυτά δίκτυα.

CSMA/CA

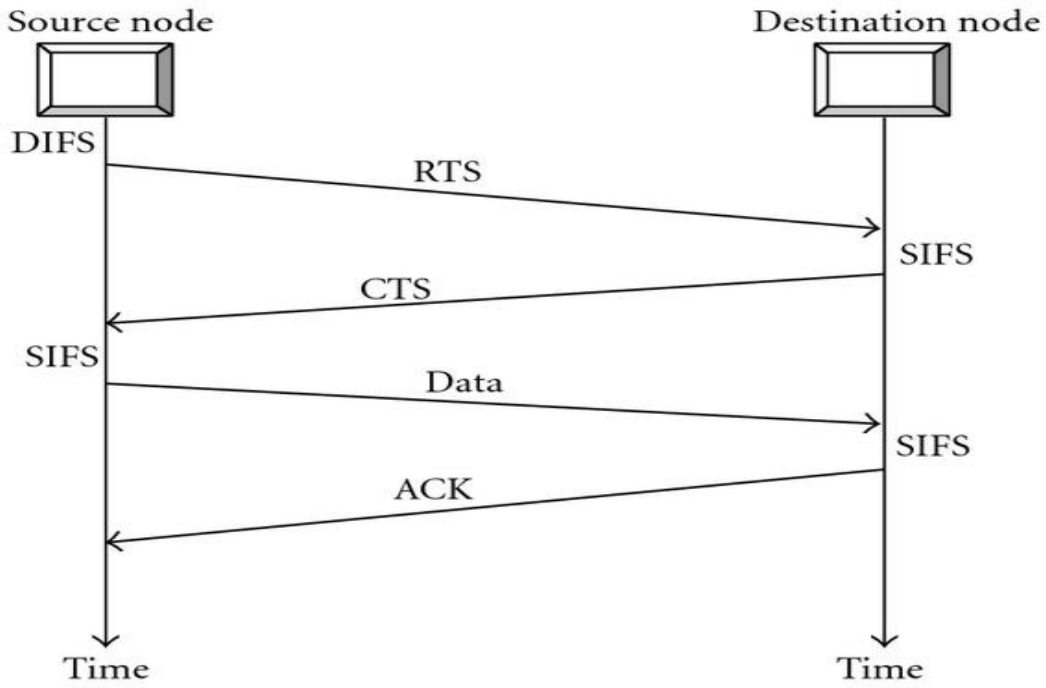
Η μέθοδος πολλαπλής πρόσβασης που έχει επιλεγεί από το 802.11 με την τεχνική αποφυγής συγκρούσεων λειτουργεί με τον εξής τρόπο.

Όταν ένας σταθμός θελήσει να εκπέμψει σε ένα κανάλι, ελέγχει εάν αυτό είναι κατειλημμένο, δηλ. εάν υπάρχει εκπομπή στο μέσο. Εάν είναι, τότε υποχωρεί. Εάν όμως είναι ελεύθερο για το χρονικό διάστημα DIFS, τότε ο σταθμός μπορεί να ξεκινήσει την εκπομπή. Ο λήπτης με τη σειρά του όταν λάβει το μήνυμα, θα στείλει ένα πακέτο επιβεβαίωσης της λήψης, το οποίο ονομάζεται ACK. Εάν ο αποστολέας λάβει το ACK πακέτο, θα γνωρίζει πως δε συνέβει καμία σύκγρουση και πως τελικά το πακέτο του έφτασε στον δέκτη. Αντιθέτως, εάν δε λάβει ACK μήνυμα εντός ορισμένου χρονικού διαστήματος, τότε θα συνεχίσει να στέλνει το πακέτο του μέχρι να λάβει ACK ή θα σταματήσει μετά από συγκεκριμένο αριθμό επαναλήψεων.

Ενώ ένας σταθμός μεταδίδει ένα πακέτο, δεν μπορεί να ελέγχει το μέσο. Στα ενσύρματα δίκτυα αυτό είναι εφικτό και μόλις ένας σταθμός διαπιστώσει την ύπαρξη σύγκρουσης σταματάει την μετάδοση. Στα ασύρματα όμως δίκτυα αυτό δεν ισχύει. Για την αποφυγή λοιπόν των συγκρούσεων χρησιμοποιείται ο μηχανισμός RTS/CTS.

Request To Send/ Clear To Send

Ο αποστολέας αρχικά στέλνει ένα πακέτο ελέγχου(RTS) το οποίο περιλαμβάνει συγκεκριμένες πληροφορίες, όπως η προέλευση, ο προορισμός και η διάρκεια αποστολής του πακέτου. Στην ουσία ο αποστολέας με αυτό το αρχικό πλαίσιο, δεσμεύει το μέσο για όση χρονική διάρκεια απαιτείται και ενημερώνει τους υπόλοιπους σταθμούς μέσω του μετρητή NAV(Network Allocation Vector) στο πλαίσιο RTS. Ο δέκτης λαμβάνοντας το RTS απαντάει με ένα CTS, στον ελάχιστο χρόνο αναμονής SIFS. Τότε λοιπόν ο αποστολέας ξεκινάει τη διαδικασία στης μετάδοσης του πακέτου και περιμένει την λήψη επιβεβαίωσης ACK.



Εικόνα 4: Μέθοδος CTS/RTS

Σύμφωνα με το CSMA/CA έχουν οριστεί δύο τρόποι λειτουργίας, ένας αποκεντρωμένος μέσω του αλγορίθμου DCF(Distributed Coordination Function) καθώς και ένας με κεντρικό έλεγχο μέσω του αλγορίθμου PCF(Point Coordination Function) ο οποίος είναι επέκταση του DCF.

Αλγόριθμος DCF(Distributed Coordination Function)

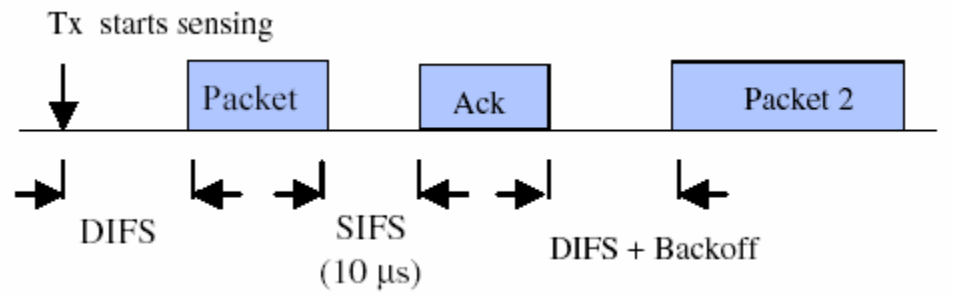
Ο αλγόριθμος DCF είναι, όπως αναφέραμε, αποκεντρωμένος και μπορεί να χρησιμοποιηθεί σε κάθε είδους ασύρματο δίκτυο. Κάθε σταθμός πριν ξεκινήσει την αποστολή πακέτων ακολουθεί κάποια συγκεκριμένα βήματα τα οποία καθορίζονται από τον αλγόριθμο.

Όταν ένας σταθμός θελήσει να εκπέμψει δεδομένα ελέγχει έαν το μέσο μετάδοσης είναι διαθέσιμο. Αν το μέσο δεν είναι διαθέσιμο, τότε ο σταθμός συνεχίζει να ελέγχει το μέσο έως ότου ελευθερωθεί. Αν το μέσο είναι διαθέσιμο, τότε ο σταθμός περιμένει χρονικό διάστημα DIFS και το ελέγχει ξανά.

Αν το μέσο είναι ξανά διαθέσιμο τότε ο σταθμός στέλνει το πακέτο. Ο παραλήπτης από τη μεριά του, επιβεβαιώνει τη σωστή παραλαβή του πακέτου αφού έχει περάσει χρονικό διάστημα SIFS.

Αν το μέσο δεν είναι διαθέσιμο, τότε ο σταθμός περιμένει μέχρι το μέσο να είναι ελεύθερο για χρόνο ίσο με DIFS. Για τον καθορισμό του επιπλέον χρόνου αναμονής, ο σταθμός ξεκινάει τη διαδικασία δυαδικής εκθετικής υποχώρησης(binary exponentioal backoff). Αυτό γίνεται επιλέγοντας τυχαία μία σχισμή του παραθύρου ανταγωνισμού(contention window). Με το τέλος αυτού του χρονικού διαστήματος, ο σταθμός είναι έτοιμος για την αποστολή του πακέτου. Αν η μετάδοση αποτύχει, τότε θεωρείται ότι έχει υπάρξει σύγκρουση(collision) και ο σταθμός καλείται να επιλέξει και πάλι τυχαία μία σχισμή του παραθύρου ανταγωνισμού και να προσπαθήσει να εκπέμψει ξανά. Αυτή η διαδικασία διακόπτεται όταν υπάρχει επιτυχής αποστολή του πακέτου ή απόρριψή του. Μία αποστολή θεωρείται επιτυχής εάν έχει ληφθεί σωστά και το αντίστοιχο ACK από τον παραλήπτη.

Η διαδικασία πρόσβασης στο μέσο με τη χρήση του αλγορίθμου DCF, είναι πολύ σημαντική για την απόκτηση του ελέγχου του μέσου καθώς επίσης και την αποφυγή των συγκρούσεων.



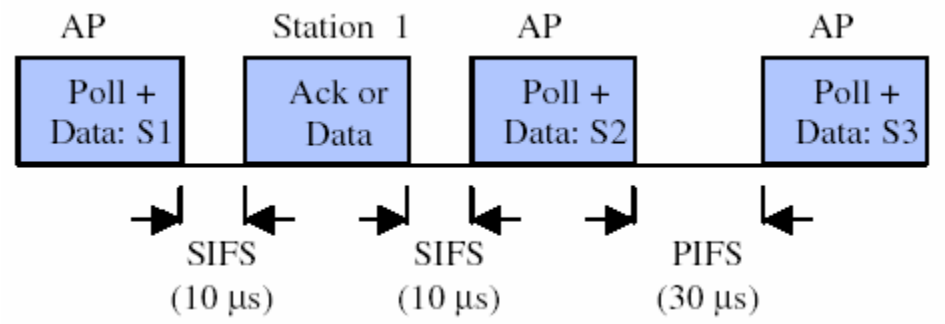
Εικόνα 5: Μέθοδος DCF

Αλγόριθμος PCF(Point Coordination Function)

Σύμφωνα με τον αλγόριθμο PCF, ένα σημείο πρόσβασης μέσα στο ασύρματο δίκτυο παίζει το ρόλο του κεντρικού διαχειριστή(Point Coordination, PC). Ο ρόλος του PC είναι να καθορίσει την περίοδο χωρίς ανταγωνισμό(Contention Free Period) που κάποιος σταθμός έχει στη διάθεσή του για τη μετάδοση των δεδομένων. Στην αρχή κάθε τέτοιας περιόδου χωρίς ανταγωνισμό, το AP στέλνει σε κάθε σταθμό ένα πλαίσιο συγχρονισμού(Beacon), το οποίο περιέχει τη μέγιστη διάρκεια της CFP. Στη συνέχεια ο χρόνος διαμοιράζεται σε θυρίδες και οι θυρίδες αυτές ανατίθενται στους σταθμούς. Τα πλαίσια από έναν κόμβο Χ σε έναν κόμβο Ψ μπορούν να μεταδοθούν από τον Χ στον Ψ κατά τη θυρίδα του Χ. Ή μπορούν να μεταδοθούν από τον Χ στο AP κατά τη θυρίδα του Χ και από το AP στον Ψ κατά τη θυρίδα του Ψ.

Με το που το AP έχει τον έλεγχο του μέσου, στέλνει σε κάθε έναν σταθμό ένα πλαίσιο ελέγχου(polling) δίνοντάς του την άδεια για μετάδοση. Το πλαίσιο αυτό δίνει το δικαίωμα σε κάθε σταθμό της μετάδοσης ενός μόνο πλαισίου.

Η χρήση αυτού του αλγορίθμου δεν είναι ευρεία και δεν υποστηρίζεται από τους κατασκευαστές. Αυτό διότι μπορεί να χρησιμοποιηθεί μόνο σε δίκτυα υποδομής καθώς απαιτεί κεντρικό έλεγχο από κάποιο AP και αποτελεί προαιρετικό μέρος του προτύπου 802.11



Εικόνα 6: Μέθοδος PCF

Εφαρμογές Ασύρματων Δικτύων

Το πρωτόκολλο 802.11 έχει πολλά πλεονεκτήματα που χάρη σε αυτά το ασύρματο δίκτυο είναι πλεόν το πιο εύχρηστο και διαδεδομένο σε διάφορους χώρους και τομείς.

Στο ν εργασιακό το μέα 'λύνει τα χέρια' όπως θα λέγαμε στην καθομιλουμένη. Οι εργαζόμενοι μπορούν να έχουν τους προσωπικούς τους φορητούς υπολογιστές, με όλα τα απαραίτητα αρχεία τους και να είναι σε συνεχή σύνδεση με το δίκτυο της εταιρείας αλλά και το διαδίκτυο. Έτσι λοιπόν οι εργαζόμενοι γίνονται πιο αποδοτικοί καθώς έχουν άμεση πρόσβαση σε απαραίτητες πληροφορίες και στον χώρο εργασίας τους αλλά και στο σπίτι τους.

Στον ιδιωτικό χώρο του καθενός, δηλαδή στο σπίτι, ένα ασύρματο δίκτυο μας απαλλάσει από την περίπλοκη εγκατάσταση ενός δικτύου με πολλά καλώδια. Μας επιτρέπει την περιήγηση στο διαδίκτυο, την παρακολούθηση βίντεο, την οπτική επικοινωνία μέσω διαφόρων εφαρμογών και όλα αυτά σε οποιοδήποτε σημείο του σπιτιού. Και αυτό μόνο με την ύπαρξη ενός ή περισσοτέρων AP.

Βέβαια δεν πρέπει να ξεχνάμε πως υπάρχουν φυσικά και κάποιοι κίνδυνοι που πρέπει να προσέχουμε και να αποφεύγουμε όταν χρησιμοποιούμε ένα ασύρματο δίκτυο, τους οποίους θα αναλύσουμε παρακάτω.

Προβλήματα του 802.11b

Ένα από τα πιο σημαντικά προβλήματα των ασύρματων δικτύων είναι κατά πόσο ασφαλή είναι. Αρκετές έρευνες πάνω στο συγκεκριμένο θέμα έδωσαν απαντήσεις στα περισσότερα ζητήματα που τέθησαν μετά την ύπαρξη αυτών των δικτύων.

Μία απλή και γρήγορη λύση που βρέθηκε για την επίτευξη της ασφάλειας ήταν η πιστοποίηση των χρηστών μέσω επιτρεπόμενων λιστών με MAC διευθύνσεις. Η MAC διεύθυνση, για να εξηγήσουμε, είναι ένας μοναδικός δεκαεξαδικός αριθμός που είναι γραμμένος στο υλικό κάθε δικτυακής συσκευής. Λειτουργούσε αρκετά απλά, δηλαδή κάθε AP κρατούσε μία λίστα με διευθύνσεις MAC, που ο διαχειριστής του δικτύου επέτρεπε να συνδεθούν με αυτό. Αν δεν ανήκε κάποια δεύθυνση στη λίστα, δεν μπορούσε να συνδεθεί. Το πρόβλημα είναι πως κάποιος εκτός λίστας ο οποίος όμως είχε αρκετά διακιώματα σε ένα unix- like λειτουργικό σύστημα, μπορεί να αλλάξει τη MAC διεύθυνσή του με αποτέλεσμα να καταφέρει τελικά να γίνει αποδεκτός και να συνδεθεί. Οι συγκεκριμένες επιθέσεις ονομάζονται mac spoofing attacks. Η πιθανή βέβαια ερώτηση είναι πώς μπορεί να γίνει αυτό. Εάν χρησιμοποιήσει εξειδικευμένο ανιχνευτικό υλικό μπορεί να φτιάξει μία λίστα με τις διευθύνσεις MAC που συνδέονται σε ένα AP και να διαλέξει οποιαδήποτε από αυτές τις διευθύνσεις, χωρίς να καταλάβει κανένας κάτι.

Υπήρξαν κάποιες προσπάθειες για την αποφυγή όσων προαναφέραμε αλλά δυστυχώς μέχρι σήμερα καμία από αυτές δεν ήταν επαρκής. Από τον ορισμό του WEP(wired equivalent privacy) μέχρι τη μελέτη νέων προτύπων, όπως το 802.11i.

Ένα άλλο μειονέκτημα είναι το πρόβλημα του κρυμμένου κόμβου. Το πρόβλημα αυτό έγγυται στον σχεδιασμό του πρωτοκόλλου και οφείλεται πιθανότατα στους στόχους που έθεσε η ίδια η ομάδα της IEEE για το WiFi σαν εναλλακτικό τρόπο δικτύωσης σε τοπικό επίπεδο.

Φανταστείτε ένα AP και πολλούς clients σε διάφορες τοποθεσίες, που όλοι βλέπουν το AP αλλά όχι ο κάθε ένας τον άλλον. Για να μπορέσετε να το φανταστείτε πιο εύκολα πάρτε για παράδειγμα ένα ενδοπανεπιστημιακό δίκτυο. Σύμφωνα με τον ορισμό του 802.11b, θα λειτουργούσε σε κλειστό περιβάλλον και έτσι θεωρούνταν λογικό ότι όλοι οι clients θα μπορούσαν να ακούν έναν client να στέλνει στο AP, χωρίς όμως να λαμβάνουν και αυτοί την πληροφορία. Να γνωρίζουν δηλαδή ότι απλά το κανάλι είναι κατηλλειμένο.

Η πιο σημαντική μέθοδος αποφυγής συγκρούσεων είναι η CSMA/CA(Carrier Sense Multiple Access with Colission Avoidance). Η λειτουργία αυτή πραγματοποιείται με την παρακολούθηση του καναλιού πριν την έναρξη της εκομπής. Εάν κάποιος σταθμός θελήσει να εκπέμψει αλλά κάποιος άλλος τυχαίνει να εκπέμπει, τότε ο πρώτος περιμένει μέχρι να ελευθερωθεί το κανάλι. Βέβαια για την καλή λειτουργία του δικτύου αυτό σημαίνει πως όλοι οι σταθμοί πελάτες πρέπει να ακούν όλους τους υπόλοιπους σταθμούς πελάτες. Εάν όμως δεν ισχύει αυτό τότε κάποιος σταθμός μπορεί εσφαλμένα να οδηγηθεί στο συμπέρασμα ότι το μέσο είναι ελεύθερο, ενώ στην ουσία να είναι απασχολημένο από κάποιον άλλον σταθμό. Το αποτέλεσμα αυτού είναι οι συνεχείς συγκρούσεις. Βέβαια το AP ευνοεί τον εκπομπό με το καλύτερο σήμα και αγνοεί το ασθενέστερο σαν θόρυβο.

Μία λύση στο συγκεκριμένο πρόβλημα είναι η χρήση ακόμα και στους σταθμούς πολυκετευθυντήριων κεραιών που εκπέμπουν κυκλικά το σήμα τους. Δυστυχώς όμως με αυτόν τον τρόπο μόνο ένας client μονοπωλεί όλο το εύρος ζώνης του AP, προκαλώντας έτσι συμφόρηση στη διακίνηση πακέτων των υπόλοιπων κόμβων. Το θετικό είναι πως υπάρχουν ειδικές συσκευές, οι οποίες εφαρμόζουν ένα είδος polling στο δίκτυο, αν και έχουν μεγάλο κόστος.

Μετρικές Απόδοσεις Πρωτοκόλλου

Για να εκτιμήσουμε ή ακόμα και να μετρήσουμε την απόδοση του πρωτοκόλλου 802.11 υπάρχουν κάποιες μετρικές οι οποίες είναι οι αξής:

i) Καθυστέρηση Πακέτων(Packet Delay) => Ορίζεται ως ο χρόνος που μεσολαβεί από τη στιγμή πο υ ένα πακέτο φτάνει στη πρώτη θέση της ουράς στο επίπεδο MAC μέχρι τη στιγμή που ένα πλαίσιο επιβεβαίωσης ACK λαμβάνεται στον αποστολέα.

ii) Ρυθμοαπόδοση(Throughput) => Ορίζεται ως το πλήθος των διαδικών ψηφίων ή πακέτων που μπορεί να δεχτεί και να μεταδώσει το δίκτυο στη μονάδα του χρόνου και μετράται σε bits/sec.

iii) Διακύμανση καθυστερήσεων(Jitter) => Ορίζεται ως η απόκλιση των τιμών καθυστέρησης από τη μέση τιμή.

iv) Πιθανότητα Σύγκρουσης(Collision Propability) => Ορίζεται ως η πιθανότητα να υποστεί ένα πακέτο σύγκρουση όταν εκπέμπεται στο μέσο.

v) Πιθανότητα Απόρριψης Πακέτων(Packet Drop Probability) => Ορίζεται ως η πιθανότητα να απορριφθεί ένα πακέτο, δηλαδή να υποστεί R+1 συγκρούσεις. Ένα πακέτο απορρίπτεται όταν ο αριθμός επαναληπτικών εκπομπών του έχει υπερβεί το μέγιστο όριο των προσπαθειών του. Εξαρτάται ευθέως από την πιθανότητα σύγκρουσης.

vi) Κατανομή πιθανότητας συγκρούσεων(delay distribution) => Ορίζεται ως η πιθανότητα να αποσταλεί με επιτυχία ένα πακέτο, αφού έχει υποστεί χρονική καθυστέρηση ίση με μία δοθείσα τιμή.

vii) Αθροιστική κατανομή πιθανότητας συγκρούσεων(cumulative delay distribution) => Ορίζεται ως η πιθανότητα να αποσταλεί με επιτυχία ένα πακέτο, αφού έχει υποστεί χρονική καθυστέρηση μικρότερης μιας δοθείσας τιμής.

viii) Χωριτικότητα Φωνητικών Συνδιαλέξεων(voice capacity) => Ορίζεται ως ο μέγιστος αριθμός συνομιλιών ανά ζεύγη που μπορεί να υποστηρίξει ένα ασύρματο δίκτυο με καθορισμένες προδιαγραφές στην ποιότητα φωνής.

Κεφάλαιο 3: Συστήματα δικτυακής προσομοίωσης

Στο κεφάλαιο αυτό θα κάνουμε μια εισαγωγή στην διαδικτυακή προσομοίωση. Θα ορίσουμε το συγκεκριμένο πεδίο και θα περιγράψουμε μια σειρά εργαλείων που προσομοιώνουν δίκτυα. Μέσω αυτής της διαδικασίας, θα εξοικειωθούμε με το αντικείμενο της εργασίας και θα αναδείξουμε την χρησιμότητα του.

Δικτυακή προσομοίωση

Η **προσομοίωση δικτύων** είναι μια μέθοδος με την οποία δημιουργούμε το μοντέλο ενός δικτύου μέσω λογισμικού και το τροφοδοτούμε με δεδομένα εισόδου. Παρατηρώντας τα αποτελέσματα αυτών των δεδομένων πάνω στο μοντέλο μπορούμε να οδηγηθούμε σε συμπεράσματα για την λειτουργία των πραγματικών δικτύων.

Δικτυακοί προσομοιωτές

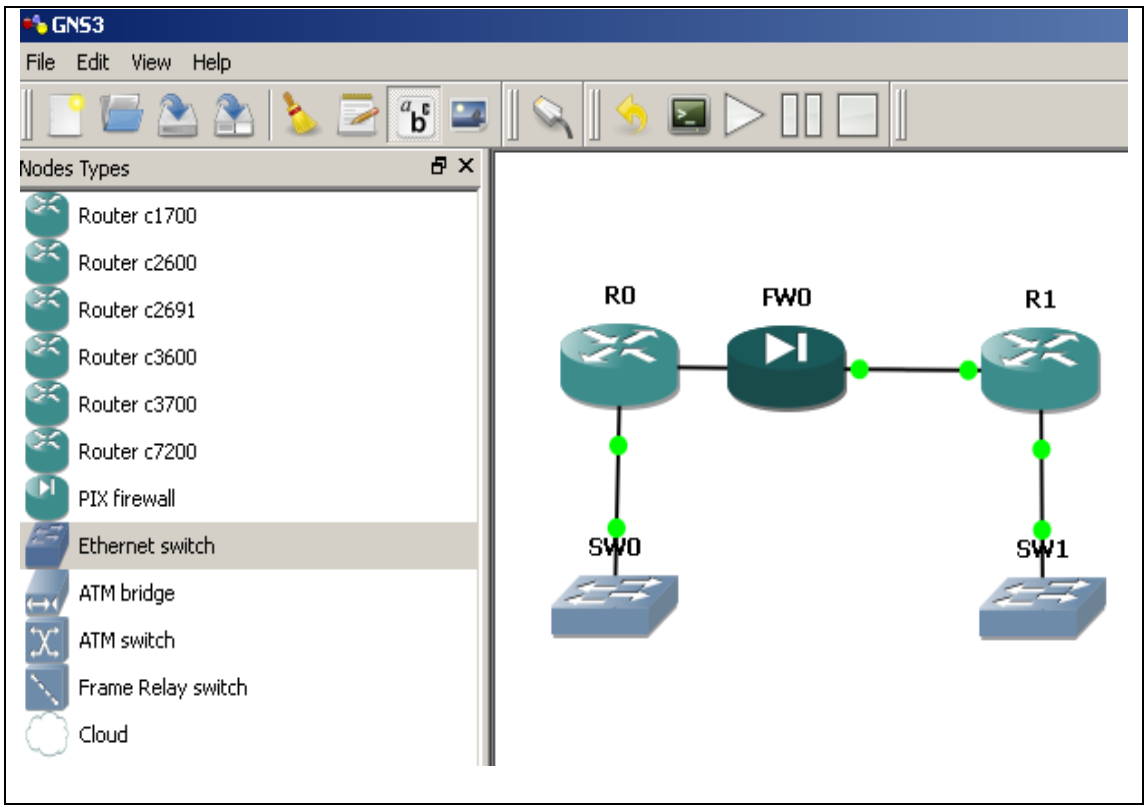
Καθώς η τεχνολογία των δικτύων αναπτύσσεται με ταχύτατους ρυθμούς, είναι επιτακτική η ανάγκη κατασκευής υψηλού επιπέδου λογισμικού για την υποστήριξή τους.

Η ανάπτυξη και επικράτηση νέων συστημάτων και βελτιωμένων σχεδιασμών ανέκαθεν βασιζόταν και θα συνεχίσει να βασίζεται στην ικανότητα παροχής εκτιμήσεων της απόδοσης τους μέσω αναλυτικών μεθόδων ή προσομοίωσης.

Έτσι και στα ασύρματα δίκτυα η μοντελοποίηση (modeling) και η προσομοίωση μπορούν να χρησιμοποιηθούν για την αξιολόγηση σχεδιασμών που απευθύνονται σε αυτά.

MIT's NETSIM (Network Simulator)

Ο NETSIM [1] είναι ένας εξομοιωτής για packet-switches δίκτυα που δουλεύει με βάση τα events. Δεν έχει δυνατότητα αναβάθμισης- εξέλιξης. Χρησιμοποιείται για όλα τα δίκτυα των οποίων τα στοιχεία ανταλλάσσουν μηνύματα. Παρέχει μόνο τα μέσα να προγραμματίζουμε τα events και εξασφαλίζει την επικοινωνία με το χρήστη.



Εικόνα 7

Το πακέτο αποτελείται από έναν event manager, I/O ρουτίνες, διάφορα δομικά εργαλεία όπως ουρές και λίστες που χρησιμοποιούνται για το σχεδιασμό στοιχείων και ένα toolkit.

Το toolkit είναι μια βιβλιοθήκη με συναρτήσεις της C που διευκολύνουν τη διαχείριση των στοιχείων και επιτρέπουν τη δημιουργία, αποθήκευση, φόρτωση και εμφάνιση των δικτυακών ρυθμίσεων.

Για να τρέξει μια εξομοίωση ο χρήστης πρέπει να γράψει καινούρια στοιχεία σε C, να αλλάξει μερικά αρχεία, να τα κάνει compile και να τα συνδέσει. Ύστερα δημιουργεί την εξομοίωση με το toolkit.

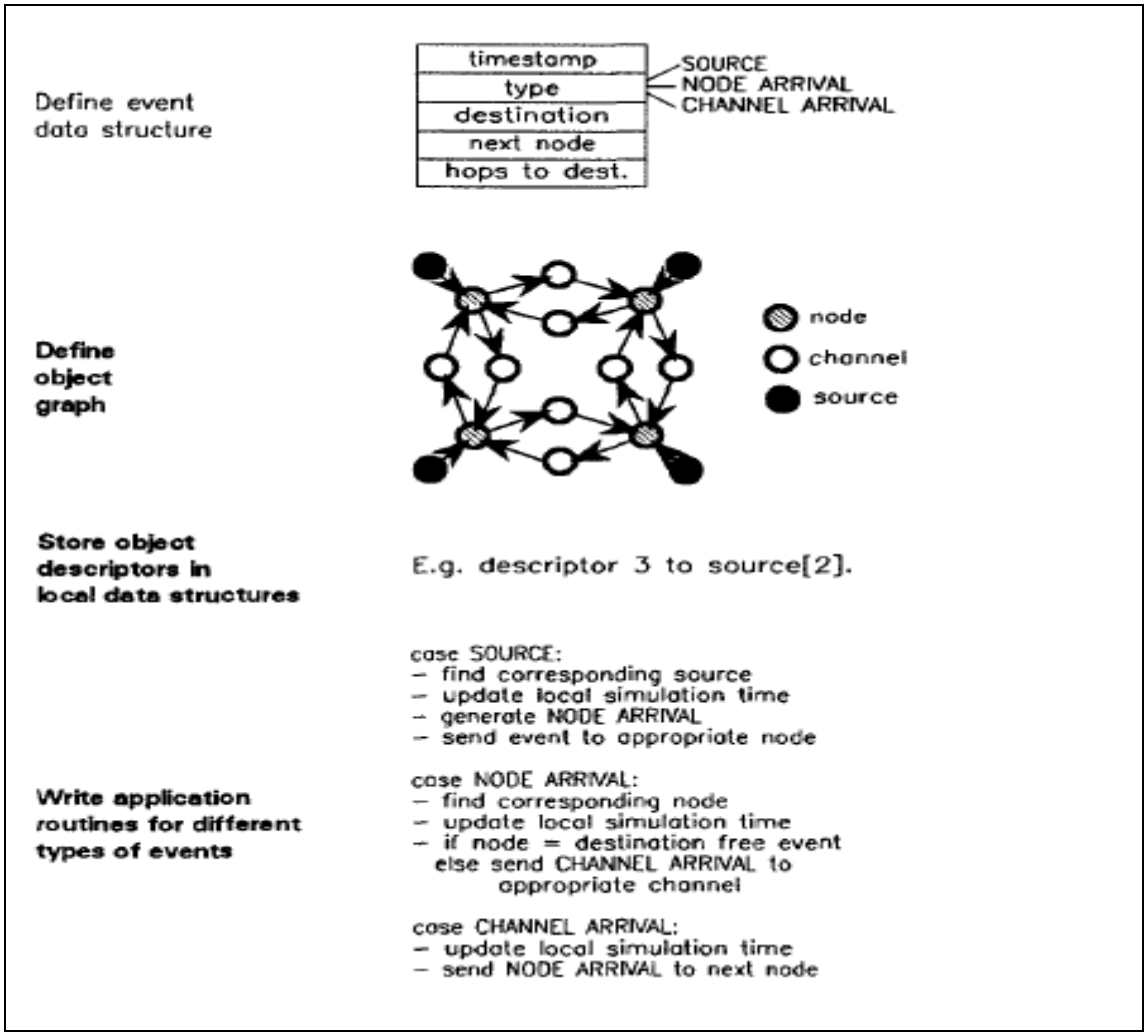
CPSIM

Είναι ένα εργαλείο παράλληλης γενικού σκοπού εξομοίωσης [2]. Χρησιμοποιείται δηλαδή για εξομοίωση παράλληλων διακριτών events και είναι κατάλληλος για μεγάλες εξομοιώσεις διακριτού event όπως σε δίκτυα υπολογιστών. Η έκδοση που διατίθεται περιορίζεται από 256 αντικείμενα εξομοίωσης.

Υπάρχουν δύο είδη εξομοίωσης: η αισιόδοξη και η συντηρητική. Ο CPsim υιοθετεί το συντηρητικό κανόνα. Στην αισιόδοξη προσέγγιση, τα events δρομολογούνται σε κάθε αντικείμενο από τη στιγμή που γίνεται διαθέσιμο, ενώ στη συντηρητική προσέγγιση γίνονται στο χρόνο εξομοίωσης t αν δεν αναμένονται άλλα events μέσα σ’ αυτό το χρόνο.

Στόχος των σχεδιαστών είναι η απόδοση. Δεν υπάρχει γραφικό περιβάλλον για το χρήστη και οι δημιουργοί ισχυρίζονται ότι δεν υπάρχει σημαντική διαφορά στον καθορισμό ενός μοντέλου με το να σέρνεις εικόνες στην οθόνη.

Ο εξομοιωτής σχεδιάστηκε για να δίνει στο χρήστη απόλυτη ελευθερία στο να δημιουργεί μοντέλα. Δεν υπάρχουν επίσης και built- in συναρτήσεις. Γι’ αυτό το λόγο έχει σχεδιαστεί για ειδικούς στην εξομοίωση χρήστες.



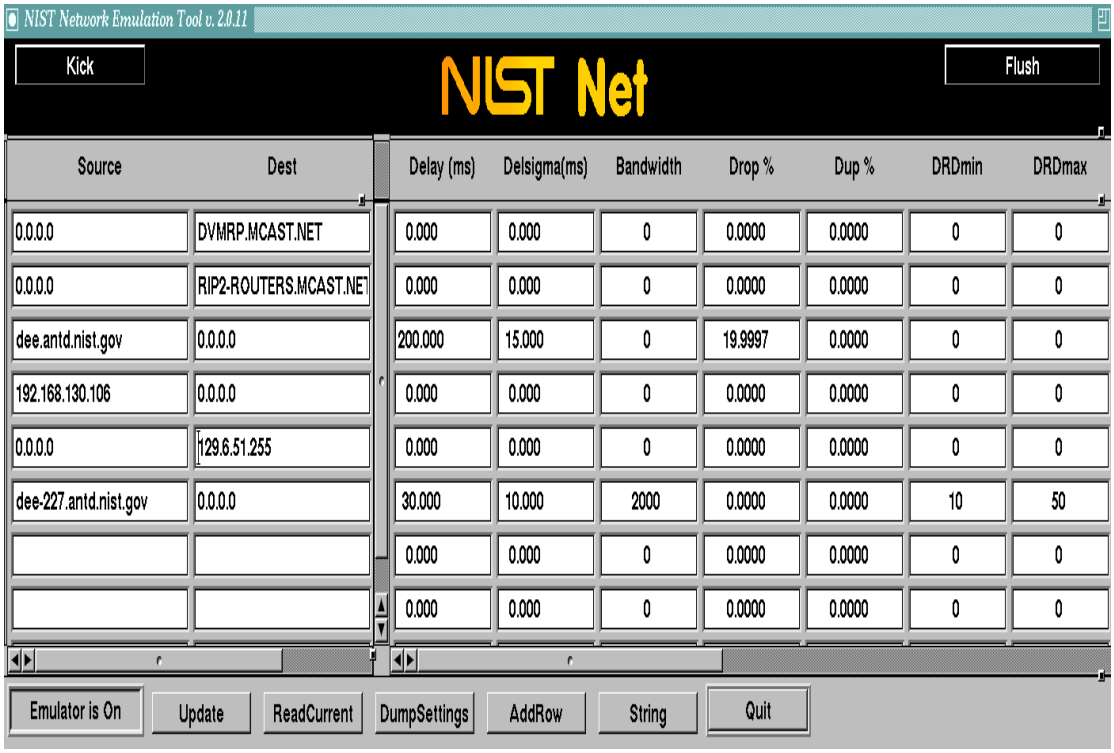
Εικόνα 8

NIST (National Institute of Standards and Technology)

Ο NIST [3] είναι ένας εξομοιωτής ATM δικτύων για την μελέτη και την εκτίμηση της απόδοσης των ATM δικτύων και στηρίζεται στον NETSIM του MIT. Τα στοιχεία του είναι τα εξής:

Χρήση: Έχει δύο βασικές χρήσεις: τον σχεδιασμό ATM δικτύων και την ανάλυση επίδοσης του ATM πρωτοκόλλου. Σαν σχεδιαστικό εργαλείο, ο εξομοιωτής τρέχει με διάφορες ρυθμίσεις δικτύου και φόρτο κίνησης και εξάγει στατιστικά στοιχεία όπως είναι τα link utilization, throughput κτλ.

Σαν εργαλείο ανάλυσης πρωτοκόλλου, χρησιμοποιείται για τη μελέτη της συνολικής επίδρασης στο σύστημα ενός συγκεκριμένου πρωτοκόλλου, όπως είναι οι μηχανισμοί για δίκαιη δέσμευση bandwidth, το protocol overhead, το bandwidth utilization, και η αποτελεσματικότητα των μηχανισμών ελέγχου κίνησης. Οι σχεδιαστές υποστηρίζουν ότι τα μοντέλα τμημάτων εξομοίωσης μπορούν εύκολα να αλλάξουν ή να προστεθούν.



Εικόνα 9

Τα πακέτα: Δεν υπάρχει βασικά καμιά σημαντική διαφορά με το NETSIM του MIT. Το εργαλείο είναι γι' αυτό το λόγο γραμμένο στη C και παρέχει ένα περιβάλλον μοντελοποίησης αλληλεπίδρασης με ένα GUI που χρησιμοποιεί το X παραθυρικό σύστημα που τρέχει σε πλατφόρμα UNIX. Αυτό εμφανίζει την τοπολογία του δικτύου και τις παραμέτρους που συνδέονται με τα επιμέρους τμήματα (components) που εξομοιώνονται.

Παράμετροι εισόδου και εξόδου των επιμέρους τμημάτων εμφανίζονται σε information windows (παράθυρα πληροφοριών), ενώ η δραστηριότητα του δικτύου καταγράφεται σε παράθυρα μετρικών (meter windows-binary meters, bar graph, histogram, κτλ).

Ο εξομοιωτής επίσης έχει έναν διαχειριστή γεγονότων (event manager), I/O ρουτίνες και άλλα εργαλεία για να φτιάχνει τα επιμέρους τμήματα. Ωστόσο ο χρήστης μπορεί να φτιάξει απευθείας με ένα GUI μια νέα τοπολογία δικτύου.

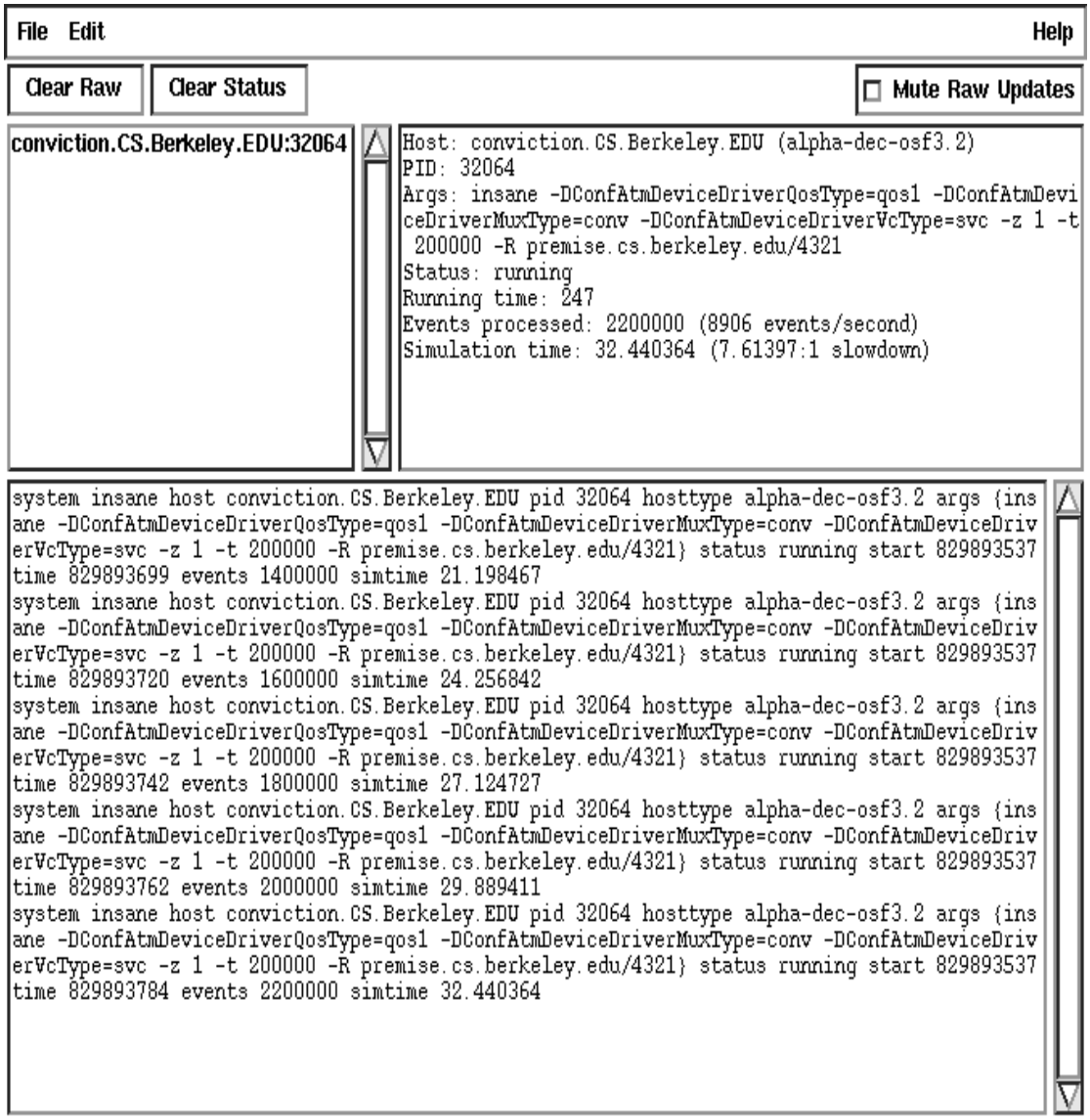
Υλοποίηση του εξομοιωτή: Τα περισσότερα αρχεία είναι περίπου ίδια με τον NETSIM, όπως και ο scheduler, αλλά επιμέρους τμήματα έχουν αντικατασταθεί για να ταιριάζουν περισσότερο με τις ATM απαιτήσεις. Τα επιμέρους τμήματα στέλνουν μηνύματα το ένα στο άλλο και αναπαρίστανται από μια ρουτίνα δράσης και μια δομή δεδομένων. Κάθε component έχει για κάθε κατηγορία έναν τύπο και ειδικά μια class μπορεί να περιέχει διάφορους τύπους από components.

Στον NIST διαθέσιμες κλάσεις από components είναι: Φυσικοί Σύνδεσμοι, ATM switches, Broadband Terminal Equipment (B-TE) και ATM αιτήσεις. Ακριβέστερα, οι αιτήσεις ATM είναι γεννήτριες ροής (traffic generators) τύπου TCP/IP, CBR, VBR (batch ή poisson) και ABR (constant, batch ή poisson).

Σαν παράδειγμα παραμέτρων, οι παράμετροι εισόδου από ένα switch που παρέχονται από το χρήστη μπορεί να είναι η καθυστέρηση (delay) αποστολής ενός cell, ο slot time, το μέγεθος της ουράς εξόδου, κτλ., ενώ οι παράμετροι εξόδου που παρακολουθούνται από τον εξομοιωτή μπορεί να είναι ο αριθμός των cell που παραδίδονται ή απορρίπτονται, κτλ.

INSANE

Κατασκευάστηκε για την εκτίμηση απόδοσης διαφόρων IP over ATM κανόνων σε ετερογενή εσωτερικά δίκτυα [4]. Χρησιμοποιήθηκε για την εξομοίωση ευρείας περιοχής ATM κορμών. Να και τα αποτελέσματα προχωράνε off- line, τρέχουν μια σειρά από μεταδιαδικαστικά scripts στο αρχείο εξόδου για να αναλύσουν την απόδοση του δικτύου.



Εικόνα 10

Ο INSANE είναι ένας αντικειμενοστραφής, διακριτών events εξομοιωτής. Το βασικό μέρος και ορισμένα βασικά αντικείμενα έχουν γραφεί σε C++. Built-in αντικείμενα συμπεριλαμβάνουν ουρές διαφόρων τύπων και μερικά modules πρωτοκόλλων (IP, TCP, UDP) και εξάγουν ένα σύνολο εντολών που επιτρέπουν τη δημιουργία και τη διαχείριση νέων σύνθετων αντικειμένων όπως είναι ένα ATM switch χρησιμοποιώντας Tcl scripts που υλοποιήθηκαν από αρχεία ρυθμίσεων.

Μια βιβλιοθήκη από Tcl scripts παρέχεται μαζί με τον κώδικα. Εξομοίωση και προσαρμογή εκτελούνται γι’ αυτό το λόγο μαζί με τα Tcl scripts, έχοντας ως πλεονέκτημα ότι δεν χρειάζεται κάποια επεξεργασία (compilation) για την κατασκευή νέων εξομοιώσεων.

Η Tcl είναι μια μεταφρασμένη γλώσσα, αλλά επειδή δεν υπάρχει κάποια μείωση της απόδοσης με τη χρήση ο βασικός υπολογισμός γίνεται από τον ήδη compiled κώδικα της C++.

Χρησιμοποιεί μοντέλα κίνησης από τον Danzig και κίνηση που μιμείται τις αιτήσεις (applications). Οι αιτήσεις τρέχουν πάνω από το εξομοιούμενο IP που με τη σειρά του μπορεί να χρησιμοποιήσει δυο διαφορετικά επίπεδα σύνδεσης δεδομένων (ATM ή LAN γενικά).

Η μονάδα ATM χρησιμοποιεί FIFO και RCSP (Rate-Controlled Static Priority) ουρές. Παρέχει ένα απλό AAL πρωτόκολλο παρόμοιο με το AAL5 που κάνει διάσπαση και συναρμολόγηση πακέτων, και ένα πρωτόκολλο σηματοδοσίας για να γίνεται έλεγχος πρόσβασης και δέσμευση πόρων για νέα κανάλια.

Το εξομοιούμενο switch υλοποιεί Early Packet Discard (EPD). Το IP module χρησιμοποιεί ένα στατικό πίνακα δρομολόγησης που φορτώνεται κατά το χρόνο ρύθμισης.

Τα αντικείμενα του INSANE υλοποιούνται βασικά όπως ένα μηχάνημα πεπερασμένης κατάστασης. Τα αντικείμενα επικοινωνούν στέλνοντας events το ένα στο άλλο.

Αντιδρούν στα events ενημερώνοντας για τη δική τους κατάσταση και προκαλώντας events. Τα events είναι μηνύματα που περιέχουν το χρόνο στον οποίο το event πρέπει να ξεκινήσει, το αντικείμενο-αποστολέα, τον τύπο και ένα πεδίο δεδομένων.

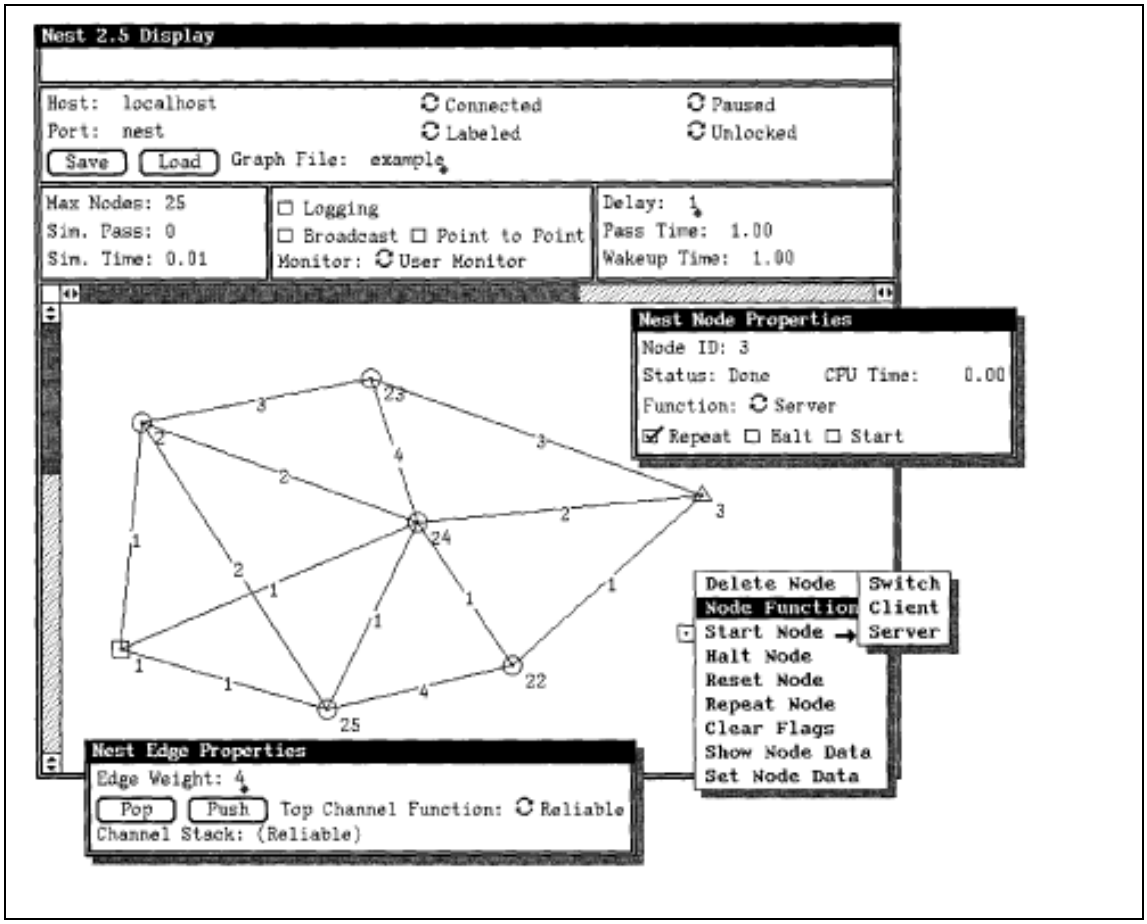
Ο προγραμματιστής (scheduler) παραδίδει events στο σχετικό αντικείμενο σύμφωνα με χρονολογική σειρά. Στηρίζεται προφανώς σε μια ημερολογιακή ουρά. Οι ημερολογιακές ουρές είναι μια συγκεκριμένη υλοποίηση ουρών προτεραιοτήτων και χρειάζεται ιδιαίτερη επεξεργασία, με τον τρόπο που υποστηρίζεται.

NEST

Προορίζεται [5] για την εξομοίωση και την προτυποποίηση κατανεμημένων αλγορίθμων και συστημάτων. Χρησιμοποιήθηκε για την εξομοίωση συστημάτων εξομάλυνσης φόρτου που βασίζονται σε μικροοικονομικές αρχές, για την μελέτη του προβλήματος ενημέρωσης της τοπολογίας στον ARPANET και άλλα.

Είναι καλός για την κατανόηση της συμπεριφοράς πρωτοκόλλων δρομολόγησης, όπως είναι για παράδειγμα οι επαναλήψεις δρομολόγησης. Περιγράφεται καλύτερα σαν ένα μοντέλο πελάτη-εξυπηρετητή: παρουσιάζει τους πελάτες που συνδέονται σε έναν εξυπηρετητή εξομοίωσης μέσω ενός socket.

Ο εξυπηρετητής εξομοίωσης είναι υπεύθυνος για την εκτέλεση των εξομοιώσεων. Οι πελάτες είναι ανεξάρτητα προγράμματα που χρησιμοποιούνται για τη δημιουργία, τις ρυθμίσεις ενός μοντέλου εξομοίωσης και τον έλεγχο της εκτέλεσής του, χρησιμοποιώντας ένα GUI που επικοινωνεί με την εξομοίωση μέσω μιας TCP/IP σύνδεσης και επιτρέπει τη δυναμική δημιουργία και διαμόρφωση των ρυθμίσεων του δικτύου. Το σύστημα απαιτεί μικρό bandwidth επικοινωνίας.



Εικόνα 11

Ξεχωριστά από την εξομοίωση το GUI προσφέρει αρκετά πλεονεκτήματα. Πρώτα απ' όλα, επιτρέπει εξοικονόμηση πόρων της CPU με το να τρέχει την εξομοίωση σε μια dedicated CPU ενώ το GUI τρέχει σε έναν σταθμό εργασίας.

Βασικά, μια μελέτη σύνθετης εξομοίωσης μπορεί να εκτελεστεί σε έναν απομακρυσμένο υπέρ-υπολογιστή. Αυτό είναι εξαιρετικά χρήσιμο για μια εργασία πολλών site και αυτό επιτρέπει να υποστηρίζει πελάτες σε ένα δίκτυο ευρείας περιοχής (WAN).

Επίσης, αυτός ο διαχωρισμός επιτρέπει πολλαπλά GUIs να αλληλεπιδρούν ταυτόχρονα με την εξομοίωση. Χρήστες μπορούν να συνδεθούν και να αποσυνδεθούν οποιαδήποτε στιγμή.

Ο NEST έχει υλοποιηθεί σαν βιβλιοθήκη από συναρτήσεις συνδεδεμένες μεταξύ τους με τον κώδικα του χρήστη. Είναι γραμμένος στη C αν και θα μπορούσε στη θεωρία οποιαδήποτε γλώσσας να χρησιμοποιηθεί εφόσον διέπεται από τη βασική αρχή του σωρού (π.χ. C, Pascal).

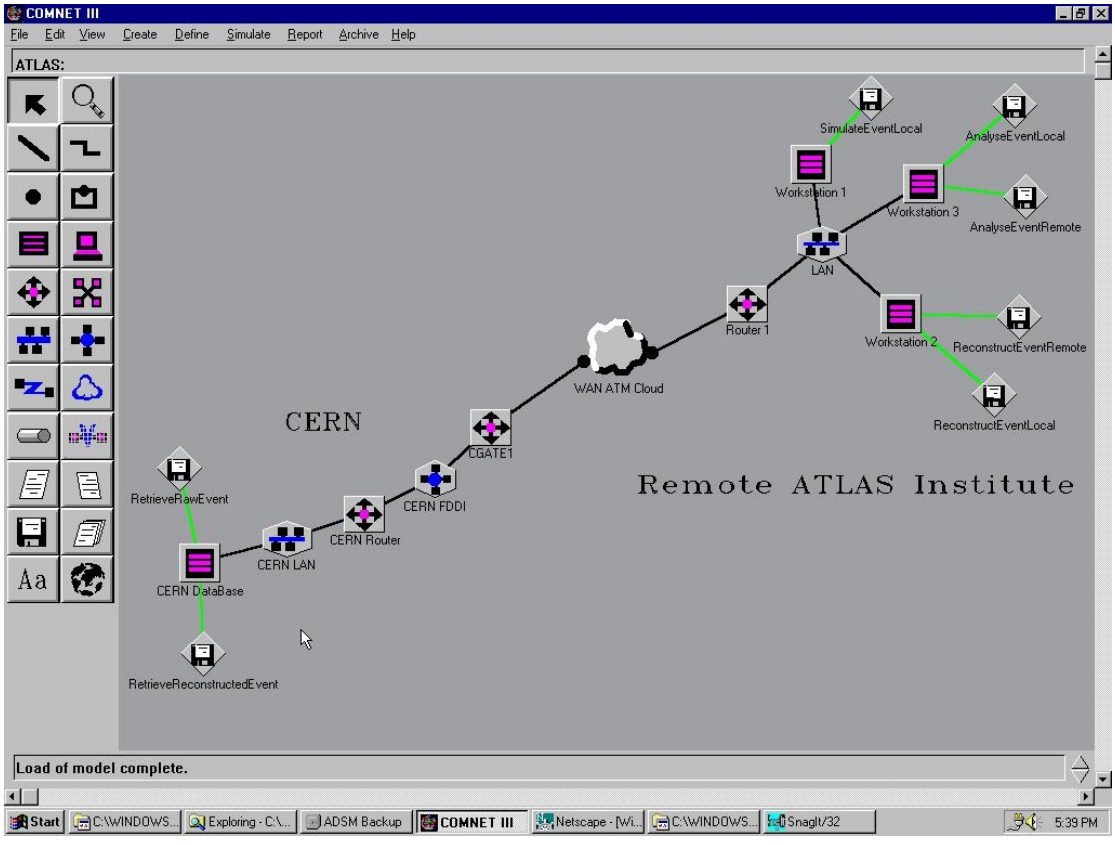
Οι απλές δραστηριότητες επικοινωνίας του NEST επιτρέπουν στο χρήστη να συνδιαλέγεται μόνο με το υψηλότερο επίπεδο ενός πρωτοκόλλου ή κατανεμημένου συστήματος.

Η τοπολογία αναπαρίσταται σαν ένας γράφος διασύνδεσης συνδέσμων και κόμβων που αποθηκεύονται εσωτερικά σαν ένα δέντρο. Ο χρήστης υλοποιεί την τοπολογία ενός δικτύου επικοινωνίας χρησιμοποιώντας ένα σύνολο γραφικών εργαλείων. Νέες λειτουργίες και συμπεριφορές συνδέσμων επικοινωνίας δημιουργούνται από το χρήστη και συνδέονται με το μοντέλο δικτύου.

COMNET III

Αποτελεί εργαλείο για τους σχεδιαστές δικτύων [6] που μπορεί και υπολογίζει εκ των προτέρων την επίδοση των LAN, WAN, των δικτύων ήχου και δεδομένων μέσω της εξομοίωσης. Το CACI' s COMNETIIITM είναι ένα λογισμικό αντικειμενοστραφούς εξομοίωσης δικτύων.

Είναι ένα αρκετά καλό εργαλείο εξομοίωσης που μοντελοποιεί την επίδοση των δικτύων και μειώνει το ρίσκο των δικτυακών επενδύσεων. Επιτρέπει στους πελάτες να προβλέψουν και να υπολογίσουν την επίδοση των LAN, WAN και σύνθετων εσωτερικών δικτύων επιχειρήσεων.



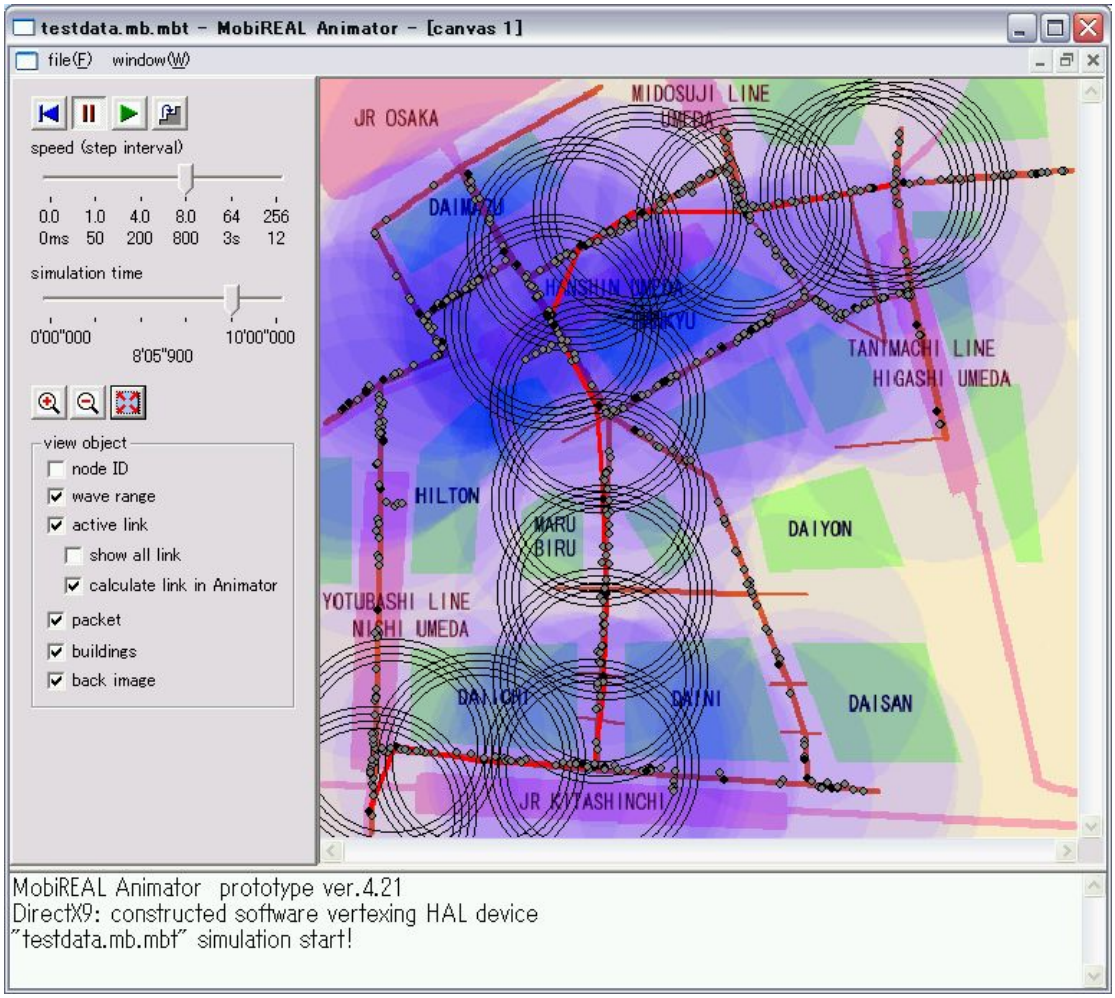
Εικόνα 12

REAL

Ο REAL (Realistic And Large) [7] είναι ένας εξομοιωτής δικτύου που προορίζεται για τη μελέτη της δυναμικής συμπεριφοράς της ροής και των σχημάτων ελέγχου συμφόρησης σε δίκτυα μεταγωγής πακέτου.

Παρέχει 30 στοιχεία γραμμένα σε C που εξομοιώνουν τα πρωτόκολλα ελέγχου ροής όπως το TCP, και 5 αρχές χρονοπρογραμματισμού όπως FIFO, Fair Queueing, DEC congestion avoidance και Hierarchical Round Robin.

Η περιγραφή της τοπολογίας του δικτύου, πρωτόκολλα φόρτου εργασίας και παράμετροι ελέγχου μεταφέρονται στον εξυπηρετητή χρησιμοποιώντας απλή ascii αναπαράσταση που ονομάζεται NetLanguage, όπου το δίκτυο παριστάνεται ως γράφος. Η τελευταία έκδοση συμπεριλαμβάνει ένα GUI γραμμένο σε JAVA. Η τοπολογία δημιουργείται γραφικά.


testdata.mb.mbt - MobiREAL Animator - [canvas 1]
file(F) window(W)
speed (step interval)
0.0 1.0 4.0 8.0 64 256
0ms 50 200 800 3s 12
simulation time
0'00"000
10'00"000
8'05"900
view object
node ID
wave range
active link
show all link
calculate link in Animator
packet
buildings
back image
JR OSAKA
MIDOSUJI LINE
UMEDA
DAIMARU
HANSHIN UMEDA
HANKYU
TANTMACHI LINE
HIGASHI UMEDA
HILTON
MARU
BIRU
DAIYON
YOTSUBASHI LINE
NISHI UMEDA
DAI ICHI
DAI NI
DAISAN
JR KITASHINCHI
MobiREAL Animator prototype ver.4.21
DirectX9: constructed software vertexing HAL device
"testdata.mb.mbt" simulation start!

Εικόνα 13

NS (Network Simulator)

Πρόκειται για έναν αντικειμενοστραφή διακριτών- event εξομοιωτή για την έρευνα δικτύων που βασίζεται στον REAL. Ο NS είναι κατάλληλος για packet- switched δίκτυα, και χρησιμοποιείται κυρίως για μικρής κλίμακας εξομοιώσεις αλγορίθμων ουρών, έλεγχο συμφόρησης πρωτοκόλλων μεταφοράς, και κάποια σχετική με multicast δουλειά. Παρέχει υποστήριξη για διάφορες υλοποιήσεις του TCP, δρομολογήσεις, πρωτοκόλλων multicast, στρωμάτων σύνδεσης, MAC, κτλ.

TeD (Telecommunications Description Language)

Ο TeD [8] ξεκίνησε από μια ανάγκη για μια γλώσσα μοντελοποίησης δικτύου που θα είχε τη δυνατότητα παράλληλης εξομοίωσης. Ο TeD είναι μια γλώσσα που έχει σχεδιαστεί κυρίως για τη μοντελοποίηση δικτυακών στοιχείων και πρωτοκόλλων.

Τεχνικά, είναι μια αντικειμενοστραφής γλώσσα στην οποία η συμπεριφορά κάθε αντικειμένου, κάθε οντότητας, εκφράζεται μέσω διαδικασιών που τρέχουν μέσα στα πλαίσια αυτού του αντικειμένου.

Τα αντικείμενα επικοινωνούν μεταξύ τους ανταλλάσσοντας events πάνω σε προκαθορισμένα και εκ των προτέρων δρομολογημένα κανάλια. Μπορεί να ενσωματωθεί C++ κώδικας μέσα στα εκτελέσιμα κομμάτια των μοντέλων του TeD.

Συμπεριλαμβάνει έναν compiler που μεταφράζει τα μοντέλα του TeD σε C++ κώδικα που χρησιμοποιεί τον GTW (Georgia Tech Time Warp) για παράλληλη εξομοίωση.

OPNET (Optimized Network Engineering Tool)

Ο OPNET [9] παρέχει ένα βοηθητικό περιβάλλον ανάπτυξης για τον καθορισμό, την εξομοίωση και την ανάλυση επιδόσεων των επικοινωνιακών δικτύων. Μπορεί να υποστηριχθεί ένα μεγάλο εύρος επικοινωνιακών δικτύων από ένα απλό LAN μέχρι και ένα μεγάλο δορυφορικό δίκτυο.

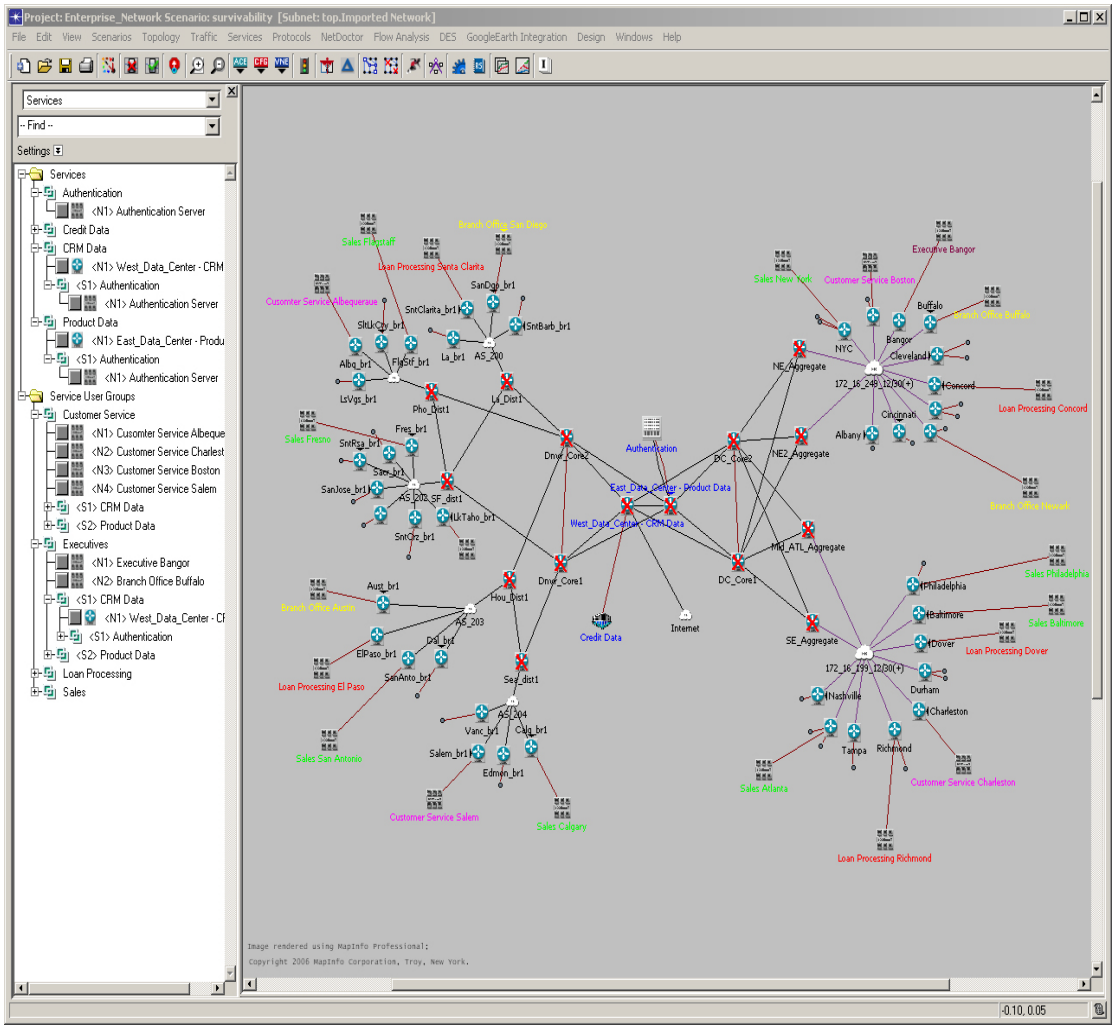
Εξομοιώσεις διακριτών γεγονότων χρησιμοποιούνται ως μέσο για την ανάλυση της επίδοσης συστημάτων και της συμπεριφοράς τους. Τα ιδιαίτερα χαρακτηριστικά του OPNET συνοψίζονται στα εξής:

- Μοντελοποίηση και κύκλος εξομοίωσης. Ο OPNET παρέχει ισχυρά εργαλεία που βοηθούν το χρήστη να προχωρήσει στις τρεις από τις πέντε φάσεις ενός κύκλου σχεδίασης (π.χ. η δημιουργία μοντέλων, η εκτέλεση μιας εξομοίωσης και η ανάλυση των αποτελεσμάτων).

- Ιεραρχική μοντελοποίηση. Ο OPNET χρησιμοποιεί μια ιεραρχική δομή για μοντελοποίηση. Κάθε επίπεδο της ιεραρχίας περιγράφει διαφορετικές προσεγγίσεις του συνολικού μοντέλου που εξομοιώνεται.

- Ειδικεύεται στα επικοινωνιακά δίκτυα. Αναλυτικά μοντέλα βιβλιοθηκών παρέχουν υποστήριξη για τα υπάρχοντα πρωτόκολλα και επιτρέπει στους ερευνητές και τους κατασκευαστές είτε να ρυθμίσουν τα υπάρχοντα μοντέλα ή να αναπτύξουν νέα μοντέλα από μόνοι τους.

- Αυτόματη δημιουργία εξομοίωσης. Τα μοντέλα του OPNET μπορούν να μετατραπούν σε εκτελέσιμο κώδικα. Μια εκτελέσιμη διακριτών-γεγονότων εξομοίωση μπορεί να αποκωδικοποιηθεί ή και να εκτελεστεί απλά, δίνοντας τα τελικά αποτελέσματα.



Εικόνα 14

JiST

Μια νέα προσέγγιση στο χώρο των δικτυακών προσομοιώσεων είναι το JiST (JavainSimulationTime) [10] που όπως είναι φανερό και από το όνομα του, χρησιμοποιεί την Java για την υλοποίηση των προσομοιώσεων.

Οι προσομοιώσεις στο JiST αποτελούνται από οντότητες που αναπαριστούν στοιχεία του δικτύου με τα γεγονότα της προσομοίωσης να σχηματίζονται από κλήσεις μεθόδων αυτών των οντοτήτων.

OMNeT++

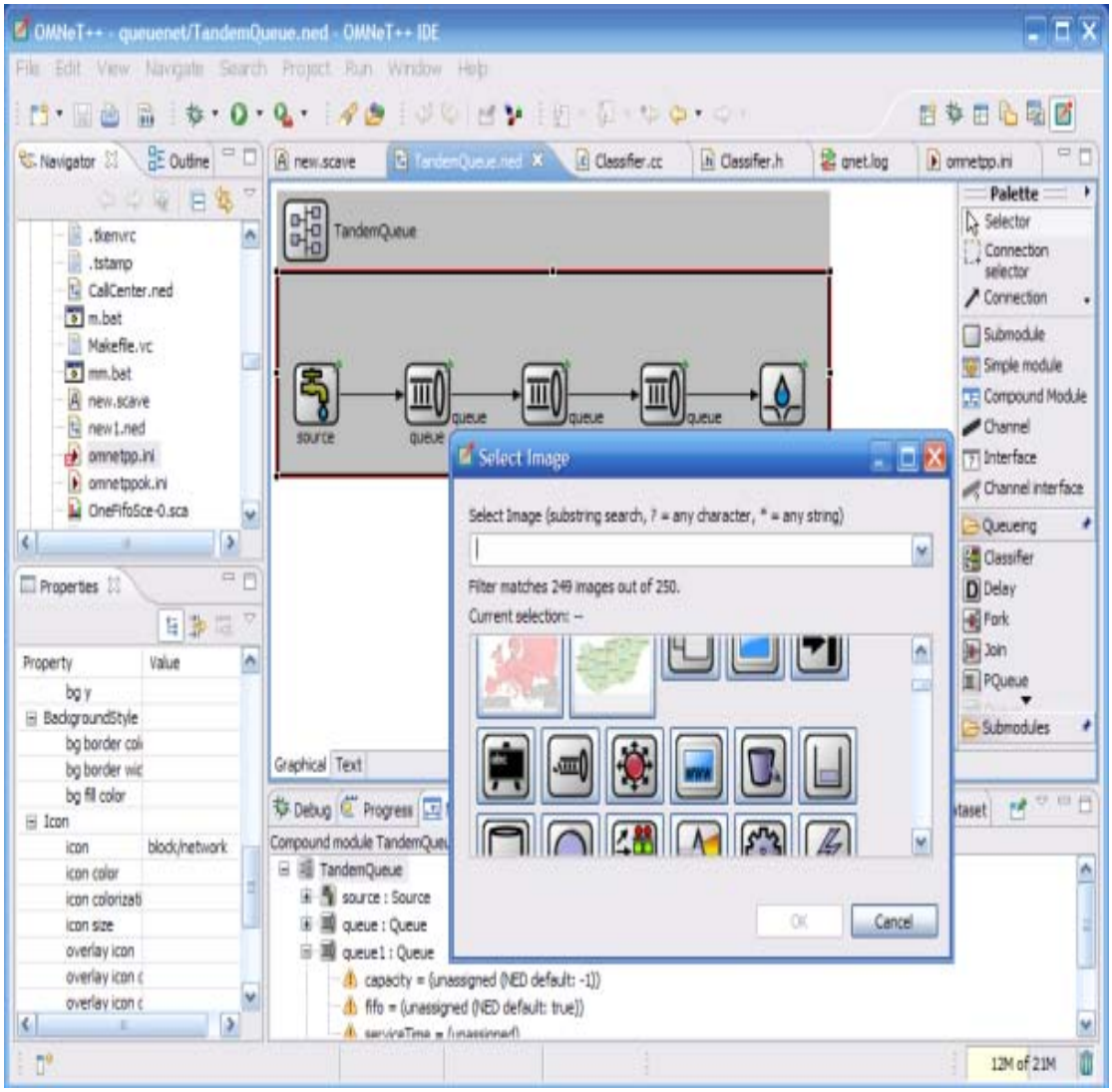
Σε αντίθεση με τον NS-2 και τον NS-3, ο OMNeT++ [11] δεν είναι μόνο δικτυακός προσομοιωτής, αλλά ένας προσομοιωτής διακεκριμένων γεγονότων γενικής χρήσης. Περισσότερο όμως χρησιμοποιείται για προσομοιώσεις δικτύων κάνοντας χρήση του πακέτου INET το οποίο περιέχει μια εκτεταμένη βιβλιοθήκη πρωτοκόλλων.

Το OMNeT++ είναι ένας αντικειμενοστραφής προσομοιωτής διακεκριμένων 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 διακεκριμένων γεγονότων και το οποίο μπορεί να αντιστοιχηθεί σε οντότητες που επικοινωνούν μεταξύ τους ανταλλάσοντας μηνύματα.



Εικόνα 15

Αρχικά, πρέπει να αναφέρουμε το περιβάλλον ανάπτυξης του OMNeT++. Αυτό αποτελείται κατά βάση από το Eclispe, το help του οποίου παραθέτει όλες τις απαιτούμενες πληροφορίες για να εξοικειωθεί ο ενδιαφερόμενος αναγνώστης σε μία πρώτη προσέγγιση. Πιο συγκεκριμένα αυτά είναι το WorkbenchUserGuide και το C/C++ DevelopmentUserGuide.

Στην συνέχεια, είναι απαραίτητη η εγκατάσταση και η μελέτη του INETFramework. Αυτό στην ουσία αποτελεί μια βιβλιοθήκη από έτοιμα simple και compoundmodules που προσομοιώνουν διαδικτυακές συσκευές και πρωτόκολλα.

Κεφάλαιο 4: Ο δικτυακός προσομοιωτής Network Simulator(NS)

Στο κεφάλαιο αυτό, θα μελετήσουμε τα ενδότερα του δικτυακού προσομοιωτή Network Simulator (NS). Θα αναφερθούμε στις προηγούμενες εκδόσεις του και θα μιλήσουμε πιο αναλυτικά για τα ιδιαίτερα χαρακτηριστικά και την λειτουργία της τρέχουσας έκδοσης που είναι η έκδοση 3 [12].

Εισαγωγή

Όπως προαναφέραμε, η Προσομοίωση Δικτύων είναι μια μέθοδος με την οποία δημιουργούμε το μοντέλο ενός δικτύου μέσω λογισμικού και το τροφοδοτούμε με δεδομένα εισόδου.

Παρατηρώντας τα αποτελέσματα αυτών των δεδομένων πάνω στο μοντέλο μπορούμε να οδηγηθούμε σε συμπεράσματα για την λειτουργία των πραγματικών δικτύων. Ο NS ή αλλιώς, network simulator είναι ένας δικτυακός προσομοιωτής διακριτού χρόνου ή γεγονότων.

Ο NS χρησιμοποιείται στην προσομοίωση των πρωτοκόλλων δρομολόγησης και σε μεγάλο βαθμό σε έρευνα ad-hoc δικτύων. Ο NS υποστηρίζει δημοφιλή πρωτόκολλα δικτύου, παρέχοντας επίσης τα αποτελέσματα της προσομοίωσης τόσο για ενσύρματα όσο και για ασύρματα δίκτυα. Είναι δημοφιλής στην έρευνα μιας και βασίζεται σε open source μοντέλο και παρέχει online ηλεκτρονικής τεκμηρίωσης.

Οι πρώτες προσπάθειες

Ο NS άρχισε την ανάπτυξη το 1989 ως μια παραλλαγή του εξομοιωτή δικτύων REAL και σήμερα συντηρείται πλέον από εθελοντές. Επιπλέον, μακροχρόνιες εισφορές προέρχονται από τη Sun Microsystems, καθώς και από τα projects του UCBDaedelus και του Carnegie Mellon Monarch, τα οποία βρίσκονται στην αρχική ιστοσελίδα του NS για προσθήκες ασύρματου κώδικα.

Network Simulator (NS – 2)

Οι προσομοιώσεις δικτύων στον NS-2 [13] δημιουργούνται με την βοήθεια κώδικα σε C++, ο οποίος χρησιμοποιείται για να μοντελοποιήσει τη συμπεριφορά των κόμβων του δικτύου, αλλά και της γλώσσας scriptOTcl που ελέγχει την προσομοίωση και καθορίζει περαιτέρω πτυχές όπως για παράδειγμα η τοπολογία του δικτύου.

Ωστόσο, η μοντελοποίηση στον NS-2 είναι μία ιδιαίτερα πολύπλοκη και χρονοβόρα διαδικασία, μιας και δεν έχει κανένα GUI και κάποιος πρέπει να μάθει μία γλώσσα προγραμματισμού, τη θεωρία ουρών καθώς και τεχνικές μοντελοποίησης.

Τα τελευταία χρόνια, υπήρξαν παράπονα σχετικά με την συνέπεια των αποτελεσμάτων (πιθανόν λόγω των συνεχών αλλαγών στον κώδικα του συστήματος) και ότι ορισμένα πρωτόκολλα έχουν σφάλματα. Ο NS έχει την άδεια χρήσης της έκδοσης 2 της GNU General Public License.

Αυτός ο σχεδιασμός αρχικά επιλέχθηκε για να αποφεύγονται οι άσκοπες μεταγλωττίσεις αν γινόταν κάποια αλλαγή στο δίκτυο. Το 1996 (όταν εμφανίστηκε ο NS-2 για πρώτη φορά) αυτό ήταν αρκετά σημαντικό αφού οι μεταγλωττίσεις με C++ κώδικα επιβράδυναν αρκετά τα συστήματα.

Όμως με τα σημερινά δεδομένα αυτός ο σχεδιασμός δεν θεωρείται αποδοτικός καθώς μειώνει σημαντικά την απόδοση της προσομοίωσης, γεγονός πολύ σημαντικό για προσομοιώσεις ευρείας κλίμακας με εκατοντάδες κόμβους.

Ο χρήστης δεν χρειάζεται να γνωρίζει C++, εκτός αν θέλει να προσθέσει κάποιο καινούργιο πρωτόκολλο ή λειτουργία που δεν υπάρχει ήδη. Σε γενικές γραμμές τα βασικά βήματα για τη δημιουργία μιας προσομοίωσης σε NS-2 είναι τα εξής:

- Δημιουργία της τοπολογίας του δικτύου και ορισμός παραμέτρων.
- Ρύθμιση της δρομολόγησης μεταξύ των κόμβων.
- Δημιουργία γεννητριών κινήσεως.
- Εκτέλεση της προσομοίωσης.

Network Simulator(NS – 3)

Ο NS-3 έχει ξεκινήσει την ανάπτυξη του από την 1 Ιουλίου του 2006 και είχε αρχικά προβλεφθεί να διαρκέσει τέσσερα χρόνια. Λόγω όμως διαφόρων δυσχερειών κατά τη διάρκεια του σχεδιασμού, της γραφής και του debugging, το λογισμικό εξακολουθεί να βρίσκεται σε φάση ανάπτυξης.

Όπως και ο προκάτοχός του, το NS-3 στηρίζεται στην C++ για την υλοποίηση των μοντέλων προσομοίωσης. Η διαφορά όμως έγκειται στο γεγονός ότι δεν χρησιμοποιεί πια την OTcl για να ελέγχει την προσομοίωση, λύνοντας έτσι αρκετά προβλήματα.

Έτσι, για μια προσομοίωση μπορεί να χρησιμοποιηθεί μόνο C++ και σε μερικές περιπτώσεις και Python. Επίσης έχει πολύ καλές δυνατότητες επέκτασης και αρκετά βελτιωμένη απόδοση.

Εγκατάσταση

Ο NS-3 διανέμει τον πηγαίο του κώδικα, βάσει της άδειας χρήσης GNU Public License. Πρόκειται για ένα κλασσικό παράδειγμα ανοικτού λογισμικού. Συνεπώς, η εγκατάσταση έγκειται στην απόκτηση του πηγαίου κώδικα και στην μεταγλώττιση του.

Για την επιτυχή μεταγλώττιση και χρήση του λογισμικού, απαιτείται ένα υπολογιστικό σύστημα με λειτουργικό σύστημα UNIX/ Linux ή Microsoft Windows. Για το τελευταίο, προϋποτίθεται η ύπαρξη της πλατφόρμας Cygwin.

Επίσης, είναι απαραίτητη η παρουσία στο σύστημα που θα φιλοξενήσει τον NS-3, των λογισμικών που φαίνονται παρακάτω.

- gcc
- g++
- python
- python-dev
- mercurial
- bzr
- gdb
- valgrind
- gsl-bin
- libgsl0-dev
- libgsl0ldbl
- flex
- bison
- tcpdump
- sqlite
- sqlite3
- libsqlite3-dev
- libxml2
- libxml2-dev
- libgtk2.0-0
- libgtk2.0-dev
- vtun
- lxc
- uncrustify
- doxygen

- graphviz
- imagemagick
- texlive
- texlive-pdf
- texlive-latex-extra
- texlive-generic-extra
- texlive-generic-recommended
- python-pygraphviz
- python-kiwi
- python-pygoocanvas
- libgoocanvas-dev

Στην συνέχεια, πλοηγούμαστε στον ριζικό φάκελο του πηγαίου κώδικα του NS-3 και εκτελούμε την εντολή

```
$ ./build.py
```

για να ξεκινήσει η μεταγλώττιση. Αυτή η διαδικασία διαρκεί για αρκετή ώρα και εφόσον ολοκληρωθεί επιτυχώς, εκτελούμε τις εντολές

```
$ cd ns-<αριθμός_έκδοσης>
$ ./waf configure
$ ./waf
```

Το λογισμικό είναι έτοιμο για χρήση.

Χρήση

Το σενάριο προσομοίωσης ουσιαστικά είναι ένα πρόγραμμα γραμμένο σε C++, που ακολουθεί μια σειρά συμβάσεων που ορίζουν οι συγγραφείς του NS-3. Η πιο απλή και εύκολη δομή είναι αυτή στην οποία φτιάχνουμε ένα φάκελο και μέσα σε αυτόν δημιουργούμε τρία αρχεία. Τον φάκελο αυτόν τον τοποθετούμε στον φάκελο examples του NS-3.

Το πρώτο αρχείο είναι το C++ αρχείο, το οποίο περιέχει όλον τον κώδικα – σενάριο και το οποίο το σώζουμε με κατάληξη .cc. Το δεύτερο είναι το αρχείο waf το οποίο περιέχει μια και μόνο εντολή, η οποία ουσιαστικά «δείχνει» την θέση του εκτελέσιμου του NS-3 και το κάνει invokeγια τον συγκεκριμένο κώδικα που θέλουμε να εκτελέσουμε. Η εντολή αυτή φαίνεται παρακάτω.

```
exec "`dirname "$0"`"/../../waf "$@"
```

Αν μετατρέψουμε κατάλληλα την διαδρομή στην οποία «δείχνει» η παραπάνω εντολή, τότε μπορούμε να εκτελούμε το σενάριο προσομοίωσης μας από οποιοδήποτε σημείο του συστήματος θέλουμε, χωρίς να είναι πλέον απαραίτητο να έχουμε τα αρχεία μας αυτά στον φάκελο examples.

Το τρίτο αρχείο είναι ένα python script, το οποίο ουσιαστικά λειτουργεί σαν Makefile για τον NS – 3 και του υποδεικνύει ότι πρέπει να γίνει μεταγλώττιση και εκτέλεση του σεναρίου που έχουμε δημιουργήσει. Αν το σενάριο μας έχει συντακτικά ή λογικά λάθη, αυτά θα φανούν στην οθόνη, ακριβώς σαν να κάναμε μεταγλώττιση οποιουδήποτε πηγαίου κώδικα σε οποιοδήποτε προγραμματιστικό περιβάλλον.

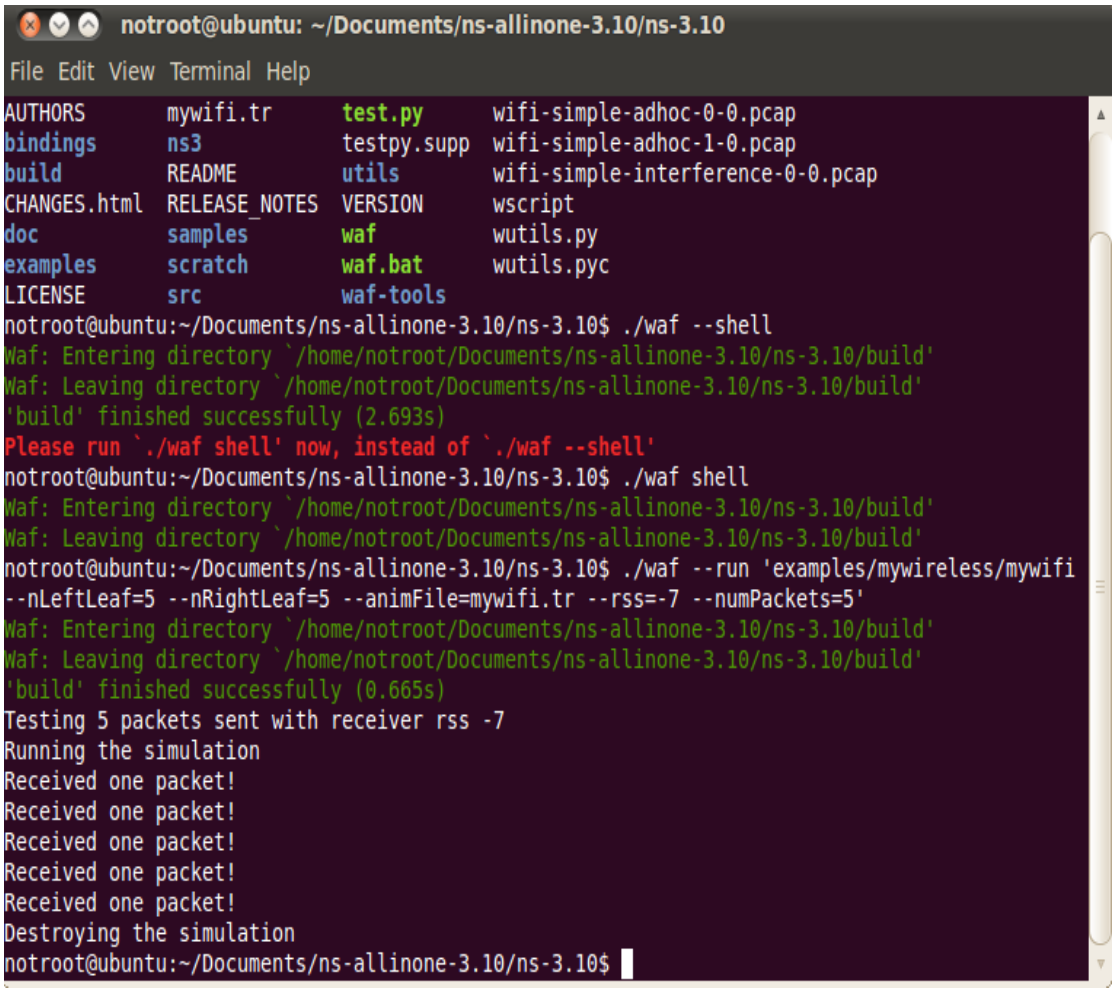
```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil;
coding: utf-8; -*-
```

```
def build(bld):

    obj = bld.create_ns3_program('mywifi', ['core', 'simulator',
'mobility', 'wifi', 'flow-monitor'])

obj.source = 'mywifi.cc'
```

Αν η μεταγλώττιση είναι επιτυχής, τότε στο τερματικό μας θα δούμε το αποτέλεσμα, όπως στην παρακάτω εικόνα.



Εικόνα 16

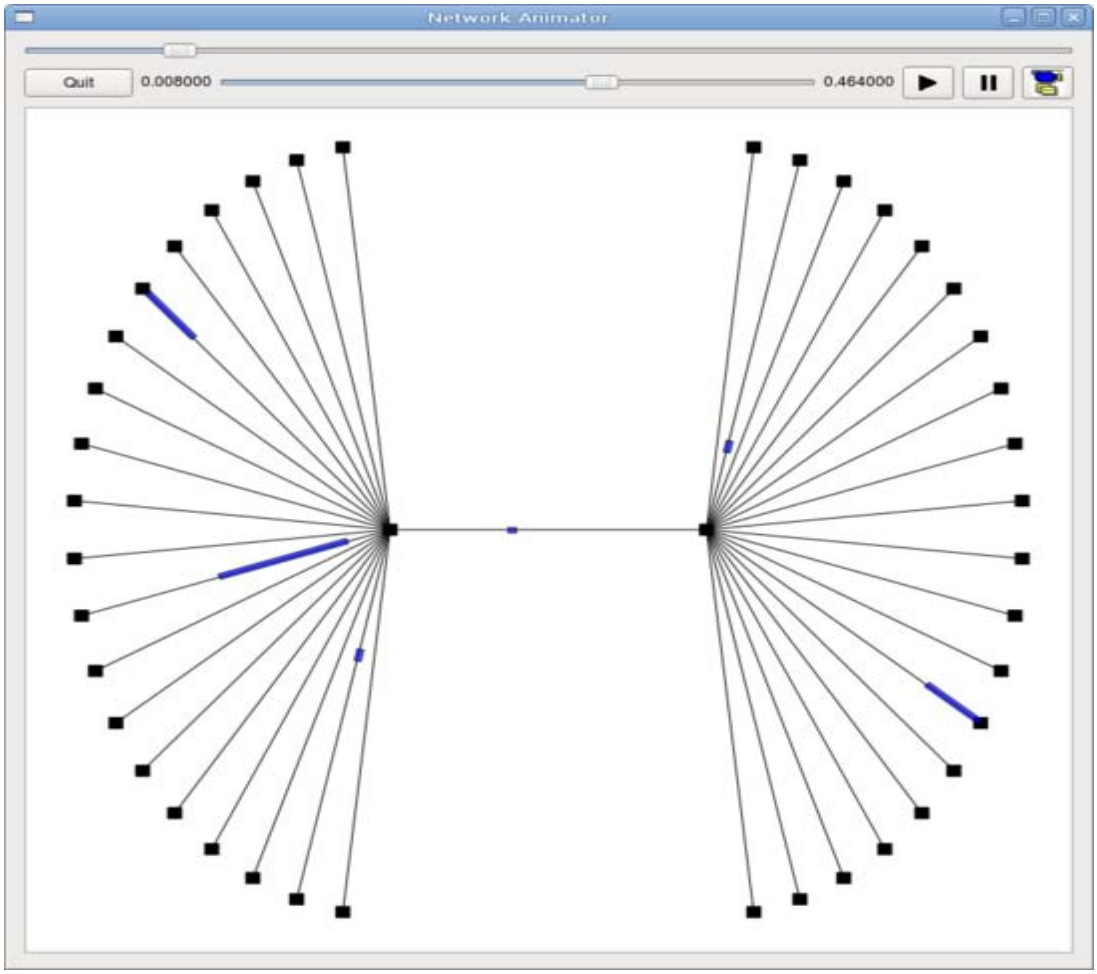
Έστω ότι έχουμε να εκτελέσουμε το σενάριοa scenario.cc το οποίο βρίσκεται στον φάκελο /home/user/Documents/ns-allinone-3.10/ns-3.10/examples/ascenario. Τότε η σειρά των εντολών είναι η εξής:

```
$ cd "/home/user/Documents/ns-allinone-3.10/ns-3.10/"
$ ./waf shell

$ ./waf --run 'examples/ascenario/ascenario --όρισμα_1=τιμή_1 ... --όρισμα_ν=τιμή_ν'
```

Οπτικοποίηση προσομοίωσης

Όταν εκτελούμε μια προσομοίωση, είναι επιθυμητό να βλέπουμε μηνύματα τα οποία να μας πληροφορούν για την εξέλιξη της και φυσικά για την δραστηριότητα του δικτύου που προσομοιώνουμε.



Εικόνα 17

Αυτό μπορεί να γίνει μέσω του τερματικού, της κονσόλας δηλαδή, στο οποίο εκτελούμε την προσομοίωση. Επίσης, μπορεί να γίνει και οπτικά, δηλαδή με την χρήση επιπρόσθετου λογισμικού το οποίο κάνει animationτο δίκτυο προσομοίωσης.

Το πιο σημαντικό λογισμικό οπτικοποίησης για τον NS-3 αυτή τη στιγμή είναι το NetworkAnimator (NetAnim). Όπως και ο NS-3, διατίθεται σε μορφή πηγαίου κώδικα και εγκαθίσταται με μεταγλώττιση.

Αυτό έχει σαν προαπαιτούμενο τις εξαρτήσεις του NS-3 και επιπλέον τα QT3/QT4 GUI Toolkits.Για την εγκατάσταση του, πλοηγούμαστε στον ριζικό φάκελο του πηγαίου κώδικα

```
cd NetAnim
```

και εκτελούμε τις εντολές

```
qmake-qt4
make
```

Το NetAnim [14]δέχεται στην είσοδο του αρχεία της μορφής .tr για να πραγματοποιήσει το επιθυμητό animation. Δηλαδή, είναι ευθύνη του χρήστη ώστε η προσομοίωση του να παράγει ένα τέτοιο αρχείο, το οποίο στην συνέχεια θα εκτελέσει στον NetAnim, με την χρήση της εντολής

```
./NetAnim <όνομα_αρχείου>.tr
```

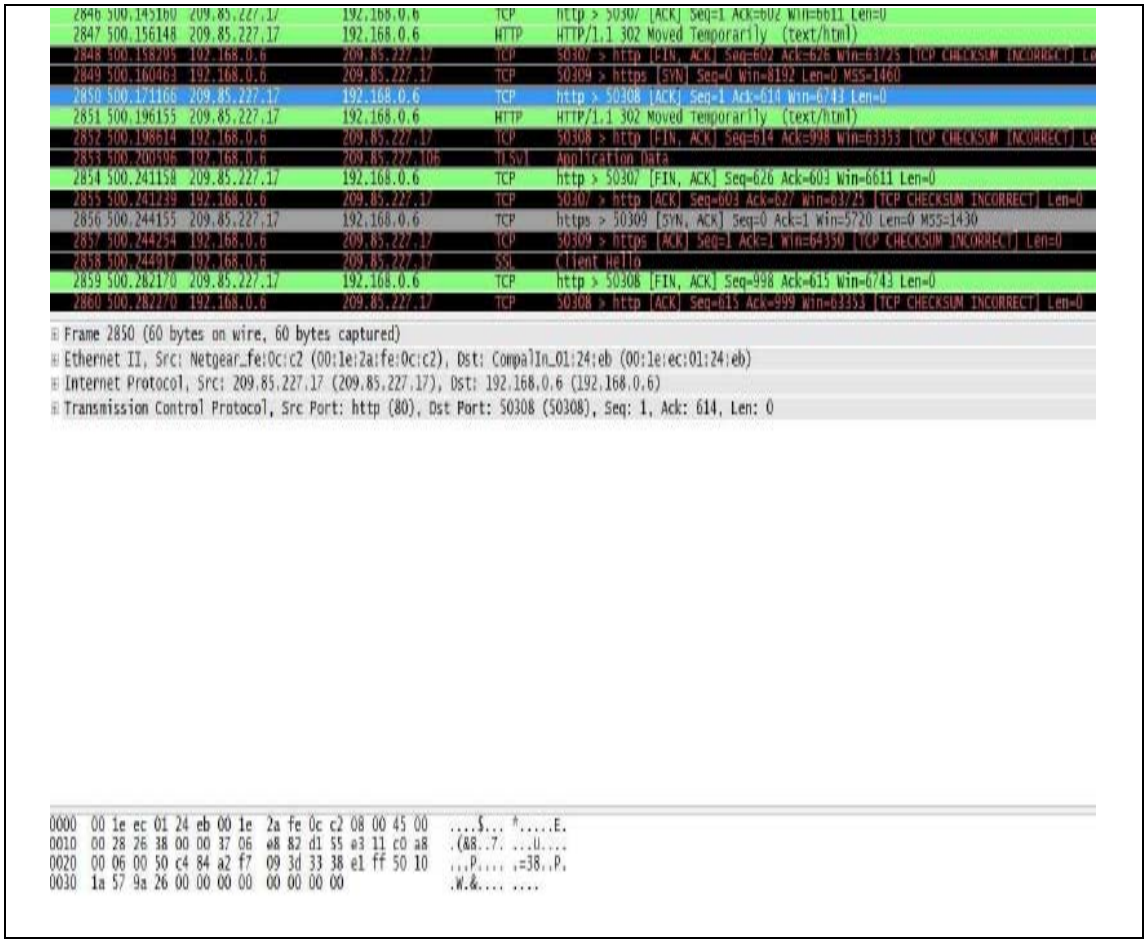
Δυστυχώς, στην τρέχουσα μορφή του, το NetAnim έχει ένα σημαντικό μειονέκτημα. Συγκεκριμένα, αδυνατεί να απεικονίσει την κίνηση σε ένα ασύρματο δίκτυο επικοινωνιών, όπως ένα MANET.

Έτσι λοιπόν, καταφεύγουμε στην άλλη σημαντική λύση που μας προσφέρει την δυνατότητα επεξεργασίας των δεδομένων μας, το λογισμικό wireshark, που περιγράφουμε παρακάτω.

Wireshark

Το Wireshark [15] είναι ένα λογισμικό ανάλυσης πακέτων (packet analyzer), με κύριο στόχο την εύρεση μηνυμάτων, χρήσιμα για το χειριστή. Μπορεί να διαβάζει πακέτα είτε online σε ένα δίκτυο, είτε αποθηκευμένα σε κάποιο αρχείο αργότερα.

Το πρότυπο που έχουν τα αρχεία που μπορεί να διαχειριστεί το λογισμικό είναι το λεγόμενο pcap. Έτσι λοιπόν, μπορεί να αναγνωρίσει και να διαχειριστεί τα pcap αρχεία που μπορεί να παράγει στην έξοδο του.



Εικόνα 18: Το wireshark παρακολουθεί την ροή των πακέτων.

Αυτό το χαρακτηριστικό είναι πολύ χρήσιμο και ιδιαίτερα για την περίπτωση όπου με τον NS-3 προσπαθούμε να εξομοιώσουμε κάποιο ασύρματο δίκτυο. Σε αυτήν την περίπτωση, δεν μπορούμε να χρησιμοποιήσουμε τον NetAnim.

Τα online δεδομένα, μπορούν να συγκεντρώνονται από Ethernet, Wirelless, PPP και loopback επικοινωνίες. Τα δεδομένα που συλλέγονται μπορούν να επεξεργαστούν ή να μετατραπούν για να αναγνωρίζονται από διαφορετικά προγράμματα καθώς είναι εφικτή και η αναβάθμιση με plugins για την εγκατάσταση νέων πρωτοκόλλων.

Το Wireshark είναι ένα ολοκληρωμένο πρόγραμμα ανάλυσης των πρωτοκόλλων των πακέτων μιας ροής κίνησης σε πραγματικό ή όχι πραγματικό χρόνο. Παρέχει ειδικά φίλτρα απόρριψης πακέτων ή σήμανση της, ενώ της δίνει τη δυνατότητα σχεδιασμού γραφημάτων του πλήθους πακέτων ως της το χρόνο.

Το wireshark δίνει τη δυνατότητα καταγραφής της άφιξης του χρόνου του πακέτου, του αποστολέα (source), του προορισμού (destination), του πρωτοκόλλου επιπέδου μεταφοράς, αλλά και πληροφορίες σχετικά με τα κατωτέρου επιπέδου πρωτόκολλα, IP, Ethernet κλπ.

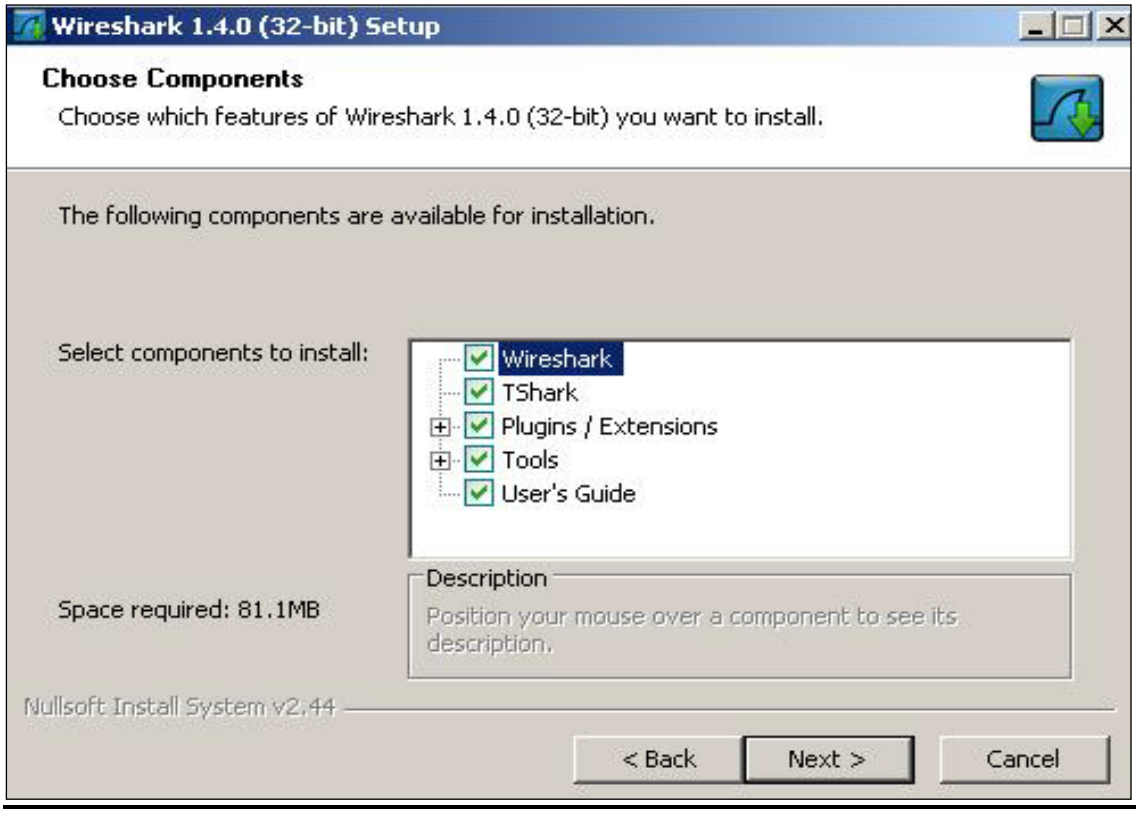
Στην παραπάνω εικόνα, φαίνεται μια ροή πακέτων σε μια κάρτα δικτύου με χρήση wireshark. Το Wireshark πριν το 2006 ονομάζονταν Ethereal. Τότε ο σχεδιαστής του, για οικονομικούς λόγους, άλλαξε την ονομασία του.

Εγκατάσταση και χρήση

Το wireshark είναι ένα ελεύθερο λογισμικό, ανοικτού κώδικα, που διανέμεται δωρεάν υπό την άδεια χρήσης GNU Public Licence. Μπορεί να αποκτηθεί με downloading από το [15].

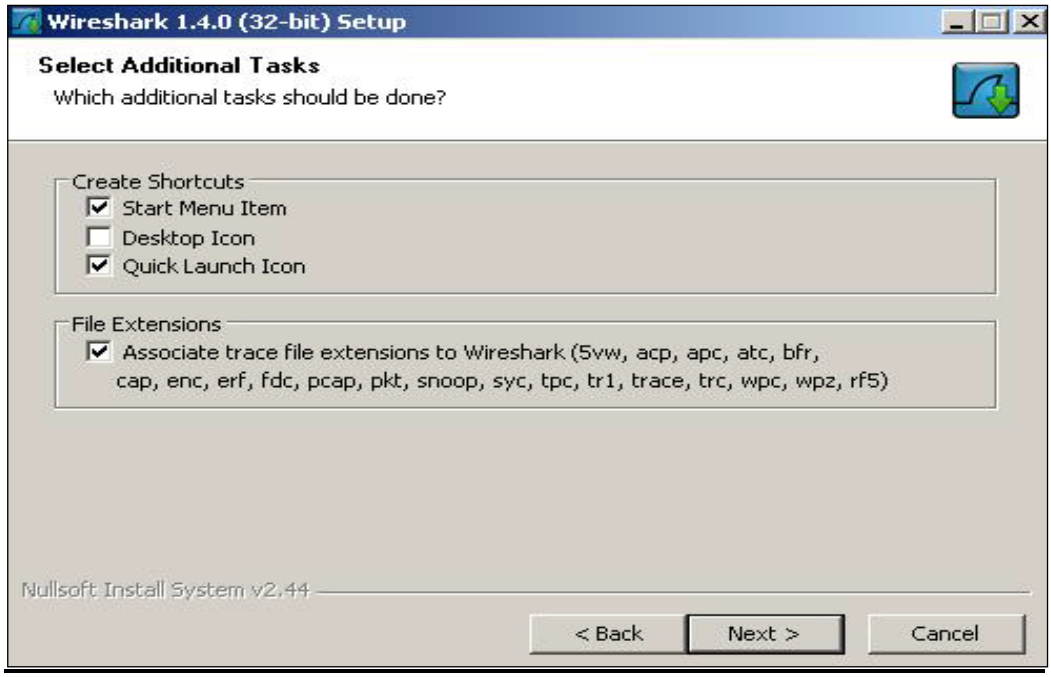
Εγκατάσταση

Εκτελώντας το setup του λογισμικού, ακολουθούμε την διαδικασία που φαίνεται της παρακάτω εικόνες. Επιλέγουμε να συμφωνήσουμε με της όρους της άδειας χρήσης και προχωράμε.



Εικόνα 19

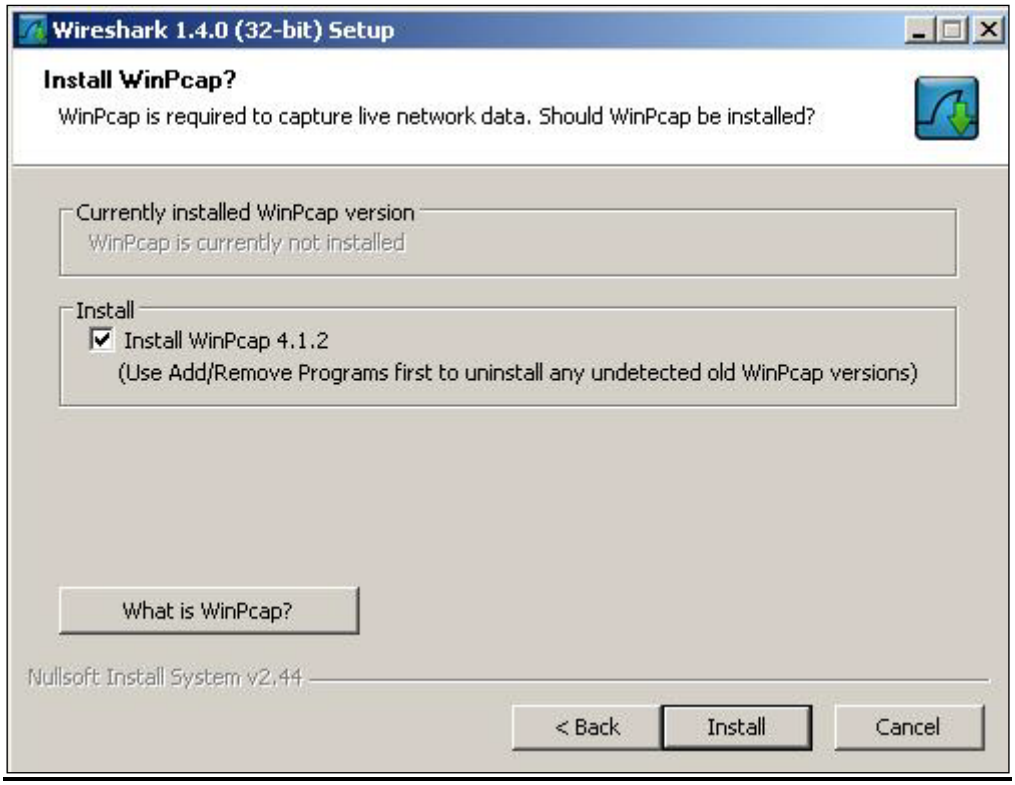
Επιλέγουμε να κάνουμε μια πλήρη εγκατάσταση και πατάμε next.



Εικόνα 20

Συσχετίζουμε το λογισμικό με τα αρχεία που μπορεί να διαβάσει και πατάμε next.

Αφήνουμε τον προεπιλεγμένο φάκελο εγκατάστασης και πατάμε next.



Εικόνα 21

Αν το πρόγραμμα εγκατάστασης μας ζητήσει την άδεια μας για να εγκαταστήσει την βιβλιοθήκη WinPcap, δεχόμαστε και επιλέγουμε install. Η εγκατάσταση εδώ ολοκληρώνεται. Πατάμε finish και πλέον μπορούμε να χρησιμοποιήσουμε το λογισμικό.

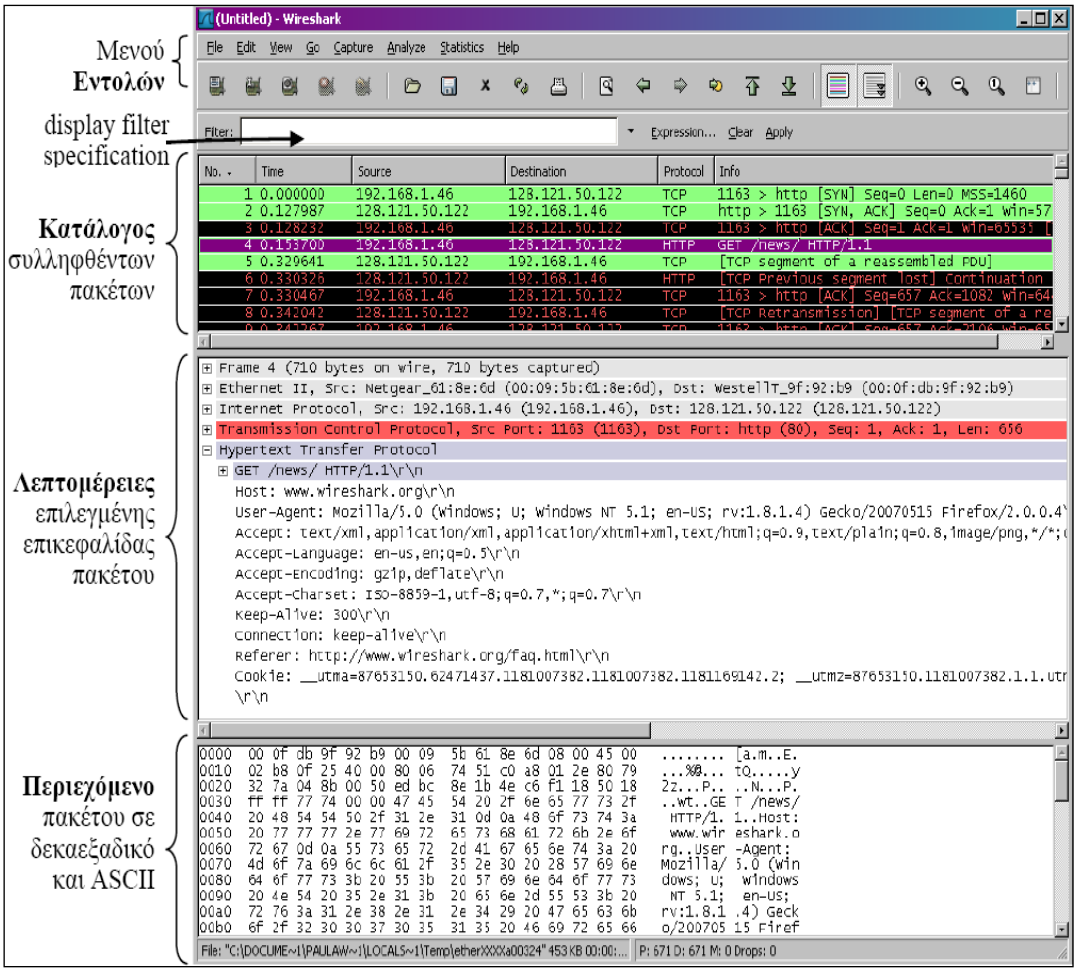
Ωστόσο, το wireshark δεν εκτελείται μόνο σε περιβάλλον Microsoft windows αλλά και σε Linux, όπως ο NS-3. Προκειμένου για διανομές της κατηγορίας Ubuntu, Debian και άλλα, η εγκατάσταση του γίνεται με την παρακάτω εντολή:

```
$sudo apt-get install wireshark
```

Χρήση

Στην παρακάτω εικόνα βλέπουμε την τυπική επιφάνεια εργασίας του λογισμικού. Η διεπαφή του Wireshark περιλαμβάνει πέντε, κύρια στοιχεία. Τα μενού των εντολών (command menus) είναι συνηθισμένα πτυσσόμενα, (pulldown) μενού που βρίσκονται στο επάνω μέρος του παραθύρου.

Προς το παρόν μας ενδιαφέρουν τα. μενού File και Capture. Το μενού File επιτρέπει την αποθήκευση δεδομένων για. πακέτα που έχουν συλληφθεί ή το άνοιγμα ενός αρχείου που περιέχει δεδομένα πακέτων που είχαν συλληφθεί προηγουμένως και την έξοδο από το Wireshark. Το μενού Capture επιτρέπει την εκκίνηση της σύλληψης πακέτων.



Εικόνα 22: Επιφάνεια εργασίας του Wireshark.

Το παράθυρο καταλόγου πακέτων (packet-listing window) παρουσιάζει μία περίληψη της μιας γραμμής για. Κάθε πακέτο που συλλαμβάνεται η οποία περιλαμβάνει τον αριθμό πακέτου (πρόκειται για. Αριθμό που απονέμεται από το Wireshark και *όχι* για έναν αριθμό πακέτου που περιέχεται στην επικεφαλίδα οποιουδήποτε πρωτοκόλλου), τον χρόνο σύλληψης του πακέτου, της διευθύνσεις πηγής και προορισμού του πακέτου, το είδος του πρωτοκόλλου και πληροφορία σχετική με το πρωτόκολλο η οποία περιέχεται στο πακέτο.

Ο κατάλογος των πακέτων μπορεί να ταξινομηθεί σύμφωνα με οποιαδήποτε από αυτές της κατηγορίες κάνοντας κλικ στο όνομα, της αντίστοιχης στήλης. Στο πεδίο είδος πρωτοκόλλου (protocol type) αναφέρεται το ανωτάτου επιπέδου πρωτόκολλο το οποίο έστειλε ή έλαβε ένα. Πακέτο, δηλαδή, το πρωτόκολλο που είναι η πηγή ή ο τελικός αποδέκτης αυτού του πακέτου.

Το παράθυρο λεπτομερειών επικεφαλίδας πακέτου (packet-header details window) παρέχει λεπτομέρειες σχετικά με το επιλεγμένο στο παράθυρο packet-listing πακέτο. (Για να επιλέξετε ένα από τα. Πακέτα του παραθύρου packet-listing, τοποθετείστε τον cursor πάνω από την μιας-γραμμής περίληψη του πακέτου στο παράθυρο packet-listing και κάντε αριστερό κλικ.)

Οι λεπτομέρειες αυτές περιλαμβάνουν πληροφορίες σχετικά με το πλαίσιο Ethernet και το IP datagram που περιέχουν αυτό το πακέτο. Το ποσό των λεπτομερειών που παρουσιάζεται για το Ethernet και το επίπεδο IP μπορεί να επεκταθεί ή να. Ελαχιστοποιηθεί κάνοντας κλικ στο βέλος που δείχνει δεξιά ή της τα. Κάτω και βρίσκεται στα αριστερά της γραμμής του πλαισίου Ethernet ή του IP datagram στο παράθυρο packet-header details.

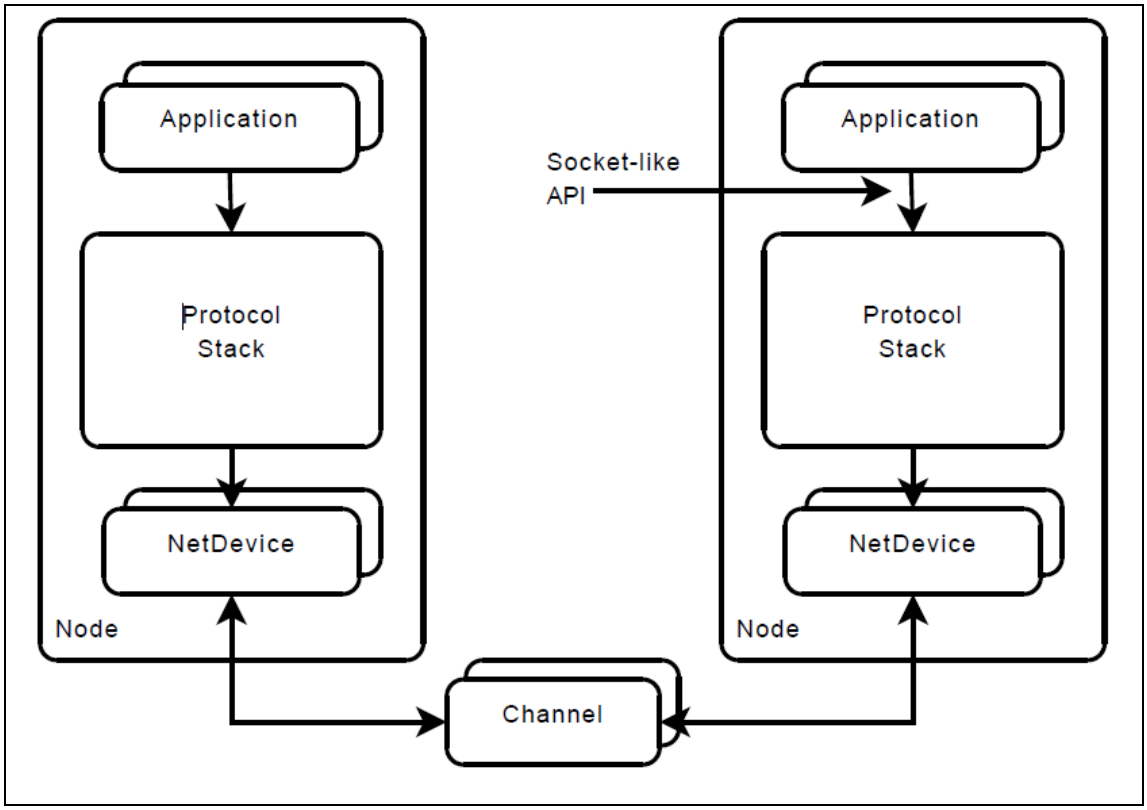
Εάν το πακέτο έχει μεταφερθεί με TCP ή UDP. Θα παρουσιαστούν και οι λεπτομέρειες που αφορούν το TCP ή το UDP. Οι οποίες μπορούν να επεκταθούν ή να ελαχιστοποιηθούν με παρόμοιο τρόπο. Τέλος, λεπτομέρειες παρέχονται της για το ανωτάτου επιπέδου πρωτόκολλο το οποίο έστειλε ή έλαβε αυτό το πακέτο.

Το παράθυρο περιεχομένου πακέτου (packet-contents window) παρουσιάζει ολόκληρο το περιεχόμενο της συλλαμβανόμενου πλαισίου και σε μορφή ASCII και σε δεκαεξαδική μορφή.

Της το επάνω μέρος της διεπαφής Wireshark βρίσκεται το πεδίο του φίλτρου παρουσίασης πακέτων (packet display filter field) στο οποίο μπορούμε να εισάγουμε το όνομα της πρωτοκόλλου ή άλλη πληροφορία έτσι ώστε να φιλτράρουμε την πληροφορία που παρουσιάζεται στο παράθυρο packet-listing (και επομένως στα παράθυρα packet-header και packet-contents).

Αρχιτεκτονική NS-3

Το βασικό μοντέλο της αρχιτεκτονικής του NS-3 [13] βρίσκεται στην παρακάτω εικόνα.

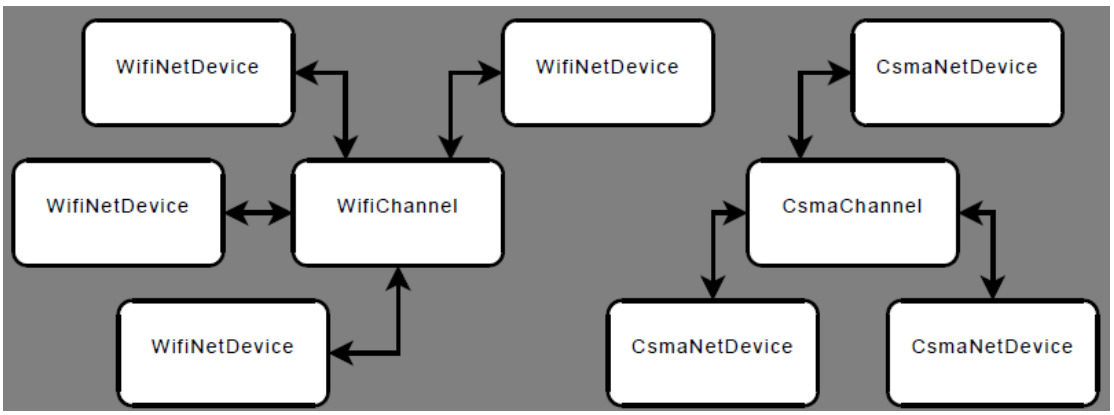


Εικόνα 23: Αρχιτεκτονική του NS-3

Από το παραπάνω σχήμα μπορούμε να αναφέρουμε τα ακόλουθα σημαντικά στοιχεία:

- Node (κόμβος): η μητρική πλακέτα του υπολογιστή με μνήμη RAM, επεξεργαστή CPU και I/O διεπαφές.
- Application (εφαρμογή): μια γεννήτρια πακέτων και καταναλωτών που μπορεί να τρέξει σ' ένα κόμβο και να επικοινωνεί μ' ένα σύνολο στοιβών δικτύου.
- Socket: η διασύνδεση μεταξύ μιας εφαρμογής και μιας στοίβας δικτύου.
- NetDevice: μια κάρτα δικτύου η οποία μπορεί να συνδεθεί με μία I/O διεπαφή ενός κόμβου καθώς και
- Channel (κανάλι): μια φυσική σύνδεση μεταξύ ενός συνόλου NetDevice αντικειμένων.

Τα NetDevices συνδέονται με τα κανάλια του αντίστοιχου τύπου, όπως φαίνεται και στην παρακάτω εικόνα:



Εικόνα 24: ΣύνδεσηNetDevices με κανάλια

Παρακάτω παρατίθενται κάποια μοντέλα της αρχιτεκτονικής του NS-3:

- Στοίβες δικτύου: arp, ipv4, icmpv4, udp, tcp (ipv6 underreview)
- Συσκευές: wifi, csma, point-to-point, bridge
- Μοντέλα λαθών και ουρές
- Εφαρμογές: udp echo, on/off, sink
- Mobility models: random walk, etc.
- Δρομολόγηση: olsr, static global.

Wifi μοντέλα

Ένα παράδειγμα των παραπάνω μοντέλων είναι τα wifi, για τα οποία ισχύουν τα παρακάτω:

- Είναι νέο μοντέλο, το οποίο δημιουργήθηκε από τις προδιαγραφές του 802.11
- Είναι ακριβές μοντέλο του MAC
- Περιέχει σύνολο αλγορίθμων για έλεγχο του ρυθμού (ARF, ideal, AARF, Minstrel, etc.)
- Δεν είναι τόσο αργά όσο τα 802.11a PHY.

Ως κάποιες νέες συνεισφορές στον τομέα των wifi μοντέλων από την πλευρά των προγραμματιστών είναι οι παρακάτω:

- Πανεπιστήμιο της of Florence: 802.11n, EDCA, frame aggregation, block ack
- Russian Academy of Sciences: 802.11s, HWMP routing protocol
- Boeing: 802.11b channel models, validation
- Deutsche Telekom Laboratories: PHY modelization, validation
- Karlsruhe Institute of Technology: PHY modelization (Rayleigh, Nakagami).

Τοπολογία

Helper/ Container API

Ο σκοπός μας στην αρχιτεκτονική του NS-3 είναι να απλοποιήσουμε τη διαδικασία δημιουργίας τοπολογιών με χρήση επαναλαμβανόμενων προτύπων και να κάνουμε την περιγραφή της υψηλότερου επιπέδου ώστε να γίνει πιο εύκολη η ανάγνωση και η κατανόησή της.

Η ιδέα για απλοποίηση της διαδικασίας είναι πολύ απλή και έγκειται στα 2 παρακάτω βήματα:

- Το σύνολο των αντικειμένων αποθηκεύεται σε Containers και
- Κάθε λειτουργία κωδικοποιείται σ' ένα αντικείμενο Helper και εφαρμόζεται σ' ένα Container.

Οι λειτουργίες του Helper συνοψίζονται στις εξής:

- Δεν είναι γενικός, δηλαδή διαφορετικοί helpers παρέχουν διαφορετικές λειτουργίες,
- Δεν επιτρέπουν επαναχρησιμοποίηση του κώδικα– αντίθετα προσπαθούν να ελαχιστοποιήσουν το ποσό του κώδικα που γράφεται και
- Παρέχουν ένα είδος συντακτικής βοήθειας προς τον χρήστη, κάνουν δηλαδή τον κώδικα περισσότερο ευανάγνωστο.

Κάποια τυπικά containers και helpers είναι τα παρακάτω:

Containers:

- NodeContainer
- NetDeviceContainer
- Ipv4AddressContainer

Helpers:

- InternetStackHelper
- WifiHelper
- MobilityHelper
- OlsrHelper
- Κάθε μοντέλο παρέχει μία helperκλάση.

Προγραμματισμός

Παρακάτω δημιουργούμε ένα ζευγάρι κόμβων:

```
NodeContainer csmaNodes;
csmaNodes.Create (2);
NodeContainer wifiNodes;
wifiNodes.Add (csmaNodes.Get (1));
wifiNodes.Create (3);
```

Όπου σε κάθε μία γραμμή έχουμε τα εξής:

1. Δημιουργία κενού κόμβου container.

2. Δημιουργία2 κόμβων.

3. Δημιουργία κενού κόμβου container.

4. Πρόσθεση σε αυτόν του ήδη υπάρχοντος κόμβου.

5. Δημιουργία κάποιων επιπρόσθετων κόμβων.

Όσον αφορά το δίκτυο csma έχουμε τα ακόλουθα:

```
NetDeviceContainer csmaDevices;
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate",
        StringValue ("5Mbps"));
csma.SetChannelAttribute ("Delay",
        StringValue ("2ms"));
csmaDevices = csma.Install (csmaNodes);
```

Όπου σε κάθε μία γραμμή έχουμε τα εξής:

1. Δημιουργία κενής συσκευής container.

2. Δημιουργία csma helper.

3. Δήλωση ρυθμού δεδομένων.

4. Δήλωση καθυστέρησης.

5. Δημιουργία csma συσκευών και καναλιού.

Όσον αφορά το ζευγάρι των wifi διεπαφών έχουμε:

1. Ρυθμίζουμε το κανάλι wifi

```
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
```

2. Δημιουργούμε ad-hoc συσκευές στο κανάλι αυτό

```
NetDeviceContainer wifiDevices;
WifiHelper wifi = WifiHelper::Default ();
wifiDevices = wifi.Install (wifiPhy, wifiNodes);
```

Κεφάλαιο 5: Προσομοίωση ασύρματων δικτύων με τον NS-3

Σε αυτό το κεφάλαιο θα περιγράψουμε την υλοποίηση ενός δικτύου που σχεδιάσαμε και προσομοιώσαμε στον NS – 3.

Γενική περιγραφή

Το σενάριο που υλοποιήσαμε είναι το εξής: δημιουργούμε και ρυθμίζουμε ένα σύνολο από κόμβους ώστε να χρησιμοποιούν το πρωτόκολλο 802.11b για να επικοινωνούν μεταξύ τους, δηλαδή να δημιουργούν διαδικτυακή κίνηση (traffic).

Με βάση το πως ο NS-3 χειρίζεται την έννοια «κόμβος», θεωρούμε πως η επικοινωνία γίνεται με ισόποσα 802.11b NIC, ένα ανά κόμβο. Το δίκτυο λειτουργεί σε ad – hoc κατάσταση, δηλαδή όπως θα λειτουργούσε και το ασύρματο δίκτυο πχ μιας πολυκατοικίας ή ενός αεροδρομίου.

Κατά την εκτέλεση του σεναρίου, κάθε κόμβος στέλνει ανά διαστήματα πακέτα των 1000 bytestον άλλον και εκτυπώνεται στην οθόνη ένα σχετικό μήνυμα κάθε φορά που γίνεται η λήψη του πακέτου από τον κόμβο - παραλήπτη.

Επίσης, το σενάριο είναι ρυθμισμένο ώστε να μην λαμβάνει υπόψη την απόσταση μεταξύ των κόμβων και την ισχύ μετάδοσης του σήματος από τον αποστολέα στον παραλήπτη.

Ανάλυση πηγαίου κώδικα

Το σενάριο της προσομοίωσης είναι, ως γνωστόν, ένα C++ πρόγραμμα που μεταγλωττίζεται και εκτελείται από τον προσομοιωτή. Συνεπώς, ακολουθούμε τους κανόνες και το συντακτικό που γνωρίζουμε από την C++.

Το πρόγραμμα ξεκινάει με την πρόταση συμπερίληψης απαραίτητων βιβλιοθηκών που παρέχει ο προσομοιωτής και αφορούν ασύρματα δίκτυα.

```
#include "ns3/core-module.h"
#include "ns3/node-module.h"
#include "ns3/simulator-module.h"
#include "ns3/common-module.h"
#include "ns3/helper-module.h"
#include "ns3/mobility-module.h"
#include "ns3/contrib-module.h"
#include "ns3/wifi-module.h"
```

Στην συνέχεια συμπεριλαμβάνουμε απαραίτητες βιβλιοθήκες της C++ που μας παρέχουν χρήσιμα προγραμματιστικά εργαλεία.

```
#include <iostream>
#include <fstream>
#include <vector>
#include<string>
```

Έπειτα ορίζουμε έναν Logger που καταγράφει την δραστηριότητα της μεταγλώττισης και της εκτέλεσης. Αυτό μας είναι χρήσιμο στις περιπτώσεις που θέλουμε να εντοπίσουμε ένα λογικό ή κάποιο δυσνόητο συντακτικό λάθος στον κώδικα μας, ένας debugger ουσιαστικά.

```
NS_LOG_COMPONENT_DEFINE ("WifiSimpleAdhoc");
```

Έπειτα ορίζουμε, το Namespace στο οποίο θα «φαίνεται» το πρόγραμμα μας.

```
usingnamespace ns3;
```

Ακολουθώντας το ANSI πρότυπο, βάζουμε τον ορισμό των συναρτήσεων πριν από την κύρια (main)συνάρτηση του προγράμματος μας.

Η πρώτη είναι η ReceivePacketπου εκτελείται κάθε φορά που λαμβάνεται ένα πακέτο και τυπώνει το μήνυμα λήψης του πακέτου. Αυτό το μήνυμα αναφέρει το μέγεθος του πακέτου, καθώς και τον αποστολέα του, με την μορφή της διεύθυνσης IP. Το όρισμα που παίρνει θα εξηγηθεί παρακάτω.

```
voidReceivePacket(Ptr<Socket>socket)
{
  Ptr<Packet> packet;
  Address from;

packet= socket->RecvFrom(from);
  Ipv4Address ipv4From =
InetSocketAddress::ConvertFrom(from).GetIpv4();

  NS_LOG_INFO("Received packet, "<< packet->GetSize()<<" bytes from
"<< ipv4From);

  NS_LOG_UNCOND(/*ipv4Dest   <<   */"Received   packet,   "<<   packet-
>GetSize()<<" bytes from "<< ipv4From);}
```

Η επόμενη είναι η Generate Traffic. Σε αυτήν εξετάζεται η ουρά ενός κόμβου. Εάν υπάρχουν μηνύματα – πακέτα να αποσταλούν καλείται η συνάρτηση Send() που κάνει την αποστολή του μηνύματος διαμέσου του socket που έχει οριστεί και το οποίο «ενώνει» αποστολέα και παραλήπτη.

Το socket ορίζεται από ένα στιγμιότυπο της κλάσης Socketπου λειτουργεί αντίστοιχα, αλλά όχι πραγματικά, με τα sockets της C, που χρησιμοποιούνται στις πραγματικές δικτυακές εφαρμογές.

```
staticvoid GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
uint32_t pktCount, Time pktInterval )
{
if(pktCount >0)
{
socket->Send (Create<Packet>(pktSize));
      Simulator::Schedule (pktInterval,&GenerateTraffic,
socket, pktSize,pktCount-1, pktInterval);
}
else
{
      socket->Close ();
}

}
```

Τέλος, ορίσαμε την συνάρτηση Run Simulation() η οποία όταν καλείται, ξεκινάει η εκτέλεση του σεναρίου. Αυτό γίνεται με την εντολή Run() που, όταν επιστρέψει, καλείται η Destroy() για να σταματήσει η εκτέλεση και να τερματιστεί η προσομοίωση.

```
void
RunSimulation (void)
{
  std::cout <<"Running the simulation"<< std::endl;
  Simulator::Run ();
```

```
  std::cout <<"Destroying the simulation"<< std::endl;
Simulator::Destroy ();

}
```

Στην συνάρτηση main, όπως είναι αναμενόμενο, γίνεται η εκτέλεση του σεναρίου. Ξεκινάμε, ορίζοντας παραμέτρους και μεταβλητές. Από αυτές, αξίζει να αναφερθούμε στις παρακάτω: η phyMode ορίζει το είδος του δικτύου μας σε ad-hoc, ασύρματο δίκτυο, εύρους 1mbps.

Η packetsize ορίζει το μέγεθος του πακέτου (εδώ είναι, όπως είπαμε, 1000Bytes), η numpakets μετράει τα πακέτα που ανταλλάσσονται στο δίκτυο και η cmd είναι η μεταβλητή που θα περιέχει της παραμέτρους που μπορεί να περνάει ο χρήστης στο πρόγραμμα.

Η μεταβλητή verbose είναι αυτή που καθορίζει αν κατά την εκτέλεση της προσομοίωσης θα εμφανίζονται όλα τα μηνύματα ή μόνο αυτά που έχουμε ορίσει εμείς. Στην πρώτη περίπτωση πρέπει η τιμή της να είναι **true**, αλλιώς **false**.

Η μεταβλητή nodes είναι αυτή που διατηρεί το πλήθος των κόμβων. Η τιμή της είναι όση ορίσει ο χρήστης κατά την εκτέλεση ή 2, αν ο χρήστης δεν καθορίζει τιμή με το όρισμα –nodes=<τιμή>.

```
std::string phyMode ("DsssRate1Mbps");
double rss =-80;// -dBm
uint32_t packetSize =1000;// bytes
uint32_t numPackets =1;
double interval =1.0;// seconds
bool verbose =true;
```

```
uint32_t    nLeftLeaf =5;
uint32_t    nRightLeaf =5;
uint32_t    nLeaf =0;// If non-zero, number of both left and right
uint16_t       port  =0;//  If  non  zero,  port  to  bind  to  for  anim
connection
std::string animFile;// Name of file for animation output
ApplicationContainer clientApps;
uint32_t nodes = 2;
CommandLine cmd;
```

Η εντολή addvalue που καλείται επί της μεταβλητής cmd, ορίζει σαν παραμέτρους χρήσης τα παρακάτω. Για λόγους μελλοντικής χρήσης, ορίσαμε να μπορούν όλες οι μεταβλητές να μπορούν να πάρουν τιμές από τον χρήστη, από την γραμμή εντολών.

```
cmd.AddValue ("phyMode","Wifi Phy mode", phyMode);
  cmd.AddValue ("rss","received signal strength", rss);
  cmd.AddValue  ("packetSize","size  of  application  packet  sent",
packetSize);
  cmd.AddValue   ("numPackets","number   of   packets   generated",
numPackets);
  cmd.AddValue  ("interval","interval  (seconds)  between  packets",
interval);
  cmd.AddValue ("verbose","turn on all WifiNetDevice log components",
verbose);
  cmd.AddValue  ("nLeftLeaf","Number  of  left  side  leaf  nodes",
nLeftLeaf);
  cmd.AddValue  ("nRightLeaf","Number  of  right  side  leaf  nodes",
nRightLeaf);
  cmd.AddValue ("nLeaf","Number of left and right side leaf nodes",
nLeaf);
  cmd.AddValue ("port","Port Number for Remote Animation", port);
```

```
  cmd.AddValue ("animFile","File Name for Animation Output",
animFile);

cmd.AddValue ("nodes","Number of nodes", nodes);
```

Έπειτα, ζητάμε από το πρόγραμμα να είναι σε θέση να διαβάσει αυτές τις παραμέτρους την ώρα της εκτέλεσης.

```
cmd.Parse (argc,argv);
if(nLeaf >0)
{
nLeftLeaf=nLeaf;
nRightLeaf=nLeaf;

}
```

Ορίζουμε το χρονικό διάστημα μεταξύ της αποστολής δύο πακέτων.

```
TimeinterPacketInterval=Seconds(interval);
```

Απενεργοποιούμε το fragmentation για πακέτα μικρότερα των 2200 bytes και ορίζουμε το non-unicast data rate να είναι ίδιο με το unicast.

```
Config::SetDefault
("ns3::WifiRemoteStationManager::FragmentationThreshold", StringValue
("2200"));
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold",
StringValue ("2200"));



Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
```

```
StringValue(phyMode));
```

Δημιουργούμε τους κόμβους.

```
  NodeContainer c;

  c.Create (nodes);
```

Ορίζουμε οι κόμβοι μας να έχουν ασύρματες δικτυακές διεπαφές (NIC).

```
WifiHelper wifi;
if(verbose)
{
      wifi.EnableLogComponents ();// Turn on all Wifi logging
}

  wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();

wifiPhy.Set ("RxGain", DoubleValue (0));

wifiPhy.SetPcapDataLinkType(YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
```

Δημιουργούμε το κανάλι επικοινωνίας, που στην περίπτωση μας είναι ασύρματο.

```
YansWifiChannelHelper wifiChannel ;
  wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
  wifiChannel.AddPropagationLoss
("ns3::FixedRssLossModel","Rss",DoubleValue(rss));
wifiPhy.SetChannel (wifiChannel.Create ());
```

Απενεργοποιούμε τα QoS και το rate control.

```
  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
```

```
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode",StringValue(phyMode),"ControlMode",StringValue(phyMode));
```

Ορίζουμε το δίκτυο μας να λειτουργεί ad – hoc.

```
wifiMac.SetType ("ns3::AdhocWifiMac");

  NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, c);
```

Στην συνέχεια ορίζουμε το μοντέλο βάσει του οποίου κινούνται οι κόμβοι εντός του δικτύου.

```
MobilityHelper mobility;
  Ptr<ListPositionAllocator>              positionAlloc              =
CreateObject<ListPositionAllocator>();
positionAlloc->Add (Vector (0.0,0.0,0.0));
positionAlloc->Add (Vector (5.0,0.0,0.0));
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install (c);
```

Ορίζουμε το TCP/IP πρωτόκολλο που χρησιμοποιεί το δίκτυο. Έτσι ορίζουμε IP διευθύνσεις, sockets επικοινωνίας μεταξύ των κόμβων, τα προσδένουμε (bind) στους αντίστοιχους κόμβους, broadcast διευθύνσεις και άλλα.

```
InternetStackHelper internet;
  internet.Install (c);


  Ipv4AddressHelper ipv4;
  NS_LOG_INFO ("Assign IP Addresses.");
```

```
  ipv4.SetBase ("10.1.1.0","255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);

  TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
  Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);
  InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny
(),80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

  Ptr<Socket> source = Socket::CreateSocket (c.Get (1), tid);
  InetSocketAddress remote = InetSocketAddress (Ipv4Address
("255.255.255.255"),80);
source->SetAllowBroadcast (true);

source->Connect(remote);
```

Έπειτα, ορίζουμε να γίνεται καταγραφή της προσομοίωσης ώστε να ήμαστε αργότερα σε θέση, αν το επιθυμούμε, να κάνουμε οπτικοποίηση του σεναρίου μας. Σημειωτέον, το δίκτυο μας δεν θα μπορεί να οπτικοποιηθεί στον NetAnim, αλλά μπορούμε να εισάγουμε τα παραγόμενα pcap αρχεία στο λογισμικό wireshark.

```
wifiPhy.EnablePcap ("my-ad-hoc-wifi", devices);
```

Ορίζουμε ένα διευκρινιστικό μήνυμα.

```
  NS_LOG_UNCOND ("Testing "<< numPackets <<" packets sent with
receiver rss "<< rss );
```

Παρακάτω, ορίζουμε την χρονική διάρκεια της προσομοίωσης. Η μονάδα μέτρησης είναι ο χρόνος που χρειάζεται για να μειωθεί κατά ένα η τιμή της μεταβλητής duration.

```
int duration =20;
        Address dest;
        Ipv4Address ipv4Dest;

Ptr<Packet> packet;
```

Οι υπόλοιπες μεταβλητές, μας βοηθούν στον καθορισμό της IP διεύθυνσης του παραλήπτη του κάθε πακέτου. Στην συνέχεια, για όσο διαρκεί η προσομοίωση, αλλάζουμε τυχαία ζεύγη αποστολέα παραλήπτη.

```
            recvSink= Socket::CreateSocket (c.Get (rand()%nodes),
tid);
            local= InetSocketAddress (Ipv4Address::GetAny (),80);
            recvSink->Bind (local);
            recvSink->SetRecvCallback (MakeCallback
(&ReceivePacket)/*,recvSink*/);

            source= Socket::CreateSocket (c.Get (rand()%nodes), tid);
            remote= InetSocketAddress (Ipv4Address
("255.255.255.255"),80);
            source->SetAllowBroadcast (true);

            source->Connect (remote);
```

Ζητάμε από τον προσομοιωτή να αναλάβει το scheduling των γεγονότων που θα συμβαίνουν κατά την προσομοίωση και μειώνουμε την duration.

```
Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
                                Seconds (1.0),&GenerateTraffic,
source, packetSize, numPackets, interPacketInterval);

duration--;
```

```

```

Τέλος, ζητάμε να ξεκινήσει η προσομοίωση και όταν αυτή τελειώσει, το πρόγραμμα τερματίζει επιτυχώς.

```
RunSimulation();

return0;
```

Κεφάλαιο 6: Επίλογος– Συμπεράσματα

Στην παρούσα πτυχιακή εργασία παρουσιάζονται βασικά χαρακτηριστικά των ασύρματων τοπικών δικτύων με βάση το πρωτόκολλο IEEE 802.11b. Μελετάμε και αναλύουμε μεθόδους και τρόπους λειτουργίας των δικτύων καθώς επίσης και εφαρμογές τους στην καθημερινότητά μας.

Πιο συγκεκριμένα παρουσιάζονται οι τρόποι επικοινωνίας των κόμβων ενός ασύρματου δικτύου, αναφέρονται οι μετρικές απόδοσεις των ασύρματων δικτύων, αναλύεται το 802.11 πρωτόκολλο με παρουσίαση όλων των στοιχείων, πλεονεκτημάτων και μη.

Αφού κατανοήσαμε και αναλύσαμε τα παραπάνω μεταφερθήκαμε στην δικτυακή προσομοίωση και τα εργαλεία της. Εξηγήσαμε μία σειρά από εργαλεία που χρησιμοποιούνται για αυτό το σκοπό, τη δικτυακή προσομοίωση. Φέραμε παραδείγματα χρήσης τους, παρουσιάσαμε εικόνες.

Επικεντρωθήκαμε όμως στον προσομοιωτή NS-3, με τον οποίο πραγματεύεται η πτυχιακή εργασία. Παρουσιάσαμε παλαιότερες εκδόσεις και αναλύσαμε την τρέχουσα με σκοπό να την κατανοήσουμε ώστε να μπορέσουμε να υλοποιήσουμε το πρόγραμμα προσομοίωσης ενός δικτύου και πιο συγκεκριμένα ενός ad- hoc δικτύου.

Δημιουργήσαμε και ρυθμίσαμε ένα σύνολο από κόμβους ώστε να χρησιμοποιούν το πρωτόκολλο 802.11b για να επικοινωνούν μεταξύ τους, δηλαδή να δημιουργούν διαδικτυακή κίνηση (traffic).

Κατά την εκτέλεση του σεναρίου, κάθε κόμβος στέλνει ανά διαστήματα πακέτα των 1000 byte στον άλλον και εκτυπώνεται στην οθόνη ένα σχετικό μήνυμα κάθε φορά που γίνεται η λήψη του πακέτου από τον κόμβο - παραλήπτη.

Βιβλιογραφία – Πηγές

[1]http://www.inrialpes.fr/planete/people/ernst/Documents/simulator.html#NETSIM

[2]Bojan Groelj. 1995. CPSim: a tool for creating scalable discrete event simulations. In *Proceedings of the 27th conference on Winter simulation* (WSC '95), Christos Alexopoulos and Keebom Kang (Eds.). IEEE Computer Society, Washington, DC, USA,579-583.

[3] http://www.nist.gov/itl/antd/emntg/rtdo_mesh.cfm

[4] http://www.kitchenlab.org/www/bmah/Software/Insane/

[5] Alexander Dupuy, Jed Schwartz, Yechiam Yemini, and David Bacon. 1990. NEST: a network simulation and prototyping testbed. Commun. ACM 33, 10 (October 1990), 63-74. DOI=10.1145/84537.84549 http://doi.acm.org/10.1145/84537.84549.

[6] S.P. Ahuja, "COMNET III: a network simulation laboratory environment for a course in communications networks," Frontiers in Education, Annual, pp. 1085-1088, 28th Annual Frontiers in Education - Vol 3 (FIE'98), 1998.

[7] http://www.cs.cornell.edu/skeshav/real/overview.html

[8] David M. Nicol. 1998. Special Issue on the Telecommunications Description Language. SIGMETRICS Perform. Eval.Rev. 25, 4 (March 1998), 3-. DOI=10.1145/274084.581192 http://doi.acm.org/10.1145/274084.581192.

[9] http://www.opnet.com/

[10] http://jist.ece.cornell.edu/

[11] http://www.omnetpp.org/

[12] http://www.nsnam.org/

[13] http://www.isi.edu/nsnam/ns/

[14] http://www.nsnam.org/wiki/index.php/NetAnim

[15] Wireshark, http://www.wireshark.org/

Παράρτημα

Εδώ μπορούμε να δούμε τον πηγαίο κώδικα της εφαρμογής.

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */

...

#include "ns3/core-module.h"
#include "ns3/node-module.h"
#include "ns3/simulator-module.h"
#include "ns3/common-module.h"
#include "ns3/helper-module.h"
#include "ns3/mobility-module.h"
#include "ns3/contrib-module.h"
#include "ns3/wifi-module.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

NS_LOG_COMPONENT_DEFINE ("WifiSimpleAdhoc");

usingnamespace ns3;

void ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  Address from;
```

```
packet= socket->RecvFrom(from);
  Ipv4Address ipv4From =
InetSocketAddress::ConvertFrom(from).GetIpv4();

  NS_LOG_INFO("Received packet, "<< packet->GetSize()<<" bytes from
"<< ipv4From);
  NS_LOG_UNCOND(/*ipv4Dest << */"Received packet, "<< packet-
>GetSize()<<" bytes from "<< ipv4From);

}

staticvoid GenerateTraffic (Ptr<Socket> socket, Ptr<Socket> recvSink,
uint32_t pktSize,
uint32_t pktCount, Time pktInterval )
{
if(pktCount >0)
{
socket->Send (Create<Packet>(pktSize));
      Simulator::Schedule (pktInterval,&GenerateTraffic,
socket, recvSink, pktSize,pktCount-1, pktInterval);
}
else
{
socket->Close ();
}
}

void
RunSimulation (void)
{
  std::cout <<"Running the simulation"<< std::endl;
```

```
  Simulator::Run ();

  std::cout <<"Destroying the simulation"<< std::endl;
  Simulator::Destroy ();
}

int main (int argc,char*argv[])
{

std::string phyMode ("DsssRate1Mbps");
double rss =-80;// -dBm
uint32_t packetSize =1000;// bytes
uint32_t numPackets =1;
uint32_t nodes =2;
double interval =1.0;// seconds
bool verbose =true;
uint32_t    nLeftLeaf =5;
uint32_t    nRightLeaf =5;
uint32_t    nLeaf =0;// If non-zero, number of both left and right
uint16_t    port =0;// If non zero, port to bind to for anim
connection
std::string animFile;// Name of file for animation output
  ApplicationContainer clientApps;
  CommandLine cmd;

  cmd.AddValue ("phyMode","Wifi Phy mode", phyMode);
  cmd.AddValue ("rss","received signal strength", rss);
  cmd.AddValue ("packetSize","size of application packet sent",
packetSize);
  cmd.AddValue ("numPackets","number of packets generated",
numPackets);
```

```
  cmd.AddValue ("interval","interval (seconds) between packets",
interval);
  cmd.AddValue ("verbose","turn on all WifiNetDevice log components",
verbose);
  cmd.AddValue ("nLeftLeaf","Number of left side leaf nodes",
nLeftLeaf);
  cmd.AddValue ("nRightLeaf","Number of right side leaf nodes",
nRightLeaf);
  cmd.AddValue ("nLeaf","Number of left and right side leaf nodes",
nLeaf);
  cmd.AddValue ("port","Port Number for Remote Animation", port);
  cmd.AddValue ("animFile","File Name for Animation Output",
animFile);
  cmd.AddValue ("nodes","Number of nodes", nodes);

  cmd.Parse (argc,argv);
if(nLeaf >0)
{
nLeftLeaf= nLeaf;
nRightLeaf= nLeaf;
}

// Convert to time object
  Time interPacketInterval = Seconds (interval);

// disable fragmentation for frames below 2200 bytes
  Config::SetDefault
("ns3::WifiRemoteStationManager::FragmentationThreshold", StringValue
("2200"));
// turn off RTS/CTS for frames below 2200 bytes
```

```
  Config::SetDefault
("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200"));
// Fix non-unicast data rate to be the same as that of unicast
  Config::SetDefault
("ns3::WifiRemoteStationManager::NonUnicastMode",
                      StringValue (phyMode));

  NodeContainer c;
  c.Create (nodes);// create the nodes

// The below set of helpers will help us to put together the wifi
NICs we want
  WifiHelper wifi;
if(verbose)
{
      wifi.EnableLogComponents ();// Turn on all Wifi logging
}
  wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

  YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
// This is one parameter that matters when using FixedRssLossModel
// set it to zero; otherwise, gain will be added
  wifiPhy.Set ("RxGain", DoubleValue (0));
// ns-3 supports RadioTap and Prism tracing extensions for 802.11b
  wifiPhy.SetPcapDataLinkType
(YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

  YansWifiChannelHelper wifiChannel ;
  wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
```

```
// The below FixedRssLossModel will cause the rss to be fixed
regardless
// of the distance between the two stations, and the transmit power
  wifiChannel.AddPropagationLoss
("ns3::FixedRssLossModel","Rss",DoubleValue(rss));
  wifiPhy.SetChannel (wifiChannel.Create ());

// Add a non-QoS upper mac, and disable rate control
  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode",StringValue(phyMode),
"ControlMode",StringValue(phyMode));
// Set it to adhoc mode
  wifiMac.SetType ("ns3::AdhocWifiMac");
  NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, c);

// Note that with FixedRssLossModel, the positions below are not
// used for received signal strength.
  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc =
CreateObject<ListPositionAllocator>();
positionAlloc->Add (Vector (0.0,0.0,0.0));
positionAlloc->Add (Vector (5.0,0.0,0.0));
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (c);

  InternetStackHelper internet;
  internet.Install (c);

  Ipv4AddressHelper ipv4;
```

```
  NS_LOG_INFO ("Assign IP Addresses.");
  ipv4.SetBase ("10.1.1.0","255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);

  TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
  Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);
  InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny
(),80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

  Ptr<Socket> source = Socket::CreateSocket (c.Get (5), tid);
  InetSocketAddress remote = InetSocketAddress (Ipv4Address
("255.255.255.255"),80);
source->SetAllowBroadcast (true);
source->Connect (remote);

// Tracing
  wifiPhy.EnablePcap ("my-ad-hoc-wifi", devices);

// Output what we are doing
  NS_LOG_UNCOND ("Testing "<< numPackets  <<" packets sent with
receiver rss "<< rss );


int duration =20;
        Address dest;
        Ipv4Address ipv4Dest;
        Ptr<Packet> packet;

while(duration >0){
```

```
recvSink= Socket::CreateSocket (c.Get (rand()%nodes), tid);
local= InetSocketAddress (Ipv4Address::GetAny (),80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback
(&ReceivePacket)/*,recvSink*/);

source= Socket::CreateSocket (c.Get (rand()%nodes), tid);
remote= InetSocketAddress (Ipv4Address ("255.255.255.255"),80);
source->SetAllowBroadcast (true);
source->Connect (remote);


/*ipv4Dest = InetSocketAddress::ConvertFrom(local).GetIpv4();


                NS_LOG_UNCOND("Destination:" << ipv4Dest << "-->");*/

                  Simulator::ScheduleWithContext (source->GetNode ()-
>GetId (),
                                    Seconds (1.0),&GenerateTraffic,
source, recvSink, packetSize, numPackets, interPacketInterval);

duration--;
}

  clientApps.Start (Seconds (0.0));
  clientApps.Stop (Seconds (10.0));

// Create the point-to-point link helpers
  PointToPointHelper pointToPointRouter;
```

```
pointToPointRouter.SetDeviceAttribute  ("DataRate", StringValue
("10Mbps"));
  pointToPointRouter.SetChannelAttribute ("Delay", StringValue
("1ms"));
  PointToPointHelper pointToPointLeaf;
  pointToPointLeaf.SetDeviceAttribute    ("DataRate", StringValue
("10Mbps"));
  pointToPointLeaf.SetChannelAttribute   ("Delay", StringValue
("1ms"));

  PointToPointDumbbellHelper d(nLeftLeaf, pointToPointLeaf,
nRightLeaf, pointToPointLeaf,
pointToPointRouter);

// Install Stack
  InternetStackHelper stack;
  d.InstallStack (stack);

// Assign IP Addresses
  d.AssignIpv4Addresses (Ipv4AddressHelper
("10.4.1.0","255.255.255.0"),
                    Ipv4AddressHelper ("10.2.1.0","255.255.255.0"),
                    Ipv4AddressHelper ("10.3.1.0","255.255.255.0"));



  std::cout <<"Stopping the Simulation..."<< std::endl;
  RunSimulation ();

return0;
```

```
}
```