ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

" Μελέτη, σχεδίαση, υλοποίηση και αξιολόγηση δικτυακής υποδομής βασισμένης στο πρότυπο DVB-T για την βελτιστοποίηση των επιδόσεων αξιοποιώντας μηχανισμούς παροχής ποιότητας υπηρεσίας (DiffServ)"

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: **ΓΕΩΡΓΑΡΑΣ ΕΛΕΥΘΕΡΙΟΣ**
ΗΜΕΡΟΜΗΝΙΑ: **19/05/2008**

ΕΙΣΗΓΗΤΗΣ: **ΠΑΛΛΗΣ ΕΥΑΓΓΕΛΟΣ**

Σε αυτούς που στάθηκαν δίπλα μου,
με ιδιαίτερη εκτίμηση και αγάπη

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής αυτής εργασίας που υλοποιήθηκε στο εργαστήριο έρευνας και τεχνολογίας του Α.Τ.Ε.Ι. Κρήτης (ΠΑΣΙΦΑΗ) θα ήθελα να ευχαριστήσω όλους όσους παρείχαν υλικοτεχνική και ψυχολογική βοήθεια όποτε αυτή ήταν απαραίτητη.

Ειδικότερα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Δρ. Πάλλη Ευάγγελο για την βοήθεια και την υποστήριξη που μου παρείχε κατά την διάρκεια της πτυχιακής, ακόμη θα ήθελα να ευχαριστήσω τους ερευνητές καθηγητές του εργαστηρίου ΠΑΣΙΦΑΗ κ. Μαρκάκη Ευάγγελο, κ. Μαστοράκη Γεώργιο, κ. Ξυλούρη Γεώργιο και κ. Σιδέρη Ανάργυρο για την πολύτιμη βοήθεια τεχνική και ψυχολογική που έλαβα αλλά και για τις γνώσεις που μου πρόσφεραν απλόχερα κατά την διάρκεια υλοποίησης αυτής της πτυχιακής εργασίας.

Ακόμη θα ήθελα να ευχαριστήσω την οικογένεια μου για την πολύτιμη υλική και ψυχολογική υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια αλλά και κατά τη διάρκεια υλοποίησης αυτής της πτυχιακής.

Ηράκλειο, Μάιος 2008

Γεωργαράς Ελευθέριος

Περίληψη

Η Πτυχιακή αυτή εργασία επικεντρώνεται στη μελέτη, σχεδίαση, υλοποίηση και αξιολόγηση δικτυακής υποδομής βασισμένης στο πρότυπο DVB-T. Έχοντας ένα δίκτυο με περιορισμένο εύρος ζώνης θα πρέπει να διασφαλίσουμε την ποιότητα των παρεχόμενων υπηρεσιών και να βελτιστοποιήσουμε τις επιδόσεις του δικτύου μας αξιοποιώντας μηχανισμούς παροχής ποιότητας υπηρεσιών (DiffServ). Εκτός από την διασφάλιση της ποιότητας των υπηρεσιών θα προσπαθήσουμε να κατηγοριοποιήσουμε τις υπηρεσίες/χρήστες μας και να δείξουμε την δυναμικότητα της διαχείρισης των διαθέσιμων πόρων. Η απόδοση του δικτύου θα αξιολογηθεί με αντικειμενικά κριτήρια μέσα από τα αποτελέσματα των πειραματικών μετρήσεων που θα γίνουν κατά την περάτωση της πτυχιακής αυτής εργασίας. Η υλοποίηση του πειραματικού δικτύου έγινε με εξοπλισμό του **ερευνητικού εργαστηρίου Τηλεπικοινωνιών και Δικτύων του Α.Τ.Ε.Ι Κρήτης (ΠΑΣΙΦΑΗ)**.

Περιεχόμενα

Λίστα Σχημάτων

1. Εισαγωγή

1.1. Γενικά

Η εισαγωγή της ψηφιακής τηλεόρασης εκτός από τα πλεονεκτήματα που έχει εισάγει στον χώρο της εκπομπής τηλεοπτικών προγραμμάτων συγκριτικά με την αναλογική τηλεόραση μπορούμε πλέον να έχουμε με ένα κανάλι επιστροφής και αμφίδρομες υπηρεσίες. Ένα ακόμη πλεονέκτημα είναι ότι μπορούμε εκτός από τηλεοπτικά προγράμματα να μεταδώσουμε και IP δεδομένα το πλεονέκτημα αυτό της ψηφιακής τηλεόραση είναι που εκμεταλλευτούμε για να δημιουργήσουμε ένα υβριδικό δίκτυο και να παρέχουμε δικτυακές υπηρεσίες πάνω από ένα σύστημα ψηφιακής τηλεόρασης.

1.2. Στόχοι

Η Πτυχιακή αυτή εργασία επικεντρώνεται στη μελέτη, σχεδίαση, υλοποίηση και αξιολόγηση δικτυακής υποδομής βασισμένης στο πρότυπο DVB-T. Έχοντας ένα δίκτυο με περιορισμένο εύρος ζώνης θα πρέπει να διασφαλίσουμε την ποιότητα των παρεχόμενων υπηρεσιών και να βελτιστοποιήσουμε τις επιδόσεις του δικτύου μας αξιοποιώντας μηχανισμούς παροχής ποιότητας υπηρεσιών (DiffServ). Εκτός από την διασφάλιση και ποιότητας των παρεχόμενων υπηρεσιών θα προσπαθήσουν να κάνουμε και μια κατηγοριοποίηση με βάση την υπηρεσία ή τον τελικό χρήστη που λαμβάνει μια υπηρεσία.

1.3. Οργάνωση

Αρχικά θα αναφερθούμε στις τεχνολογίες (ψηφιακή τηλεόραση, WLAN, VoIP, DiffServ) και στα πρωτόκολλα (UDP, TCP, NTP) που θα χρησιμοποιηθούν για την περάτωση αυτής της πτυχιακής εργασίας. Στη συνέχεια θα περιγράψουμε την τοπολογία του πειραματικού δικτύου και το πως αυτές οι τεχνολογίες μπορούν να συνεργαστούν αρμονικά αλλά και να έχουμε τον έλεγχο της διαχείρισης των διαθέσιμων πόρων. Τα σενάρια των πειραματικών μετρήσεων που θα υλοποιηθούν θα μας δώσουν αποτελέσματα σε μετρήσιμα μεγέθη για να μπορέσουμε να τα αξιολογήσουμε με αντικειμενικά κριτήρια όπως είναι useful Throughput (ωφέλιμος ρυθμός διαμεταγωγής), RTT (Round Trip Time - χρόνος

πλήρους διαδρομής), One Way Delay (μονόδρομη καθυστέρηση)[2] και Losses (Απώλειες) [3].

2. Τεχνολογίες και Πρωτόκολλα

2.1. Εισαγωγή

Σε αυτό το κεφάλαιο θα εξετάσουμε και θα αναλύσουμε ορισμένες τεχνολογίες όπως είναι η ψηφιακή τηλεόραση και ειδικότερα τα πρότυπα DVB-T (Digital Video Broadcasting-Terrestrial) και DVB-RCS (Digital Video Broadcasting - Return Channel via Satellite), WLAN (Wireless Local Area Network), VoIP (Voice Over IP) και η τεχνολογία DiffServ (Differentiated Services). Ενώ τα πρωτόκολλα που θα εξετάσουμε είναι τα πρωτόκολλα TCP (Transmission Control Protocol), UDP (User Datagram Protocol), NTP (Network Time Protocol).

2.2. Τεχνολογίες

2.2.1. Ψηφιακή Τηλεόραση και Broadcasting Δίκτυα

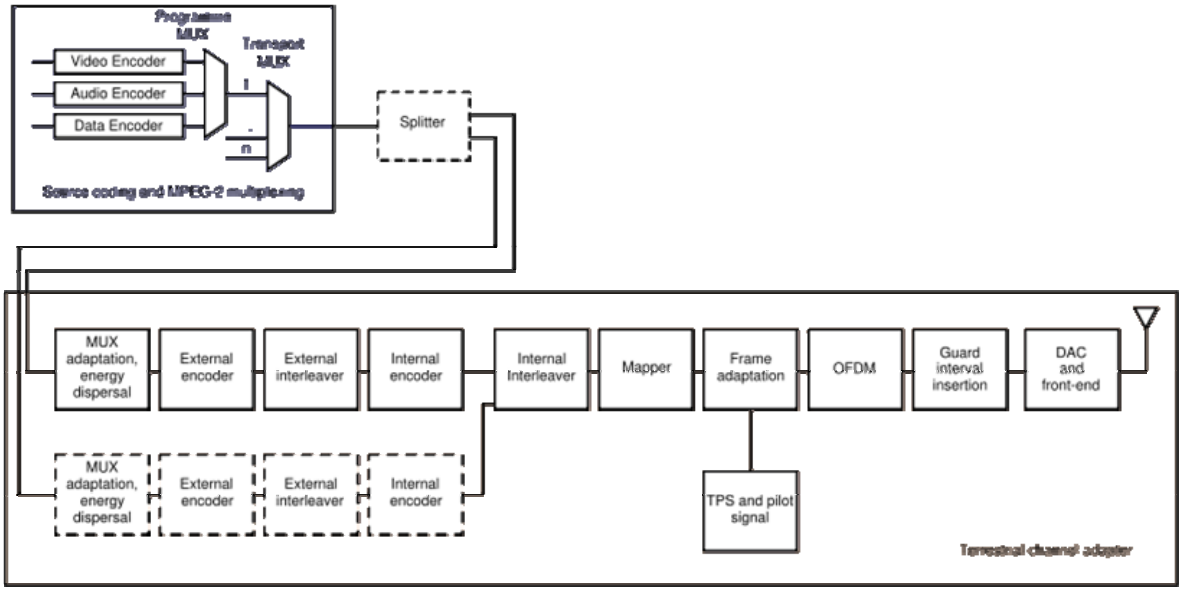
Η ψηφιακή τηλεόραση έχει εισέλθει στη ζωή μας εδώ και μερικά χρόνια φέρνοντας μαζί τα πλεονεκτήματα της. Λέγοντας ψηφιακή τηλεόραση μας έρχονται στο νου τα πρότυπα DVB (Digital Video Broadcasting), ATSC(Advanced Television Systems Committee) και ISDB (Integrated Services Digital Broadcasting) η πρώτη οικογένεια προτύπων χρησιμοποιείται στην Ευρώπη και είναι αυτό που θα ασχοληθούμε, η δεύτερη χρησιμοποιείται στη Αμερική και η τρίτη στην Ιαπωνία.

Η οικογένεια προτύπων DVB έχει ως μέλη της τα πρότυπα DVB-T(Digital Video Broadcasting-Terrestrial), DVB-S (Digital Video Broadcasting-Satellite), DVB-RCS (Digital Video Broadcasting - Return Channel via Satellite) και DVB-H (Digital Video Broadcasting - Handheld).

2.2.1.1. DVB-T

Το πρότυπο DVB-T (Digital Video Broadcasting-Terrestrial), χρησιμοποιείται για την μετάδοση ψηφιακής επίγειας τηλεόρασης. Σε ένα σύστημα DVB-T το οπτικοακουστικό

σήμα μεταδίδεται συμπιεσμένο, χρησιμοποιώντας διαμόρφωση πολλαπλών φερόντων στο σχήμα της πολυπλεξίας με ορθογώνια διαίρεση συχνότητας και κωδικοποίηση καναλιού (COFDM – Coded Orthogonal Frequency Division Multiplexing). Η μέθοδος κωδικοποίησης πηγής που χρησιμοποιείται είναι το πρότυπο MPEG-2, ενώ πρόσφατα υιοθετήθηκε και το H.264. Στα DVB-T συστήματα η μετάδοση επιτυγχάνεται εκπέμποντας σε ένα από τα κανάλια 21-69 της μπάντας των UHF, όπως και τα "παραδοσιακά" συστήματα αναλογικής μετάδοσης, έχοντας διαθέσιμο εύρος ζώνης 8 MHZ.



Σχήμα 1. Διάγραμμα ενός DVB Πομπού

Το πρότυπο αυτό υποστηρίζει μόνο μονόδρομη κίνηση όπου μπορεί να εκπεμφθούν επιπλέον υπηρεσίες όπως IPTV και IPRadio όπου δεν χρειάζεται αλληλεπίδραση με τον τελικό χρήστη. Για να μπορέσουμε να έχουμε επιπλέον IP [1] υπηρεσίες όπως HTTP, FTP, SMTP, Video On Demand ή Audio On Demand θα πρέπει να έχουμε ένα κανάλι επιστροφής ώστε να γίνονται οι αιτήσεις για την υπηρεσία που ζητά ο τελικός χρήστης. Αυτό το κανάλι επιστροφής θα μπορούσε να είναι ένα δίκτυο PSTN, ISDN, GSM, GPRS, DVB-S, χDSL ή οποιαδήποτε άλλη τεχνολογία εξυπηρετεί την εκάστοτε εφαρμογή.

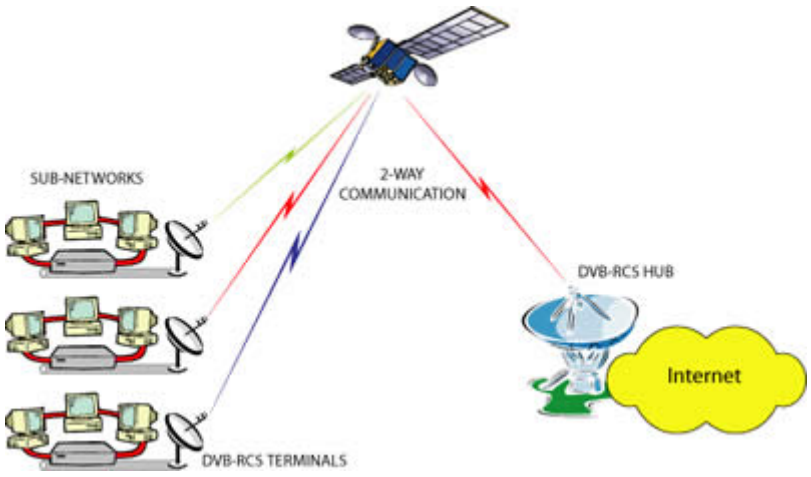
2.2.1.1.1. *Το MPEG Πρότυπο*

Η Ομάδα Ειδικών Κινούμενης Εικόνας ή MPEG(Moving Picture Experts Group) είναι μια ομάδα εργασίας ISO/ IEC, υπεύθυνη για την ανάπτυξη των τηλεοπτικών και

ακουστικών προτύπων κωδικοποίησης. Ένα από τα πρότυπα της οικογένειας MPEG το οποίο αξιοποιείται για τη δημιουργία τηλεοπτικών σημάτων είναι το MPEG-2. Το πρότυπο αυτό υποστηρίζει εφαρμογές με μεγάλες απαιτήσεις στην ποιότητα του video και προσφέρει μεταβλητούς ρυθμούς μετάδοσης από τη στιγμή που ο βαθμός συμπίεσης μεταβάλλεται αντιστρόφως ανάλογα με την πολυπλοκότητα των πλαισίων που κωδικοποιούνται. Έχει καθιερωθεί σε παγκόσμιο επίπεδο ως το πρότυπο για συμπίεση της ψηφιακής τηλεόρασης αφού παρέχει υψηλό βαθμό συμπίεσης διατηρώντας την εικόνα σε υψηλό επίπεδο ποιότητας. Το πρότυπο MPEG-2 σε συνδυασμό με το πρότυπο Multi Protocol Encapsulation (MPE) ενθυλακώνει τα IP δεδομένα σε ροές μεταφοράς οι οποίες αποστέλλονται μέσω δικτυακών πλατφόρμων DVB. Εκτός από την επίγεια ψηφιακή τηλεόραση το πρότυπο MPEG-2 αξιοποιείται ακόμα στην δορυφορική τηλεόραση (Satellite Digital TV), στην καλωδιακή τηλεόραση (Cable TV) και στους δίσκους DVD (Digital Video Disk).

2.2.1.2. DVB-RCS

Η τεχνολογία DVB-RCS[4] (Digital Video Broadcasting - Return Channel via Satellite) προσφέρει αμφίδρομες ευρυζωνικές υπηρεσίες μετάδοσης φωνής, δεδομένων, εικόνας και video μέσω του δορυφόρου. Το δίκτυο, το οποίο συνίσταται από το δορυφόρο, τον Κομβικό Σταθμό Εδάφους (HUB) και τα τερματικά των χρηστών (σταθερών και κινητών), διατάσσεται σε τοπολογία αστέρα και απεικονίζεται στο Σχήμα 2.



Σχήμα 2. Αρχιτεκτονική δικτύου DVB-RCS

Η τεχνολογία DVB-RCS χαρακτηρίζεται από δύο κανάλια που συνδέουν τον πάροχο υπηρεσιών και τον τελικό χρήστη. Τα δύο αυτά κανάλια είναι το κανάλι εκπομπής (Broadcast Channel), ένα ευρυζωνικό κανάλι εκπομπής μονής κατεύθυνσης, που περιλαμβάνει βίντεο, ήχο και εικόνα, εγκαθίσταται από τον φορέα παροχής υπηρεσιών προς τους χρήστες. Μπορεί να περιλαμβάνει και το Forward Interaction Path. Και το διαδραστικό Κανάλι (Interaction Channel), ένα αμφίδρομο διαδραστικό κανάλι εγκαθίσταται μεταξύ του φορέα παροχής υπηρεσιών / χρήστη και του χρήστη για σκοπούς αλληλεπίδρασης. όπου αποτελείται από το Return Interaction Path (Return Channel), αυτή η διαδρομή καθιερώνεται από τον χρήστη προς τον φορέα παροχής υπηρεσιών. Χρησιμοποιείται για να υποβάλει αιτήματα στον παροχέα υπηρεσιών / χρήστη, να απαντά σε ερωτήσεις ή να μεταφέρει δεδομένα. Και από το Forward Interaction Path αυτή η διαδρομή καθιερώνεται από τον φορέα παροχής υπηρεσιών προς τον χρήστη. Χρησιμοποιείται για να παρέχει πληροφορία από τον παροχέα υπηρεσιών / χρήστη προς το χρήστη(ες) και για οποιαδήποτε άλλη απαιτούμενη επικοινωνία για την παροχή διαδραστικών υπηρεσιών. Αυτή η διαδρομή μπορεί να εμπεριέχεται στο κανάλι εκπομπής. Είναι πιθανό αυτό το κανάλι να μη χρειάζεται σε συγκεκριμένες εφαρμογές που κάνουν χρήση του Broadcast Channel ως φορέα δεδομένων προς το χρήστη.

Το καινοτόμο σύστημα καναλιών επιστροφής διευκολύνει την αμφίδρομη επικοινωνία υψηλού ρυθμού μετάδοσης δεδομένων και δίνει πλέον τη δυνατότητα να χρησιμοποιηθεί για τη γρήγορη πρόσβαση στο Διαδίκτυο καθώς και για τις μεγάλες ανταλλαγές δεδομένων. Το σύστημα DVB-RCS το οποίο υποβλήθηκε στην τελική τυποποίηση από το ETSI το 2000, περιλαμβάνει το σύστημα δεδομένων DVB/MPEG-2 για την προωστική σύνδεση καθώς και το πρωτόκολλο πολλαπλής πρόσβασης MF-TDMA για τη σύνδεση επιστροφής. Πιο συγκεκριμένα, το φέρον μετάδοσης στην προωστική όδευση χρησιμοποιεί τη διαμόρφωση QPSK καθώς και συνδεμένους συνελικτικούς κώδικες Reed Solomon. Επιπλέον, μηνύματα σηματοδοσίας μεταφέρονται στα επιμέρους τερματικά που αφορούν λάθη συχνότητας και συγχρονισμού καθώς και την κατανομή του εύρους ζώνης (θυρίδες χρόνου και συχνότητας). Αυτά τα μηνύματα μεταφέρονται μέσω ενός ή περισσοτέρων πολυπλεγμένων καναλιών ελέγχου του δικτύου. Επομένως, το κάθε δορυφορικό τερματικό για τη μετάδοσή του στο κανάλι επιστροφής δεν έχει σταθερή συχνότητα ούτε σταθερό εύρος φάσματος εκπομπής αλλά οι προαναφερθείσες παράμετροι καθορίζονται από τον Κεντρικό Δορυφορικό Σταθμό Εδάφους. Επομένως, τα τερματικά λαμβάνουν πίνακες με πληροφορίες για την εύρεση των καναλιών ελέγχου τους και είναι παρόμοιοι με τον πίνακα πληροφοριών δικτύου (NIT), πίνακα περιγραφής υπηρεσιών (SDT), και τον πίνακα πληροφοριών γεγονότος (EIT) στη

μετάδοση DVB. Αναφορικά με την πορεία επιστροφής από τον επιμέρους χρήστη μέσω ενός δορυφορικού τερματικού, το τελευταίο λειτουργεί ως δρομολογητής-πολυπλέκτης για τις διάφορες πηγές δεδομένων, προς το διαδραστικό κεντρικό υπολογιστή στον Κεντρικό Δορυφορικό Σταθμό Εδάφους χρησιμοποιώντας ένα σχέδιο πολλαπλής πρόσβασης, MF-TDMA. Το MF-TDMA επιτρέπει σε μία ομάδα τερματικών να επικοινωνεί με τον κεντρικό κόμβο χρησιμοποιώντας συγκεκριμένες θυρίδες χρόνου/συχνότητας που απορρέουν από τη δυναμική ανάθεση εύρους ζώνης από τον κεντρικό σταθμό στα τερματικά με αποτέλεσμα το διαθέσιμο εύρος ζώνης να χρησιμοποιείται αποτελεσματικά.

Ο απαιτούμενος εξοπλισμός για τα άκρα του δικτύου περιγράφονται στην παράγραφο αυτή. Το δορυφορικό τερματικό(SIT) αποτελείται τυπικά από μια εξωτερική μονάδα (ODU) και από εσωτερική μονάδα (IDU).

Η εξωτερική μονάδα αποτελείται από μία κεραία που μπορεί να λειτουργεί στην Ku μπάντα συχνοτήτων. Τα μεγέθη που μπορεί να υποστηρίξει το προτεινόμενο μοντέλο εκτείνονται από 0.96m ως 1.8m. Οι συχνότητες λήψης βρίσκονται στην μπάντα μεταξύ 10.95 ως 12.75GHz. Η λήψη πραγματοποιείται με την χρήση ενός LNB που λειτουργεί επίσης στην ίδια μπάντα συχνοτήτων. Η μονάδα αυτή διαθέτει διεπαφή L-band η οποία συνδέεται απευθείας στην εσωτερική μονάδα μέσω ομοαξονικού καλωδίου. Η μετάδοση πραγματοποιείται με τη χρήση ενός High Power Block Up Converter που λειτουργεί επίσης στην Ku μπάντα συχνοτήτων. Στις περισσότερες των περιπτώσεων μετάδοσης χρησιμοποιείται ένας 2-Watt ενισχυτής. Οι συχνότητες μετάδοσης βρίσκονται στην μπάντα 14 ως 14.5 GHz. Παρομοίως με τη λήψη η έξοδος του συστήματος μετάδοσης μέσω ομοαξονικού καλωδίου συνδέεται με την εσωτερική μονάδα.

2.2.2. WLAN

Ως ασύρματο τοπικό δίκτυο (Wireless LAN - WLAN) εννοούμε τη σύνδεση δύο η περισσοτέρων υπολογιστών χωρίς την χρήση καλωδίων. Το WLAN χρησιμοποιεί την διαμόρφωση σήματος Spread Spectrum ή/και OFDM και τα ραδιοκύματα για να πετύχει επικοινωνία μεταξύ συσκευών σε μια μικρή περιοχή. Αυτό δίνει την δυνατότητα στους χρήστες να μπορούν να μετακινούνται μέσα στην ορισμένη περιοχή και να συνεχίζουν να είναι συνδεδεμένοι στο δίκτυο.

Για τους οικιακούς χρήστες, το ασύρματο δίκτυο έχει γίνει δημοφιλές λόγω της ευκολίας που έχει στην εγκατάσταση και την ελευθερία που δίνει στην τοποθέτηση του υπολογιστή, σε συνδυασμό με την διείσδυση στην αγορά τον φορητών υπολογιστών.

Το πρώτο WLAN αναπτύχθηκε από το Πανεπιστήμιο της Χαβάη και ονομαζόταν ALOHAnet. Το συγκεκριμένο δίκτυο περιλάμβανε πολλές καινοτομίες και υπήρξε η βάση για το τοπικό δίκτυο Ethernet και για τα ασύρματα δίκτυα. Χρησιμοποιούσε τοπολογία αμφίδρομου αστέρα για την επικοινωνία επτά υπολογιστών, οι οποίοι ήταν εγκατεστημένοι σε τέσσερα νησιά, με ένα κεντρικό υπολογιστή, χωρίς να χρησιμοποιούνται ενσύρματες γραμμές.

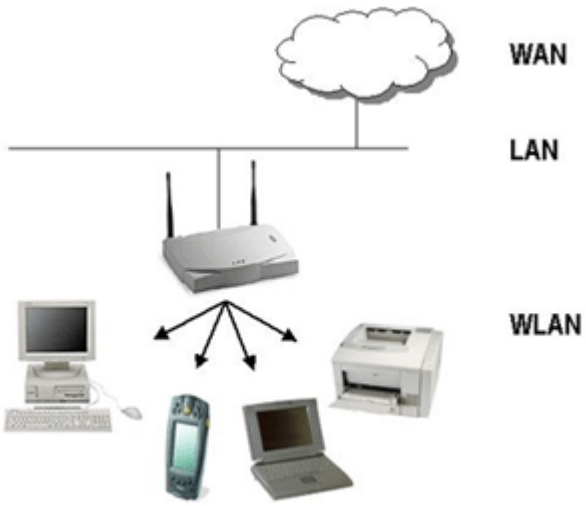
Στην αρχή της πορείας των ασύρματων δικτύων, το υλικό τους ήταν τόσο ακριβό που χρησιμοποιούνταν μόνο όπου η εγκατάσταση δομημένης καλωδίωσης ήταν δύσκολη ή αδύνατη. Κατά την ανάπτυξη των WLAN κυριάρχησαν τα πρότυπα IEEE 802.11χ (Σχήμα 3)

Protocol	Release Date	Op. Frequency	Throughput (Typ)	Data Rate (Max)	Modulation Technique	Range (Radius Indoor)	Range (Radius Outdoor)
Legacy	1997	2.4 GHz	0.9 Mbit/s	2 Mbit/s		~20 Meters	~100 Meters
802.11a	1999	5 GHz	23 Mbit/s	54 Mbit/s	OFDM	~35 Meters	~120 Meters
802.11b	1999	2.4 GHz	4.3 Mbit/s	11 Mbit/s	DSSS	~38 Meters	~140 Meters
802.11g	2003	2.4 GHz	19 Mbit/s	54 Mbit/s	OFDM	~38 Meters	~140 Meters
802.11n	June 2009	2.4 GHz 5 GHz	74 Mbit/s	248 Mbit/s		~70 Meters	~250 Meters
802.11y	June 2008	3.7 GHz	23 Mbit/s	54 Mbit/s		~50 Meters	~5000 Meters

Σχήμα 3. Περιγραφή των Πρωτοκόλλων 802.11

Τα WLAN έγιναν δημοφιλή κυρίως για την ευκολία χρήσης τους, το μικρό κόστος και την ευκολία ενσωμάτωσης τους με υπάρχοντα δίκτυα και εξοπλισμό. Πολλοί υπολογιστές που πωλούνται σήμερα και σχεδόν το σύνολο των φορητών υπολογιστών έρχονται με εξοπλισμό πρόσβασης σε ασύρματο δίκτυο.

Η τεχνολογία των ασύρματων τοπικών δικτύων, αν και παρέχει πολλές ευκολίες και παρέχει τα πλεονεκτήματα που αναφέρονται παραπάνω, έχει και τις αδυναμίες της. Για ορισμένες εγκαταστάσεις, η χρήση τοπικών δικτύων μπορεί να μην είναι η επιθυμητή λύση. Αυτό κυρίως οφείλετε σε περιορισμούς που εισαγάγει η ίδια η τεχνολογία όπως η ασφάλεια, ακτίνα χρήσης, αξιοπιστία, ταχύτητα και πολλούς άλλους περιορισμούς.



Σχήμα 4. Αρχιτεκτονική Ασύρματου Δικτύου

Όλα τα συστήματα που μπορούν να συνδεθούν στο ασύρματο μέσο σε ένα δίκτυο αναφέρονται ως σταθμοί. Όλοι οι σταθμοί είναι εξοπλισμένοι με ασύρματες κάρτες δικτύου. Οι σταθμοί χωρίζονται σε δύο κατηγορίες σημεία πρόσβασης και πελάτες.

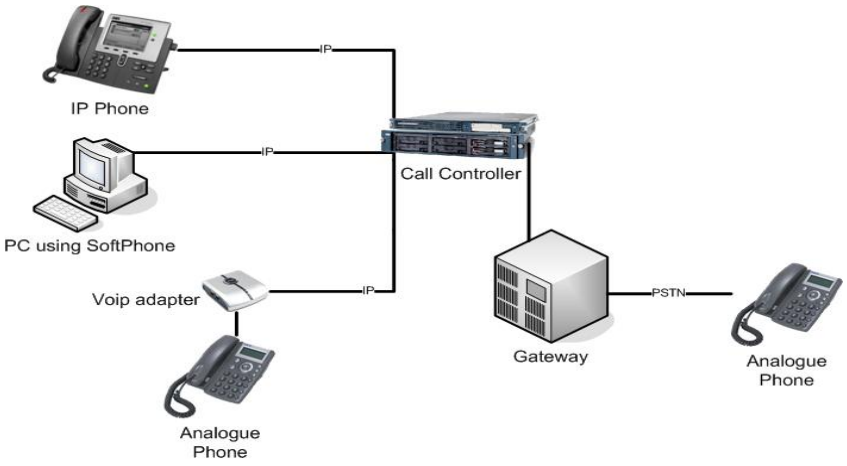
Τα σημεία πρόσβασης (Access Point - AP) είναι σταθμοί βάσης για το ασύρματο τοπικό δίκτυο. Εκπέμπουν και λαμβάνουν ραδιοσυχνότητες έτσι ώστε οι ασύρματες συσκευές να επικοινωνούν μεταξύ τους (δηλ. όλες οι ασύρματες συσκευές επικοινωνούν μέσω των AP). Το AP είναι το κέντρο και ο ελεγκτής του ασύρματου δικτύου, ορίζει την ακτίνα δράσης και τις παραμέτρους των πρωτοκόλλων που θα χρησιμοποιηθούν. Το AP συνδέει το ασύρματο δίκτυο με υπάρχοντα ενσύρματα δίκτυα. Πολλά AP συνδεδεμένα με ενσύρματο μέσο και λειτουργώντας σε διαφορετικά κανάλια συχνότητας συνδυάζονται για να επεκτείνουν την ακτίνα ενός ασύρματου τοπικού δικτύου.

Οι πελάτες σε ένα ασύρματο τοπικό δίκτυο μπορεί να είναι μεταφερόμενες συσκευές όπως φορητοί υπολογιστές, PDA (Personal digital Assistants), τηλέφωνα IP ή σταθερές συσκευές όπως προσωπικοί υπολογιστές, κ.α., εξοπλισμένα με μια ασύρματη κάρτα δικτύου.

2.2.3. VoIP

Το Voice over IP ή VoIP ή τηλεφωνία μέσω διαδικτύου χαρακτηρίζει μια ομάδα πρωτοκόλλων-τεχνολογιών (H.323, SIP), η οποία προσφέρει φωνητική συνομιλία σε πραγματικό χρόνο με σχετικά καλή ποιότητα πλέον και στην ουσία χωρίς κόστος. Οι

συνομιλίες αυτές παραδοσιακά γίνονταν αποκλειστικά μέσω PC που ήταν συνδεδεμένο με το Διαδίκτυο (Internet) και διέθετε μικρόφωνο, ακουστικά και το κατάλληλο λογισμικό. Η κλήση κατέληγε σε ένα άλλο, ανάλογα εξοπλισμένο, PC χωρίς να υπάρχει κάποια επιπλέον χρώση. Εκτός από κλήσεις από PC σε PC (Σχήμα 5)υπάρχει η δυνατότητα για σύνδεση μια VoIP συσκευής όπου συνδέεται απευθείας σε ένα δίκτυο (ενσύρματο ή ασύρματο) ή ακόμη και να χρησιμοποιηθεί μια αναλογική συσκευή με έναν VoIP adapter για να συνδεθεί στο δίκτυο ή να γίνουν κλήσεις σε ένα αναλογικό τηλέφωνο πάνω από το δίκτυο PSTN.



Σχήμα 5. Γενική Αρχιτεκτονική VoIP

2.2.4. Πρότυπα Ποιότητας Υπηρεσίας (QoS) - Πρότυπο Διαφοροποιημένων Υπηρεσιών (DiffServ)

2.2.4.1. Εισαγωγή

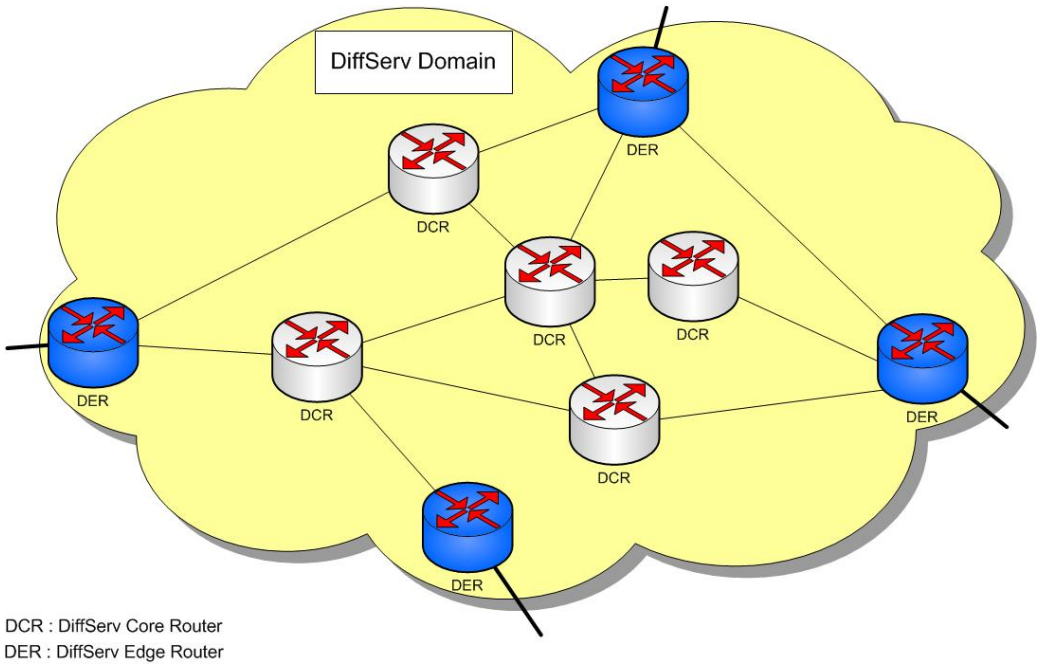
Στα σύγχρονα δίκτυα όπου μεταφέρονται διάφοροι τύποι υπηρεσιών (WWW, HTTP, FTP, SMTP) υπάρχει η ανάγκη για την εξασφάλιση της ποιότητας υπηρεσίας. Τα πρότυπα ποιότητας υπηρεσίας (Quality of Service – QoS)[7] μπορούν να εξασφαλίσουν την ποιότητα μια υπηρεσίας ή ενός χρήστη κάνοντας διαχείριση των δικτυακών πόρων διατηρώντας τις επιθυμητές τιμές στα μετρίσιμα μεγέθη όπως αυτά του ωφέλιμου ρυθμού διαμεταγωγής (useful throughput), του χρόνου πλήρους διαδρομής (Round Trip Time), της μονόδρομης καθυστέρησης (one way delay), της διακύμανσης της καθυστέρησης (jitter) και των απωλειών (losses) σε επιτρεπτά επίπεδα για την συγκεκριμένη δικτυακή κίνηση. Στα πλαίσια της κοινοπραξίας IETF (Internet Engineering Task Force) [8] έχουν αναπτυχθεί τα πρότυπα IntServ (Integrated Services) [9], DiffServ (Differentiated Services) [12] και

MPLS (Multi-protocol Label Switching) [13] για την αξιοποίησή τους σε θέματα Ποιότητας Υπηρεσίας σε δίκτυα βασισμένα στο πρωτόκολλο IP. Οι προϋποθέσεις που πρέπει να πληροί μια ροή για να ταξινομηθεί προσδιορίζονται σε μια συμφωνία μεταξύ του παροχέα της υπηρεσίας και του πελάτη – χρήστη, την Συμφωνία Στάθμης (παρεχόμενης) Υπηρεσίας (Service Level Agreement - SLA). Η συμφωνία αυτή μπορεί να περιέχει και λεπτομερείς κανόνες ρύθμισης της κίνησης, οι οποίοι με την σειρά τους συντάσσουν ένα Συμφωνητικό Ρύθμισης Κίνησης (Traffic Conditioning Agreement – TCA). Το Συμφωνητικό Ρύθμισης Κίνησης προσδιορίζει πότε η κίνηση είναι εντός των συνθηκών (in profile), πότε εκτός συνθηκών (out-of-profile) και τι ενέργειες πρέπει να ληφθούν ώστε να ταξινομηθεί και να ρυθμιστεί αναλόγως Εμείς παρακάτω θα αναφερθούμε στην αρχιτεκτονική του προτύπου DiffServ (το πρότυπο του χρησιμοποιείται σε αυτή την πτυχιακή εργασία) και στους αλγόριθμους μορφοποίησης κίνησης.

2.2.4.2. Αρχιτεκτονική Διαφοροποιημένων Υπηρεσιών

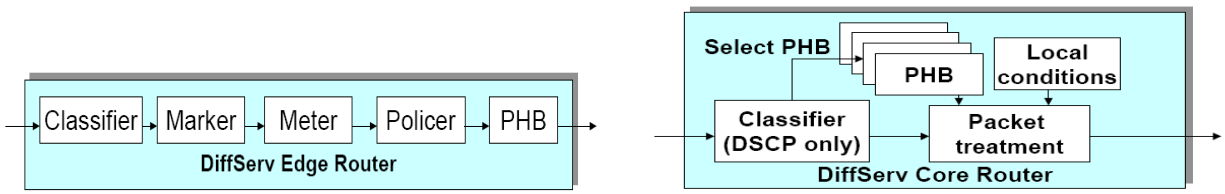
Η αρχιτεκτονική των Διαφοροποιημένων Υπηρεσιών είναι βασισμένη σε ένα απλό μοντέλο όπου η κίνηση που εισέρχεται σε ένα δίκτυο που υλοποιεί διαφοροποίηση υπηρεσιών (DiffServ Domain), κατηγοριοποιείται, ρυθμίζεται και ταξινομείται στα διάφορα σύνολα συμπεριφοράς (Behavior Aggregates – BA). Οι ενέργειες αυτές πραγματοποιούνται στα όρια του δικτύου από τους Δρομολογητές Παρυφής (DiffServ Edge Routers - DER). Πιο αναλυτικά, οι λειτουργίες που εκτελεί ένας DER είναι :

- Ταξινόμηση της δικτυακής κίνησης : Η ταξινόμηση επιτυγχάνεται μαρκάροντας κατάλληλα το Κωδικοσημείο Διαφοροποιημένων Υπηρεσιών (Differentiated Services CodePoint – DSCP) κάθε IP πακέτου. Τα IP πακέτα που φέρουν την ίδια τιμή στο πεδίο DSCP ανήκουν στο ίδιο σύνολο συμπεριφοράς.
- Μέτρηση, αστυνόμευση και διαμόρφωση της εισερχόμενης κίνησης με τέτοιο τρόπο ώστε να παρέχεται εγγυημένη Ποιότητα Υπηρεσίας στα σύνολα ροών.



Σχήμα 6. Σχηματική αναπαράσταση μιας περιοχής Διαφοροποιημένων Υπηρεσιών.

Μέσα στο δίκτυο τα πακέτα διαβιβάζονται στον προορισμό τους από τους Δρομολογητές Πυρήνα (DiffServ Core Routers - DCR), οι οποίοι ελέγχουν την τιμή του DSCP κάθε πακέτου και το προωθούν σύμφωνα με τους κανόνες που διέπουν το σύνολο συμπεριφοράς στο οποίο ανήκουν (Per-Hop Behavior - PHB)[14],[15]. Για να επιτευχθεί το επιθυμητό επίπεδο ποιότητας εφαρμόζονται διάφορες μέθοδοι διαχείρισης ουρών (Queuing Disciplines), πιθανό επαναπροσδιορισμό (remarking) της τιμής του DSCP και συντονισμό (scheduling) της κίνησης (Σχήμα 7).

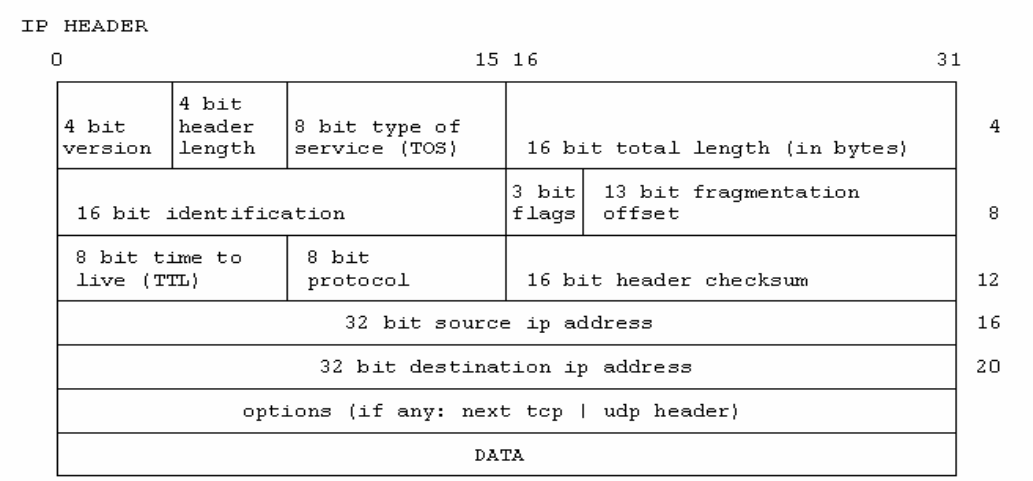


α β

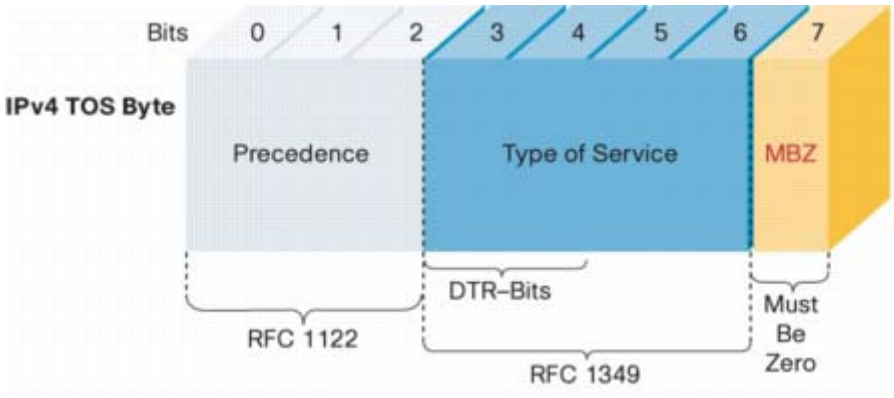
Σχήμα 7. Ενέργειες για την εφαρμογή DiffServ σε : α)DER, β)DCR

Κάθε σύνολο συμπεριφοράς προσδιορίζεται από την τιμή ενός πεδίου των 8 bits, που υπάρχει μέσα στην επικεφαλίδα των IP πακέτων (IP header). Οι σχεδιαστές των

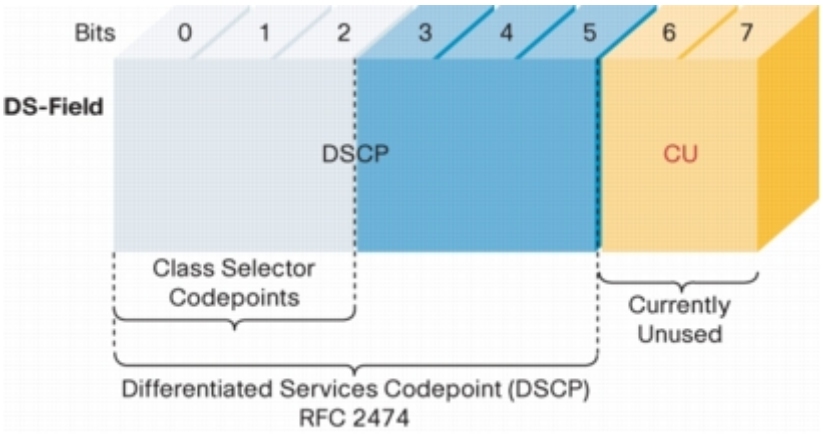
Διαφοροποιημένων Υπηρεσιών αποφάσισαν να χρησιμοποιήσουν την δεύτερη οκτάδα bits της IP επικεφαλίδας μετονομάζοντας την από πεδίο ToS (Type of Service Field) σε DS πεδίο (Differentiated Services Field). Για την διαφοροποίηση των υπηρεσιών γίνεται χρήση μόνο των 6 πρώτων bits (από αριστερά) από τα διαθέσιμα 8 του πεδίου, που ονομάζονται Κωδικοσημείο Διαφοροποιημένων Υπηρεσιών (Differentiated Services CodePoint – DSCP), αφήνοντας τα δύο τελευταία ανεκμετάλλευτα. Θεωρητικά, ένα δίκτυο θα μπορούσε να έχει μέχρι 64 (2^6) διαφορετικά σύνολα συμπεριφοράς χρησιμοποιώντας όλες τις δυνατές τιμές που μπορεί να πάρει το DSCP. Η δομή της IP επικεφαλίδας και των πεδίων DS και ToS φαίνεται στο σχήμα Σχήμα 8.



α



β



γ

Σχήμα 8. α)Η επικεφαλίδα των IP πακέτων, β) το πεδίο ToS, γ) το πεδίο DSCP.

2.2.4.3. Per Hop Behavior

Όπως αναφέρθηκε παραπάνω, η Per Hop Behavior (PHB) καθορίζεται από την τιμή που φέρει το DSCP του κάθε πακέτου και υποδεικνύει τον τρόπο με τον οποίο αυτό θα προωθηθεί από τους δρομολογητές πυρήνα του δικτύου. Ο οργανισμός IETF έχει τυποποιήσει και προτείνει τρεις κατηγορίες PHB :

- Την Εσπευσμένη Προώθηση (Expedited Forwarding – EF – *RFC 3246*) [11] που παρέχει υψηλή ποιότητα μετάδοσης, προωθώντας τα πακέτα με μικρή καθυστέρηση, χαμηλό jitter, ελάχιστες απώλειες και εξασφαλισμένο εύρος ζώνης. Η τιμή του DSCP της EF κατηγορίας είναι : 101110
- Την Εξασφαλισμένη Προώθηση (Assured Forwarding – AF – *RFC 2597*) [10], η οποία προσφέρει την δυνατότητα ταξινόμησης των υπηρεσιών σε τέσσερις υποκατηγορίες (κλάσεις) με διαφορετική προτεραιότητα διαβίβασης. Στις τρεις πρώτες κλάσεις εφαρμόζεται το "Ολυμπιακό" πρότυπο για τον διαχωρισμό των υπηρεσιών σε : χρυσή, ασημένια και χάλκινη, παραχωρώντας την ανάλογη προτεραιότητα. Επιπρόσθετα κάθε κλάση ορίζεται από τρία ιεραρχικά επίπεδα απόρριψης πακέτων (drop precedence). Οι προτεινόμενες τιμές του DSCP των AF κλάσεων φαίνονται στο Σχήμα 9.

	Class 1	**Class 2**	**Class 3**	**Class 4**
Low Drop Precedence	*AF11* 001010	*AF21* 010010	*AF31* 011010	*AF41* 100010
Medium Drop Precedence	*AF12* 001100	*AF22* 010100	*AF32* 011100	*AF42* 100100
High Drop Precedence	*AF13* 001110	*AF23* 010110	*AF33* 011110	*AF43* 100110

Σχήμα 9. Παρουσίαση των τιμών του DSCP πεδίου των AF κλάσεων

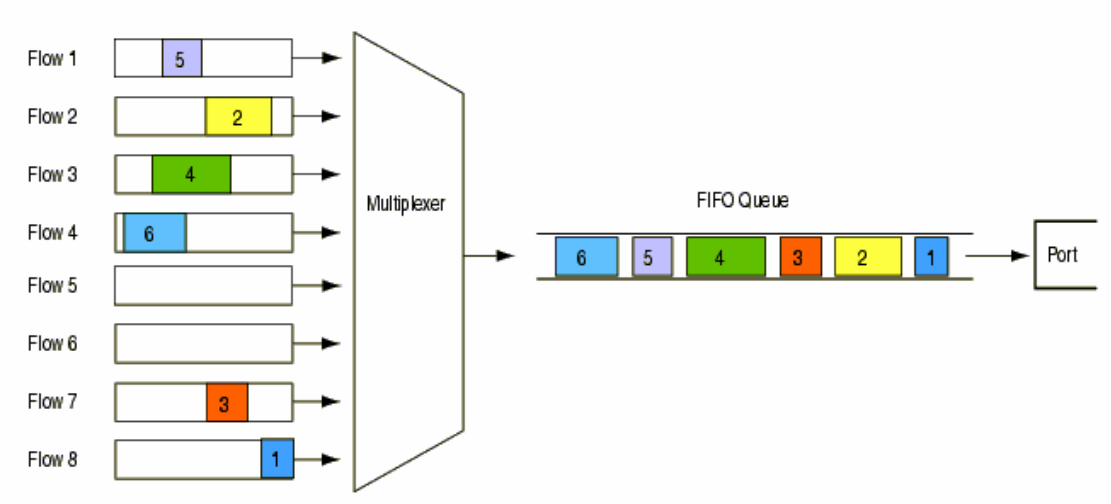
- Την Προκαθορισμένη PHB (Default), που χρησιμοποιείται για την κίνηση βέλτιστης προσπάθειας (Best Effort – BE), η οποία δεν παρέχει εγγυήσεις για την ποιότητα της υπηρεσίας. Η τιμή του DSCP είναι : 000000.

2.2.4.4. Αλγόριθμοι Μορφοποίησης Κίνησης

Μερικοί αλγόριθμοι μορφοποίησης κίνησης είναι οι FIFO, TBF, SFQ, RED, GRED, PRIO, HTB και DSMARK εμείς θα εξετάσουμε παρακάτω τους FIFO, PRIO, HTB και DSMARK όπου θα χρησιμοποιήσουμε για την υλοποίηση αυτής της πτυχιακής.

2.2.4.4.1. *FIFO*

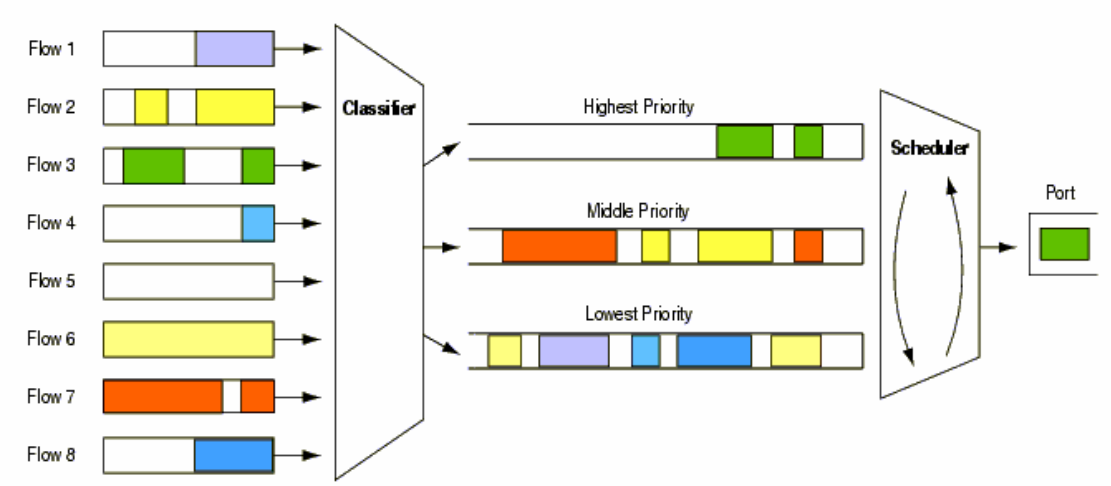
Ο αλγόριθμος FIFO (First In First Out) είναι από τους πιο απλούς. Από το όνομα του αλγορίθμου μπορούμε να καταλάβουμε ότι ο πρώτος που μπαίνει φεύγει και πρώτος δηλαδή η λειτουργία του έχει ως εξής τα πακέτα που φτάνουν μπαίνουν σε μια ουρά και περιμένουν μέχρι να έρθει η σειρά τους να φύγουν απλά φεύγουν με την σειρά που κατέφθασαν(το πρώτο πακέτο που έφτασε φεύγει και πρώτο). Όποια πακέτα φτάνουν αφού έχει γεμίσει η ουρά απλά απορρίπτονται. Στο Σχήμα 10. μπορούμε να δούμε τη σχηματική αναπαράσταση του αλγορίθμου.



Σχήμα 10. Σχηματική Αναπαράσταση του Αλγορίθμου FIFO

2.2.4.4.2. *PRIO*

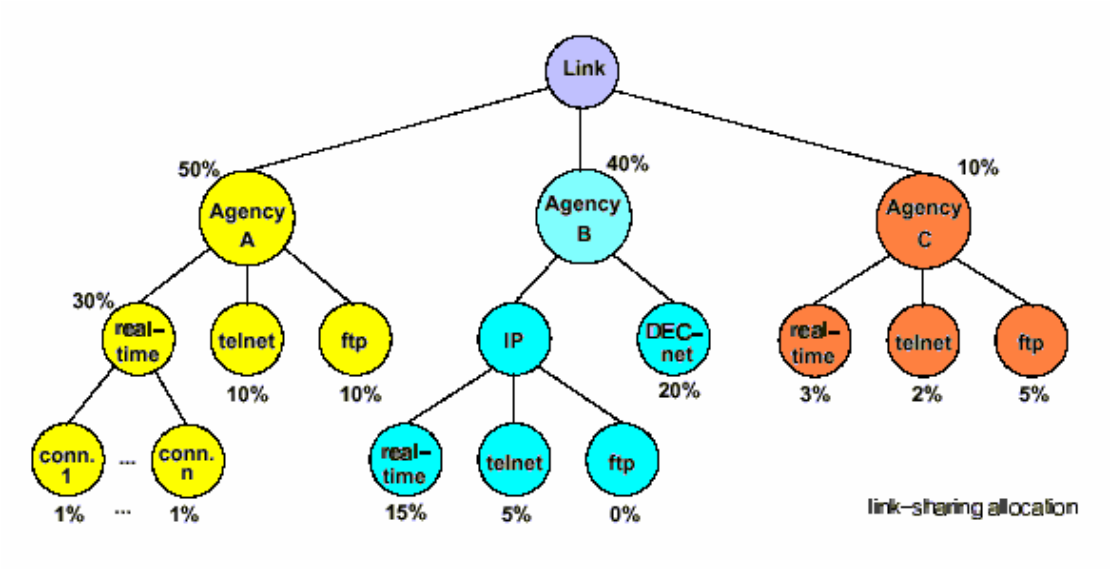
Ο αλγόριθμος PRIO (Priority Queuing) όπου μια σχηματική αναπαράσταση μπορούμε να δούμε στο Σχήμα 11 ταξινομεί τα πακέτα σε διαφορετικής προτεραιότητας ουρές ο μόνος περιορισμός είναι ότι για να εξυπηρετηθεί ένα πακέτο χαμηλότερης προτεραιότητας ουρά θα πρέπει όλες υψηλότερης προτεραιότητας ουρές να μην έχουν άλλα πακέτα προς εξυπηρέτηση.



Σχήμα 11. Σχηματική Αναπαράσταση του Αλγορίθμου PRIO

2.2.4.4.3. *HTB*

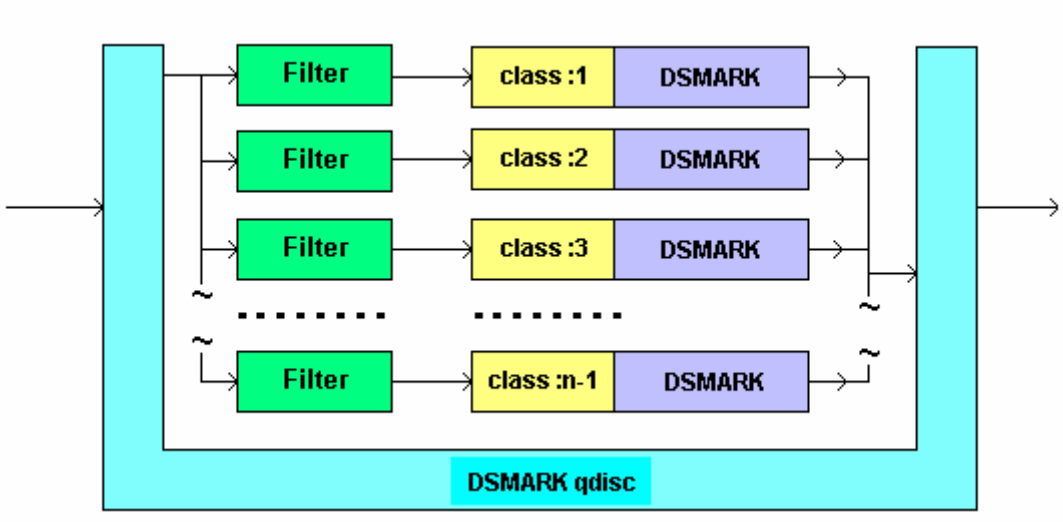
Ο αλγόριθμός HTB (Hierarchical Token Bucket) χρησιμοποιείται για την μορφοποίηση της δικτυακής κίνησης σε ένα ιεραρχικό μοντέλο (Σχήμα 12) όπου κάθε κλάση παίρνει το δικαίωμα να στείλει τα πακέτα που έχει στις ουρές με ένα μέσο ρυθμό μετάδοσης και με ελεγχόμενους καταιγισμούς (bursts) της ανάλογα με το ποσοστό εύρους που διαθέτει. Ο αλγόριθμος αυτός χρησιμοποιείται για τον καταμερισμό του bandwidth και μπορεί να χρησιμοποιηθεί για να διαφοροποιήσουμε υπηρεσίες.



Σχήμα 12. Ιεραρχικός Διαμοιρασμός του Διαθέσιμου Bandwidth

2.2.4.4.4. *DSMARK*

Ο αλγόριθμος DSMARK χρησιμοποιείται για να μαρκάρει τα πακέτα στο πεδίο DS της επικεφαλίδας ενός IP πακέτου. Στο Σχήμα 13 μπορούμε να διακρίνουμε σε ποίο σημείο της υλοποίησης του DiffServ γίνεται το marking (μαρκάρισμα) της υπηρεσίας-κίνησης.



Σχήμα 13. Σχηματική Αναπαράσταση του Αλγορίθμου DSMARK

2.2.4.4.5. *Ταξινομητές (Classifiers)*

Τέλος, ένας σημαντικός παράγοντας για την αστυνόμευση και την κατηγοριοποίηση των πακέτων είναι οι ταξινομητές (classifiers). Η αστυνόμευση προϋποθέτει μέτρηση της δικτυακής κίνησης, ώστε να μπορεί να γίνει έλεγχός αν συμφωνεί με τους όρους που αναφέρονται στο συμβόλαιο για την παρεχόμενη ποιότητα υπηρεσίας. Μια από τις θεμελιώδης αρχές της αρχιτεκτονικής των Διαφοροποιημένων Υπηρεσιών είναι να μην επιτρέπεται περισσότερη κίνηση από αυτή για την οποία σχεδιάστηκε το δίκτυο, για να μην υπερφορτώνονται οι ουρές αναμονής. Οι ταξινομητές επιλέγουν, ελέγχουν και κάνουν κατηγοριοποίηση των ροών κίνησης με βάση τα κριτήρια που καθορίζονται από τις παραμέτρους τους. Μέσα στις αρμοδιότητες τους έγκειται και να αποφασίζουν για τις ενέργειες που θα ληφθούν όταν μια ροή δεν πληροί ή αντίθετα, πληροί τους κανόνες που προ-συμφωνήθηκαν. Κάποιοι από τους ταξινομητές που χρησιμοποιούνται είναι : ο fw, ο u32, ο route και ο tcindex.

2.3. Πρωτόκολλα

Εδώ θα μελετήσουμε τα πρωτόκολλα μεταφοράς δεδομένων UDP και TCP και το πρωτόκολλο NTP.

2.3.1. Το Πρωτόκολλο UDP

Το Πρωτόκολλο UDP (User Datagram Protocol), χρησιμοποιείται όταν σε μια επικοινωνία δεν απαιτείται η επιβεβαίωση των δεδομένων η απλά ο χρόνος που απαιτείται είναι απαγορευτικός. Ένα καλό παράδειγμα χρήσης του πρωτοκόλλου αυτού είναι στη μετάδοση πολυμεσικού περιεχομένου όπως βίντεο, ήχος που είναι σε πραγματικό χρόνου και οι καθυστερήσεις ή οι επιβεβαιώσεις είναι περιττές.

+	Bits 0 - 15	16 - 31
0	Source Port Θύρα Προέλευσης	Destination Port Θύρα Προορισμού
32	Length μέγεθος	Checksum
64	Data Δεδομένα	

Σχήμα 14. Δομή ενός UDP πακέτου

Στο Σχήμα 14 βλέπουμε την δομή ενός UDP πακέτου, παρατηρούμε ότι η επικεφαλίδα αποτελείται από τέσσερα πεδία (source port, destination port, length, checksum) και μάλιστα τα δύο από αυτά δεν είναι υποχρεωτικά να συμπληρωθούν (source port, checksum). Το πεδίο source port περιέχει την θύρα αποστολέα, το πεδίο destination port την θύρα του παραλήπτη, το πεδίο length το μέγεθος του πακέτου σε bytes και το checksum πεδίο 16-bit το οποίο χρησιμοποιείται για επαλήθευση της ορθότητας του πακέτου στο σύνολό του, δηλαδή τόσο της κεφαλίδας όσο και των δεδομένων.

2.3.2. Το Πρωτόκολλο TCP

Το TCP *(Transmission Control Protocol - Πρωτόκολλο Ελέγχου Μεταφοράς)* είναι ένα από τα κυριότερα πρωτόκολλα της Σουίτας Πρωτοκόλλων Διαδικτύου. Βρίσκεται πάνω από το IP protocol *(πρωτόκολλο IP)*. Οι κύριοι στόχοι του πρωτοκόλλου TCP είναι να επιβεβαιώνεται η αξιόπιστη αποστολή και λήψη δεδομένων, επίσης να μεταφέρονται τα δεδομένα χωρίς λάθη μεταξύ του στρώματος δικτύου (network layer) και του στρώματος εφαρμογής (application layer) και, φτάνοντας στο πρόγραμμα του στρώματος εφαρμογής, να έχουν σωστή σειρά. Οι περισσότερες σύγχρονες υπηρεσίες στο Διαδίκτυο βασίζονται στο TCP. Για παράδειγμα το SMTP (port 25), το παλαιότερο (και μη-ασφαλές) Telnet (port 23), το FTP και πιο σημαντικό το HTTP (port 80), γνωστό ως υπηρεσίες World Wide Web (WWW - Παγκόσμιος Ιστός). Το TCP χρησιμοποιείται σχεδόν παντού, για αμφίδρομη επικοινωνία μέσω δικτύου. Έτσι και σε αυτή την πτυχιακή εργασία δεν θα μπορούσαμε να μην ασχοληθούμε.

+	Bits 0 - 3	4 - 9	10 - 15	16 - 31
0	**Source Port** Θύρα Προέλευσης			**Destination Port** Θύρα Προορισμού
32	**Sequence Number** Αριθμός ακολουθίας			
64	**Acknowledgment Number** Αριθμός επιβεβαίωσης			
96	**Data Offset**	**Reserved**	**Flags** Σημαίες	**Window** Παράθυρο
128	**Checksum** Άθροισμα ελέγχου			**Urgent Pointer** Επείγοντα δεδομένα
160	**Options** Επιλογές (προαιρετικές)			
160/192+	**Data** Δεδομένα			

Σχήμα 15. Δομή ενός TCP πακέτου

Στο Σχήμα 15 μπορούμε να δούμε την διάταξη ενός πακέτου TCP όπου αποτελείται από την θύρα προέλευσης που είναι το πεδίο που ορίζει την port (θύρα) του αποστολέα, την θύρα προορισμού το πεδίο με την port (θύρα) του παραλήπτη, το Sequence Number (αριθμός ακολουθίας), τον Acknowledgment Number (αριθμό επιβεβαίωσης), data offset είναι ο αριθμός από words μεγέθους 32 bit στην επικεφαλίδα TCP (TCP header), από το reserved που καθορίζει το μέγεθος της επικεφαλίδας (πολλαπλάσιο του 32) και επομένως δείχνει και την αρχή των δεδομένων, από πεδίο 6 bit "κρατημένων" για μελλοντική χρήση. Η τιμή των bit πρέπει να είναι 0. Από το πεδίο flags (σημαίες) η αλλιώς control bits (bits ελέγχου) περιέχει 6bit σημαίες (Σχήμα 16).

Σημαία	Σημασία	Προέλευση ονομασίας
URG	Το πεδίο urgent pointer είναι σημαντικό	**URG**ent
ACK	Το πεδίο επιβεβαίωσης είναι σημαντικό	**ACK**nowledgment
PSH	Λειτουργία ώθησης	**P**u**SH**
RST	Επαναρύθμιση σύνδεσης	**R**e**S**e**T**
SYN	Συγχρονισμός αριθμών ακολουθίας	**SYN**chronize
FIN	Ο αποστολέας δεν στέλνει άλλα δεδομένα	**FIN** (=τέλος)

Σχήμα 16. Flags για την κεφαλίδα TCP

Από το window που είναι ο αριθμός των byte που επιθυμεί να δεχθεί ο αποστολέας του πακέτου, το πεδίο checksum μεγέθους 16 bit χρησιμοποιείται για έλεγχο λαθών στην επικεφαλίδα και στα δεδομένα, από option μεταβλητή η οποία καθορίζει ειδικές επιλεγόμενες ρυθμίσεις και μπορεί να καταλάβει χώρο στο τέλος της επικεφαλίδας TCP και το urgent pointer όπου εάν είναι ενεργοποιημένο το URG bit ελέγχου, τότε αυτό το πεδίο δείχνει τον αριθμό ακολουθίας (sequence number) και τέλος το πεδίο data όπου περιέχονται τα δεδομένα της επικεφαλίδας.

Για την ανίχνευση χαμένων πακέτων το κάθε πακέτο αριθμείται στο πεδίο Sequence Number που αναφέραμε προηγουμένως και αυξάνεται κατά ένα σε σχέση με το προηγούμενο. Ο παραλήπτης απαντά για την επιτυχή παραλαβή πακέτου με ένα πακέτο όπου στο πεδίο Acknowledgment Number περιέχει τον αριθμό του επόμενου πακέτου που περιμένει. Ανεπιβεβαίωτα πακέτα που έχουν υπερβεί ένα χρονικό όριο γνωστό ως RTO (Retransmission TimeOut) επανεκπέμπονται.

Το TCP πρωτόκολλο ξεκινώντας μια σύνδεση στέλνει ένα αριθμό bytes χωρίς να περιμένει επιβεβαίωση αυτό ονομάζεται window size. Μετά από μια επιτυχή παράδοση πακέτου το TCP πρωτόκολλο επιχειρεί να αυξήσει το ρυθμό αποστολής αλλά σε καμία περίπτωση δεν θα πρέπει να ξεπερνά το μέγεθος του receiver advertised window που έχει δηλωθεί από τον παραλήπτη.

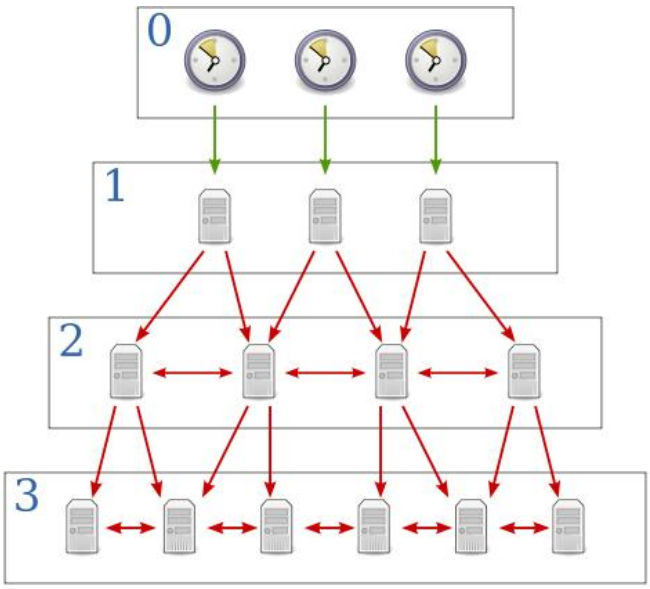
Για την έλεγχο της συμφόρησης του δικτύου το TCP έχει μερικού αλγόριθμους που εξασφαλίζουν την είτε στην αποφυγή είτε στην γρήγορη αποκατάσταση αυτό επιτυγχάνεται μέσα από ειδικούς αλγόριθμους όπως τον αλγόριθμος slow-start, τον congestion avoidance, τον fast retransmit και τον fast recovery όπως αναφέρεται στο RFC 2001 [6].

2.3.3. Το Πρωτόκολλο NTP

Το πρωτόκολλο NTP (Network Time Protocol) χρησιμοποιείται για τον συγχρονισμό υπολογιστικών συστημάτων, χρησιμοποιεί την UDP port 123 και βρίσκεται στο transport layer (επίπεδο μεταφοράς). Το NTP (Network Time Protocol) πρωτόκολλο είναι από τα παλιότερα Internet protocols (πριν από το 1985) και χρησιμοποιείται ακόμη και σήμερα.

Το Πρωτόκολλο NTP είναι ένα πρωτόκολλο το οποίο χρησιμοποιείται για να τον συγχρονισμό της ώρας ενός υπολογιστή, εξυπηρετητή, δρομολογητή ή γενικότερα μίας δικτυακής συσκευής με μία πηγή ώρας που λειτουργεί ως σημείο αναφοράς. Μέσω του NTP επιτυγχάνεται συγχρονισμός ονομαστικής ακρίβειας της τάξης του χιλιοστού του δευτερολέπτου σε τοπικά δίκτυα (LAN), και της τάξης του χιλιοστού του δευτερολέπτου για

δίκτυα μεγάλης εμβέλειας (WAN). Μία τυπική NTP διαμόρφωση χρησιμοποιεί πολλαπλούς εφεδρικούς εξυπηρετητές και διάφορες διαδρομές δικτύου, με σκοπό να πετύχει υψηλή ακρίβεια και αξιοπιστία. Μερικές παραμετροποιήσεις περιέχουν ταυτοποίηση με κρυπτογραφία για αποφυγή ατυχημάτων ή μοχθηρών επιθέσεων.



Σχήμα 17. Ιεραρχικό σύστημα Πρωτοκόλλου NTP

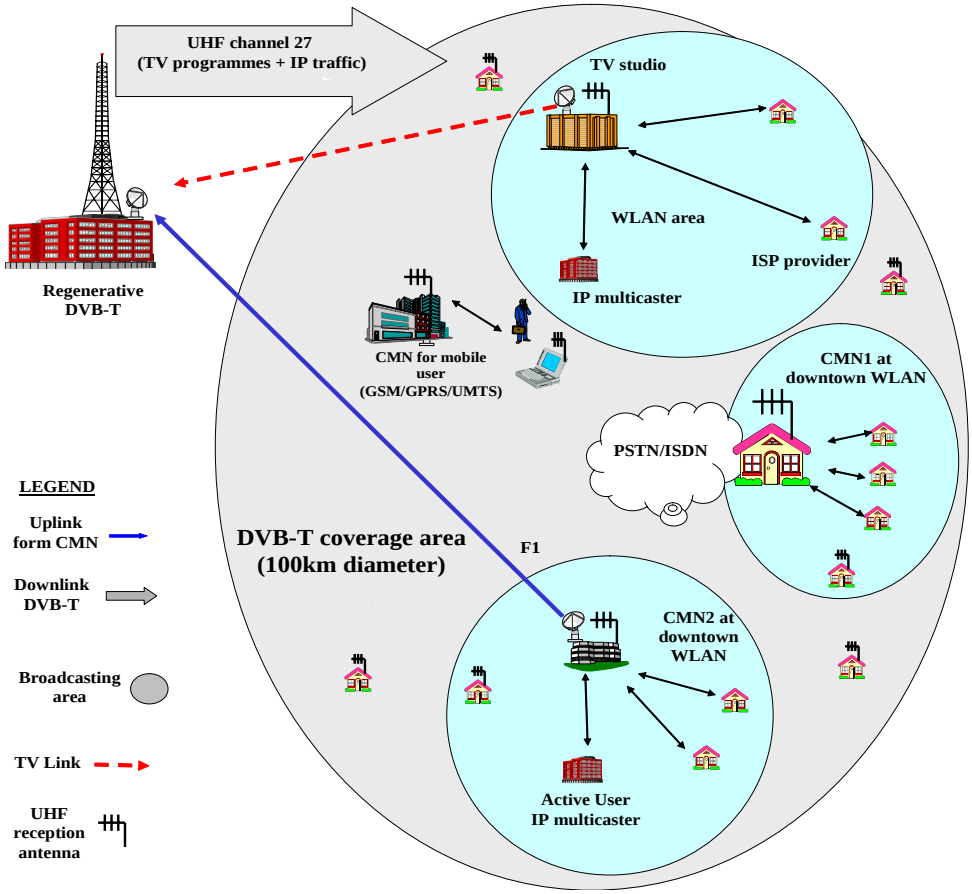
Ένας NTP εξυπηρετητής συγχρονίζει τα ρολόγια των συσκευών που συνδέονται σε αυτόν μέσω του πρωτοκόλλου NTP. Με τον τρόπο αυτό οι δικτυακές αυτές συσκευές έχουν κοινό χρονικό σημείο αναφοράς. Όμως σε πολλές περιπτώσεις και το ρολόι του εξυπηρετητή του ίδιου συγχρονίζεται με άλλο ρολόι αναφοράς. Συνήθως, οι NTP εξυπηρετητές συγχρονίζονται τελικά αναδρομικά με το UTC (Universal Time Coordinate). Αυτό επιτυγχάνεται ως εξής: Ατομικά ρολόγια δίνουν πληροφορία χρόνου από φυσικές πηγές χρόνου (π.χ. Καισίου). Αυτά τα ρολόγια θεωρούνται Stratum 0. Κύριοι πρωτεύοντες (Primary Stratum 1) εξυπηρετητές συγχρονίζουν σε εθνικά πρότυπα χρόνου (UTC, EET κλπ) με Stratum 0 εξυπηρετητές με χρήση ραδιοφωνικών κυμάτων, δορυφόρων και ατομικών ρολογιών. NTP εξυπηρετητές λαμβάνουν ώρα από άλλους NTP εξυπηρετητές, δημιουργώντας έτσι ένα ιεραρχικό μοντέλο παροχής χρονισμού. Οι εξυπηρετητές αυτοί ονομάζονται stratum 2, stratum 3 κλπ ανάλογα με την απόστασή τους από τους stratum 1 εξυπηρετητές.

Η αξιοπιστία της λειτουργίας παρέχεται από την ύπαρξη του ιεραρχικού σχήματος (Σχήμα 17), την πιθανή ύπαρξη εναλλακτικών πηγών χρονισμού και την ύπαρξη πολλαπλών διαδρομών προς αυτές.

3. Αρχιτεκτονική Του Πειραματικού Δικτύου

3.1. Εισαγωγή

Όπως αναφέραμε και παραπάνω ένα σύστημα ψηφιακής τηλεόρασης με ένα κανάλι επιστροφής (xDSL, WLAN, DVB-RCS, PSTN, ISDN, GSM, GPRS) μπορεί να γίνει διαδραστικό (interactive TV), όμως πέρα από μια απλή αλληλεπίδραση του τελικού χρήστη με το περιεχόμενο(VoD, AoD) μπορεί να μεταφέρει και IP δεδομένα. Αυτό το πλεονέκτημα είναι που θα εκμεταλλευτούμε για να παρέχουμε δικτυακές υπηρεσίες πάνω από ένα υβριδικό σύστημα όπου συνεργάζονται η ψηφιακή τηλεόρασης αλλά και άλλες τεχνολογίες.

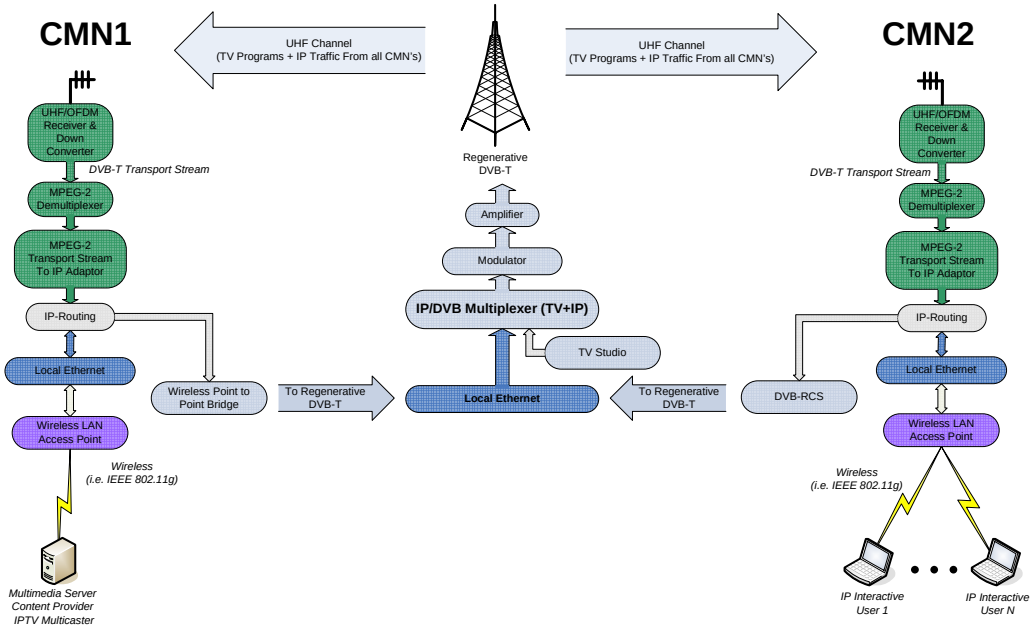


Σχήμα 18. Γενική Αρχιτεκτονική ενός DVB-T συστήματος

Ένα τέτοιο σύστημα αποτελείται από το κεντρικό σημείο εκπομπής (πλατφόρμα DVB-T), από πολλούς ενδιάμεσους κόμβους διανομής (Cell Main Node - CMN) και από τους τελικούς χρήστες (End User). Οι CMN δρομολογούν την κίνηση από και προς τους τελικούς χρήστες δηλαδή λαμβάνουν την κίνηση από μία κάρτα τηλεόρασης (DVB-T) και την προωθούν στους τελικούς χρήστες και την κίνηση από τους τελικούς χρήστες προς την πλατφόρμα ψηφιακής τηλεόρασης μέσω του καναλιού επιστροφής που διαθέτει ο εκάστοτε CMN (WLAN, DVB-RCS). Στο Σχήμα 18 μπορούμε να δούμε την γενική αρχιτεκτονική ενός τέτοιου συστήματος.

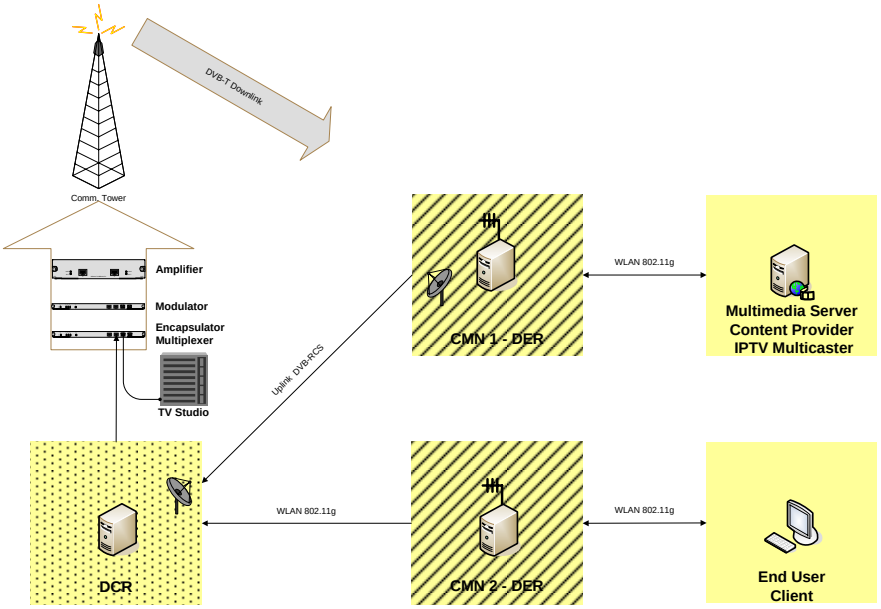
3.2. Τοπολογία Δικτύου

Το σύστημα μας (Σχήμα 19) βασίζεται στο γενική αρχιτεκτονική που περιγράψαμε χρησιμοποιώντας σαν κανάλια επιστροφής WLAN(802.11g) για τον πρώτο CMN (CMN1) και DVB-RCS για τον δεύτερο CMN (CMN2). Ενώ στο δίκτυο πρόσβασης του τελικού χρήστη χρησιμοποιείται WLAN τεχνολογίας 802.11g και για τους δύο CMN που υπάρχουν στο πειραματικό δίκτυο. Το δίκτυο μας αποτελείται από το κεντρικό σημείο εκπομπής (πλατφόρμα DVB-T) τους CMN και τους τελικούς χρήστες. Οι τελικοί χρήστες συνδέονται με τους CMN και οι CMN με το κεντρικό σύστημα διανομής.



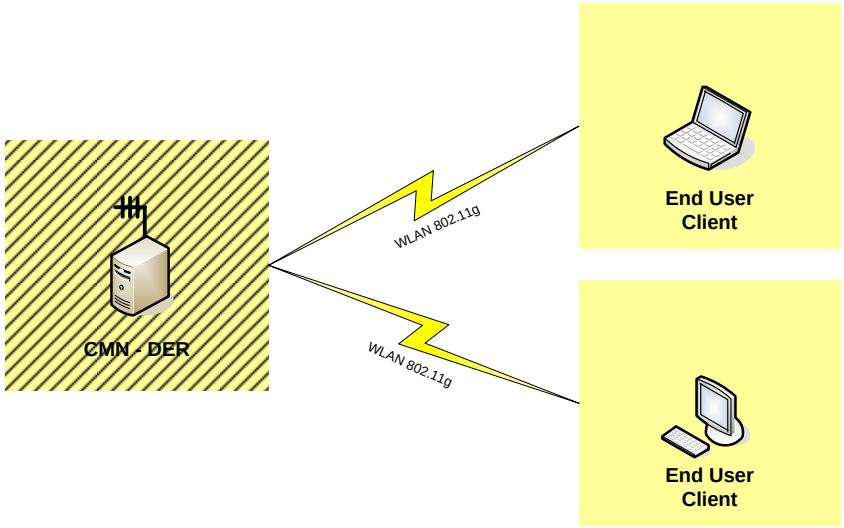
Σχήμα 19. Συνολικό Δίκτυο DVB-T

Για να έχουμε αμφίδρομη επικοινωνία μεταξύ της πλατφόρμας DVB-T και των CMN χρησιμοποιούμε σαν κανάλι καθόδου το DVB-T όπου εκτός από τα IP δεδομένα αποστέλλονται και τα ψηφιακά τηλεοπτικά προγράμματα και ως κανάλι επιστροφής ένα ασύρματο δίκτυο τεχνολογίας 802.11g και ένα DVB-RCS δίκτυο όπως βλέπουμε στο Σχήμα 20.



Σχήμα 20. Σύστημα(DCR) με CMN-DER

Τέλος οι τελικοί χρήστες συνδέονται στους CMN μέσω ασύρματου δικτύου WLAN 802.11g (Σχήμα 21) και όλη η κίνηση τους περνά από τον CMN τους όπου μαρκάρεται και δρομολογείται.



Σχήμα 21. Access Network

Η κίνηση δημιουργείται από τον service provider δρομολογείται μέσω του CMN1 προς την πλατφόρμα DVB-T και από εκεί στον CMN2 για να προωθηθεί στους τελικούς χρήστες.

3.3. Υλοποίηση

Η πτυχιακή αυτή εργασία υλοποιήθηκε με εξοπλισμό του ερευνητικού εργαστηρίου Τηλεπικοινωνιών και Δικτύων του Α.Τ.Ε.Ι Κρήτης **ΠΑΣΙΦΑΗ**. Σε όλους τους υπολογιστές που χρησιμοποιήθηκαν στο πειραματικό δίκτυο ήταν εγκατεστημένο δωρεάν λειτουργικό σύστημα Linux διανομής Debian.

3.3.1. Τελικός Χρήστης (End User)

Οι τελικοί χρήστες χρησιμοποίησαν ασύρματες κάρτες PCI ή PCMCI στην περίπτωση φορητού υπολογιστή για να συνδεθούν στο AP (Access Point) όπου συνδέεται καλωδιακά με τους CMN's. Η κίνηση δημιουργείται από τον service provider και δρομολογείται μέσω του CMN1 στην πλατφόρμα και από εκεί στον CMN2 για να φτάσει η κίνηση στους τελικούς χρήστες.

3.3.2. Ενδιάμεσος Κόμβος Διανομής (Cell Main Node)

Στους CMN's-DER's (DiffServ Edge Router) είναι εγκατεστημένο επίσης λειτουργικό σύστημα Linux-Debian στον πρώτο CMN στον CMN1 όπου είναι συνδεδεμένος ο Service Provider υπάρχει εγκατεστημένη μια ενσύρματη κάρτα δικτύου όπου συνδέεται το AP, μια ασύρματη κάρτα δικτύου για σύνδεση με το κεντρικό σημείο διανομής όπου χρησιμοποιείτε σαν κανάλι επιστροφής και μία κάρτα ψηφιακής τηλεόρασης για επικοινωνία από την πλατφόρμα DVB-T προς τον CMN1 όπου χρησιμοποιείται σαν κανάλι καθόδου.

Ο δεύτερος CMN ο CMN2 χρησιμοποιήθηκε για να συνδεθούν οι τελικοί χρήστες, έχει και αυτός μια ενσύρματη κάρτα δικτύου και μια κάρτα ψηφιακής τηλεόρασης όπως CMN1 για τους ίδιους λόγους με την μόνη διαφορά στο κανάλι επιστροφής όπου αντί για WLAN έχουμε DVB-RCS για την επικοινωνία με την πλατφόρμα.

Οι CMN's αναλαμβάνουν να κάνουν το μαρκάρισμα (Marking) των πακέτων ανάλογα με την υπηρεσία ή τον προορισμό(CMN1 σελ 54, CMN2 σελ 55) και την προώθηση τους.

3.3.3. Κεντρικό Σημείο Εκπομπής (Πλατφόρμα DVB-T) - DCR

Το κεντρικό σημείο εκπομπής αποτελείται από τον DiffServ Core Router, τον ενθυλακωτή-πολυπλέκτη (Encapsulator-Multiplexer), τον διαμορφωτή (Modulator) από τον ενισχυτή (Amplifier) και από έναν MPEG 2 TS Server που περιέχει τηλεοπτικά προγράμματα.

Στον DiffServ Core Router (DCR) λειτουργούν τα scripts για το DiffServ(σελ 53). Όπως αναφέραμε ο DCR λαμβάνει την κίνηση από τους CMN μέσω του WLAN από τον CMN1 και μέσω του DVB-RCS από τον CMN2 και ελέγχει και κατηγοριοποιεί την κίνηση ανάλογα το εκάστοτε σενάριο ελέγχοντας τα πακέτα σύμφωνα με το marking που έχει γίνει στους CMN's και στη συνέχεια προωθεί την κίνηση στον Encapsulator-Multiplexer μέσω μιας ενσύρματης κάρτας δικτύου για να ενθυλακωθούν τα πακέτα στο MPEG2-TS μαζί με τα τηλεοπτικά προγράμματα από τον MPEG 2 TS Server. Ο Encapsulator-Multiplexer ρυθμίστηκε να δεσμεύει για τις IP υπηρεσίες 8 Mbps, ενώ για τα ψηφιακά τηλεοπτικά προγράμματα 13 Mbps.

Ο διαμορφοτής DVB-T (COFDM) ρυθμίστηκε σε διαμόρφωση 16QAM, ρυθμό κώδικα 7/8 και διάστημα φύλαξης (guard interval) ίσο με το 1/32 του μήκους συμβόλου. Οι παράμετροι αυτές αντιστοιχούν σε ωφέλιμο ρυθμό δεδομένων ίσο με 21.11 Mbps. Επίσης ως φέροντα σήματα δηλώνονται 8K . Η συχνότητα του σήματος εκπομπής είναι στα 626 MHz (κανάλι 40) με επίπεδο δύναμης (power level) 1Watt και το συγκεκριμένο κανάλι έχει εύρος 8MHz.

3.3.4. DCR

3.3.5. Προγράμματα που Χρησιμοποιήθηκαν

Η δημιουργία της κίνησης έγινε με τα εξής προγράμματα :

- *MEGN[16]*: Το MGEN (Multi-Generator) είναι ένα λογισμικό ανοιχτού κώδικα και παρέχει τη δυνατότητα να εκτελεσθούν μετρήσεις για την απόδοση δικτύων που παρέχουν IP υπηρεσίες, δημιουργώντας UDP κίνηση.

- *Iperf[17]*: Το Iperf είναι ένα εργαλείο για δημιουργία TCP και UDP κινήσεων, το οποίο λειτουργεί σε όλα τα συστήματα (Unix, Windows, MacOS κλπ.). Μερικά από τα χαρακτηριστικά του γνωρίσματα είναι ότι παρέχει χρήσιμες πληροφορίες και αποτελέσματα για το εύρος ζώνης, τις απώλειες, την διακύμανση της καθυστέρησης και γενικότερα για την απόδοση του δικτύου
- *D-ITG* [18] : Το D-ITG είναι ένα πρόγραμμα ικανό να παράγει δικτυακές κινήσεις στα επίπεδα δικτύου, μεταφοράς και εφαρμογής. Στις πειραματικές μετρήσεις της συγκεκριμένης πτυχιακής το εργαλείο D-ITG θα χρησιμοποιηθεί για την δημιουργία της VoIP επικοινωνίας.

Η σύλληψη της δικτυακής κίνησης με :

- *Tcpdump[23]*: Το Tcpdump είναι ένα εργαλείο "σύλληψης" και παρακολούθησης της δικτυακής κυκλοφορίας, το οποίο σε συνεργασία με άλλα προγράμματα βοηθάει στην ανάλυση των διαφόρων χαρακτηριστικών των δικτυακών κινήσεων.

Άλλα προγράμματα που χρησιμοποιήθηκαν :

- *Tcptrace [25]*: Το Tcptrace είναι ένα εργαλείο που χρησιμοποιείται για την ανάλυση αρχείων που έχουν δημιουργηθεί από διάφορα προγράμματα "σύλληψης" δικτυακής κίνησης, όπως είναι το tcpdump. Το tcptrace μπορεί να παράγει αρχεία τα οποία περιέχουν διαφόρους τύπους πληροφοριών για κάθε υπαρκτή κίνηση, όπως επαναμεταδόσεις, καθυστέρηση, ρυθμοαπόδοση και άλλα. Μπορεί επίσης να παράγει γραφικές παραστάσεις για κάθε μια από τις παραπάνω πληροφορίες, για περαιτέρω ανάλυση.
- *Iproute2 και tc[19]*: Το iproute2 είναι μια συλλογή εφαρμογών για την διαχείριση IP δικτυακής κίνησης σε περιβάλλον Linux. Από τα εργαλεία που προσφέρει πιο σημαντικά θεωρούνται το ip και το tc. Το εργαλείο tc δίνει την δυνατότητα εισαγωγής ουρών, φίλτρων και την διαχείριση του εύρους ζώνης μιας ζεύξης.
- *Iptables [24]*: το εργαλείο iptables χρησιμοποιείται και αυτό για την διαχείριση της δικτυακής κίνησης. Μέσα στις λειτουργίες που παρέχει είναι το φιλτράρισμα πακέτων με την εισαγωγή κανόνων και η σήμανση των πεδίων ToS και DSCP της επικεφαλίδας των IP πακέτων.

3.4. Σενάρια Πειραματικών Μετρήσεων

Για να μπορέσουμε να αξιολογήσουμε το δίκτυο που αναφέρθηκε παραπάνω θα μετρήσουμε την δικτυακή κίνηση που δημιουργείται από τον service provider δημιουργούμε UDP κίνηση από τον Service Provider στον τελικό χρήστη και TCP και VoIP μεταξύ του Service Provider των τελικών χρηστών. Η διάρκεια των μετρήσεων θα είναι 180sec και η παρουσίαση των αποτελεσμάτων θα γίνει σε πραγματικό χρόνο (real-time) και παρουσιάζοντας τις μέσες τιμές από τα 180 δευτερόλεπτα της μέτρησης. Τα μεγέθη που θα εξετάσουμε για την αξιολόγηση της απόδοσης του δικτύου είναι ο ωφέλιμος ρυθμός διαμεταγωγής (useful throughput) ο χρόνος πλήρους διαδρομής (round trip time), η μονόδρομη καθυστέρηση (one way delay) και οι απώλειες (losses). Όλες τις μετρήσεις έχουν ληφθεί από τον CMN1 μέχρι τον CMN2 για να αξιολογήσουμε την ποιότητα υπηρεσίας στο κομμάτι του δικτύου όπου εφαρμόζεται η τεχνολογία DiffServ.

3.4.1. Σενάριο 1

Στο πρώτο σενάριο θα υπάρχουν οχτώ χρήστες που θα ζητούν TCP υπηρεσίες, οχτώ VoIP υπηρεσίες (G.729.2) με τον Service Provider (όπου συνδέεται στον CMN1) και μια multicast εκπομπή μιας IPTV (0.7Mbps) υπηρεσίας προς τους τελικούς χρήστες (όπου συνδέονται στον CMN2), σε αυτό το σενάριο θα έχουμε τις υπηρεσίες χωρίς ενεργοποιημένο τον μηχανισμό DiffServ για να αξιολογήσουμε τη συμπεριφορά του δικτύου χωρίς κανένα μηχανισμό παροχής ποιότητας υπηρεσίας.

3.4.2. Σενάριο 2

Στο δεύτερο σενάριο θα έχουμε τις ίδιες υπηρεσίες με ενεργοποιημένο τον μηχανισμό DiffServ. Η κατηγοριοποίηση θα γίνει με βάση τον τύπο της υπηρεσίας (UDP, TCP, VoIP) αλλά και βάση του χρήστη-πελάτη (User1, User2 κτλ). Οι υπηρεσίες IPTV και VoIP από όλους τους χρήστες θα ανήκουν στην κλάση EF ενώ οι TCP υπηρεσίες θα κατηγοριοποιηθούν στις κλάσεις AF και BE δηλαδή η TCP κίνηση από τους δύο πρώτους χρήστες θα κατηγοριοποιηθεί σαν AF1 κλάση δεσμεύοντας 3Mbps bandwidth (εύρος ζώνης) και για τους δύο χρήστες συνολικά, οι δυο επόμενοι χρήστες θα κατηγοριοποιηθούν

σαν AF2 κλάση δεσμεύοντας 2Mbps, οι άλλοι δύο χρήστες σαν AF3 κλάση με 1Mbps και οι δύο τελευταίοι χρήστες σαν BE κλάση όπου χρησιμοποιούν το υπόλοιπο bandwidth.

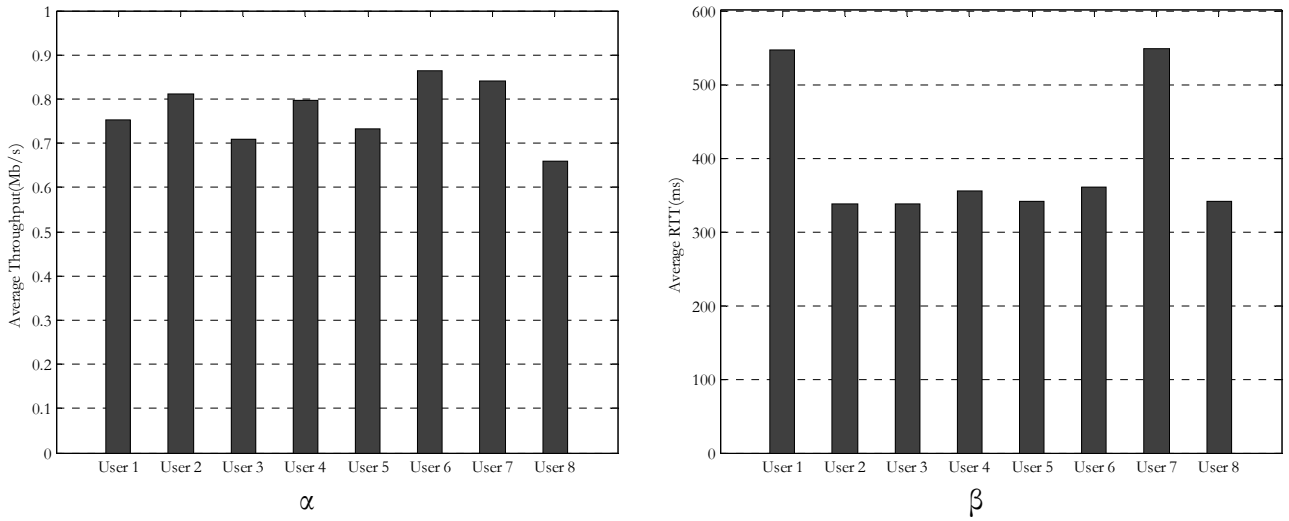
3.4.3. Σενάριο 3

Στο τρίτο σενάριο θα προσπαθήσουμε να δείξουμε την δυναμικότητα του συστήματος στην κατηγοριοποίηση των υπηρεσιών – χρηστών. Το σενάριο αυτό έχει τις ίδιες παραμέτρους με το σενάριο 2 για τα πρώτα 90sec θα υπάρξει μια αλλαγή θα υπάρξει μια αλλαγή στους χρήστες και οι δύο πρώτοι χρήστες που έχουν κατηγοριοποιηθεί σαν AF1 θα σταματήσουν να ζητούν TCP υπηρεσίες. Στο σημείο αυτό ο μηχανισμός θα αλλάξει το δεσμευμένο bandwidth από κάθε κλάση και εφόσον η AF1 κλάση δεν χρειάζεται πλέον το διαθέσιμο bandwidth για αυτήν το παραχωρεί στις επόμενες κλάσεις που βρίσκονται κάτω από αυτή. Η διάθεση των πόρων μετά τα 90sec θα είναι ως εξής η AF2 κλάση θα πάρει 3.5Mbps η AF3 2Mbps και τέλος η BE το bandwidth που υπολείπεται.

4. Παρουσίαση Των Πειραματικών Μετρήσεων

4.1. Σενάριο 1

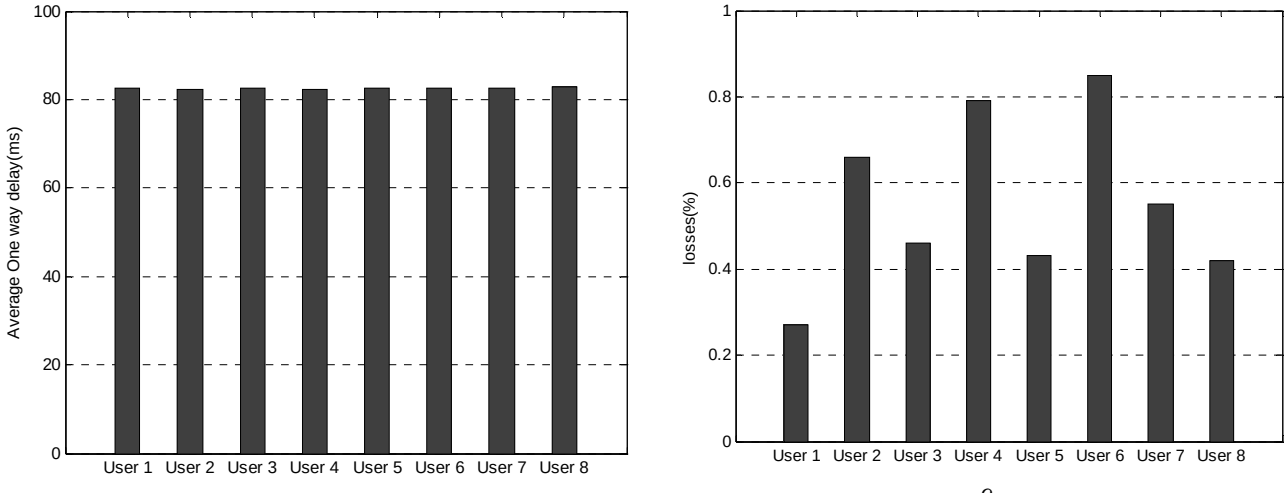
Στο πρώτο σενάριο όπου δεν είχαμε ενεργοποιημένο τον μηχανισμό DiffServ παρατηρήσαμε ότι το throughput για τις TCP ροές έχει κατανεμηθεί ομοιόμορφα και στις οχτώ ροές παίρνοντας η κάθε μία από περίπου 0,77Mbps(Σχήμα 22α). Ακόμη σχεδόν ομοιόμορφα έχει κατανεμηθεί και το RTT με 396ms(Σχήμα 22β) για κάθε ροή.



α β

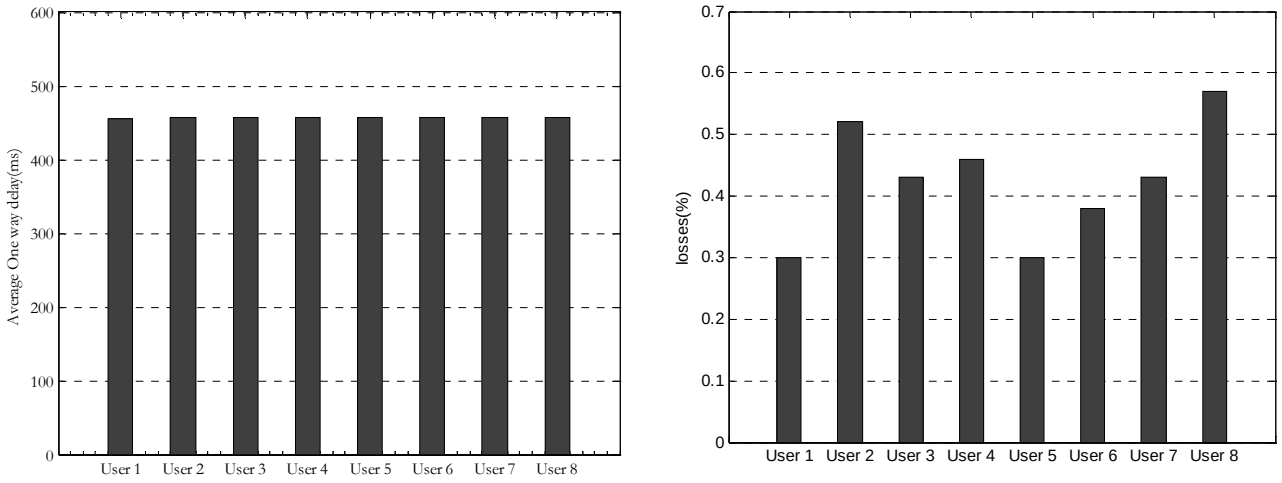
Σχήμα 22. TCP α)Average Throughput β)Average RTT

Η IPTV έχει μια καθυστέρηση 83ms και απώλειες 0,51% και η VoIP κίνηση που δημιουργείται από τον service provider προς τους τελικούς χρήστες έχει καθυστέρηση περίπου 83.5ms(Σχήμα 23α) και απώλειες 0,55%(Σχήμα 23β). Ενώ η VoIP κίνηση που δημιουργείται από τους τελικούς χρήστες προς τον service provider έχει καθυστέρηση περίπου 456ms(Σχήμα 24α) και απώλειες 0,42%(Σχήμα 24β).



α β

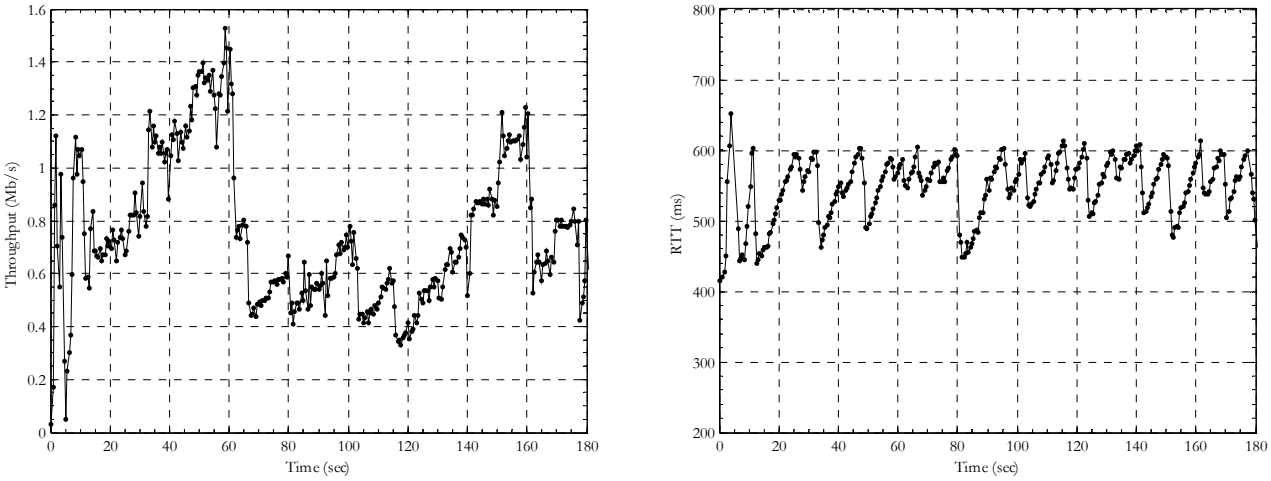
Σχήμα 23. VoIP server to client α) Average One Way Delay β) Average Losses



α β

Σχήμα 24. VoIP client to server α) Average One Way Delay β) Average Losses

Παρακάτω θα δούμε μια TCP ροή σε πραγματικό χρόνο (real-time) για 180sec που διαρκεί μια μέτρηση για τον πρώτο χρήστη. Στο Σχήμα 25α βλέπουμε το Throughput να μην είναι σταθερό και να εξαρτάται άμεσα από το Throughput που θα λάβουν οι υπόλοιποι χρήστες και στο Σχήμα 25β βλέπουμε το RTT σε real-time για τον πρώτο χρήστη.

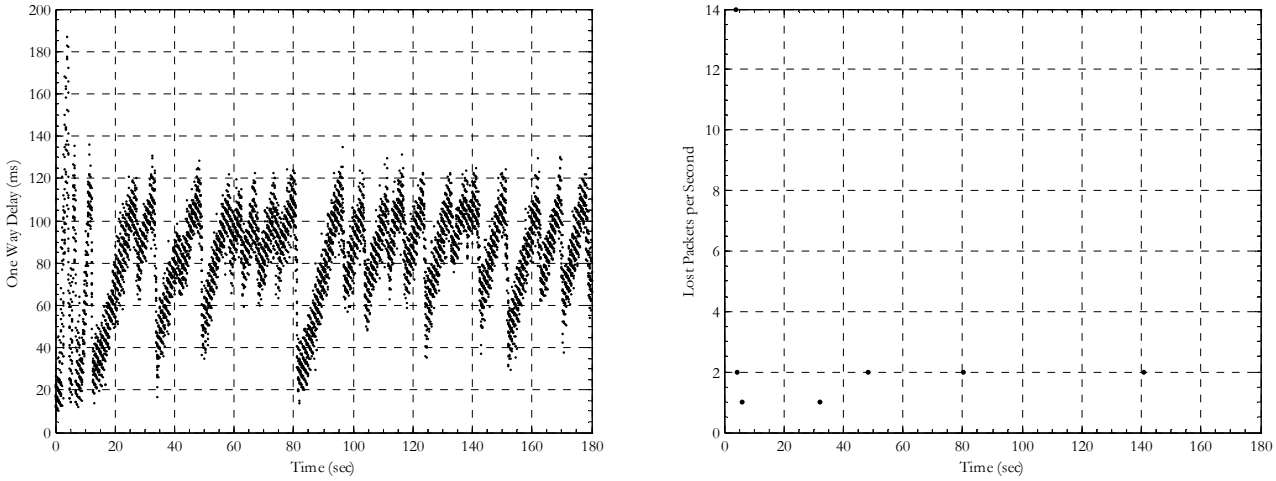


α β

Σχήμα 25. α)Throughput για τον 1ο χρήστη β) RTT για τον 1ο χρήστη

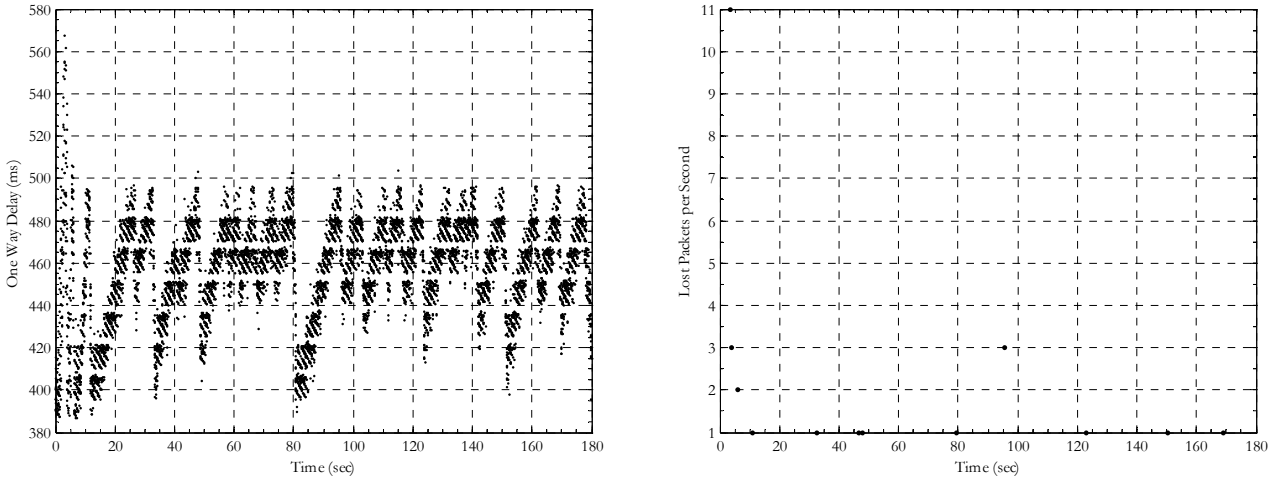
Στο Σχήμα 26 βλέπουμε την VoIP κίνηση (One Way Delay, losses) σε real-time για πρώτο χρήστη από τον service provider προς τον τελικό χρήστη αλλά και την κίνηση από τον τελικό χρήστη προς τον service provider. Και στο Σχήμα 27 βλέπουμε την VoIP κίνηση σε real-time από τον τελικό χρήστη προς τον service provider για τον πρώτο χρήστη. Παρατηρούμε ότι παρότι έχουμε σχετικά μικρό ποσοστό απωλειών για την VoIP υπηρεσία η καθυστέρηση δεν είναι ομοιόμορφα κατανεμημένη. Και στο Σχήμα 28 βλέπουμε την multicast υπηρεσία IPTV σε real-time και εδώ παρατηρούμε όπως και στην

VoIP ότι και εδώ το ποσοστό απωλειών είναι μικρό αλλά η καθυστέρηση δεν είναι ομοιόμορφα κατανεμημένη ούτε σε αυτή την υπηρεσία



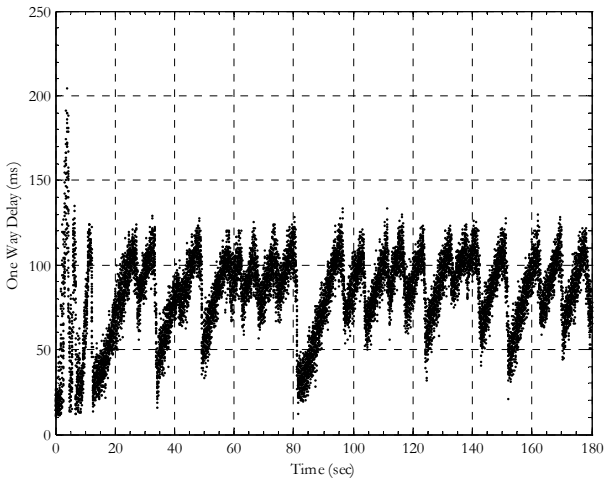
α β

Σχήμα 26. Server to Client α) One Way Delay β) Losses

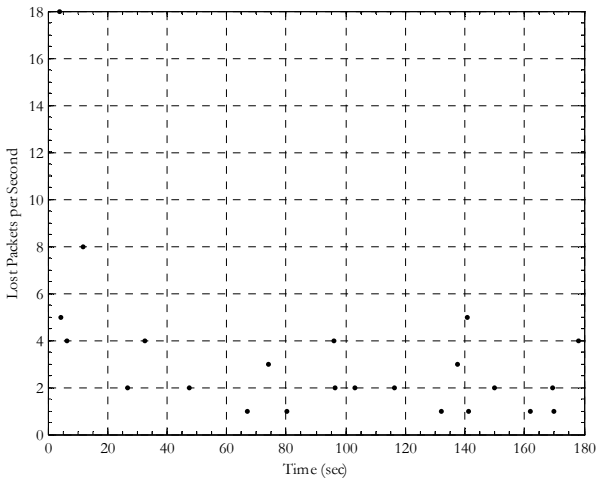


α β

Σχήμα 27. Client to Server α) One Way Delay β) Losses



α

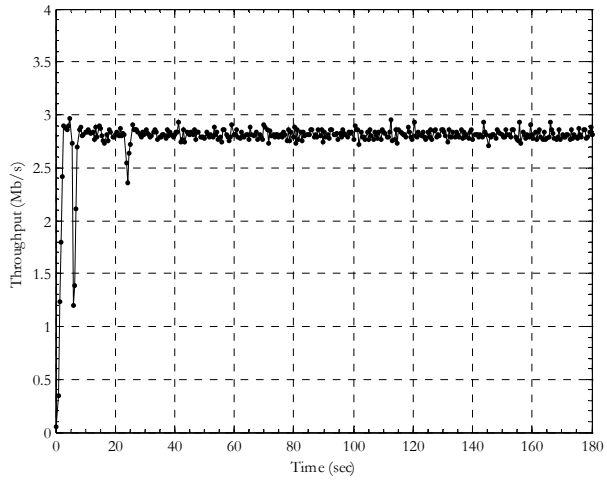


β

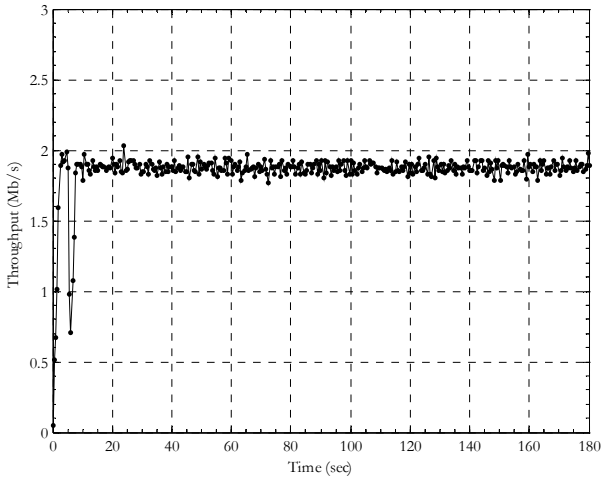
Σχήμα 28. IPTV α) One Way Delay β) Losses

4.2. Σενάριο 2

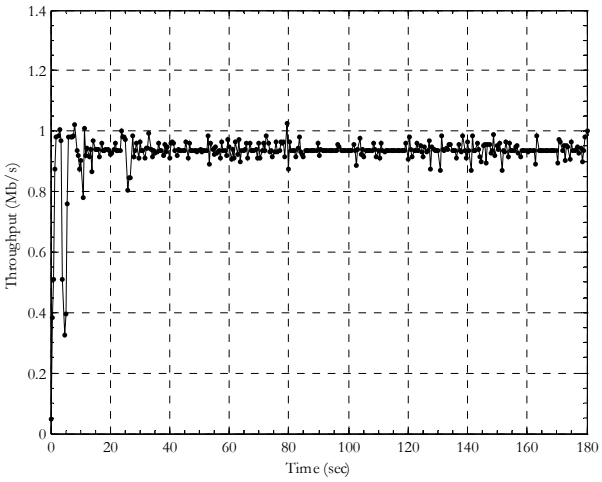
Στο δεύτερο σενάριο ενεργοποιήσαμε τον μηχανισμό DiffServ και πραγματοποιήσαμε τις ίδιες μετρήσεις. Η παρουσίαση των μετρήσεων για τις TCP ροές θα γίνει ανά κατηγορία δηλαδή AF1, AF2, AF3, BE ενώ για την EF κλάση η παρουσίαση θα γίνει ανά χρήστη. Στο Σχήμα 29 βλέπουμε το throughput σε real-time για κάθε κλάση ξεχωριστά. Για την AF1 κλάση έχουμε Throughput 2,77Mbps, στην κλάση AF2 1,85 Mbps, στην κλάση AF3 0,92Mbps και τέλος η κλάση BE 0.09Mbps



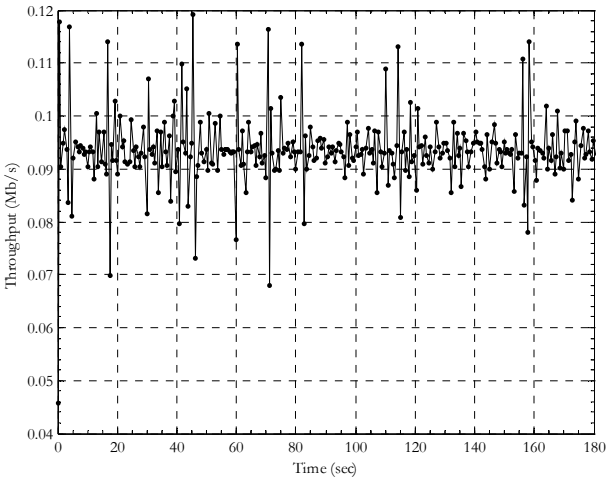
α



β



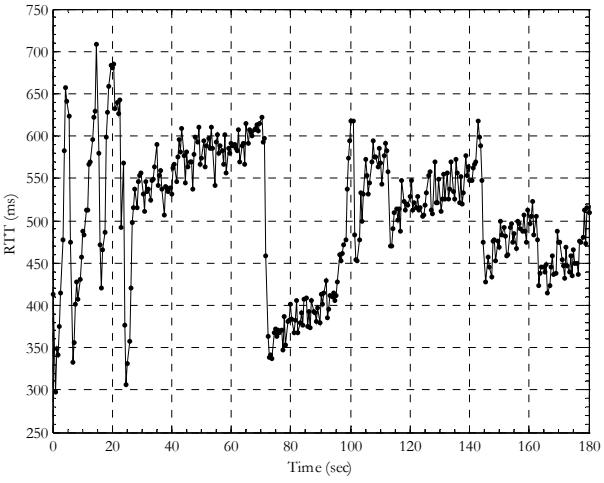
γ



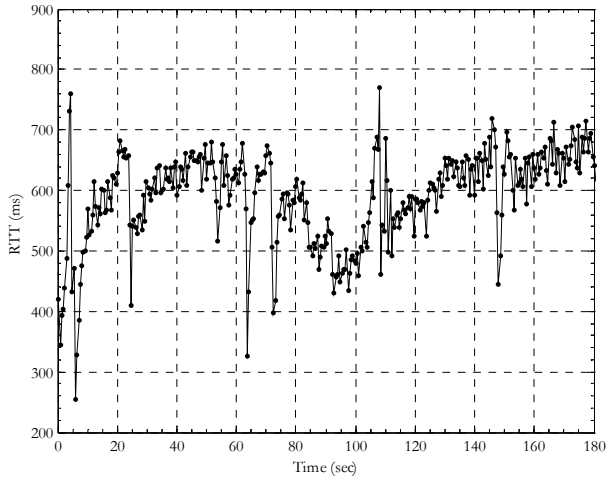
δ

Σχήμα 29. α)Throughput για AF1 class β) Throughput για AF2 class γ) Throughput για AF3 class δ) Throughput για BE class

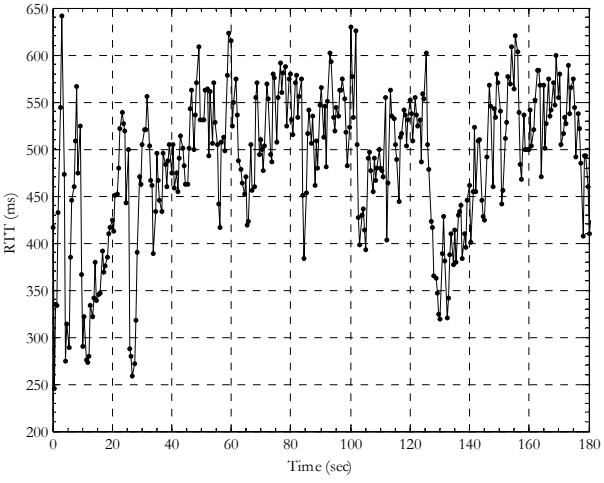
Και στο Σχήμα 30 βλέπουμε το RTT σε real-time για την κάθε κλάση AF1, AF2, AF3 και BE.



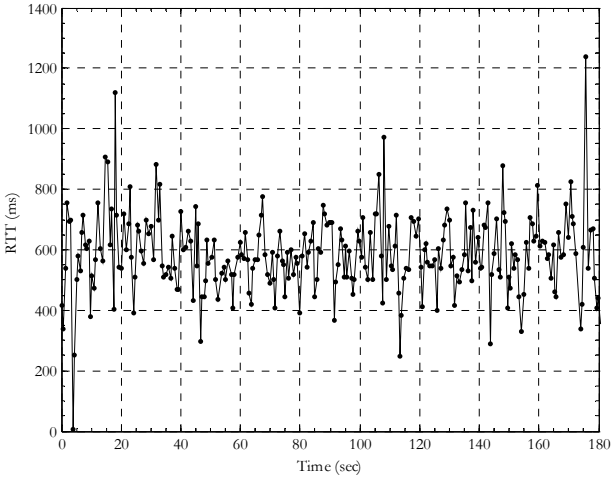
α



β



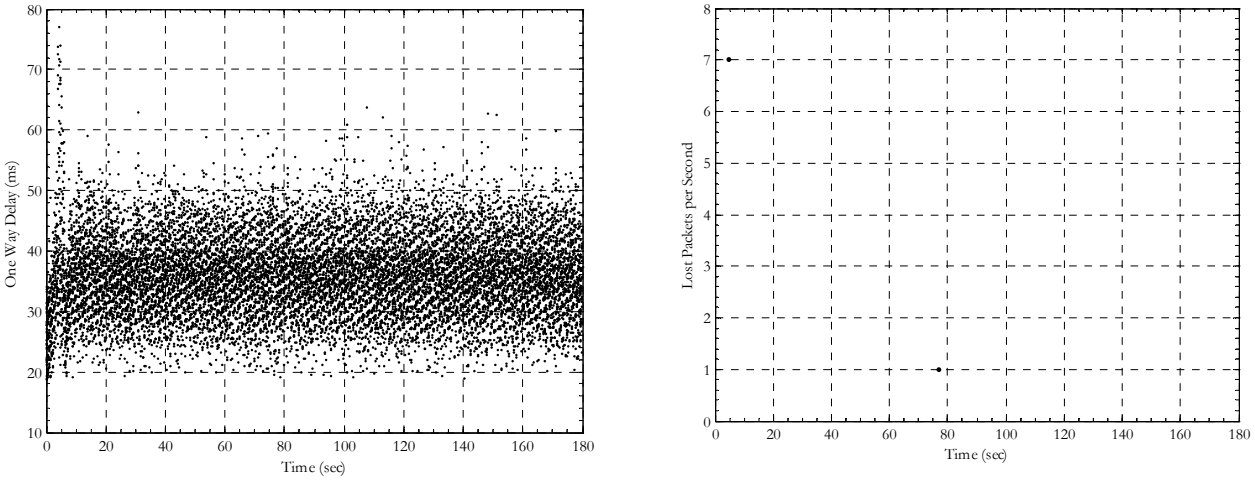
γ



δ

Σχήμα 30. α)RTT για AF1 class β) RTT για AF2 class γ) RTT για AF3 class δ) RTT για BE class

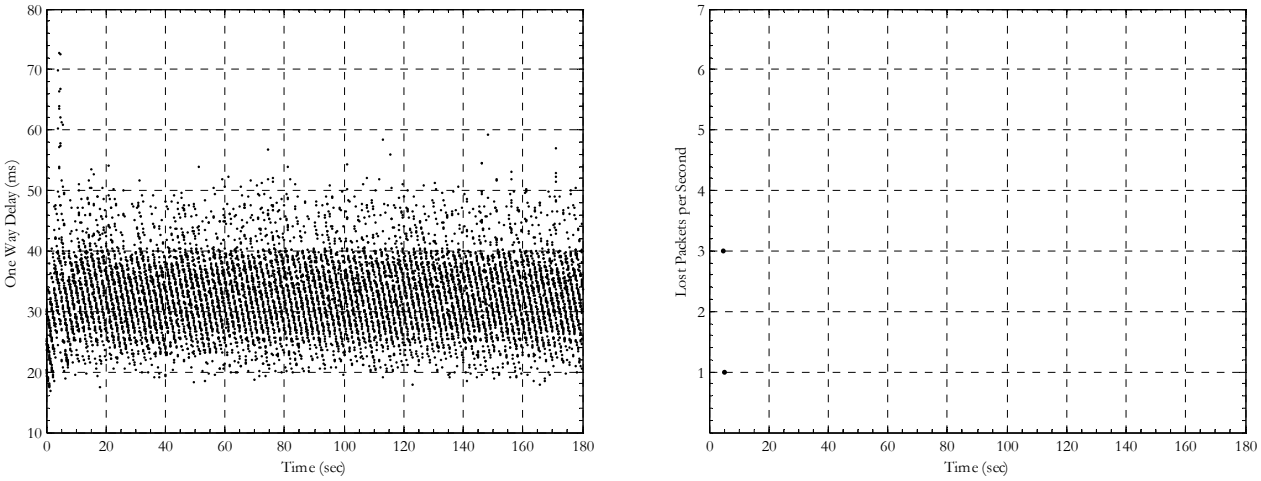
Ενώ για την IPTV υπηρεσία στο Σχήμα 31α βλέπουμε το one way delay που έχει ομοιόμορφη κατανομή σε σχέση με το πρώτο σενάριο.



α β

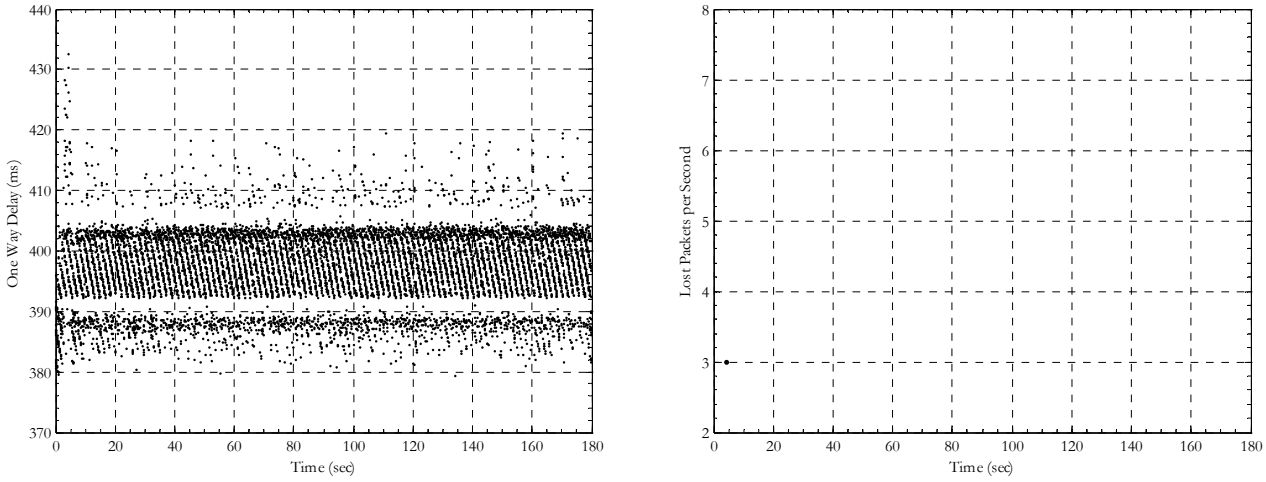
Σχήμα 31. IPTV με DiffServ α) One Way Delay β) Losses

Και για την VoIP υπηρεσία παρατηρούμε επίσης μια ομοιομορφία και από τον service provider προς τον τελικό χρήστη(Σχήμα 32) και από τον τελικό χρήστη προς τον service provider(Σχήμα 33).



α β

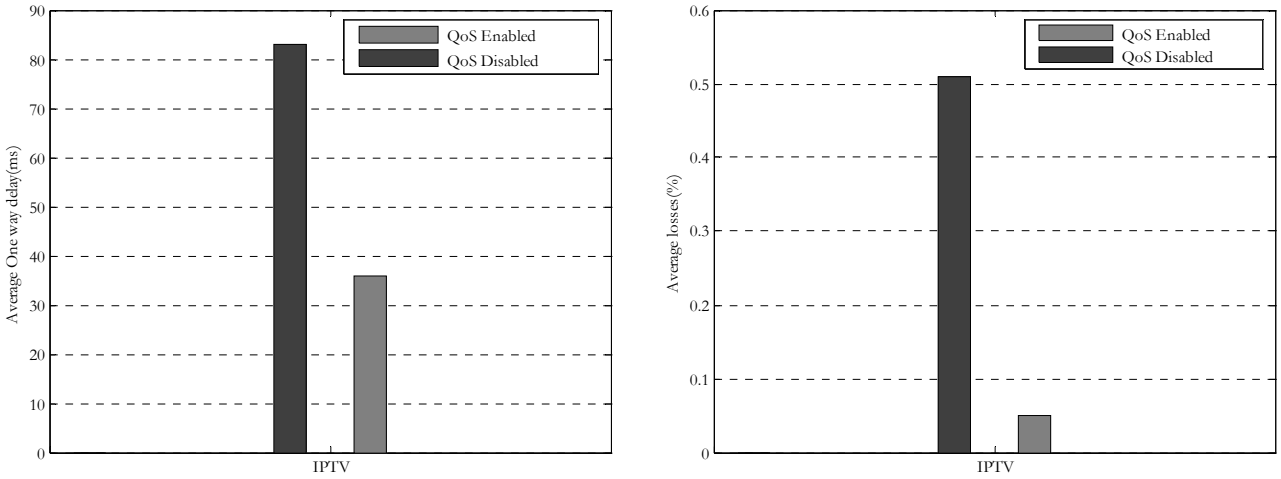
Σχήμα 32. VoIP server to client για τον 1ο χρήστη με DiffServ α) One Way Delay β) Losses



α β

Σχήμα 33. VoIP client to server για τον 1ο χρήστη με DiffServ α) One Way Delay β) Losses

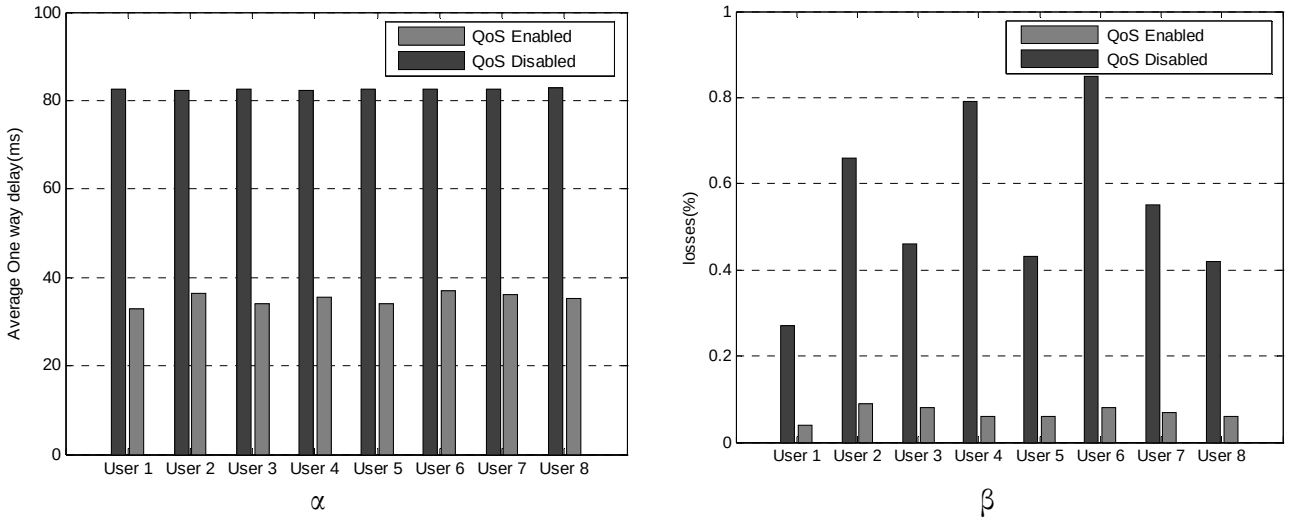
Σε αυτό το σημείο θα κάνουμε μια σύγκριση του πρώτου με το δεύτερο σενάριο για να δούμε τις βελτιώσεις που έχει επιφέρει η εφαρμογή του μηχανισμού στο δίκτυο μας στο Σχήμα 34α παρατηρούμε μία μείωση της καθυστέρησης κατά 49,7ms και τα losses μειώνονται κατά 0,23% (Σχήμα 34β).



α β

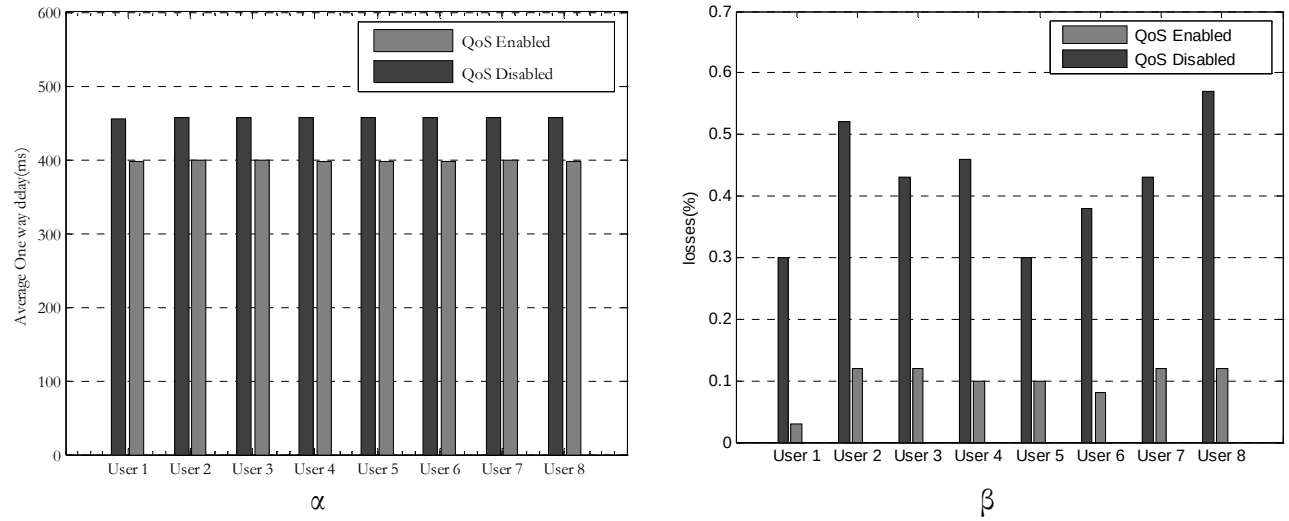
Σχήμα 34. IPTV α)One Way Delay β)Losses

Στην VoIP υπηρεσία από τον service provider προς τον τελικό χρήστη(Σχήμα 35) παρατηρούμε μια μείωση του one way delay περίπου κατά 47,4ms και losses κατά 0,48%



Σχήμα 35. VoIP server to client α)One Way Delay β)Losses

Και VoIP κίνηση από τον τελικό χρήστη προς τον service provider(Σχήμα 36) έχουμε μια μείωση του one way delay περίπου κατά 58,75ms και μείωση των απωλειών κατά 0,32% περίπου.



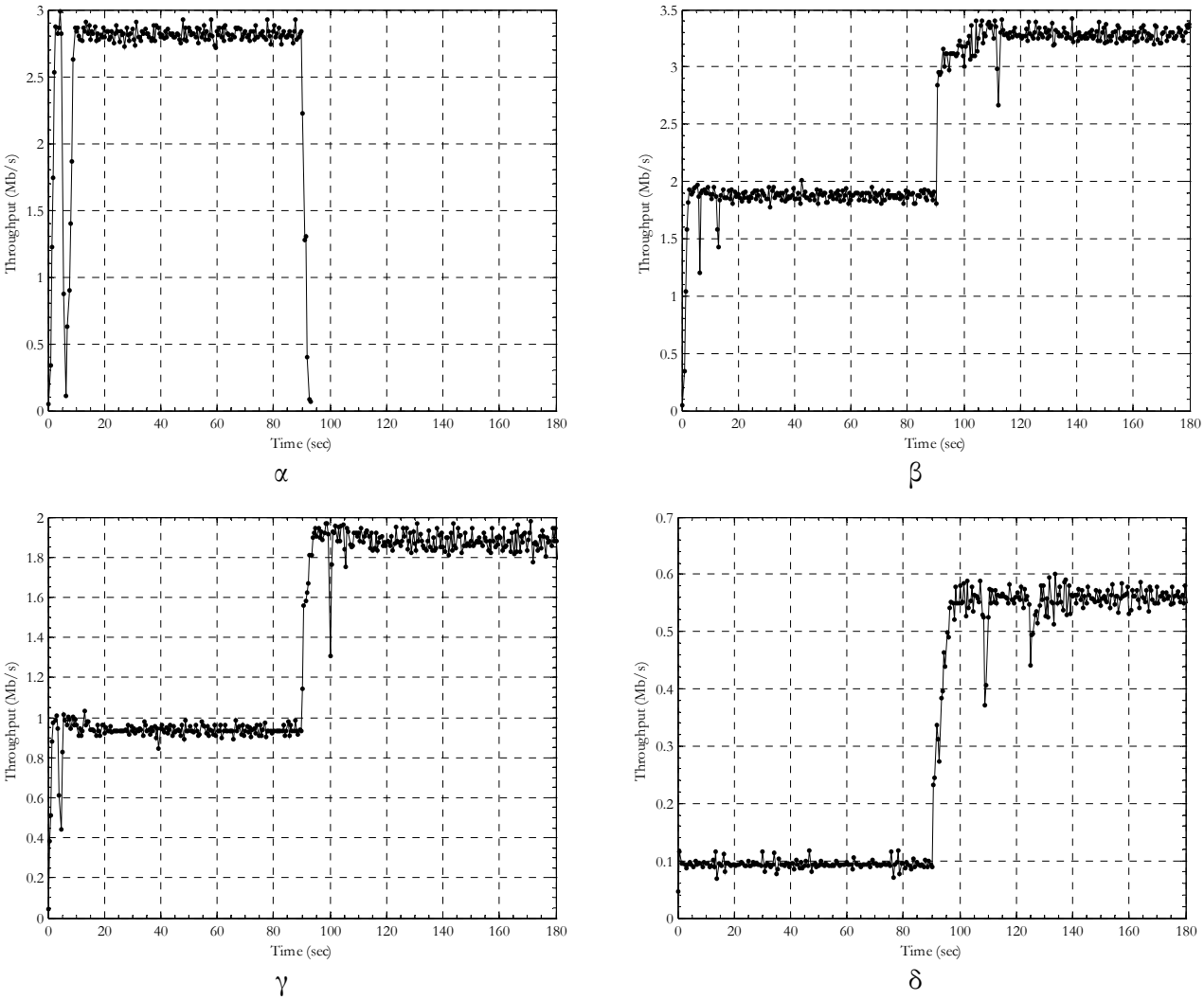
Σχήμα 36. VoIP client to server α)One Way Delay β)Losses

Αυτό που καταφέραμε με την εφαρμογή του DiffServ στο σύστημα μας είναι να μειώσουμε και να σταθεροποιήσουμε την καθυστέρηση αλλά και να μειώσουμε τις απώλειες κάτι που τελικά αποδείχτηκε μέσα από τα αποτελέσματα των πειραματικών μετρήσεων που είδαμε παραπάνω.

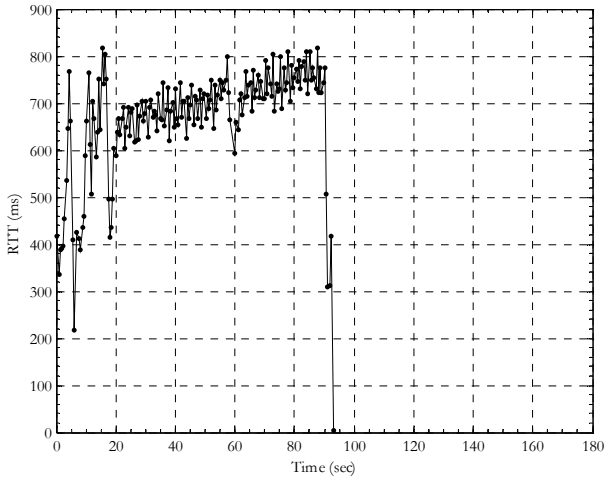
4.3. Σενάριο 3

Σε αυτό το σενάριο δείξαμε την δυναμικότητα του συστήματος κρατώντας τις ίδιες παραμέτρους με το σενάριο 2 για τα πρώτα 90sec όπου υπάρχουν όλες οι υπηρεσίες και μετά τα 90sec που σταμάτησαν ο πρώτος και ο δεύτερος χρήστης που είναι AF1 κλάση να

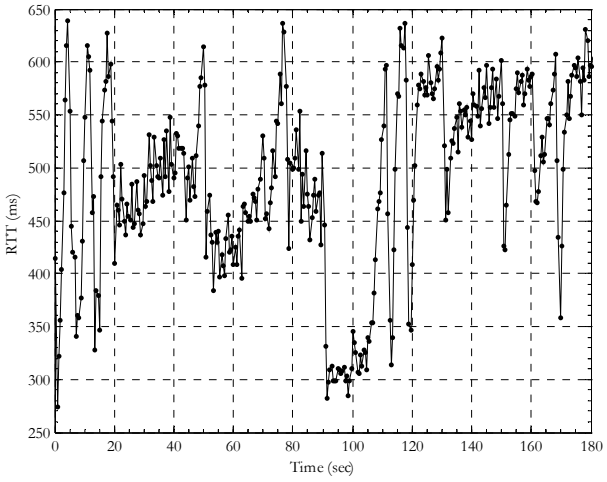
ζητά TCP υπηρεσίες βλέπουμε τις υπόλοιπες κλάσεις AF2, AF3 και BE να μοιράζονται δυναμικά το διαθέσιμο εύρος. Η κλάση AF1 να σταματά να ζητά κίνηση μετά τα 90sec (Σχήμα 37α) ενώ την ίδια χρονική στιγμή παρατηρούμε την κλάση AF2 από 1,84Mbps throughput να ανεβαίνει στα 3,34Mbps (Σχήμα 37β) την κλάση AF3 από 0,92Mbps να ανεβαίνει στα 1,88Mbps (Σχήμα 37γ) και την κλάση BE από 0,09Mbps να ανεβαίνει στα 0,54Mbps (Σχήμα 37γ) και να σταθεροποιούνται σε αυτές τις τιμές. Ενώ στο RTT (Σχήμα 38) δεν είδαμε και πολλές αλλαγές. Για τις υπηρεσίες IPTV και VoIP δεν έχουμε κάποιες αλλαγές για το λόγο αυτό και δεν θα παρουσιάσουμε και τις γραφικές παραστάσεις λόγο του ότι παραμένουν σταθερές σε σχέση με το σενάριο 2.



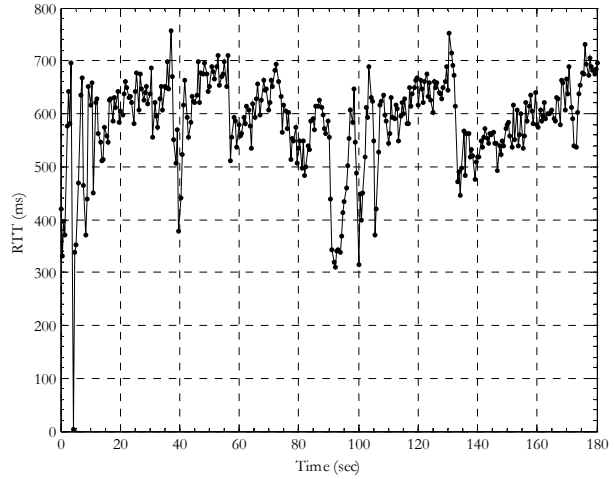
Σχήμα 37. α)Throughput για AF1 class β) Throughput για AF2 class γ) Throughput για AF3 class δ) Throughput για BE class



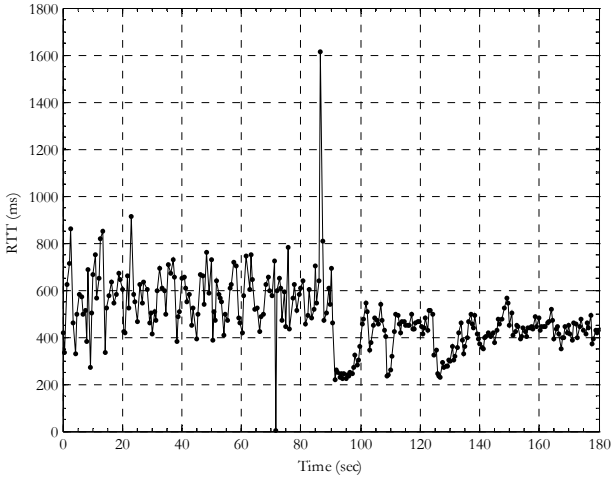
α



β



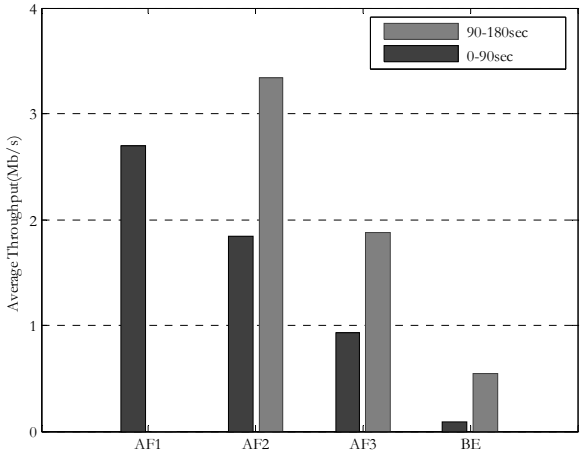
γ



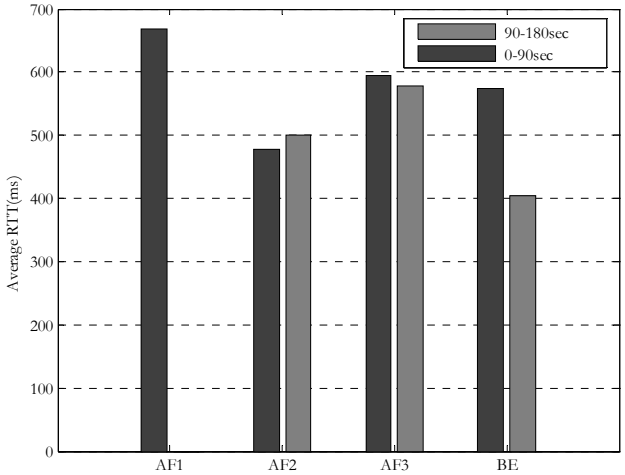
δ

Σχήμα 38. α)RTT για AF1 class β) RTT για AF2 class γ) RTT για AF3 class δ) RTT για BE class

Στο Σχήμα 39 μπορούμε να δούμε την average διαφορά του throughput και του RTT πριν τα 0sec ως τα 90sec μετά τα 90sec ως 180sec



α



β

Σχήμα 39. Σύγκριση α) Average Throughput β) Average RTT

5. Συμπεράσματα

Με την ολοκλήρωση την ολοκλήρωση της πτυχιακής αυτής εργασίας μπορέσαμε να δούμε την λειτουργία ενός υβριδικού συστήματος που αποτελείται από πολλές και διαφορετικές τεχνολογίες όπως είναι (WLAN, DVB-T, DVB-RCS) και το πώς μπορούν να συνυπάρχουν και να συνεργάζονται.

Ενώ με την τεχνολογία DiffServ μπορέσαμε να δείξουμε ότι μπορούμε να διαχειριστούμε και να κατηγοριοποιήσουμε την κίνηση ανάλογα με τους διαθέσιμους πόρους και σύμφωνα με το σενάριο που επιθυμούμε. Επίσης μας έδωσε την δυνατότητα να βελτιώσουμε τις επιδώσεις του δικτύου μας και να βελτιστοποιήσουμε τις υπηρεσίες που έχουν προτεραιότητα σε σχέση με τις άλλες υπηρεσίες. Ειδικότερα είδαμε μια βελτιστοποίηση στην IPTV υπηρεσία κατά 49,7ms για την καθυστέρηση και μείωση των απωλειών κατά 0,23% ακόμη είδαμε και την ομοιόμορφη κατανομή της μονόδρομης καθυστέρησης των πακέτων κατά την διάρκεια των 180 δευτερολέπτων που διαρκούσε η κάθε μέτρηση κάτι που στην περίπτωση που δεν είχαμε ενεργοποιημένη την τεχνολογία DiffServ δεν ήταν εφικτό. Και στην VoIP υπηρεσία είδαμε την ίδια κατανομή της μονόδρομης καθυστέρησης όπως και στην IPTV υπηρεσία αλλά και την βελτίωση της μοόδρομης καθυστέρησης κατά 47,4ms και μείωση των απωλειών κατά 0,48% από τον service provider προς τους τελικούς χρήστες αλλά και 58,75ms μείωση της καθυστέρησης και 0,32% μείωση των απωλειών για την κίνηση από τους τελικούς χρήστες προς τον service provider. Ακόμη στη TCP κίνηση είδαμε την κατηγοριοποίηση των χρηστών και το πώς μπορεί να γίνει η διαχείριση των διαθέσιμων πόρων από το σύστημα με την χρήση της τεχνολογίας DiffServ. Και στο τρίτο σενάριο μέσα από τις πειραματικές μετρήσεις είδαμε την δυναμική διαχείριση των διαθέσιμων πόρων του δικτύου μας σύμφωνα πάντα με την διάθεση των πόρων που επιθυμούμε στο εκάστοτε σενάριο.

Τέλος είδαμε ότι με την τεχνολογία DiffServ μπορούμε να διαμορφώσουμε ή να κατηγοριοποιήσουμε και να βελτιώσουμε τις παρεχόμενες υπηρεσίες ανάλογα με το είδος της υπηρεσίας ή ανάλογα με τον χρήστη της υπηρεσίας και να έχουμε την διαφοροποίηση που επιθυμούμε στο δίκτυο μας.

6. Βιβλιογραφία

[1]. Information Sciences Institute University of Southern California, "INTERNET PROTOCOL", RFC 791, September 1981

[2]. Almes, G., Kalidindi, S., Zekauskas, M. (1999, September). A One-way Delay Metric for IPPM. *RFC 2679, IETF*

[3]. Almes, G., Kalidindi, S., Zekauskas, M. (1999, September). A One-way Packet Loss Metric for IPPM. *RFC 2680, IETF*

[4]. ETSI, "Interaction channel for satellite distribution. systems (DVB-RCS)", ETSI EN-301.790, 2002

[5]. Towards a theory of differentiated services http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=847957

[6]. W. Stevens, NOAO, RFC 2001, January 1997

[7]. Huston G., "Next Steps for the IP QoS Architecture", RFC 2990, November 2000

[8]. The Internet Engineering Task Force (IETF), June 2007, http://www.ietf.org

[9]. R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, IETF, June 1994.

[10]. Heinanen J., Baker, W. Weiss F. and Wroclawski J., "Assured Forwarding PHB Group", RFC 2597, June 1999

[11]. B. Davie, A. Charny, Cisco Systems, Inc., J.C.R. Bennett, Motorola, K. Benson, Tellabs, J.Y. Le Boudec,EPFL, W. Courtney, TRW,S. Davari, PMC-Sierra, V. Firoiu, Nortel Networks, D. Stiliadis, Lucent Technologies, RFC 3246, March 2002

[12]. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, IETF, December 1998.

[13]. E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, IETF, January 2001.

[14]. Jacobson V., Nichols K. and Poduri K., "An Expedited Forwarding PHB", RFC 2598, June 1999.

[15]. Heinanen J., Baker, W. Weiss F. and Wroclawski J., "Assured Forwarding PHB Group", RFC 2597, June 1999.

[16]. MGEN - The Multi-Generator Toolset,(http://mgen.pf.itd.nrl.navy.mil/).

[17]. IPERF – Traffic Generator (Tool http://dast.nlanr.net/Projects/Iperf)

[18]. D-ITG – Traffic Generator Tool (http://www.grid.unina.it/software/ITG/index.php)

[19]. iproute2+tc notes, (http://snafu.freedom.org/linux2.2/iproute-notes.html).

[20]. Linux Advanced Routing & Traffic Control, (http://lartc.org/).

[21]. Differentiated Service on Linux HOWTO, (http://opalsoft.net/qos/DS.htm).

[22]. IP Performance Metrics (IPPM) Working group, (http://www.ietf.org/html.charters/ippm-charter.html).

[23]. TCPdump - dump traffic on a network, (http://www.tcpdump.org/

[24]. Iptables, (http://www.netfilter.org/projects/iptables/index.html).

[25]. Tcptrace, (http://jarok.cs.ohiou.edu/software/tcptrace/).

7. Συντομογραφίες

AF	Assured Forwarding
AoD	Audio On Demand
AP	access point
BA	Behavior Aggregate
BE	Best Effort
CMN	Cell Main Node
DCR	DiffServ Core Routers
DER	DiffServ Edge Routers
DiffServ	Differentiated Services
DS	Differentiated Services
DSCP	Differentiated Services CodePoint
DSL	Digital Subscriber Loop
DVB	Digital Video Broadcasting
DVB-H	Digital Video Broadcasting - Handheld
DVB-RCS	Digital Video Broadcasting - Return Channel via Satellite
DVB-S	Digital Video Broadcasting - Satellite
DVB-T	Digital Video Broadcasting - Terrestrial
EF	Expedited Forwarding
FIFO	First In First Out
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GRED	Generalized Random Early Detection
GSM	Global System for Mobile communications
HTB	Hierarchical Token Bucket
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol
ISDN	Integrated Services Digital Network / Isolated Subscriber Digital Network
LAN	Local Area Network

MPLS	Multi-Protocol Label Switching
NTP	Network Time Protocol
OFDM	Orthogonal frequency-division multiplexing
PC	Personal Computer
PDA	Personal digital Assistants
PHB	Per-Hop Behavior
PRIO	Priority Queuing
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RED	Random Early Detection
RTT	Round Trip Time
SFQ	Stochastic Fair Queuing
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
TBF	Token Bucket Filter
TCA	Traffic Conditioning Agreement
TCP	Transmission Control Protocol
Telnet	TELecommunication NETwork
ToS	Type of Service Field
UDP	User Datagram Protocol
VoD	Video On Demand
VoIP	Voice over Internet Protocol
WAN	Wide Area Network
WLAN	wireless local area network
WWW	World Wide Web

8. Παράρτημα

8.1. Παραδείγματα Δημιουργίας Κίνησης

- Παράδειγμα UDP κίνηση

  Στον υπολογιστή που δημιουργεί την κίνηση

  mgen input script_file

  script_file

  0.0 ON 1 UDP DST 192.168.0.5/7000 SRC 9500 PERIODIC [pack_per_sec size]

  180.0 OFF 1

  Στον υπολογιστή που λαμβάνει την κίνηση δεν χρειάζεται κάποια εντολή

- Παράδειγμα TCP κίνηση

  Στον υπολογιστή που δημιουργεί την κίνηση

  Iperf –c 192.xx.xx.xx

  Και στον υπολογιστή που λαμβάνει την κίνηση

  Iperf –s

- Παράδειγμα VoIP κίνηση

  Στον υπολογιστή που δημιουργεί την κίνηση

  ITGSend -a 172.16.0.5 -sp 7000 -t 120000 VoIP -x G.729.2

  Και στον υπολογιστή που λαμβάνει την κίνηση

  ITGRecv

8.2. Παράδειγμα Σύλληψης Κίνησης

- Παράδειγμα tcpdump

  tcpdump –w filename -n -vv -i eth0 port 7000

8.3. Scripts

8.3.1. DCR

8.3.1.1. Script για Σενάριο 2

```
#tc qdisc del dev eth0 root

tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 set_tc_index
tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2

tc filter add dev eth0 parent 1:0 protocol ip prio 1 handle 0x2e tcindex classid 1:111
tc filter add dev eth0 parent 1:0 protocol ip prio 1 handle 10 tcindex classid 1:121
tc filter add dev eth0 parent 1:0 protocol ip prio 1 handle 18 tcindex classid 1:131
tc filter add dev eth0 parent 1:0 protocol ip prio 1 handle 26 tcindex classid 1:141
tc filter add dev eth0 parent 1:0 protocol ip prio 2 handle 0 tcindex mask 0 classid 1:1

tc qdisc add dev eth0 parent 1:0 handle 2:0 htb
tc class add dev eth0 parent 2:0 classid 2:1 htb rate 7.35Mbit burst 12k ceil 7.55Mbit
tc filter add dev eth0 parent 2:0 protocol ip prio 1 tcindex mask 0xf0 shift 4 pass_on
#EF
tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 1 tcindex classid 2:10
tc class add dev eth0 parent 2:1 classid 2:10 htb rate 1250Kbit burst 15k ceil 1300Kbit
prio 1
tc qdisc add dev eth0 parent 2:10 pfifo limit 30
#AF1
tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 2 tcindex classid 2:20
tc class add dev eth0 parent 2:1 classid 2:20 htb rate 3000Kbit burst 8k ceil 3050Kbit prio
2
tc qdisc add dev eth0 parent 2:20 pfifo limit 95
#AF2
tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 3 tcindex classid 2:30
tc class add dev eth0 parent 2:1 classid 2:30 htb rate 2000Kbit burst 6k ceil 2050Kbit prio
3
tc qdisc add dev eth0 parent 2:30 pfifo limit 60
#AF3
tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 4 tcindex classid 2:40
tc class add dev eth0 parent 2:1 classid 2:40 htb rate 1000Kbit burst 4k ceil 1050Kbit prio
4
tc qdisc add dev eth0 parent 2:40 pfifo limit 30
#BE
tc filter add dev eth0 parent 2:0 protocol ip prio 2 handle 0 tcindex classid 2:50
tc class add dev eth0 parent 2:1 classid 2:50 htb rate 100Kbit burst 4k ceil 100Kbit prio 5
tc qdisc add dev eth0 parent 2:50 pfifo limit 5
```

8.3.1.2. Script για Σενάριο 3

```
#!/bin/bash

#AF1
tc class del dev eth0 parent 2:1 classid 2:20 htb rate 3000Kbit burst 6k ceil 3050Kbit prio
2
#AF2
tc class change dev eth0 parent 2:1 classid 2:30 htb rate 3500Kbit burst 6k ceil 3600Kbit
prio 3
tc qdisc change dev eth0 parent 2:30 pfifo limit 100
#AF3
tc class change dev eth0 parent 2:1 classid 2:40 htb rate 2000Kbit burst 4k ceil 2050Kbit
prio 4
tc qdisc change dev eth0 parent 2:40 pfifo limit 65
#BE
tc class change dev eth0 parent 2:1 classid 2:50 htb rate 600Kbit burst 4k ceil 600Kbit
prio 5
tc qdisc change dev eth0 parent 2:50 pfifo limit 15
```

8.3.2. DER

8.3.2.1. CMN 1 (Server Side)

```
#!/bin/bash

clear

#delete any qdisc
tc qdisc del dev ath0 ingress
tc qdisc del dev ath0 root
tc qdisc del dev eth2 ingress
tc qdisc del dev eth2 root


#clear iptables mangle chain
iptables -t mangle -F
iptables -t mangle -L -n

iptables -t mangle -A PREROUTING -p UDP -s 172.16.0.5 -d 192.168.0.5 -m multiport
--destination-ports 7000,7001 -j DSCP --set-dscp-class EF

iptables -t mangle -A PREROUTING -p TCP -s 172.16.0.5 -d 192.168.0.5 -m multiport -
-destination-ports 9500,9501 -j DSCP --set-dscp-class AF11
```

iptables -t mangle -A PREROUTING -p TCP -s 172.16.0.5 -d 192.168.0.5 -m multiport --destination-ports 9502,9503 -j DSCP --set-dscp-class AF21

iptables -t mangle -A PREROUTING -p TCP -s 172.16.0.5 -d 192.168.0.5 -m multiport --destination-ports 9504,9505 -j DSCP --set-dscp-class AF31

iptables -t mangle -A PREROUTING -p TCP -s 172.16.0.5 -d 192.168.0.5 -m multiport --destination-ports 9506,9507 -j DSCP --set-dscp-class BE

8.3.2.2. CMN 2 (Client Side)

```
#!/bin/bash

clear

#delete any qdisc
tc qdisc del dev eth1 ingress
tc qdisc del dev eth1 root
tc qdisc del dev eth2 ingress
tc qdisc del dev eth2 root

#clear iptables mangle chain
iptables -t mangle -F
iptables -t mangle -L -n

iptables -t mangle -A PREROUTING -p UDP -s 192.168.0.5 -d 172.16.0.5 --dport 7002 -j DSCP --set-dscp-class EF

iptables -t mangle -A PREROUTING -p TCP -s 192.168.0.5 -m multiport --source-ports 9500,9501 -d 172.16.0.5 -j DSCP --set-dscp-class AF11

iptables -t mangle -A PREROUTING -p TCP -s 192.168.0.5 -m multiport --source-ports 9502,9503 -d 172.16.0.5 -j DSCP --set-dscp-class AF21

iptables -t mangle -A PREROUTING -p TCP -s 192.168.0.5 -m multiport --source-ports 9504,9505 -d 172.16.0.5 -j DSCP --set-dscp-class AF31

iptables -t mangle -A PREROUTING -p TCP -s 192.168.0.5 -m multiport --source-ports 9506,9507 -d 172.16.0.5 -j DSCP --set-dscp-class BE
```

8.4. Προγράμματα Ανάλυσης Κίνησης

8.4.1. UDP κίνηση

8.4.1.1. Script replicid (ipv4_replicid.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w


open(AFILE, "<udp_sender.txt") || die ("cannot open udp_sender file\n");
open(BFILE, "<udp_receiver.txt") || die ("cannot open udp_receiver file \n");
my ($temp);
my ($temp1);
my (@rec_lines) = <AFILE>;

 $start = time;


for ($j=0;$j<@rec_lines;$j++)
{

 @udp_id=split(/[\t +]/,$rec_lines[$j]);
 for ($k=0;$k<@udp_id;$k++)
 {
        $temp=0;
        if ($udp_id[$k] eq "cid")
                {
                        $temp=$udp_id[$k+1];
                        last;
                }

 }
```

```
for ($k=0;$k<@udp_id;$k++)
 {

          if ($udp_id[$k] eq "seq")
                  {

                          $temp=$temp.($udp_id[$k+1]);
                          last;
                  }

 }
 for ($k=0;$k<@udp_id;$k++)
 {

          if ($udp_id[$k] eq "ser")
                  {

                          $temp=$temp.($udp_id[$k+1]);
                          last;
                  }

 }
 for ($k=0;$k<@udp_id;$k++)
 {

          if ($udp_id[$k] eq "id")
                  {

                          $temp=$temp.($udp_id[$k+1]);
                          last;
                  }

 }
#print "$temp\n";
for ($i=$j+1;$i<@rec_lines;$i++)
{
  @udp_id_compare=split(/[\t +]/,$rec_lines[$i]);
          for ($k=0;$k<@udp_id;$k++)
          {
           $temp1=0;
           if ($udp_id_compare[$k] eq "cid")
                  {
                          $temp1=$udp_id_compare[$k+1];
                          last;
                  }

          }
          for ($k=0;$k<@udp_id;$k++)
 {

           if ($udp_id_compare[$k] eq "seq")
```

```
                {

                        $temp1=$temp1.($udp_id_compare[$k+1]);
                        last;
                }

}
for ($k=0;$k<@udp_id;$k++)
{

        if ($udp_id_compare[$k] eq "ser")
                {

                        $temp1=$temp1.($udp_id_compare[$k+1]);
                        last;
                }

}
 for ($k=0;$k<@udp_id;$k++)
{

        if ($udp_id_compare[$k] eq "id")
                {

                        $temp1=$temp1.($udp_id_compare[$k+1]);
                        last;
                }

}
#print "$temp1\n";
if ($temp eq $temp1)
{
  print "Found two instances (sender file) of $temp1 at line $i $j\n";
exit(12);
}

}

#print "$sen_line\n";


}

$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
  print "Time elapsed for senders file: $hour hours:$minute min:$second sec\n";

close(AFILE);
```

```
my (@rec_lines) = <BFILE>;


for ($j=0;$j<@rec_lines;$j++)
{




@udp_id=split(/[\t +]/,$rec_lines[$j]);
 for ($k=0;$k<@udp_id;$k++)
 {
        $temp=0;
        if ($udp_id[$k] eq "cid")
                {
                        $temp=$udp_id[$k+1];
                        last;
                }

 }

for ($k=0;$k<@udp_id;$k++)
 {

        if ($udp_id[$k] eq "seq")
                {

                        $temp=$temp.($udp_id[$k+1]);
                        last;
                }

 }
 for ($k=0;$k<@udp_id;$k++)
 {

        if ($udp_id[$k] eq "ser")
                {

                        $temp=$temp.($udp_id[$k+1]);
                        last;
                }

 }

#print "$temp\n";
for ($i=$j+1;$i<@rec_lines;$i++)
{
  @udp_id_compare=split(/[\t +]/,$rec_lines[$i]);
for ($k=0;$k<@udp_id;$k++)
```

```
        {
         $temp1=0;
         if ($udp_id_compare[$k] eq "cid")
                {
                        $temp1=$udp_id_compare[$k+1];
                        last;
                }

        }
        for ($k=0;$k<@udp_id;$k++)
{

         if ($udp_id_compare[$k] eq "seq")
                {

                        $temp1=$temp1.($udp_id_compare[$k+1]);
                        last;
                }

}
for ($k=0;$k<@udp_id;$k++)
{

         if ($udp_id_compare[$k] eq "ser")
                {

                        $temp1=$temp1.($udp_id_compare[$k+1]);
                        last;
                }

}
#print "$temp1\n";

if ($temp eq $temp1)
{
  print "Found two instances (receiver file) of $temp1 at line $i $j\n";
exit(12);
}

}

#print "$sen_line\n";


}


$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
```

```
my $hour = ($elapsed_sec/(60*60))%24;
  print "Total Time elapsed: $hour hours:$minute min:$second sec\n";

close(BFILE);
```

8.4.1.2. Script createedfiles (ipv4_createendfiles.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w

open(INPUTFILE, "<udp_sender.txt") || die ("cannot open udp_sender file 1\n");

unless (open (OUTFILE, ">final.tx"))
{

        die ("cannot open output file outfile\n");

}


$start = time;

my (@rec_lines) = <INPUTFILE>;

foreach $rec_line (@rec_lines)
{
        chomp($rec_line);
  @line=split(/[\t +]/,$rec_line);

        for ($k=0;$k<@line;$k++)
 {
         $id=0;
         if ($line[$k] eq "cid")
                 {
                         $id=$line[$k+1];
                         last;
                 }
```

```
}

for ($k=0;$k<@line;$k++)
 {

        if ($line[$k] eq "seq")
                {

                        $id=$id.($line[$k+1]);
                        last;
                }

 }
 for ($k=0;$k<@line;$k++)
 {

        if ($line[$k] eq "ser")
                {

                        $id=$id.($line[$k+1]);
                        last;
                }

 }
 for ($k=0;$k<@line;$k++)
 {

        if ($line[$k] eq "id")
                {

                        $id=$id.($line[$k+1]);
                        last;
                }

 }


for ($k=0;$k<@line;$k++)
 {

        if ($line[$k] eq "length:")
                {

                        $size=$line[$k+1];
                        last;
                }

 }
  $time=$line[0];
  #$id=substr($id,0,length($id)-1);
```

```
   $size=substr($size,0,length($size)-1);
#print ("$id $size \n");
 print OUTFILE ("$id\t$id\t$time\t$size\n");

}

close(INPUTFILE);
close(OUTPUTFILE);

#second file

open(INPUTFILE, "<udp_receiver.txt") || die ("cannot open udp_receiver file 1\n");

unless (open (OUTFILE, ">final.rx"))
{

        die ("cannot open output file outfile\n");

}


my (@rec_lines) = <INPUTFILE>;

foreach $rec_line (@rec_lines)
{
        chomp($rec_line);
  @line=split(/[\t +]/,$rec_line);
        for ($k=0;$k<@line;$k++)
 {
         $id=0;
         if ($line[$k] eq "cid")
                {
                        $id=$line[$k+1];
                        last;
                }

 }

for ($k=0;$k<@line;$k++)
 {

         if ($line[$k] eq "seq")
                {

                        $id=$id.($line[$k+1]);
                        last;
                }

 }
 for ($k=0;$k<@line;$k++)
 {
```

```
        if ($line[$k] eq "ser")
                {

                        $id=$id.($line[$k+1]);
                        last;
                }

}
for ($k=0;$k<@line;$k++)
{

        if ($line[$k] eq "id")
                {

                        $id=$id.($line[$k+1]);
                        last;
                }

}

for ($k=0;$k<@line;$k++)
{

        if ($line[$k] eq "length:")
                {

                        $size=$line[$k+1];
                        last;
                }

}
  $time=$line[0];
  #$id=substr($id,0,length($id)-1);
  $size=substr($size,0,length($size)-1);

 print OUTFILE ("$id\t$id\t$time\t$size\n");

}


$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
  print "Total Time elapsed: $hour hours:$minute min:$second sec\n";

close(INPUTFILE);
close(OUTPUTFILE);
```

8.4.1.3. Script losses (ipv_all_losses.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
#Author irons
#Mail irons@pasiphae.teiher.gr
#This file calculates the losses in a udp transmission.

#!/usr/bin/perl -w

$start = time;

& calc_loss;

$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
  print "Total Time elapsed: $hour hours:$minute min:$second sec\n";


sub calc_loss
# Simple loss calculation
{
        my ($sender_packets);
        my ($receiver_packets);
        my ($loss_rate);

        $sender_packets = 0;
        $receiver_packets = 0;
        $loss_rate = 0;

# Sender file
        open (SENDER, "<final.tx") || die ("cannot open input file 1\n");
```

```
        while (<SENDER>)
        {

                $sender_packets++;


        }

        close (SENDER);

# Receiver file
        open (RECEIVER, "<final.rx") || die ("cannot open input file 2\n");


        while (<RECEIVER>)
        {

                $receiver_packets++;


        }


        close (RECEIVER);

#calculation
        $loss_rate =
                (($sender_packets -
                  $receiver_packets) / $sender_packets) * 100;

        print "sender packets $sender_packets, receiver packets $receiver_packets, losses
$loss_rate%\n";

     if (($sender_packets - $receiver_packets)!=0)
{
  & lossvstime;
}

}

sub lossvstime
{

unless (open (OUTFILE, ">pack_num_seq_loss_vs_time"))
{

        die ("cannot open output file outfile\n");

}
```

```
        open (SENDER, "<final.tx");
open (RECEIVER, "<final.rx");
my (@sen_lines) = <SENDER>;
my ($sen_line);
my (@rec_lines) = <RECEIVER>;
my ($rec_line);
my ($temp);
my ($temp1);
my ($lock);
my ($lock_ref_time);
my ($send_time);
my ($start_ref_time);
my ($packet_counter);
close (SENDER);
close (RECEIVER);
$size=@rec_lines;
$size1=@sen_lines;
$packet_counter=0;
$lock_ref_time=0;

foreach $sen_line (@sen_lines)
        {
$lock=0;
$packet_counter++;
      chomp($sen_line);
                @line_sender=split(/[\t +]/,$sen_line);
                $temp1=$line_sender[0].$line_sender[1];

       if ($lock_ref_time==0)
{
 $start_ref_time=$line_sender[2];
$lock_ref_time=1;
#print ("\n$start_ref_time\n");

}
foreach $rec_line (@rec_lines)
{
        chomp($rec_line);
        @line_receiver=split(/[\t +]/,$rec_line);
        $temp=$line_receiver[0].$line_receiver[1];
if ($temp eq $temp1)
                {

                        #unless (open (OUTFILE, ">>aligned_sender"))
                        #{

                        #       die ("cannot open output file outfile\n");

                        #}
```

```
                        #close (OUTFILE);
                $lock=1;
                        last;
#print " sender $sen_line receiver $rec_line\n";
                }


        }
if ($lock==0)
{
$send_time=$line_sender[2]-$start_ref_time;
print OUTFILE "$send_time\t$packet_counter \n";

}
#if ($lock==1)
#{
#$send_time=$line_sender[2]-$start_ref_time;
#print OUTFILE "$send_time\t0 \n";

#}
}
close (OUTFILE);
& avg_lossvstime;
}
sub avg_lossvstime{

unless (open (OUTFILE, ">lossvstime"))
{

        die ("cannot open output file outfile\n");

}


open (LOSSES, "<pack_num_seq_loss_vs_time");

my (@sen_lines) = <LOSSES>;
my ($sen_line);
my ($temp1);
my ($temp);
my ($lock);
my ($lock1);
my ($send_time);
my ($packet_counter);
my ($line_counter);

close (LOSSES);

$size=@rec_lines;
```

```
$size1=@sen_lines;
$packet_counter=0;
$lock_ref_time=0;
$line_counter=0;
$lock=0;

$lock1=0;
foreach $sen_line (@sen_lines)
          {

$packet_counter++;
       chomp($sen_line);
                  @line_sender=split(/[\t +]/,$sen_line);
                  $temp=$line_sender[0];
             if ($lock==0)
{
 $temp1=$line_sender[0];
#print "$temp1\n";
@line_sender_last=split(/[\t +]/,$sen_lines[(@sen_lines) -1]);
#print "$line_sender_last[0]\n";
 #$packet_counter=0;
$lock=1;
}

if (($temp1+1.0) < ($temp))
{
  if ($lock1==0)
   {
   $pack_count=$packet_counter-1;
  print OUTFILE "$send_time\t$pack_count \n";
  #print "$send_time\t$packet_counter\t$temp \n";
  $packet_counter=1;
$temp1=$temp;
}
}
if (($temp1+1) > ($line_sender_last[0]))
{
 $lock1=1;
   if ($temp==$line_sender_last[0])
{
$send_time=$temp;
# print "$send_time\t$packet_counter\t$temp \n";
  print OUTFILE "$send_time\t$packet_counter \n";
}


}

$send_time=$temp;
$line_counter++;
}
```

```
close (OUTFILE);
#print "$pack_count\t$packet_counter\t$line_counter\n";
}
```

8.4.1.4. Script throughput (ipv_all_sender_receiver_rate.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w


my($sum_pack_size);
my($lock);
my($start_time);
my($end_time);
my($transffer_time);
my($data_rate);
$data_rate=0;
$transffer_time=0;
$sum_pack_size=0;
$lock=0;
$start_time=0;
$end_time=0;

open(SENDER, "<final.tx") || die ("cannot open input file 1\n");

$start = time;

while (<SENDER>)
{
 my($sen_line) = $_;
 chomp($sen_line);

 @line_sender=split(/[\t +]/,$sen_line);
 $sum_pack_size+=$line_sender[3];
  if ($lock==0)
{
 $start_time=$line_sender[2];
```

```
  $lock=1;
}

#print "$papa[3]\n";


}
$end_time=$line_sender[2];
#print "stime $start_time etime $end_time\n";
$transffer_time=$end_time-$start_time;
$data_rate=$sum_pack_size/$transffer_time;
print "SENDER RESULTS\n";
print "total bytes transferred $sum_pack_size in $transffer_time sec. Sender output data
rate is $data_rate bytes/sec \n";
close(SENDER);


open(RECEIVER, "<final.rx") || die ("cannot open input file 2\n");
$data_rate=0;
$transffer_time=0;
$sum_pack_size=0;
$lock=0;
$start_time=0;
$end_time=0;
while (<RECEIVER>)
{
  my($sen_line) = $_;
  chomp($sen_line);

  @line_receiver=split(/[\t +]/,$sen_line);
  $sum_pack_size+=$line_receiver[3];
   if ($lock==0)
{
  $start_time=$line_receiver[2];
  $lock=1;
}

#print "$papa[3]\n";


}
$end_time=$line_receiver[2];
print "stime $start_time etime $end_time\n";
$transffer_time=$end_time-$start_time;
$data_rate=$sum_pack_size/$transffer_time;
print "RECEIVER RESULTS\n";
print "total bytes transferred $sum_pack_size in $transffer_time sec. Receiver input data
rate is $data_rate bytes/sec \n";

close(RECEIVER);
```

```
$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

8.4.1.5. Script align for jitter (ipv_all_align_for_delay_jitt.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w

unless (open (OUTFILE, ">aligned_sender"))
{

        die ("cannot open output file outfile\n");

}




open (SENDER, "<final.tx");
open (RECEIVER, "<final.rx");
my (@sen_lines) = <SENDER>;
my ($sen_line);
my (@rec_lines) = <RECEIVER>;
my ($rec_line);
my ($temp);
my ($temp1);

close (SENDER);
close (RECEIVER);
$size=@rec_lines;
$size1=@sen_lines;
print "Receiver packets $size --- Sender packets $size1\n";
```

```
$start = time;


foreach $rec_line (@rec_lines)
{
        chomp($rec_line);
        @line_receiver=split(/[\t +]/,$rec_line);
        $temp=$line_receiver[0].$line_receiver[1];
        foreach $sen_line (@sen_lines)
        {
      chomp($sen_line);
                @line_sender=split(/[\t +]/,$sen_line);
                $temp1=$line_sender[0].$line_sender[1];
                if ($temp eq $temp1)
                {

                        #unless (open (OUTFILE, ">>aligned_sender"))
                        #{

                        #       die ("cannot open output file outfile\n");

                        #}

                        print OUTFILE "$sen_line\n";

                        #close (OUTFILE);

                        last;
#print " sender $sen_line receiver $rec_line\n";
                }

        }

}

$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
  print "Total Time elapsed: $hour hours:$minute min:$second sec\n";

close (OUTFILE);
```

8.4.1.6. Script timestamp (ipv_all_timestamp.pl)

```
#!/usr/bin/perl -w

unless (open (SENDER, "<aligned_sender"))
        {
```

```
                die ("cannot open input file outfile\n");

        }


unless (open (OUTFILE, ">sender_timestamp"))
                        {

                                die ("cannot open output file outfile\n");

                        }


$start = time;

while (<SENDER>)
{
 my($sen_line) = $_;
 chomp($sen_line);

 @line_sender=split(/[\t +]/,$sen_line);


 print OUTFILE "$line_sender[2]\n";




#print "$papa[3]\n";


}

close(OUTFILE);
close(SENDER);

unless (open (RECEIVER, "<final.rx"))
        {

                die ("cannot open input file outfile\n");

        }

unless (open (OUTFILE, ">receiver_timestamp"))
                        {

                                die ("cannot open output file outfile\n");

                        }
```

```
while (<RECEIVER>)
{
 my($sen_line) = $_;
 chomp($sen_line);

 @line_receiver=split(/[\t +]/,$sen_line);


 print OUTFILE "$line_receiver[2]\n";



#print "$papa[3]\n";


}


close(OUTFILE);

close(RECEIVER);

$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

8.4.1.7. Script smoothed jitter(RFC 3550) (ipv_all_inter_arrival_jitter.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w

unless (open (SENDER, "<sender_timestamp"))
```

```
        {

                die ("cannot open input file outfile\n");

        }
unless (open (RECEIVER, "<receiver_timestamp"))
        {

                die ("cannot open input file outfile\n");

        }

my (@sen_lines) = <SENDER>;
my ($sen_line);
my (@rec_lines) = <RECEIVER>;
my ($rec_line);
my($transit);
my($delta_transit);
my($last_transit);
my($jitter);
my($counter);
$counter=0;
$transit=0;
$delta_transit=0;
$last_transit=0;
$jitter=0;


close(SENDER);
close(RECEIVER);

unless (open (OUTFILE, ">final_jitter"))
                        {

                                die ("cannot open output file outfile\n");

                        }

unless (open (OUTFILE1, ">final_pack2packdelay"))
                        {

                                die ("cannot open output file outfile\n");

                        }

$start = time;

foreach $sen_line (@sen_lines)
{
  chomp($sen_line);
```

```
   $transit=$rec_lines[$counter]-$sen_line;
if ($last_transit!=0)
{
$delta_transit=$transit-$last_transit;
if ( $delta_transit < 0 ) {
                $delta_transit = -$delta_transit;
              }


$jitter+=($delta_transit-$jitter)/16.0;


}
$last_transit=$transit;
   $result=$jitter*1000;
$timerec=$rec_lines[$counter];
chomp($timerec);
print OUTFILE "$timerec $result\n";
$pack_delay=$delta_transit*1000;
print OUTFILE1 "$timerec $pack_delay\n";
   #print "$sen_line $rec_lines[$counter] $result\n";
   $counter++;

}


close(OUTFILE);
close(OUTFILE1);

unless (open (INFILE, "<final_jitter"))
          {

                    die ("cannot open input file outfile\n");

          }
$min=100000;
$max=0;
$counter=0;
$result=0;
$lock=0;
while (<INFILE>)
{

my($sen_line) = $_;
  chomp($sen_line);
@values=split(/[\t +]/,$sen_line);
$value=$values[1];
$result+=$value;
 if ($counter==1)
   {
$min=$value;
```

```
}

if ($value>$max)
{
  $max=$value;

}
if (($value<$min))
{
  $min=$value;

}
$counter++;


}


close (INFILE);
$result=$result/$counter;

print "aver jitter is $result max is $max min $min\n";

unless (open (INFILE, "<final_pack2packdelay"))
        {

                die ("cannot open input file outfile\n");

        }
$min=100000;
$max=0;
$counter=0;
$result=0;
$lock=0;
while (<INFILE>)
{

my($sen_line) = $_;
  chomp($sen_line);

@values=split(/[\t +]/,$sen_line);
$value=$values[1];
$result+=$value;
 if ($counter==1)
   {
$min=$value;

}

if ($value>$max)
```

```
{
 $max=$value;

}
if (($value<$min))
{
 $min=$value;

}
$counter++;


}


close (INFILE);
$result=$result/($counter);

print "aver pack2packdelay is $result max is $max min $min\n";



unless (open (INFILE, "<final_pack2packdelay"))
        {

                die ("cannot open input file outfile\n");

        }

unless (open (OUTFILE, ">timed_final_pack2packdelay"))
        {

                die ("cannot open input file outfile\n");

        }

my (@times) = <INFILE>;
close(INFILE);
$time=0;
$lock=0;
for ($i=0;$i<@times-1;$i++)
{

 if ($lock==0)
{
  @valuesplits=split(/[\t +]/,$times[$i]);
  $valuesplit=$valuesplits[1];
chomp($valuesplit);
  print OUTFILE "$time $valuesplit\n";
$lock=1;
}
```

```
  #chomp($sen_line);
 @timesplit=split(/[\t +]/,$times[$i+1]);
 $temp_time1=$timesplit[0];
@timesplit=split(/[\t +]/,$times[$i]);
$temp_time2=$timesplit[0];
$time=($temp_time1-$temp_time2)+$time;
 #$time=$timesplit[0];
 @valuesplits=split(/[\t +]/,$times[$i+1]);
  $valuesplit=$valuesplits[1];
  chomp($valuesplit);
  print OUTFILE "$time $valuesplit\n";
 #print "$time\n";

}

close(OUTFILE);


unless (open (INFILE, "<final_jitter"))
        {

                die ("cannot open input file outfile\n");

        }

unless (open (OUTFILE, ">timed_final_jitter"))
        {

                die ("cannot open input file outfile\n");

        }

my (@times) = <INFILE>;
close(INFILE);
$time=0;
$lock=0;
for ($i=0;$i<@times-1;$i++)
{

 if ($lock==0)
{
  @valuesplits=split(/[\t +]/,$times[$i]);
  $valuesplit=$valuesplits[1];
chomp($valuesplit);
  print OUTFILE "$time $valuesplit\n";
$lock=1;
}
  #chomp($sen_line);
 @timesplit=split(/[\t +]/,$times[$i+1]);
 $temp_time1=$timesplit[0];
@timesplit=split(/[\t +]/,$times[$i]);
```

```
$temp_time2=$timesplit[0];
$time=($temp_time1-$temp_time2)+$time;
 #$time=$timesplit[0];
 @valuesplits=split(/[\t +]/,$times[$i+1]);
  $valuesplit=$valuesplits[1];
chomp($valuesplit);
  print OUTFILE "$time $valuesplit\n";
 #print "$time\n";

}

close(OUTFILE);

$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

8.4.1.8. Script timestamp one way delay (ipv_all_one_way_delay.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w

unless (open (SENDER, "<sender_timestamp"))
        {

                die ("cannot open input file outfile\n");

        }
unless (open (RECEIVER, "<receiver_timestamp"))
        {

                die ("cannot open input file outfile\n");

        }
```

```
my (@sen_lines) = <SENDER>;
my (@rec_lines) = <RECEIVER>;
my($delay);
my($counter);
my($avg_delay);
my($sample_time);
my($min);
my($max);
$counter=0;
$delay=0;
$avg_delay=0;
$sample_time=0;
$max=-1;
$min=1000000;
close(SENDER);
close(RECEIVER);

unless (open (OUTFILE, ">one_way_delayvstime"))
                        {

                                die ("cannot open output file jittervstime\n");

                        }


$start = time;


# One way delay formula is Di=abs(R_i-S_i)
#Avg One way Delay is Sum(Di)/n

for ($i=0;$i<@sen_lines;$i++)
{

  #print("Sender line $sen_lines[$i]\n");
 #print("Receiver line $rec_lines[$i]\n");
  $delay=abs(($rec_lines[$i])-($sen_lines[$i]))*1000;
 $avg_delay+=$delay;
 if ($min>$delay)
{
        $min=$delay;
}
if ($max<$delay)
{
        $max=$delay;
}
$sample_time=$rec_lines[$i]-$rec_lines[0];
#print ("$sample_time $jitter\n");
print OUTFILE "$sample_time $delay\n";
$counter++;
```

```
}
$avg_delay=($avg_delay/$counter);
print ("Average One way Delay is $avg_delay ms. Max One way Delay is $max ms. Min
One way Delay is $min ms\n");

close(OUTFILE);

$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

8.4.1.9. Script Jitter (ipv_all_jitter.pl)

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

#!/usr/bin/perl -w

unless (open (SENDER, "<sender_timestamp"))
        {

                die ("cannot open input file outfile\n");

        }
unless (open (RECEIVER, "<receiver_timestamp"))
        {

                die ("cannot open input file outfile\n");

        }

my (@sen_lines) = <SENDER>;
my (@rec_lines) = <RECEIVER>;
my($jitter);
my($counter);
```

```
my($avg_jitter);
my($sample_time);
my($min);
my($max);
$counter=0;
$jitter=0;
$avg_jitter=0;
$sample_time=0;
$max=-1;
$min=1000000;
close(SENDER);
close(RECEIVER);

unless (open (OUTFILE, ">jittervstime"))
                        {

                                die ("cannot open output file jittervstime\n");

                        }


$start = time;


# Jitter formula is Di=abs(R_(i)-R_(i-1))-(S_(i)-S_(i-1))
#Avg jitter is Sum(Di)/n

for ($i=0;$i<@sen_lines-1;$i++)
{

  #print("Sender line $sen_lines[$i]\n");
 #print("Receiver line $rec_lines[$i]\n");
  $jitter=abs(($rec_lines[$i+1]-$rec_lines[$i])-($sen_lines[$i+1]-$sen_lines[$i]))*1000;
 $avg_jitter+=$jitter;
 if ($min>$jitter)
{
        $min=$jitter;
}
if ($max<$jitter)
{
        $max=$jitter;
}
$sample_time=$rec_lines[$i+1]-$rec_lines[0];
#print ("$sample_time $jitter\n");
print OUTFILE "$sample_time $jitter\n";
$counter++;
}
$avg_jitter=($avg_jitter/$counter);
print ("Average Jitter is $avg_jitter ms. Max jitter is $max ms. Min jitter is $min ms\n");

close(OUTFILE);
```

```
$elapsed_sec = time - $start;

my $second = $elapsed_sec%60;
my $minute = ($elapsed_sec/60)%60;
my $hour = ($elapsed_sec/(60*60))%24;
print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

8.5. Matlab scripts

8.5.1. inter_arrival_jitter

```
%% Create tables and fill with data (x,y)
x = timed_final_jitter(:,1);
y = timed_final_jitter(:,2);

%% Create figure
figure1 = figure('Color',[1 1 1]);

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180]);
xlabel(axes1,'Time (sec)');
ylabel(axes1,'Inter-arrinal Jitter (ms)');
hold(axes1,'all');
box;

%% Create plot
plot(x,y,'Marker','none','MarkerSize',0.5,'Color','black');
```

8.5.2. losses

```
%% Create tables and fill with data (x,y)
x = lossvstime(:,1);
y = lossvstime(:,2);

%% Create figure
figure1 = figure('Color',[1 1 1]);

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180]);
xlabel(axes1,'Time (sec)');
ylabel(axes1,'Lost Packets per Second');
```

```
hold(axes1,'all');
box;

%% Create plot
plot(x,y,'LineStyle','none','Marker','.','MarkerSize',12.0,'Color','black');
```

8.5.3. one_way_delayvstime

```
%% Create tables and fill with data (x,y)
x = one_way_delayvstime(:,1);
y = one_way_delayvstime(:,2);

%% Create figure
figure1 = figure('Color',[1 1 1]);

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180]);
xlabel(axes1,'Time (sec)');
ylabel(axes1,'One Way Delay (ms)');
hold(axes1,'all');
box;

%% Create plot
plot(x,y,'LineStyle','none','Marker','.','MarkerSize',3.5,'Color','black');
```

8.5.4. Packet-to-Packet Delay Variation

```
%% Create tables and fill with data (x,y)
x = jittervstime(:,1);
y = jittervstime(:,2);

%% Create figure
figure1 = figure('Color',[1 1 1] );

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180]);
xlabel(axes1,'Time (sec)');
ylabel(axes1,'Packet-to-Packet Delay Variation (ms)');
hold(axes1,'all');
box;

%% Create plot
```

```
plot(x,y,'LineStyle','none','Marker','.','MarkerSize',3.5,'Color','black');
```

8.5.5. Retransmissions-Triple duplicete Packets

```
%% Create tables and fill with data (x,y)
x = traffic_loss_dataset_blue(:,1);
y(:,1) = traffic_loss_dataset_blue(:,2);
y(:,2) = traffic_loss_dataset_yellow(:,2);

%% Create figure
figure1 = figure('Color',[1 1 1]);

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180])
xlabel(axes1,'Time (sec)');
ylabel(axes1,'Lost Events per Second');
hold(axes1,'all');
box;

%% Create plot
plot(x,y(:,1),'LineStyle','none',...
  'Marker','.',...
  'MarkerSize',12,...
  'Parent',axes1,...
  'DisplayName','Retransmitted Packets','Color','black');

%% Create plot
plot(x,y(:,2), 'LineStyle','none',...
  'Marker','x',...
  'MarkerSize',10,...
  'Parent',axes1,...
  'DisplayName','Triple Duplicate Acknowledgments','Color','black');


%% Create legend
legend1 = legend(...
  axes1,{'Retransmitted Packets','Triple Duplicate Acknowledgments'},...
  'FontName','Garamond',...
  'FontWeight','bold');
```

8.5.6. RTT

```
%% Create tables and fill with data (x,y)
x = traffic_rtt_dataset_blue(:,1);
y = traffic_rtt_dataset_blue(:,2);
```

```
%% Create figure
figure1 = figure('Color',[1 1 1]);

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180]);
xlabel(axes1,'Time (sec)');
ylabel(axes1,'RTT (ms)');
hold(axes1,'all');
box;

%% Create plot
plot(x,y,'Marker','.','MarkerSize',10,'Color','black');
```

8.5.7. Throughput

```
%% Create tables and fill with data (x,y)
x = traffic_bytes_dataset_green(:,1);
%y = data(:,2);

%% Convert Byte/sec to Mb/sec
y = (traffic_bytes_dataset_green(:,2)*8/(1024*1024));

%% Create figure
figure1 = figure('Color',[1 1 1]);

%% Create axes
axes1 =
axes('FontName','Garamond','XGrid','on','XMinorTick','on','YGrid','on','YMinorTick','on
');
xlim(axes1,[0 180]);
xlabel(axes1,'Time (sec)');
ylabel(axes1,'Throughput (Mb/s)');
hold(axes1,'all');
box;

%% Create plot
plot(x,y,'Marker','.','MarkerSize',10,'Color','black');
```