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 κάνοντας χρήση του UDP πρωτοκόλλου για την βελτιστοποίηση των επιδόσεων αξιοποιώντας μηχανισμούς παροχής ποιότητας υπηρεσίας (DiffServ)"

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: **ΠΑΝΤΑΚΗΣ ΠΟΛΥΧΡΟΝΗΣ**
ΗΜΕΡΟΜΗΝΙΑ: **10/10/2008**

ΕΙΣΗΓΗΤΗΣ: **Δρ. ΠΑΛΛΗΣ ΕΥΑΓΓΕΛΟΣ**

Στην οικογένεια μου και
σε όσους στάθηκαν δίπλα μου,
με ιδιαίτερη εκτίμηση και αγάπη

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής αυτής εργασίας που υλοποιήθηκε στο εργαστήριο τηλεπικοινωνιών και δικτύων του Α.Τ.Ε.Ι. Κρήτης (ΠΑΣΙΦΑΗ) θα ήθελα να ευχαριστήσω όλους όσους βοήθησαν στην ολοκλήρωση αυτής της εργασίας .

Καταρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή του Α.Τ.Ε.Ι Κρήτης Δρ. Πάλλη Ευάγγελο για την βοήθεια και την υποστήριξη που μου πρόσφερε καθόλη την διάρκεια υλοποίησης αυτής της πτυχιακής εργασίας.

Επίσης θα ήθελα να ευχαριστήσω ιδιαιτέρως τους Δρ. Μαστοράκη Γεώργιο, τον υποψήφιο διδάκτορα κ. Μαρκάκη Ευάγγελο καθώς και τον ερευνητή καθηγητή του εργαστηρίου ΠΑΣΙΦΑΗ κ. Σιδέρη Ανάργυρο για την πολύτιμη βοήθεια τους τόσο τεχνική όσο και ψυχολογική που μου πρόσφεραν απλόχερα κατά την διάρκεια περάτωσης αυτής της πτυχιακής εργασίας.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την πολύτιμη ψυχολογική υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια των σπουδών μου καθώς και τους συναδέλφους συμφοιτητές μου κατά τη διάρκεια υλοποίησης αυτής της εργασίας.

Ηράκλειο, Οκτώβριος 2008

Παντάκης Πολυχρόνης

Περίληψη

Η Πτυχιακή αυτή εργασία επικεντρώνεται στη μελέτη, σχεδίαση, υλοποίηση και αξιολόγηση δικτυακής υποδομής βασισμένης στο πρότυπο DVB-T κάνοντας χρήση του UDP πρωτόκολλου. Πιο συγκεκριμένα, με το δικτυακό αυτό περιβάλλον σύγκλισης των τεχνολογιών τηλεπικοινωνιών και των δικτύων ευρείας κάλυψης (broadcasting) πραγματοποιείται μία ενοποιημένη υποδομή η οποία επιτρέπει στους παραδοσιακά παθητικούς αστικούς τελικούς χρήστες να γίνουν ενεργοί, συμμετέχοντας στην Κοινωνία της Πληροφορίας δημιουργώντας, επεξεργάζοντας και παρέχοντας το δικό τους περιεχόμενο/υπηρεσίες με τη χρήση μίας κοινά αξιοποιήσιμης υποδομής. Επιπλέον το δικτυακό αυτό περιβάλλον επιτρέπει την παροχή υπηρεσιών triple/play δηλαδή υπηρεσίες φωνής, βίντεο και δεδομένων σε απομακρυσμένες περιοχές στις οποίες δεν υπάρχει σύνδεση του τοπικού βρόχου με τα κεντρικά δίκτυα κορμού (core backbone).

Ωστόσο, σε ένα τόσο περίπλοκο περιβάλλον, όσο αυτό της διαδραστικής ψηφιακής τηλεόρασης, η χρονικά μεταβαλλόμενη κίνηση που δημιουργείται από κάθε χρήστη επηρεάζει την λειτουργία του δικτύου. Επίσης, ένας από τους σημαντικότερους παράγοντες που πρέπει να ληφθεί υπόψη είναι η "ευαισθησία" των παρεχόμενων υπηρεσιών, τόσο σε ποσοτικά όσο και ποιοτικά χαρακτηριστικά, όπως είναι η καθυστέρηση και η διακύμανση της. Το πρωτόκολλο IP, που χρησιμοποιείται σήμερα διαβιβάζει τα πακέτα βασιζόμενο στο "παραδοσιακό" μοντέλο Βέλτιστης Προσπάθειας (Best Effort), που χρησιμοποιείται και στο Διαδίκτυο. Σύμφωνα με αυτό, η εξυπηρέτηση της κίνησης γίνεται το συντομότερο δυνατό, χωρίς όμως να παρέχεται καμιά εγγύηση τόσο για το χρόνο παράδοσής της όσο και για την ίδια την παράδοση.

Ορισμένες, όμως, εφαρμογές απαιτούν να λάβουν εξυπηρέτηση στην διαβίβαση των πακέτων τους που να ικανοποιεί κάποιες προϋποθέσεις, όπως χαμηλή καθυστέρηση. Συνεπώς, κρίνεται αναγκαία η ενσωμάτωση στο δίκτυο κατάλληλων μηχανισμών παροχής Ποιότητας Υπηρεσιών (Quality of Service - QoS) με σκοπό την διασφάλισή της σε ικανοποιητικά επίπεδα. Μία από τις πιο διαδεδομένες τεχνικές παροχής QoS, η οποία προσφέρει κατηγοριοποίηση της δικτυακής κίνησης αλλά και διαφοροποίηση στον τρόπο εξυπηρέτησης αυτής είναι ο μηχανισμός των Διαφοροποιημένων Υπηρεσιών (Differentiated Services – DiffServ).

Τέλος η απόδοση του δικτύου αυτού 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).

Στόχοι Πτυχιακής Εργασίας

Στόχος της πτυχιακής αυτής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός δικτύου επίγειας ψηφιακής τηλεόρασης για την παροχή αμφίδρομων διαδραστικών υπηρεσιών κάνοντας χρήση του πρωτοκόλλου UDP . Εν συνεχεία, ακολουθεί η μελέτη του μηχανισμού λειτουργίας των Διαφοροποιημένων Υπηρεσιών (DiffServ) αλλά και ο τρόπος με τον οποίο μπορεί αυτή η τεχνολογία να ενσωματωθεί στην αρχιτεκτονική ενός διαδραστικού DVB-T δικτύου για την βελτιστοποίηση της απόδοσης και της ποιότητας των παρεχόμενων υπηρεσιών. Τέλος, μέσω μιας σειράς πειραμάτων που θα γίνουν αρχικά με εικονικά πακέτα δεδομένων και ύστερα με πραγματικά βίντεο, θα πραγματοποιηθεί ανάλυση των μετρήσεων που θα προκύψουν και θα εκτιμηθούν τα αποτελέσματα χωρίς αλλά και με την παρουσία της τεχνολογίας DiffServ στα διάφορα χαρακτηριστικά του δικτύου.

Οργάνωση Πτυχιακής Εργασίας

Θα ξεκινήσουμε αρχικά κάνοντας μια αναφορά στις διάφορες τεχνολογίες που θα χρησιμοποιήσουμε κατά τη περάτωση αυτής της εργασίας έτσι ώστε ο αναγνώστης να τις κατανοήσει και να μπορέσει να εξοικειωθεί μαζί τους. Στη συνέχεια θα περιγράψουμε την αρχιτεκτονική και την τοπολογία του πειραματικού δικτύου και τέλος θα ακολουθήσει η ανάλυση, η μελέτη και η παρουσίαση των διαφόρων πειραματικών μετρήσεων που θα υλοποιηθούν.

Περιεχόμενα

Λίστα Σχημάτων

1. ΘΕΩΡΙΤΙΚΟ ΥΠΟΒΑΘΡΟ

1.1. Εισαγωγή

Σε μια εποχή που έχει επικρατήσει για τις περισσότερες μορφές πληροφορίας η ψηφιοποιημένη αποθήκευση και μεταφορά η τηλεοπτική τεχνολογία έχει δυστυχώς διατηρήσει την αναλογική μέθοδο εκπομπής. Μόνο την τελευταία δεκαετία και χάρη στα πρότυπα συμπίεσης και πολυπλεξίας εικόνας της οικογένειας MPEG και το ευρωπαϊκό σύνολο προδιαγραφών DVB για διαμόρφωση και μετάδοση έχει αποκτήσει η ψηφιακή τηλεόραση μια συγκεκριμένη προοπτική.

Είναι αρκετά τα πλεονεκτήματα τα οποία εισάγει η ψηφιακή τεχνολογία στην διαμόρφωση και διανομή του τηλεοπτικού σήματος. Μερικά από αυτά είναι:

• Σταθερή ποιότητα εικόνας, με μεγαλύτερη ανοχή στις ατέλειες του ασύρματου ή ενσύρματου διαύλου. Εξάλειψη φαινομένων όπως αυτά της θόλωσης, των πολλαπλών ειδώλων ή του θορύβου («χιονιού»).

• Μειωμένος λόγος σήματος προς θόρυβο που απαιτείται σε σύγκριση με την αναλογική μετάδοση. Αυτό επιτρέπει τη μείωση της εκπεμπόμενης ισχύος μέχρι και κατά 30 dB χωρίς να διαταράσσεται η ποιότητα της εικόνας.

• Καλύτερη εκμετάλλευση του φάσματος λόγω της συμπίεσης του σήματος βασικής ζώνης. Για παράδειγμα, ένα επίγειο κανάλι UHF αντιστοιχεί σε ένα και μόνο αναλογικό τηλεοπτικό πρόγραμμα, ενώ το ίδιο εύρος μπορεί να φιλοξενήσει μέχρι και 4 ψηφιακά προγράμματα υψηλής ποιότητας.

• Ευέλικτες τεχνικές πολυπλεξίας των ψηφιακών συστημάτων, που επιτρέπουν την συνύπαρξη πολλών προγραμμάτων και υπηρεσιών επιλεγόμενης ποιότητας και ευκρίνειας.

• Μεταβλητή ταχύτητα (bit rate) εκπομπής, ανάλογα με τις απαιτήσεις ποιότητας του προγράμματος, κάτι που μεταξύ άλλων αυξάνει το κέρδος πολυπλεξίας (multiplexing gain) στην περίπτωση της ταυτόχρονης μετάδοσης πολλών προγραμμάτων μεταβλητού ρυθμού.

• Δυνατότητα επεξεργασίας της εικόνας στο δέκτη μετά τη λήψη μέσω αλγορίθμων ψηφιακής επεξεργασίας (digital image post-processing), όπως χρωματική διόρθωση, αποκοπή ορίων, αυξομείωση του μεγέθους ή αφαίρεση θορύβου.

• Εύκολος εμπλουτισμός των τηλεοπτικών προγραμμάτων μέσω τυποποιημένων αρχιτεκτονικών (π.χ. MHP, OpenTV) με τοπικές εφαρμογές που εκμεταλλεύονται τις δυνατότητες των σύγχρονων «έξυπνων δεκτών».

• Ενσωμάτωση διαφόρων πολυμεσικών εφαρμογών και υπηρεσιών δεδομένων, όπως αμφίδρομων υπηρεσιών και διαδικτυακής πρόσβασης σε μια κοινή ψηφιακή πλατφόρμα, με προϋπόθεση ότι υπάρχει διαθέσιμο κανάλι επιστροφής (reverse path).

1.2. DVB

Το Digital Video Broadcasting (DVB) είναι ένα σύνολο από διεθνή αναγνωρισμένα πρότυπα για ψηφιακή τηλεόραση. Τα πρότυπα αυτά αναπτύχθηκαν από το DVB Project και εγκρίθηκε από το Ευρωπαϊκό Ίδρυμα Προτύπων Τηλεπικοινωνιών (ETSI). Τα πρότυπα του DVB διακρίνονται στα δορυφορικά (DVB-S, DVB-S2, DVB-SH), στο καλωδιακό DVB-C, στο επίγειο (DVB-T, DVB-T2), στο επίγειο αλλά για κινητούς χρήστες (DVB-H, DVB-SH) καθώς και μέσω μικροκυμάτων (DVB-MT, DVB-MC, DVB-MS).

2. DVB-T

Γενικά

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.

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 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.

2.1. Το Πρότυπο MPEG

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. Η οικογένεια MPEG περιλαμβάνει διάφορα πρότυπα τα οποία είναι:

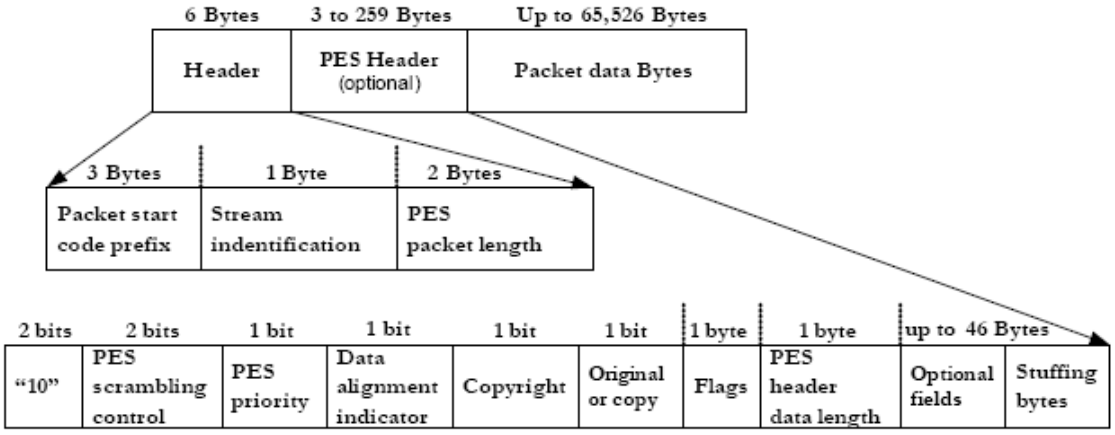
- MPEG - 1: Το αρχικό πρότυπο για συμπίεση ήχου και βίντεο. Στη πορεία χρησιμοποιήθηκε ως πρότυπο για το Video CD, και περιλαμβάνει επίσης και το δημοφιλές Layer 3 (MP3) για συμπίεση ήχου.
- MPEG – 2: Έχει καθιερωθεί σε παγκόσμιο επίπεδο ως το πρότυπο για συμπίεση της ψηφιακής τηλεόρασης αφού παρέχει υψηλό βαθμό συμπίεσης διατηρώντας την εικόνα σε υψηλό επίπεδο ποιότητας. Εκτός από την ψηφιακή τηλεόραση το MPEG-2 χρησιμοποιείτε επίσης και σε ψηφιακές υπηρεσίες δορυφορικής τηλεόρασης, σε ψηφιακά σήματα καλωδιακής τηλεόρασης, στα SVCD καθώς επίσης και στα DVD.
- MPEG – 3: Αρχικά σχεδιάστηκε για HDTV αλλά εγκαταλείφθηκε όταν ανακαλύφθηκε ότι το MPEG – 2 σε συνδυασμό με κάποια extensions επαρκούσε για HDTV.
- MPEG – 4: Επεκτείνει το MPEG – 1 έτσι ώστε να περιλαμβάνει "αντικείμενα" βίντεο/ήχου, 3D περιεχόμενο, χαμηλή ροή μεταφοράς κρυπτογράφησης καθώς επίσης και διαχείριση ψηφιακών δικαιωμάτων (Digital Rights Management)
- MPEG – 7: Πρότυπο περιγραφής πολυμεσικού περιεχομένου
- MPRG – 21: Το πρότυπο αυτό καθορίζει ένα ανοιχτό πλαίσιο εργασίας για πολυμεσικές εφαρμογές

2.2. Το Πρότυπο MPEG – 2

Το MPEG είναι ένα σύστημα κωδικοποίησης και συμπίεσης για ψηφιακό πολυμεσικό περιεχόμενο. Το MPEG – 2 επεκτείνει το βασικό MPEG σύστημα έτσι ώστε να προσφέρει συμπίεση και για τη ψηφιακή τηλεόραση. Επειδή ακριβώς το πρότυπο MPEG – 2 προσφέρει καλή συμπίεση χρησιμοποιώντας τυπικούς αλγορίθμους έχει αναδεχθεί σε ένα κυρίαρχο πρότυπο για τη ψηφιακή τηλεόραση. Τα χαρακτηριστικά του MPEG – 2 είναι:

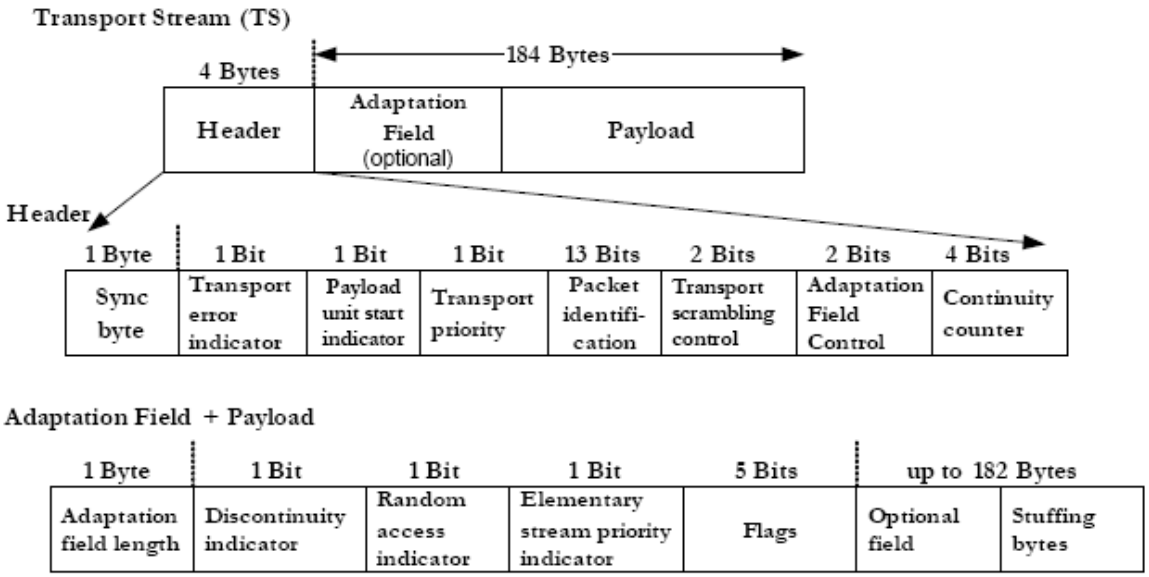
- Συμπίεση βίντεο η οποία είναι συμβατή με το MPEG – 1
- Βελτιωμένη κωδικοποίηση ήχου
- Transport Multiplexing
- Άλλες υπηρεσίες (GUI, κρυπτογράφηση, μετάδοση δεδομένων)

Έτσι τα συμπιεσμένα οπτικοακουστικά σήματα μαζί με άλλου είδους ψηφιακή πληροφορία πολυπλέκονται μαζί με βάση τις προδιαγραφές οι οποίες καθορίζονται στο πρότυπο MPEG-2 Systems για τη δημιουργία μίας κοινής Ροής Μεταφοράς MPEG-2 (MPEG-2 Transport Stream) που αποτελεί και το σήμα βασικής ζώνης για όλα τα συστήματα μετάδοσης DVB. Οι ροές αυτές μεταφοράς είναι σχεδιασμένες ώστε να μεταφέρουν δεδομένα σε πραγματικό χρόνο. Κάθε ένα από τα αναλογικά σήματα ήχου και βίντεο κωδικοποιούνται με βάση το πρότυπο MPEG – 2 και δημιουργούν ξεχωριστές Βασικές Ροές (Elementary Streams - ES) δεδομένων μία για κάθε ένα από αυτά. Στο επόμενο στάδιο οι βασικές αυτές ροές διαμορφώνονται σε μορφή πακέτων δημιουργώντας τις Βασικές Ροές Πακέτων (Packetized Elementary Streams – PES).



Σχήμα 1. Δομή Πακέτου MPEG – 2 PES

Στη συνέχεια για τη δημιουργία του σήματος βασικής ζώνης πρέπει να πολυπλεχθούν οι βασικές ροές πακέτων έτσι ώστε να μπορέσει να σχηματιστή η τελική ροή μεταφοράς MPEG -2. Η ροή μεταφοράς MPEG-2 αποτελεί μία μορφή πολυπλεξίας σχεδιασμένη για σύνθετες εφαρμογές όπως τα ψηφιακά τηλεοπτικά μπουκέτα τα οποία απαρτίζονται από πολλαπλά τηλεοπτικά προγράμματα και υπηρεσίες δεδομένων.



Σχήμα 2. Δομή Πακέτου MPEG – 2 TS

3. Γενικές Αρχές Επίγειας Ψηφιακής Μετάδοσης (DVB-T)

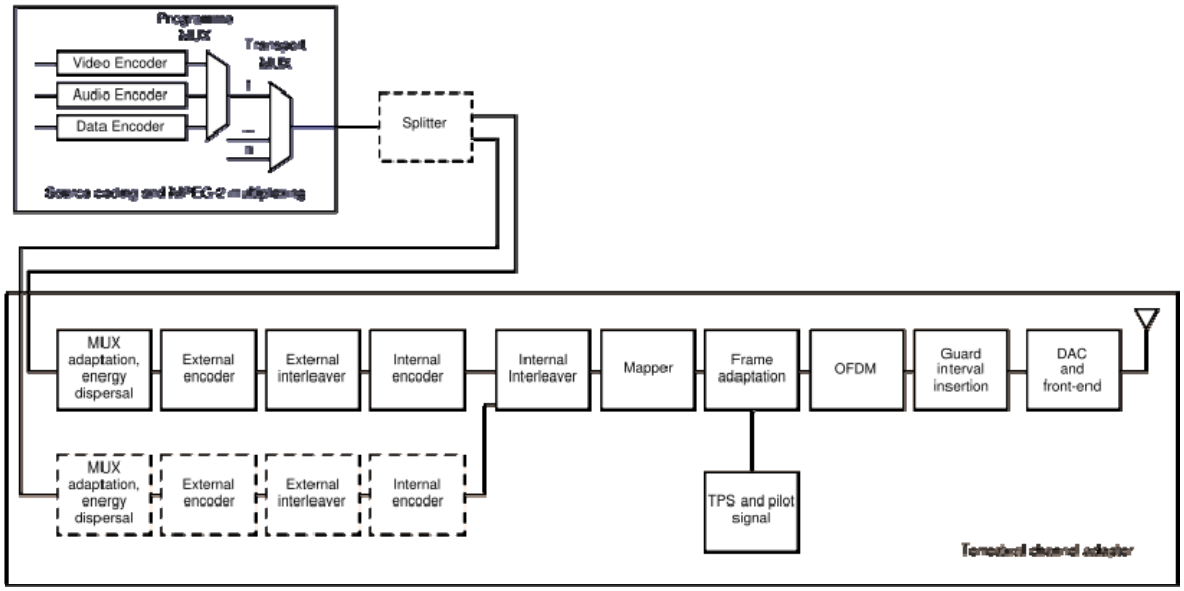
Το πρότυπο DVB-T καθορίζει τις προδιαγραφές ενός συστήματος επίγειας εκπομπής ευρείας κάλυψης και περιορίζεται αποκλειστικά στην περιγραφή των λειτουργιών του διαμορφωτή παραλείποντας τον καθορισμό των τεχνολογιών για την υλοποίηση των καναλιών επιστροφής και την παροχή αμφίδρομων πολυμεσικών και Διαδικτυακών υπηρεσιών, καθώς και τις προδιαγραφές για τη δημιουργία του σήματος βασικής ζώνης. Ο διαμορφωτής δέχεται από τον πολυπλέκτη/ενθυλακωτή την τελική ροή μεταφοράς MPEG-2 στην οποία περιέχονται πολυπλεγμένες οι υπηρεσίες εικόνας, ήχου και δεδομένων υπό τη μορφή σήματος βασικής ζώνης και στην έξοδό του παράγει το προς μετάδοση σήμα RF. Το εύρος ζώνης του σήματος αυτού είναι 8MHz και μεταδίδεται σε ένα από τα διαθέσιμα κανάλια UHF (21ch. – 69ch.).

Ο διαμορφωτής DVB-T χρησιμοποιεί σχήμα COFDM στην επίγεια μετάδοση για την αντιμετώπιση του φαινομένου των επιλεκτικών ως προς τη συχνότητα διαλείψεων. Η τεχνική COFDM περιλαμβάνει μία μέθοδο για τη διαμόρφωση πολλαπλών φερόντων και αποτελεί ένα αντικείμενο μελέτης ευρέως ενδιαφέροντος για πολλά χρόνια. Έγινε πρώτα ελκυστική για τη μετάδοση σημάτων με τη χρήση διακριτών Μετασχηματισμών Fourier και πιο πρόσφατα χρησιμοποιώντας μετασχηματισμούς FFT. Είναι αρκετά ανθεκτική στη λήψη πολυδιαδρομικών σημάτων και είναι χρήσιμη για κανάλια που παρουσιάζουν γραμμικές παραμορφώσεις. Είναι η διαμόρφωση που επιλέχθηκε από την οικογένεια προτύπων DVB για την ανάπτυξη του Ευρωπαϊκού προτύπου επίγειας ψηφιακής τηλεόρασης (DVB-T) και ως τεχνική μετάδοσης γενικά είναι η κύρια υποψήφια για τις τεχνολογίες προσωπικών επικοινωνιών 4ης γενιάς και των ασύρματων επικοινωνιών υψηλής ταχύτητας. Η τεχνική διαμόρφωσης COFDM χρησιμοποιεί χιλιάδες ξεχωριστά φέροντα σήματα για τη μεταφορά του σήματος δεδομένων, διαμοιράζοντας τα δεδομένα σε κάθε φέρων. Το σήμα δεδομένων έπειτα διαμορφώνει αυτά τα φέροντα σύμφωνα με τα πρότυπα διαμόρφωσης QPSK και QAM. Επιλέγοντας το σωστό τύπο του σήματος φέροντος είναι δυνατόν να σχηματιστεί ένα σύνολο φερόντων τα οποία είναι κοντά τοποθετημένα μαζί και δεν υπάρχει η ανάγκη για την ύπαρξη κενού εύρους συχνότητας μεταξύ τους. Αυτό είναι ένα βασικό στοιχείο της τεχνικής COFDM όπου τα φέροντα σήματα είναι κοντά τοποθετημένες με τέτοιο τρόπο σε ένα κανάλι RF δίχως την ύπαρξη κενού εύρους ενώ κάθε ξεχωριστό φέρων μπορεί ληφθεί από το σύνολο τους. Σύμφωνα με τα χαρακτηριστικά μετάδοσης και τις ρυθμίσεις λειτουργίας για τη μετάδοση DVB-T είναι δυνατόν να επιτευχθεί

ωφέλιμος ρυθμός μετάδοσης μεταξύ 4Mb/s έως και 31Mb/s. Αυτό αποτελεί μία δυνατότητα για περισσότερη ευελιξία στο σχεδιασμό δικτύων επίγειας ψηφιακής μετάδοσης ευρείας κάλυψης. Το πρότυπο DVB-T συνδυάζει την τεχνική COFDM με κωδικοποίηση και διεμπλοκή δύο επιπέδων που καθιστούν το σήμα ιδιαίτερα ανθεκτικό σε καταστάσεις πολυδιαδρομικής διάδοσης και παρεμβολών.

Το σύστημα επίγειας ψηφιακής μετάδοσης καθορίζεται στο Σχήμα 3 όπου παρουσιάζεται το λειτουργικό μπλοκ διάγραμμα με βάση το οποίο πραγματοποιείται η προσαρμογή των τηλεοπτικών σημάτων βασικής ζώνης από την έξοδο του πολυπλέκτη/ενθυλακωτή MPEG-2 TS στα χαρακτηριστικά του επίγειου καναλιού. Οι βασικές λειτουργίες της μετατροπής του σήματος βασικής ζώνης στο προς μετάδοση σήμα παρουσιάζονται παρακάτω με τη σειρά που εφαρμόζονται στη ροή μεταφοράς MPEG-2:

• Προσαρμογή ροής μεταφοράς και τυχαιοποίηση
• Εξωτερική κωδικοποίηση (προστασία έναντι λαθών με κώδικα Reed-Solomon)
• Εξωτερική συνελικτική διεμπλοκή (convolutional interleaving)
• Εσωτερική κωδικοποίηση με διατρητό συνελικτικό κώδικα (punctured convolutional code)
• Εσωτερική διεμπλοκή (inner interleaving)
• Αντιστοίχηση και διαμόρφωση φερόντων σημάτων
• Μετάδοση με τη χρήση πολυπλεξίας με διαίρεση σε ορθογώνιες συχνότητες (OFDM)
• Άνω μετατροπή στη συχνότητα RF (up conversion)



Σχήμα 3. Διάγραμμα ενός DVB Πομπού

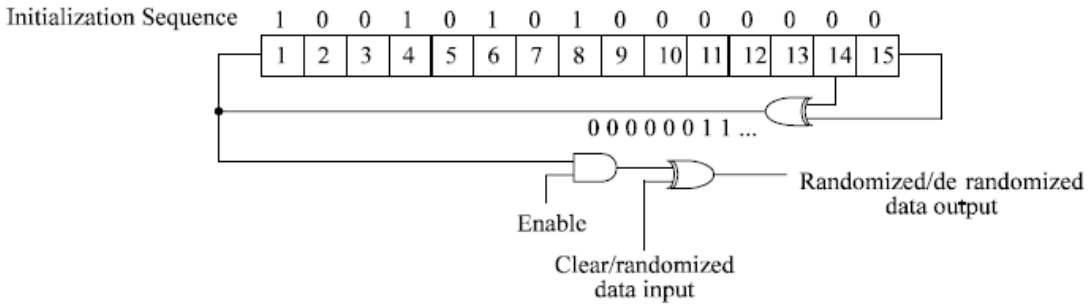
Το σύστημα αυτό είναι απόλυτα συμβατό με τα κωδικοποιημένα τηλεοπτικά σήματα MPEG-2 σύμφωνα με τα πρότυπα ISO/IEC 13818 και έχει σχεδιαστεί για τη μετάδοση επίγειων ψηφιακών τηλεοπτικών υπηρεσιών στην ήδη υπάρχουσα περιοχή φάσματος των συχνοτήτων UHF που χρησιμοποιείται για τη μετάδοση των αναλογικών τηλεοπτικών προγραμμάτων σήμερα. Επιπλέον απαιτείται από το σύστημα μετάδοσης να παρέχει επαρκή προστασία στα υψηλά επίπεδα των παρεμβολών παρακείμενου καναλιού και των ομοκαναλικών παρεμβολών που προέρχονται από τις υπάρχουσες υπηρεσίες PAL/SECAM/NTSC και επίσης απαιτείται ότι το σύστημα αυτό επιτρέπει την καλύτερη δυνατή απόδοση φάσματος όταν χρησιμοποιείται στις συχνότητες UHF. Αυτή η απαίτηση μπορεί να επιτευχθεί όταν αξιοποιηθεί η λειτουργία SFN.

Για την αξιοποίηση της κοινής χρήσης με τα πρότυπα του ETSI για τη δορυφορική και καλωδιακή μετάδοση, η εξωτερική κωδικοποίηση και διεμπλοκή είναι κοινές με αυτές του προτύπου DVB-T ενώ η εσωτερική κωδικοποίηση είναι κοινή μόνο με το πρότυπο της δορυφορικής εκπομπής. Επιπλέον για τον βέλτιστο συμβιβασμό μεταξύ της τοπολογία του δικτύου επίγειας ψηφιακής τηλεόρασης και της απόδοσης της συχνότητας, ένα διάστημα φρούρησης καθορίζεται που επιτρέπει διαφορετικές δικτυακές διατάξεις όπως δίκτυα SFN μεγάλης εμβέλειας διατηρώντας παράλληλα τη μέγιστη απόδοση συχνότητας.

Το σύστημα μετάδοσης DVB-T επιτρέπει τη χρήση διαφορετικών επιπέδων διαμόρφωσης QAM και διαφορετικούς ρυθμούς εσωτερικής κωδικοποίησης και παράλληλα επιτρέπει την αξιοποίηση δύο επιπέδων ιεραρχικής κωδικοποίησης και διαμόρφωσης καναλιού. Σε αυτήν την περίπτωση το λειτουργικό μπλοκ διάγραμμα του συστήματος επεκτείνεται περιλαμβάνοντας τα στοιχεία που φαίνονται στο Σχήμα 3 με διακεκομμένες γραμμές. Στην περίπτωση της ιεραρχικής διαμόρφωσης το σήμα βασικής ζώνης διαχωρίζεται σε δύο ροές μεταφοράς: μία υψηλής και μία χαμηλής προτεραιότητας και τα δύο αυτά σήματα διαμορφώνονται ταυτόχρονα σε ένα ιεραρχικό σχήμα QAM. Έτσι ένας δέκτης με κακές συνθήκες λήψης λαμβάνει μόνο τα δεδομένα υψηλής προτεραιότητας ενώ ένας με καλύτερες λαμβάνει το σύνολο των υπηρεσιών. Η λειτουργία αυτή δίνει τη δυνατότητα ευελιξίας στο σύστημα μετάδοσης της επίγειας ψηφιακής τηλεόρασης ειδικά όταν συνοδευτεί από κλιμακωτή κωδικοποίηση της κινούμενης εικόνας κατά MPEG-2.

3.1. Προσαρμογή Ροής Μεταφοράς και Τυχαιοποίηση για Διασπορά Ενέργειας

Η ροή μεταφοράς MPEG-2 που προέρχεται από την έξοδο του πολυπλέκτη/ενθυλακωτή και εισέρχεται στον διαμορφωτή COFDM είναι σταθερού ρυθμού (Constant Bit Rate - CBR) και αποτελείται από πακέτα σταθερού μήκους των 188 bytes (Σχ. 5 α) τα οποία περιλαμβάνουν το κάθε ένα από ένα byte συγχρονισμού. Προκειμένου να εξασφαλιστούν επαρκείς δυαδικές μεταβολές, τα δεδομένα εισόδου της ροής μεταφοράς MPEG-2 δέχονται μία διαδικασία τυχαιοποίησης η οποία παρουσιάζεται στο Σχήμα 4.



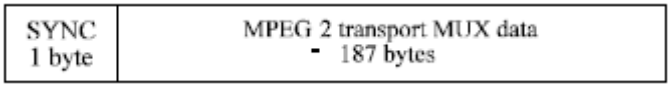
Σχήμα 4. Τυχαιοποίηση Δεδομένων Ροής Μεταφοράς MPEG-2

Με τη διαδικασία αυτή εξασφαλίζεται ότι το σήμα COFDM έχει ισότιμα κατανεμημένη την πληροφορία σε όλο το διαθέσιμο εύρος ζώνης. Αυτό επιτυγχάνεται εισάγοντας τη ροή μεταφοράς σε έναν περιπλέκτη (scrambler) ο οποίος χρησιμοποιεί μία γεννήτρια ψευδοτυχαίας δυαδικής ακολουθίας (Pseudo Random Binary Sequence – PRBS generator) έτσι ώστε όλη η πληροφορία του σήματος MPEG-2 να διαμοιραστεί τυχαία σε όλη τη ροή δεδομένων. Αυτή η διαδικασία επιπλέον εξασφαλίζει ότι θα υπάρχουν αρκετές δυαδικές μεταβολές και αυτό βοηθάει για την αντιμετώπιση λαθών κατά τη διάρκεια της μετάδοσης. Η γεννήτρια PRBS που χρησιμοποιείται έχει μία περίοδο των 1503 bytes και έχει τη δυνατότητα να διαχειρίζεται 8 πακέτα δεδομένων κάθε φορά (Σχήμα 5 b).Για την παροχή ενός σήματος έναρξης για τον δέκτη, το byte συγχρονισμού του πρώτου από τα 8 συνεχή πακέτα μέσα στη ροή μεταφοράς αναστρέφεται δυφιηδόν (bit-wise inverted) και το πρώτο bit από την έξοδο της γεννήτριας PRBS εφαρμόζεται στο πρώτο bit (MSB) του πρώτου byte που ακολουθεί το ανεστραμμένο αυτό byte συγχρονισμού. Για τη δυνατότητα αξιοποίησης πρόσθετων λειτουργιών συγχρονισμού, κατά τη διάρκεια των bytes συγχρονισμού των υπόλοιπων 7 συνεχόμενων πακέτων, η γεννήτρια PRBS συνεχίζει τη λειτουργία της αλλά η έξοδός της απενεργοποιείται

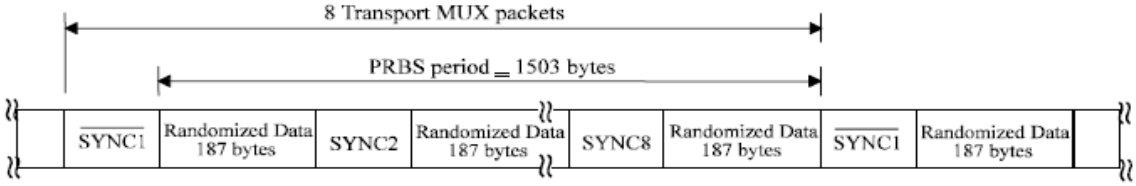
διατηρώντας αυτά τα bytes όπως είναι δίχως να έχουν υποστεί τη διαδικασία της τυχαιοποίησης. Έτσι, η περίοδος της ακολουθίας PRBS όπως αναφέρθηκε παραπάνω είναι 1503 bytes.

3.2. Εξωτερική Κωδικοποίηση και Συνελικτική Διεμπλοκή

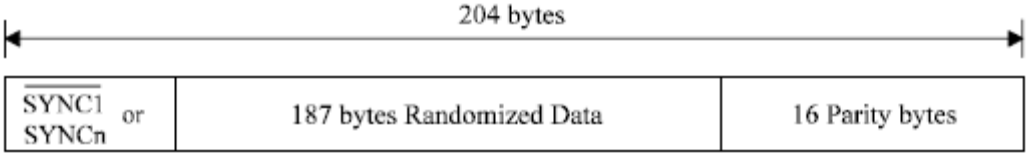
Μετά την προσαρμογή της ροής μεταφοράς και την τυχαιοποίηση ακολουθούν τα στάδια της εξωτερικής κωδικοποίησης και της συνελικτικής διεμπλοκής. Συντομευμένος κώδικας Reed-Solomon RS (204,188, t=8), ο οποίος προκύπτει από τον αρχικό συστηματικό κώδικα RS (255,239, t=8), εφαρμόζεται σε κάθε τυχαιοποιημένο πακέτο (188 bytes) έτσι ώστε να προκύψουν προστατευμένα από λάθη πακέτα των 204 bytes (Σχήμα 5 c). Ο κώδικας Reed-Solomon εφαρμόζεται επίσης στο byte συγχρονισμού των πακέτων είτε είναι ανεστραμμένο είτε όχι και επιτρέπει τη διόρθωση μέχρι και 8 λανθασμένων bytes σε τυχαίες θέσεις μέσα στο προστατευμένο πακέτο. Ο συντομευμένος κώδικας Reed-Solomon μπορεί να υλοποιηθεί προσθέτοντας 51 bytes (όλα μηδέν) πριν από τα bytes της πληροφορίας στην είσοδο του κωδικοποιητή RS (255,239, t=8). Μετά τη διαδικασία της κωδικοποίησης RS αυτά τα μηδενικά bytes απορρίπτονται οδηγώντας σε μία λέξη κώδικα RS των N=204 bytes.



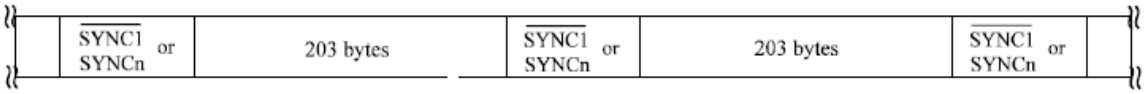
a) MPEG-2 transport MUX packet



b) Randomized transport packets: Sync bytes and Randomized Data types



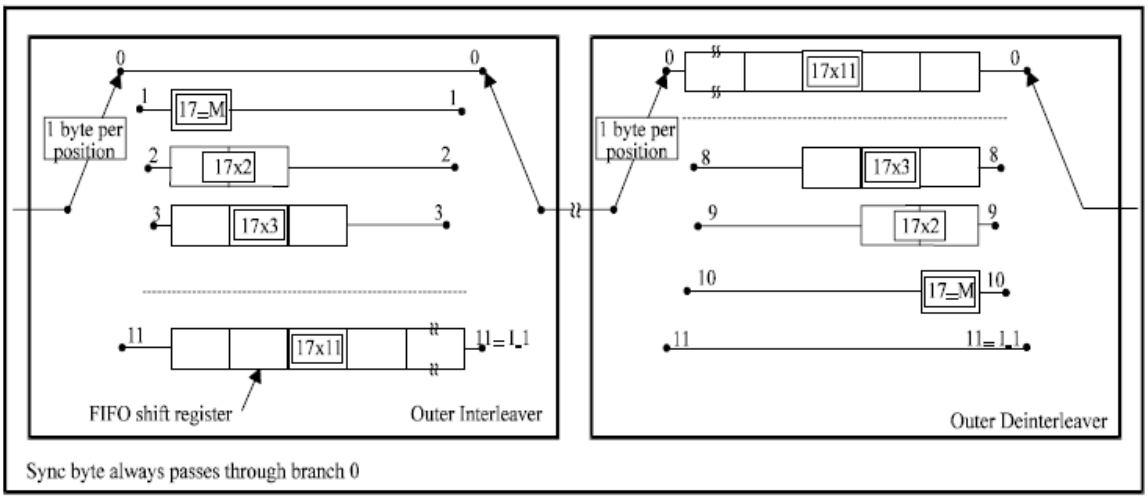
c) Reed-Solomon RS (204, 188, 8) error protected packets



d) Data structure after outer interleaving; interleaving depth I = 12 bytes

Σχήμα 5. Στάδια Προσαρμογής, Τυχαιοποίησης, Εξωτερικής Κωδικοποίησης και Διεμπλοκής

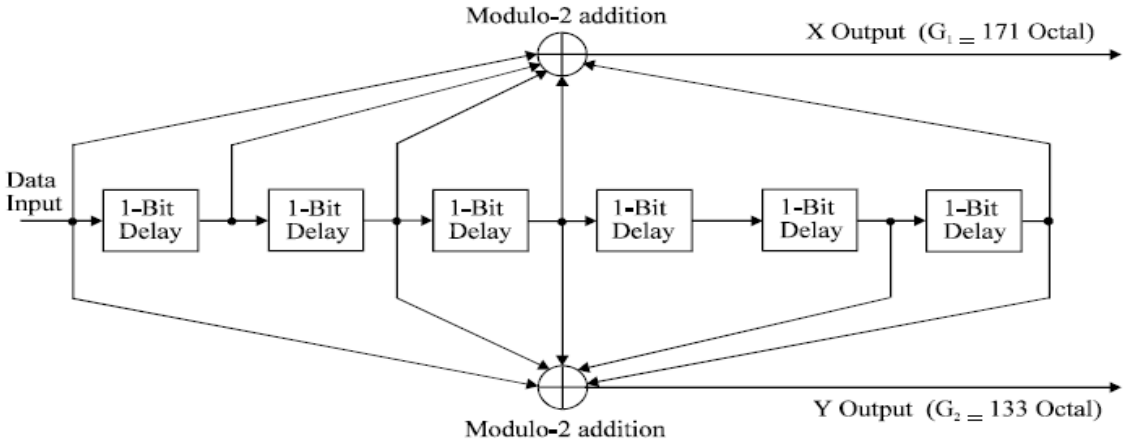
Μετά το στάδιο της εξωτερικής κωδικοποίησης, συνελικτική διεμπλοκή με βάθος I=12 (convolutional interleaving) εφαρμόζεται στα προστατευμένα από λάθη πακέτα και αυτό οδηγεί στη δομή δεδομένων του Σχήμα 5 d. Η διαδικασία της συνελικτικής διεμπλοκής βασίζεται στη μέθοδο Forney η οποία είναι συμβατή με τη μέθοδο Ramsey τύπου III. Ο διεμπλοκέας (Σχήμα 6) αποτελείται από I=12 κλάδους, κυκλικά συνδεδεμένους στην είσοδο της ροής των bytes στον διακόπτη εισόδου. Κάθε κλάδος αποτελεί έναν καταχωρητή μετατόπισης (shift register) First-In, First-Out (FIFO) του οποίου τα κελιά περιέχουν 1 byte και οι διακόπτες εισόδου και εξόδου είναι συγχρονισμένοι. Για λόγους συγχρονισμού τα bytes SYNC και δρομολογούνται πάντα στον κλάδο "0" του διεμπλοκέα.



Σχήμα 6. Διάγραμμα Εξωτερικού Διεμπλοκέα και Αποδιεμπλοκέα

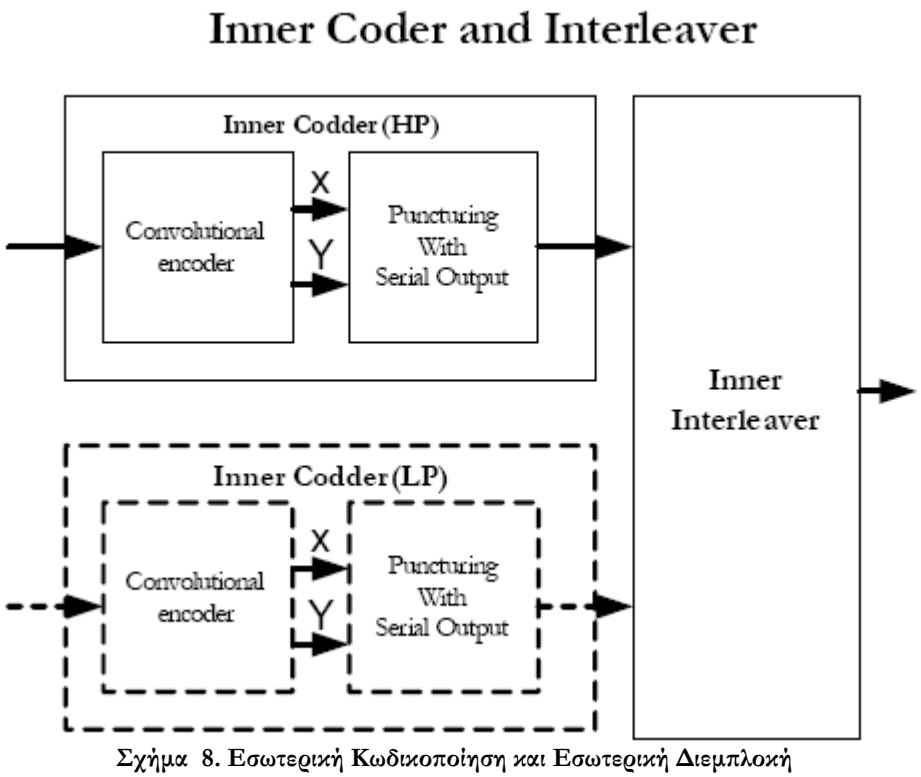
3.3. Εσωτερική Κωδικοποίηση και Διεμπλοκή

Το πρότυπο DVB-T επιτρέπει μία ευρεία επιλογή διάτρητων συνελικτικών κωδίκων (punctured convolutional codes) οι οποίοι βασίζονται σε έναν μητρικό συνελικτικό κώδικα με ρυθμό ½. Αυτό επιτρέπει την επιλογή του πιο κατάλληλου επιπέδου προστασίας λαθών για μία δεδομένη υπηρεσία ή ένα ρυθμό μετάδοσης είτε στη μέθοδο της ιεραρχικής είτε της μη-ιεραρχικής μετάδοσης. Τα πολυώνυμα-γεννήτριες του μητρικού κώδικα είναι G 1 = 171 OCT για την έξοδο X και G 2 = 133 OCT για την έξοδο Y (Σχήμα 6). Επιπρόσθετα με τον μητρικό κώδικα ρυθμού ½ το πρότυπο επιτρέπει τη χρήση διάτρητων κωδίκων με ρυθμούς 2/3, ¾, 5/6 και 7/8.



Σχήμα 7. Μητρικός Συνελικτικός Κώδικας με Ρυθμό Κώδικα 1/2

Στο επόμενο στάδιο μετά την έξοδο του εσωτερικού κωδικοποιητή τα δεδομένα υφίστανται εσωτερική διεμπλοκή (Σχήμα 8) η οποία περιλαμβάνει διεμπλοκή σε επίπεδο bits και σε επίπεδο συμβόλων. Η ακριβής αντιστοιχία των bits εισόδου στα τελικά διαμορφωμένα σύμβολα έχει άμεση σχέση με το σχήμα διαμόρφωσης το οποίο αξιοποιείται.



Σχήμα 8. Εσωτερική Κωδικοποίηση και Εσωτερική Διεμπλοκή

Μετά τη διαδικασία της εσωτερικής διεμπλοκής παράγεται μία ακολουθία δυαδικών ψηφίων (bits) η οποία είναι ήδη οργανωμένη σε σύμβολα και τα δυνατά σχήματα διαμόρφωσης τα οποία μπορούν αξιοποιηθούν με βάση τις προδιαγραφές του προτύπου DVB-T (ETSI 300 744) είναι: QPSK, 16QAM, 64QAM (2bits/symbol, 4bits/symbol και 6bits/symbol σε κάθε φέρον σήμα αντίστοιχα). Τα δεδομένα προς εκπομπή μεταδίδονται στη συνέχεια με τη χρήση της τεχνικής κωδικοποιημένης u960 πολυπλεξίας με ορθογωνική διαίρεση συχνότητας (COFDM). Με τη χρήση της τεχνικής αυτής τα δεδομένα αντιστοιχίζονται με βάση το κατάλληλο σχήμα διαμόρφωσης σε κάθε ένα από τα χιλιάδες φέροντα σήματα τα οποία χρησιμοποιούνται για τη μετάδοσή τους σε όλη την περιοχή κάλυψης DVB-T.

3.4. Ωφέλιμο Bit Rate

Ο διαθέσιμος ωφέλιμος ρυθμός μετάδοσης για την εκπομπή δεδομένων σε συστήματα DVB-Tδεν εξαρτάται από την κατάσταση λειτουργίας (2K ή 8K) του διαμορφωτή COFDM αλλά από το σχήμα διαμόρφωσης που αξιοποιείται, από τη διάρκεια του διαστήματος φρούρησης και το ρυθμό της εσωτερικής κωδικοποίησης που επιλέγεται όπως φαίνεται από τον παρακάτω πίνακα (Σχήμα 9)

Modulation	Code rate	Guard interval			
		1/4	1/8	1/16	1/32
QPSK	1/2	4,98	5,53	5,85	6,03
	2/3	6,64	7,37	7,81	8,04
	3/4	7,46	8,29	8,78	9,05
	5/6	8,29	9,22	9,76	10,05
	7/8	8,71	9,68	10,25	10,56
16-QAM	1/2	9,95	11,06	11,71	12,06
	2/3	13,27	14,75	15,61	16,09
	3/4	14,93	16,59	17,56	18,10
	5/6	16,59	18,43	19,52	20,11
	7/8	17,42	19,35	20,49	21,11
64-QAM	1/2	14,93	16,59	17,56	18,10
	2/3	19,91	22,12	23,42	24,13
	3/4	22,39	24,88	26,35	27,14
	5/6	24,88	27,65	29,27	30,16
	7/8	26,13	29,03	30,74	31,67

Σχήμα 9. Ωφέλιμος Ρυθμός Μετάδοσης (Mbps) για όλους τους Συνδυασμούς Ρυθμού Κωδικοποίησης, Σχήματος Διαμόρφωσης και Διαστήματος Φρούρησης (Κανάλι 8MHz)

3.5. WLAN

Το WLAN (Wireless Local Area Network) είναι ένα ασύρματο τοπικό δίκτυο το οποίο συνδέει δύο ή περισσότερους Η/Υ ή συσκευές χωρίς να κάνει χρήση καλωδίων και χρησιμοποιεί τις τεχνολογίες διαμόρφωσης της φασματικής εξάπλωσης (spread spectrum) ή της ορθογωνικής πολύπλεξης διαίρεσης συχνότητας (Orthogonal Frequency Division Multiplex – OFDM). Όλα αυτά δίνουν τη δυνατότητα στο χρήστη να κινείτε μέσα στη περιοχή κάλυψης του ασύρματου δικτύου και να είναι συνδεδεμένος στο δίκτυο.

Το WLAN χαρακτηρίζεται από αρκετά πλεονεκτήματα αλλά και μειονεκτήματα.

Πλεονεκτήματα:

- Κινητικότητα
- Παραγωγικότητα
- Επεκτασιμότητα
- Ευκολία στη χρήση
- Σχετικά χαμηλό κόστος

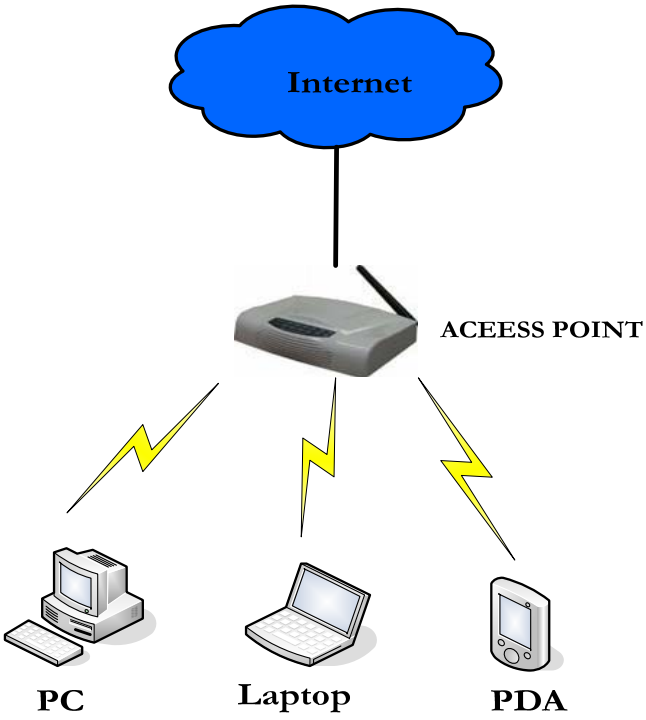
Μειονεκτήματα:

- Ασφάλεια
- Ακτίνα χρήσης
- Αξιοπιστία
- Ταχύτητα

Κατά την ανάπτυξη των WLAN κυριάρχησαν τα πρότυπα IEEE 802.11χ (Σχήμα 10)

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 10. Περιγραφή των Πρωτοκόλλων 802.11



Σχήμα 11. Αρχιτεκτονική Ασύρματου Δικτύου

3.6. Η Τεχνολογία ADSL

Το Asymmetric Digital Subscriber Line (Ασύμμετρη Ψηφιακή Συνδρομητική Γραμμή) ή ADSL είναι μια μορφή DSL, δηλαδή μια τεχνολογία μετάδοσης δεδομένων που λειτουργεί πάνω σε παραδοσιακή τηλεφωνική γραμμή αλλά πετυχαίνει υψηλότερους ρυθμούς μεταφοράς από τα παραδοσιακά modem.

Το απλό χάλκινο καλώδιο (γνωστό και ως τοπικός βρόχος, local loop ή last mile) που συνδέει σχεδόν κάθε σπίτι με το τοπικό τηλεφωνικό κέντρο, έχει πολύ περισσότερες δυνατότητες από την υποστήριξη της απλής τηλεφωνίας. Έτσι με χρήση ανώτερου τμήματος του εύρους ζώνης του βρόχου, εκείνου το οποίο μένει αναξιοποίητο από την κλασική τηλεφωνία (PSTN ή ISDN), επιτυγχάνονται υψηλές ταχύτητες μετάδοσης δεδομένων. Το γεγονός αυτό προσφέρει κι ένα ακόμη πλεονέκτημα: η παραδοσιακή τηλεφωνία και η μετάδοση δεδομένων μπορούν να λειτουργούν ταυτόχρονα και ανεξάρτητα η μία από την άλλη, εφόσον

χρησιμοποιούν διαφορετικό φάσμα συχνοτήτων στην τηλεφωνική γραμμή. Ωστόσο οι συχνότητες που χρησιμοποιεί το ADSL εξασθενούν συντομότερα από αυτές της τηλεφωνίας, με αποτέλεσμα να μπορεί να λειτουργήσει σε αποστάσεις έως 5 Χλμ. από το τηλεφωνικό κέντρο. Επιπλέον, όσο μεγαλώνει η απόσταση από το τηλεφωνικό κέντρο τόσο μειώνεται η ταχύτητα μετάδοσης δεδομένων που μπορεί να επιτευχθεί από το ADSL.

Χαρακτηριστικό του ADSL είναι το ότι οι ταχύτητες λήψης και αποστολής δεδομένων διαφέρουν - σε αυτό οφείλει και τη λέξη «ασύμμετρη» στο όνομά του. Για παράδειγμα, μια τυπική ταχύτητα για ADSL στην Ελλάδα είναι η 1024/256 Kbps, ενώ η μέγιστη ταχύτητα που μπορεί να επιτύχει είναι τα 24/1 Mbps. Ένα επιπλέον χαρακτηριστικό είναι ότι η σύνδεση ADSL είναι μόνιμη και διαθέσιμη ανά πάσα στιγμή (always-on). Δηλαδή δεν απαιτείται σύνδεση και αποσύνδεση από το δίκτυο όπως συμβαίνει με τις τηλεφωνικές κλήσεις. Εξελιγμένες εκδόσεις του ADSL είναι το ADSL2 και το ADSL2+, οι οποίες παρέχουν μεγαλύτερες ταχύτητες αξιοποιώντας διαφορετικά το εύρος ζώνης του καλωδίου. Η μέγιστη ταχύτητα που μπορεί να επιτύχει το ADSL2+ είναι τα 24/1 Mbps (ή τα 24/3,5 Mbps σε περίπτωση που υλοποιεί το πρότυπο ITU G.992.5 Annex M), αλλά στην πράξη πολύ λίγοι χρήστες μπορούν να συνδεθούν σε αυτές τις ταχύτητες, λόγω της απόστασής τους από το τηλεφωνικό κέντρο.

Standard name	Common name	Downstream rate	Upstream rate
ANSI T1.413-1998 Issue 2	ADSL	8Mbit/s	1.0Mbit/s
ITU G.992.1	ADSL (G.DMT)	12Mbit/s	1.3Mbit/s
ITU G.992.1 Annex A	ADSL OVER POTS	12Mbit/s	1.3Mbit/s
ITU G.992.1 Annex B	ADSL OVER ISDN (IDSL)	12Mbit/s	1.8Mbit/s
ITU G.992.2	ADSL Lite (G.Lite)	1.5Mbit/s	0.5Mbit/s
ITU G.992.3/4	ADSL2	12Mbit/s	1.0Mbit/s
ITU G.992.3 Annex J	ADSL2	12Mbit/s	1.0Mbit/s
ITU G.992.3 Annex L	RE-ADSL2	5Mbit/s	0.8Mbit/s
ITU G.992.5	ADSL+	24Mbit/s	1.0Mbit/s
ITU G.992.5 Annex M	ADSL2+M	24Mbit/s	3.5Mbit/s

Σχήμα 12. Τα Πρότυπα Του ADSL

4. Το Πρωτόκολλο UDP

Η στοίβα πρωτοκόλλων του Internet υποστηρίζει επίσης ένα πρωτόκολλο μεταφοράς πληροφοριών χωρίς σύνδεση, το Πρωτόκολλο Δεδομενογραφημάτων Χρήστη UDP (User Datagram Protocol). Το UDP προσφέρει έναν τρόπο για να στέλνουν οι εφαρμογές ενθυλακωμένα ακατέργαστα δεδομενογραφήματα IP χωρίς να πρέπει να εγκαταστήσουν μια σύνδεση. Πολλές εφαρμογές πελάτη-εξυπηρετητή, που έχουν μία αίτηση και μία απόκριση, προτιμούν να χρησιμοποιήσουν το UDP παρά να μπουν στον κόπο να εγκαταστήσουν και κατόπιν να απολύσουν μια σύνδεση.

Σχήμα 13. Επικεφαλίδα UDP

Ένα τεμάχιο UDP αποτελείται από μια επικεφαλίδα των 8 byte (64 bit), ακολουθούμενη από δεδομένα. Η επικεφαλίδα φαίνεται στο παραπάνω σχήμα. Οι θύρες πηγής και προορισμού χρησιμοποιούνται για την αναγνώριση των ακραίων σημείων στα μηχανήματα πηγής και προορισμού αντίστοιχα. Το πεδίο Μήκος UDP (UDP length) αφορά στην επικεφαλίδα 8 byte και στα δεδομένα. Το πεδίο Άθροισμα ελέγχου UDP (UDP checksum) το οποίο χρησιμοποιείται για επαλήθευση της ορθότητας του πακέτου στο σύνολό του, δηλαδή τόσο της κεφαλίδας όσο και των δεδομένων.

Το Άθροισμα ελέγχου UDP είναι προαιρετικό και καταχωρείται ως 0 όταν δεν υπολογίζεται. (Το πραγματικά υπολογισμένο 0 καταχωρείται με όλα τα bit ίσα με 1, που είναι το ίδιο σε συμπλήρωμα ως προς 1). Το να μην χρησιμοποιηθεί είναι ανόητο, εκτός εάν η ποιότητα των δεδομένων δεν έχει μεγάλη σημασία (π.χ. η ψηφιοποιημένη φωνή). Το UDP είναι ένα μη αξιόπιστο πρωτόκολλο, για εφαρμογές που δεν θέλουν τον έλεγχο της ακολουθίας ή της ροής που προσφέρει το πρωτόκολλο TCP και επιθυμούν να χρησιμοποιήσουν δικό τους. Επίσης χρησιμοποιείται ευρέως σε γρήγορες εφαρμογές και ερωταποκρίσεις, τύπου πελάτη-εξυπηρετητή, όπου η άμεση παράδοση είναι σπουδαιότερη από τη σωστή.

5. Διαφοροποιημένες Υπηρεσίες (Differentiated Services – DiffServ)

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) 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) έχουν αναπτυχθεί τα πρότυπα IntServ (Integrated Services), DiffServ (Differentiated Services) και MPLS (Multi-protocol Label Switching) 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.

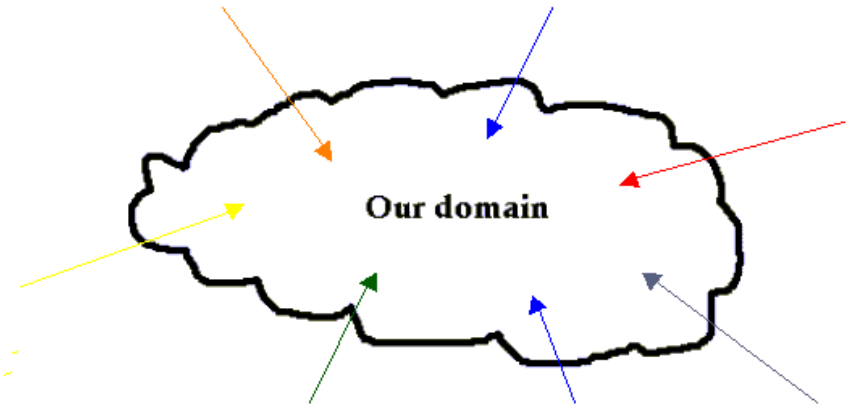
Ένα από τα προβλήματα που αντιμετωπίζει στις μέρες μας το περιβάλλον του Internet είναι πως θα μπορούσαμε να προσφέρουμε καλύτερες και διαφοροποιημένες υπηρεσίες στους χρήστες. Βασισμένη στην ιδέα ότι διαφορετικής ποιότητας υπηρεσίες πρέπει να προσφέρονται έτσι ώστε να καλύπτονται οι διάφορες απαιτήσεις και ανάγκες των χρηστών, η αρχιτεκτονική της διαφοροποιημένης υπηρεσίας (diffserv) αναπτύχτηκε. Χρησιμοποιώντας την τεχνολογία του

diffserv, επιτυγχάνουμε τη βελτιστοποίηση των διαφόρων υπηρεσιών καθώς επίσης μπορούμε να προσφέρουμε καλύτερη ποιότητα και πλουσιότερες επιλογές.

Η αρχιτεκτονική του diffserv είναι βασισμένη σε ένα μοντέλο δικτύου το οποίο εφαρμόζεται πάνω σε ολοκληρωμένο και Αυτόνομο Σύστημα ή σε ένα τομέα (domain). Έχοντας αυτό το τομέα κάτω από τον έλεγχό μας, μπορούμε να προετοιμαστούμε κατάλληλα έτσι ώστε να εγκαθιδρύσουμε ξεκάθαρους και συνεπείς κανόνες, κανόνες οι οποίοι θα διαχειρίζονται την εισερχόμενη κίνηση καθώς και τη ροή μέσα στα διάφορα δίκτυα που απαρτίζουν το τομέα.

Για να το πραγματοποιήσουμε αυτό, εφαρμόζεται μια αρχιτεκτονική όπου η κίνηση η οποία εισέρχεται στο δίκτυο από τις άκρες του τομέα κατηγοριοποιείτε και ανατίθεται σε κάποια από τις διάφορες ομάδες συμπεριφοράς (behavior aggregates) που έχουμε δημιουργήσει. Κάθε μία από αυτές τις ομάδες αναγνωρίζεται από το marking της κεφαλίδας των πακέτων (κάθε ομάδα έχει και διαφορετικό marking) που εισέρχονται στο τομέα. Μέσα στο τομέα, τα πακέτα που ανήκουν στην ίδια ομάδα προωθούνται ανάλογα με τους κανόνες που έχουν εγκαθιδρυθεί προηγουμένως. Με αυτό το τρόπο, αυτό που πραγματικά κάνουμε είναι να δημιουργούμε κλάσεις που απαρτίζονται από ροές οι οποίες 'ταξιδεύουν' μέσα από τα δίκτυά μας. Κάθε μία ροή, μέσα στο τομέα, συμπεριφέρεται ανάλογα στη κλάση στην οποία ανήκει.

Το συννεφάκι του Σχήμα 14 αντιπροσωπεύει το τομέα μας, ενώ τα βέλη που εισέρχονται σε αυτό είναι διαφορετικές ροές που δεχόμαστε από έξω. Τα χρώματα των ροών είναι διαφορετικά δηλώνοντας με αυτό το τρόπο ότι δεν έχουν την ίδια σημασία για εμάς.

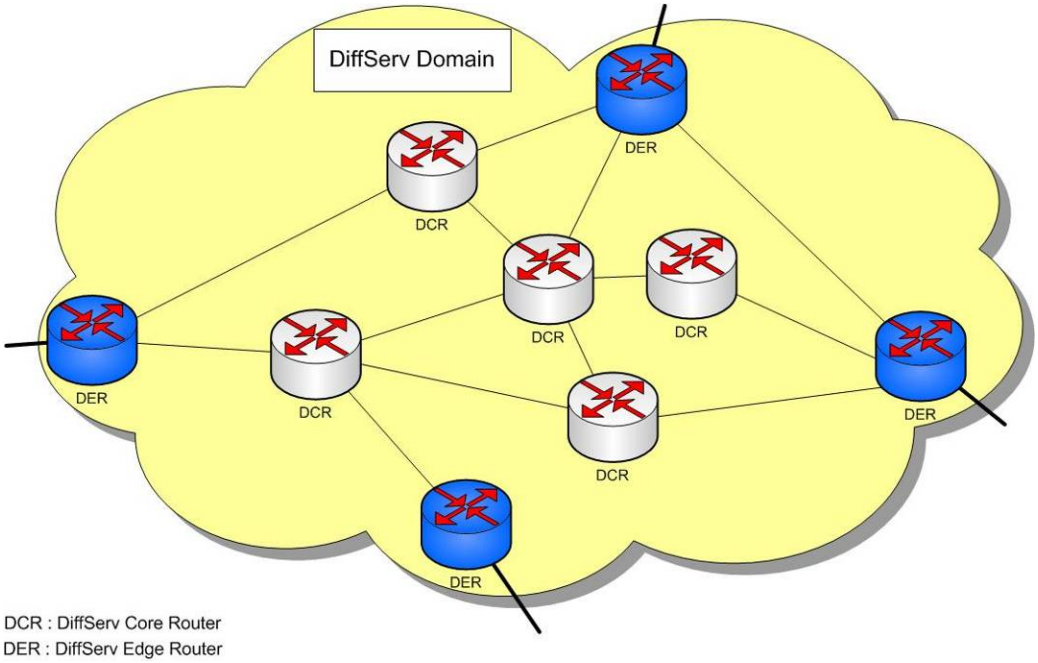


Σχήμα 14. Αναπαράσταση του τομέα και των εισερχόμενων ροών

5.1. Αρχιτεκτονική Διαφοροποιημένων Υπηρεσιών

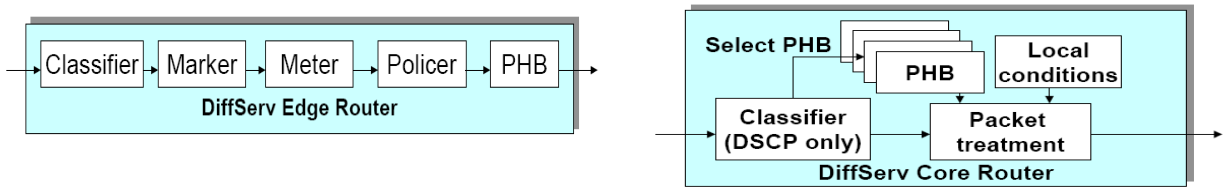
Η αρχιτεκτονική των Διαφοροποιημένων Υπηρεσιών είναι βασισμένη σε ένα απλό μοντέλο όπου η κίνηση που εισέρχεται σε ένα δίκτυο που υλοποιεί διαφοροποίηση υπηρεσιών (DiffServ Domain), κατηγοριοποιείται, ρυθμίζεται και ταξινομείται στα διάφορα σύνολα συμπεριφοράς (Behavior Aggregates – BA). Οι ενέργειες 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 15. 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)[11],[12]. 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 16).

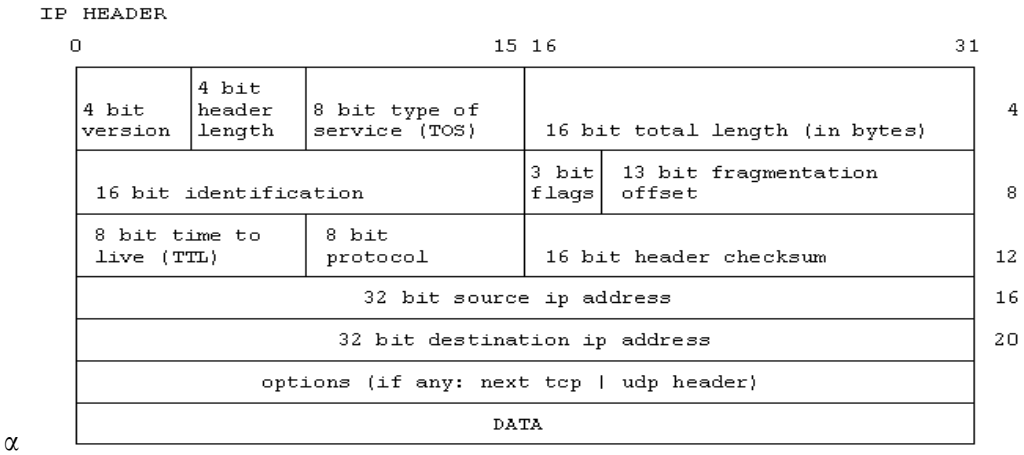


α β

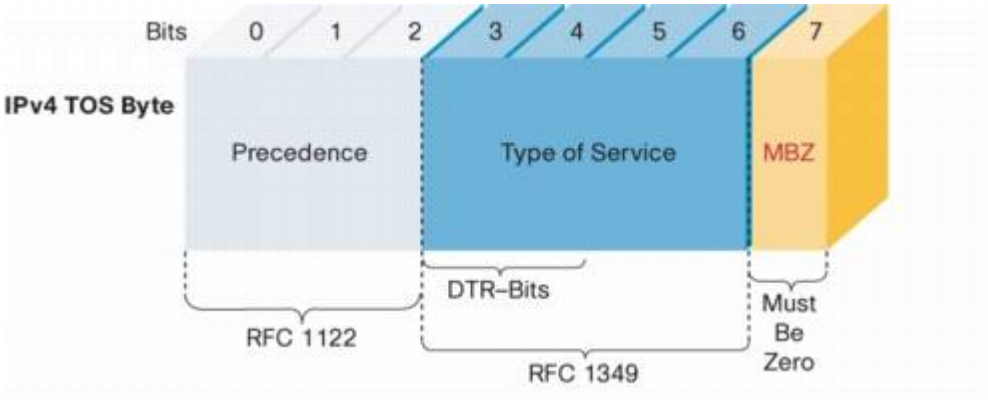
Σχήμα 16. Ενέργειες για την εφαρμογή DiffServ σε : α)DER, β)DCR

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 Κωδικό σημείο 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 17.

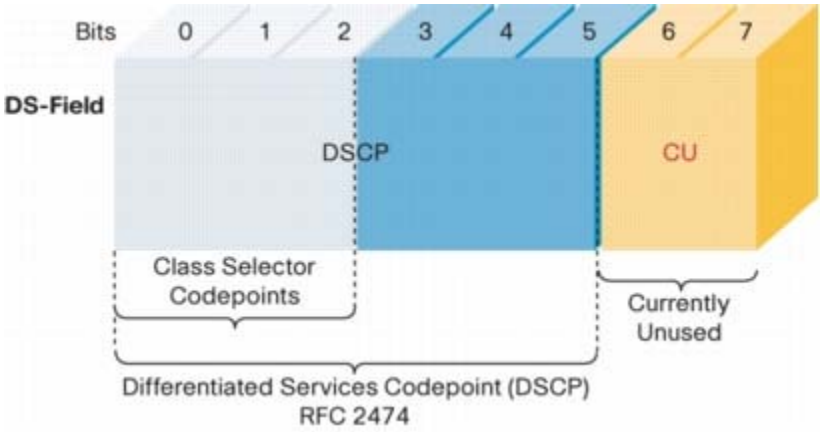
α



β



γ



Σχήμα 17. α)Η επικεφαλίδα των IP πακέτων, β) το πεδίο ToS, γ) το πεδίο DSCP

5.2. 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*) [8] 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*) [7], 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 18.
- 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.

	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

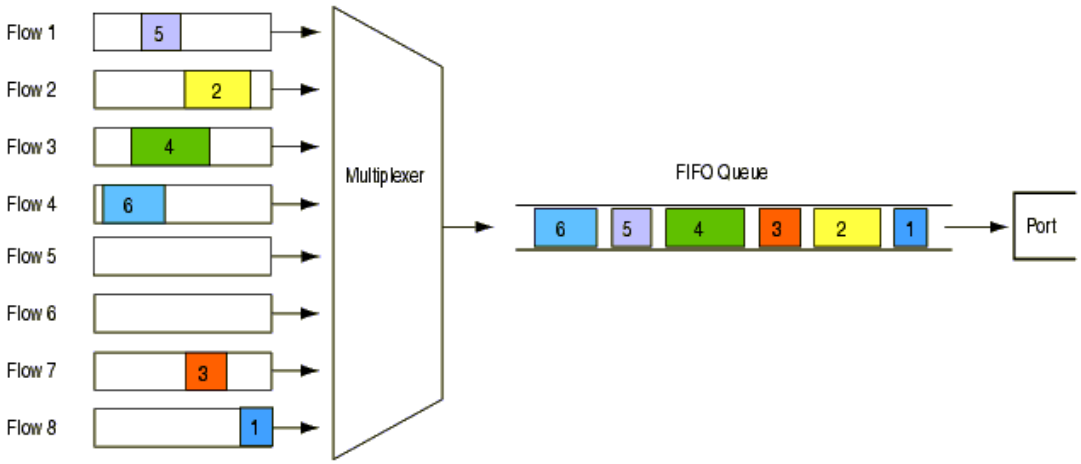
Σχήμα 18. Παρουσίαση των τιμών του DSCP πεδίου των AF κλάσεων

5.3. Αλγόριθμοι Μορφοποίησης Κίνησης

Υπάρχει ένα σύνολο από αλγόριθμους μορφοποίησης κίνησης, μερικοί από τους οποίους είναι οι FIFO, TBF, SFQ, RED, GRED, PRIO, HTB και DSMARK

5.3.1. *FIFO*

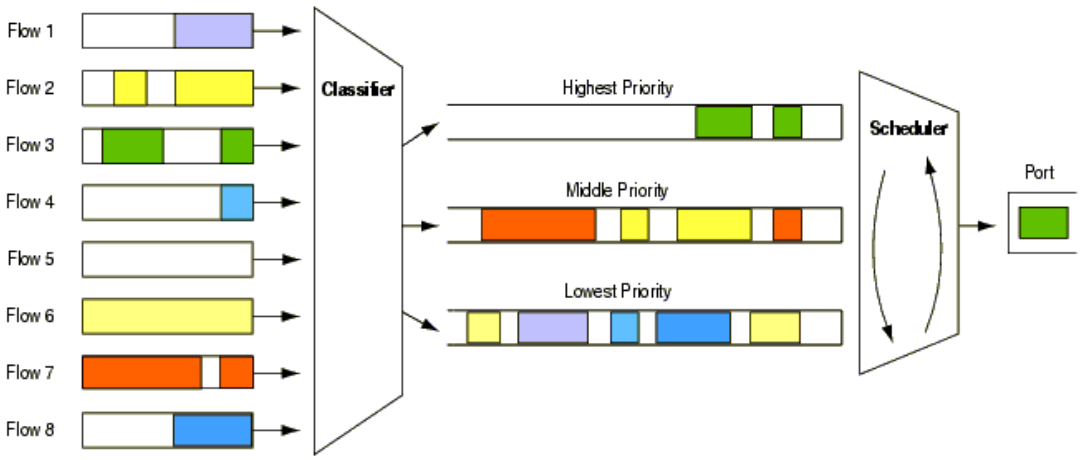
Ο αλγόριθμος FIFO (First In First Out) είναι από τους πιο απλούς. Από το όνομα του αλγορίθμου μπορούμε να καταλάβουμε ότι ο πρώτος που μπαίνει φεύγει και πρώτος δηλαδή η λειτουργία του έχει ως εξής τα πακέτα που φτάνουν μπαίνουν σε μια ουρά και περιμένουν μέχρι να έρθει η σειρά τους να φύγουν απλά φεύγουν 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 19 μπορούμε να δούμε τη σχηματική αναπαράσταση του αλγορίθμου.



Σχήμα 19. Σχηματική Αναπαράσταση του Αλγορίθμου FIFO

5.3.2. *PRIO*

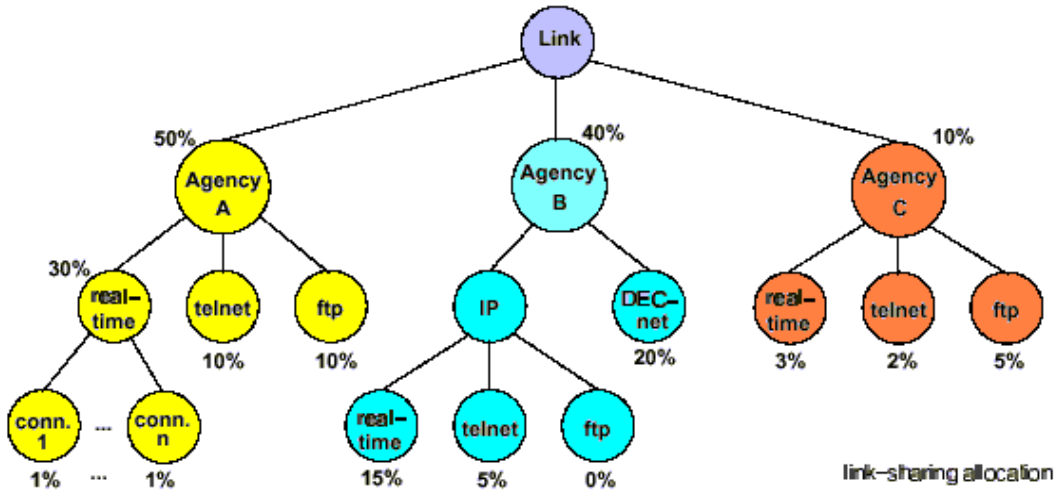
Ο αλγόριθμος PRIO (Priority Queuing) όπου μια σχηματική αναπαράσταση μπορούμε να δούμε στο Σχήμα 20 ταξινομεί τα πακέτα σε διαφορετικής προτεραιότητας ουρές ο μόνος περιορισμός είναι ότι για να εξυπηρετηθεί ένα πακέτο χαμηλότερης προτεραιότητας ουρά θα πρέπει όλες υψηλότερης προτεραιότητας ουρές να μην έχουν άλλα πακέτα προς εξυπηρέτηση.



Σχήμα 20. Σχηματική Αναπαράσταση του Αλγορίθμου PRIO

5.3.3. *HTB*

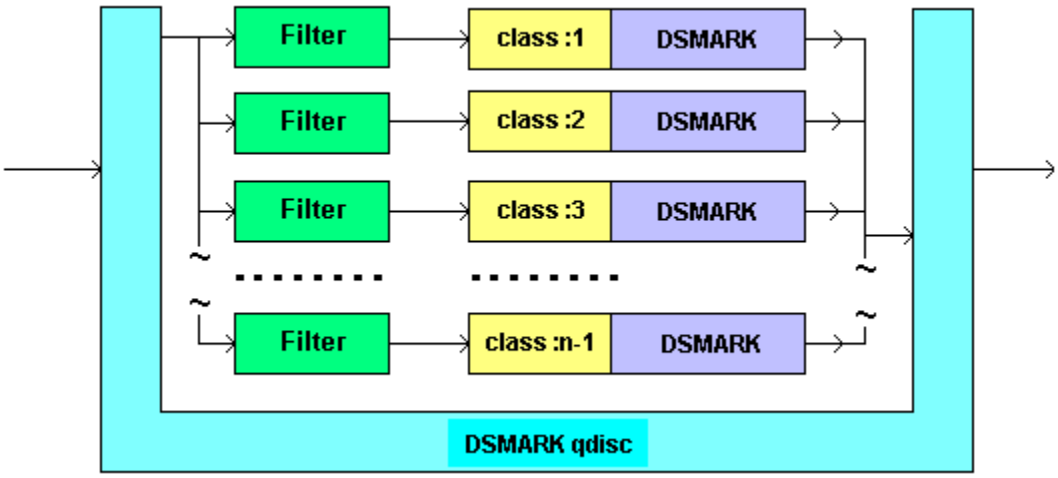
Ο αλγόριθμός HTB (Hierarchical Token Bucket) χρησιμοποιείται για την μορφοποίηση της δικτυακής κίνησης σε ένα ιεραρχικό μοντέλο (Σχήμα 21) όπου κάθε κλάση παίρνει το δικαίωμα να στείλει τα πακέτα που έχει στις ουρές με ένα μέσο ρυθμό μετάδοσης και με ελεγχόμενους καταιγισμούς (bursts) της ανάλογα με το ποσοστό εύρους που διαθέτει. Ο αλγόριθμος αυτός χρησιμοποιείται για τον καταμερισμό του bandwidth και μπορεί να χρησιμοποιηθεί για να διαφοροποιήσουμε υπηρεσίες.



Σχήμα 21. Ιεραρχικός Διαμοιρασμός του Διαθέσιμου Bandwidth

5.3.4. *DSMARK*

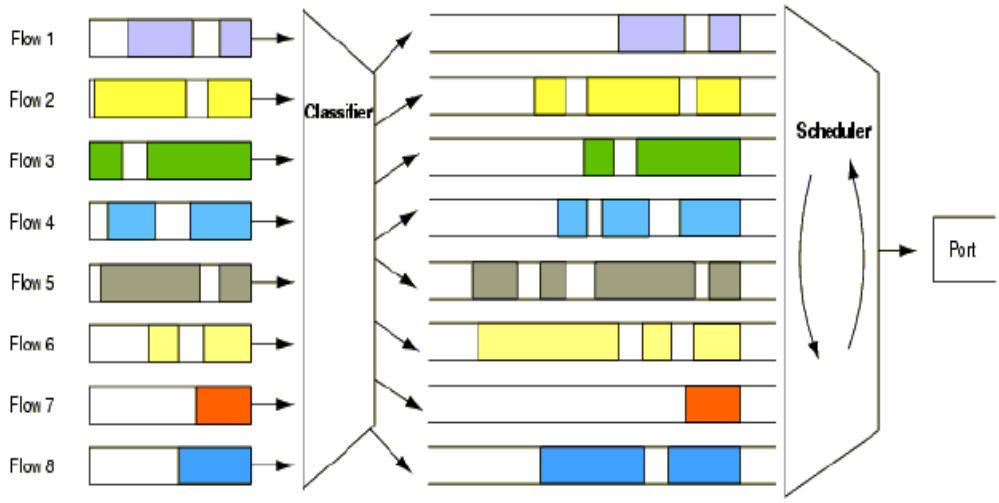
Ο αλγόριθμος DSMARK χρησιμοποιείται για να μαρκάρει τα πακέτα στο πεδίο DS της επικεφαλίδας ενός IP πακέτου. Στο Σχήμα 22 μπορούμε να διακρίνουμε σε ποίο σημείο της υλοποίησης του DiffServ γίνεται το marking (μαρκάρισμα) της υπηρεσίας-κίνησης.



Σχήμα 22. Σχηματική Αναπαράσταση του Αλγορίθμου DSMARK

5.3.5. *SFQ*

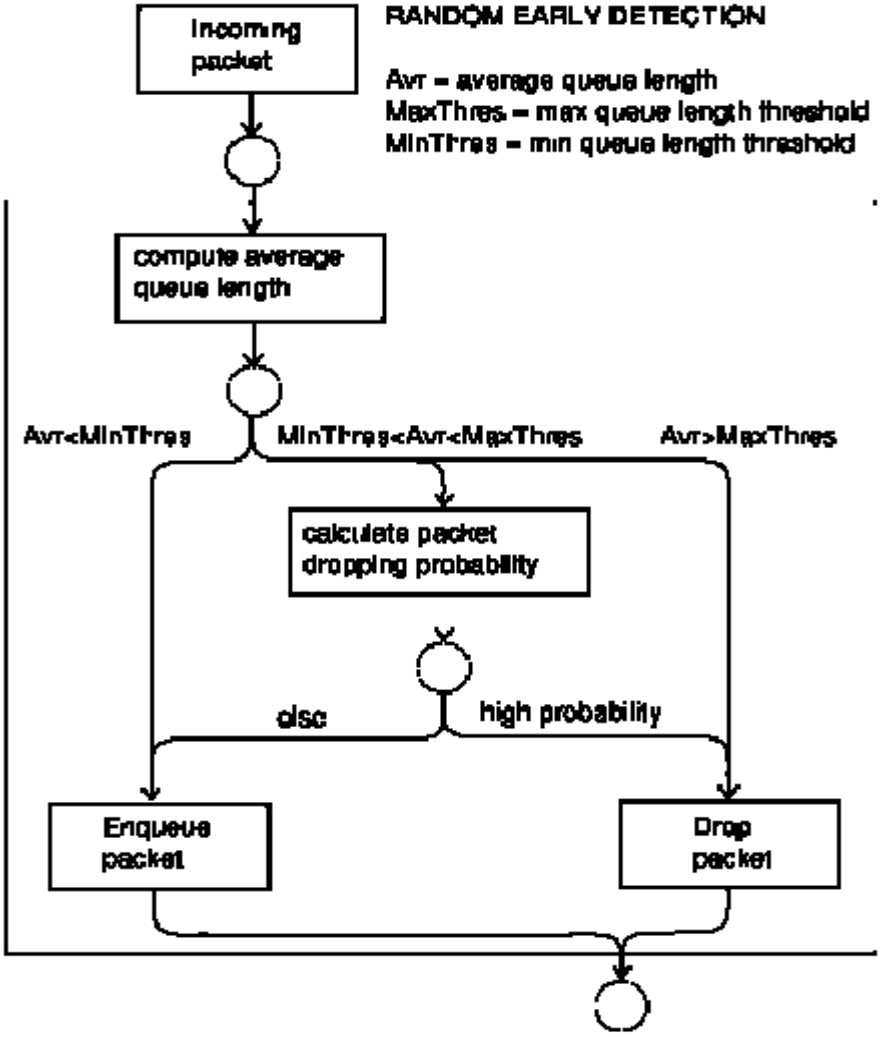
Ο αλγόριθμος SFQ (Stochastic Fair Queuing) είναι σχεδιασμένος με τέτοιο τρόπο ώστε να εξασφαλίζει κάθε ροή να έχει επαρκή πρόσβαση στους πόρους του δικτύου και παράλληλα να αποτρέπει μια καταιγιστική ροή από την κατανάλωση περισσότερου εύρους ζώνης από αυτό που της αντιστοιχεί. Τα πακέτα αρχικά ταξινομούνται από το σύστημα σε ροές και στην συνέχεια εξυπηρετείται ένα πακέτο την φορά από κάθε ουρά με κυκλική σειρά, παραλείποντας τις κενές ουρές.



Σχήμα 23. Σχηματική Περιγραφή Του Αλγορίθμου SFQ

5.3.6. *RED*

Μια εναλλακτική αντιμετώπιση της τεχνικής drop tail που εφαρμόζεται στην FIFO ουρά είναι ο αλγόριθμος RED (Random Early Detection). Σκοπός του είναι να μην αφήνει την FIFO ουρά να γεμίσει, απορρίπτοντας επιλεκτικά πακέτα όταν χρειάζεται. Με αυτό τον τρόπο αντιμετωπίζεται η συμφόρηση του δικτύου και το TCP πρωτόκολλο καταφέρνει να αποκτήσει γρηγορότερα τον κατάλληλο ρυθμό αποστολής δεδομένων. Μια πιο εξελιγμένη μορφή του RED είναι ο αλγόριθμος GRED (Generalized RED) που υποστηρίζει πολλαπλές προτεραιότητες απόρριψης, καθιστώντας τον κατάλληλο για εφαρμογή στις Διαφοροποιημένες Υπηρεσίες.



Σχήμα 24. Σχηματική περιγραφή του αλγόριθμου **RED**

5.4. Ταξινομητές (Classifiers)

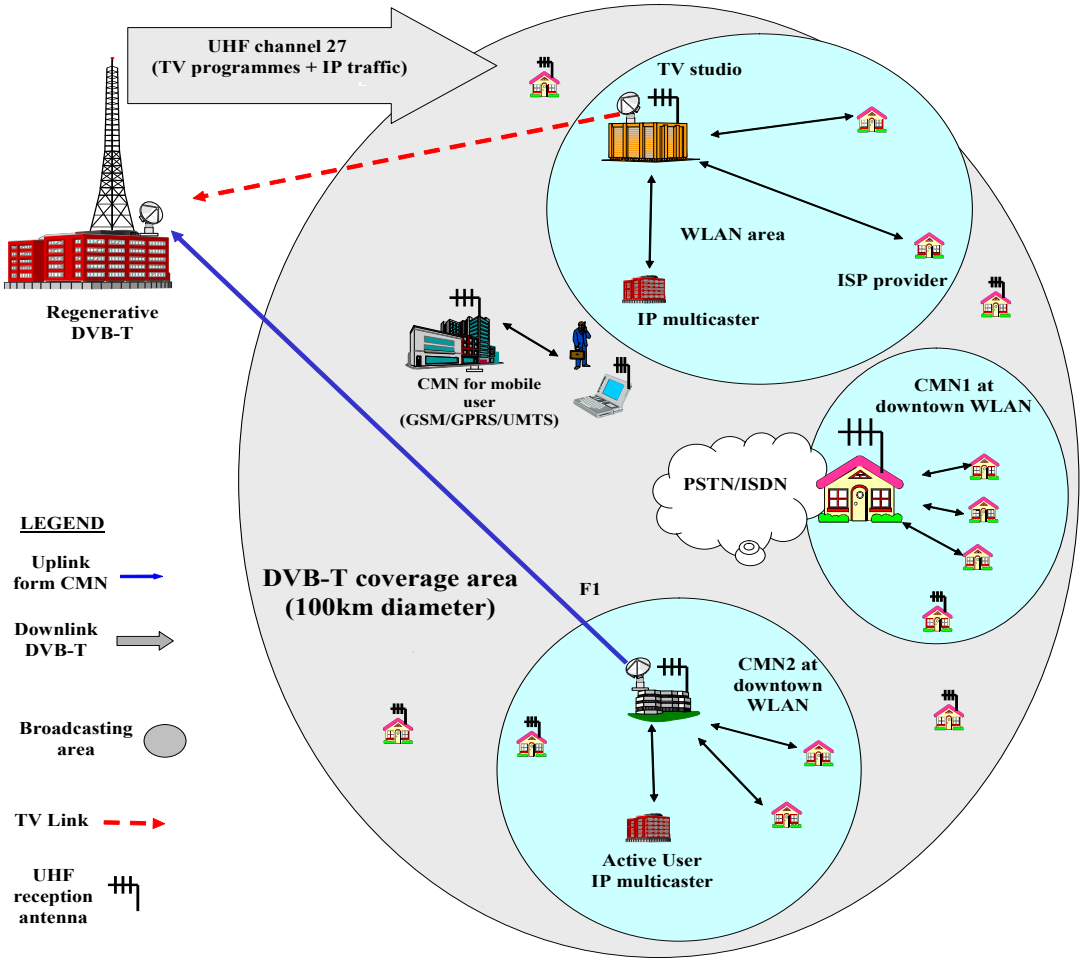
Τέλος, ένας σημαντικός παράγοντας για την αστυνόμευση και την κατηγοριοποίηση των πακέτων είναι οι ταξινομητές (classifiers). Η αστυνόμευση προϋποθέτει μέτρηση της δικτυακής κίνησης, ώστε να μπορεί να γίνει έλεγχός αν συμφωνεί με τους όρους που αναφέρονται στο συμβόλαιο για την παρεχόμενη ποιότητα υπηρεσίας. 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.

6. Αρχιτεκτονική Πειραματικού Δικτύου Επίγειας Ψηφιακής Τηλεόρασης

Εισαγωγή

Η δυνατότητα που παρέχει η πλατφόρμα επίγειας ψηφιακής τηλεόρασης να μεταφέρει ψηφιακά τηλεοπτικά προγράμματα (MPEG - 2) καθώς και υπηρεσίες IP σε συνδυασμό με την ύπαρξη καναλιών επιστροφής (Reverse Path), μας δίνει τη δυνατότητα να υλοποιήσουμε ευρυζωνικές υποδομές για την παροχή αμφίδρομων διαδραστικών υπηρεσιών.

6.1. Τοπολογία Πειραματικού Δικτύου



Σχήμα 25. Αρχιτεκτονική ενός DVB-T συστήματος

Ένα τέτοιο σύστημα φαίνεται και στο παραπάνω σχήμα (Σχήμα 25).Το σύστημα αυτό αποτελείτε από τις εξής στοιχεία:

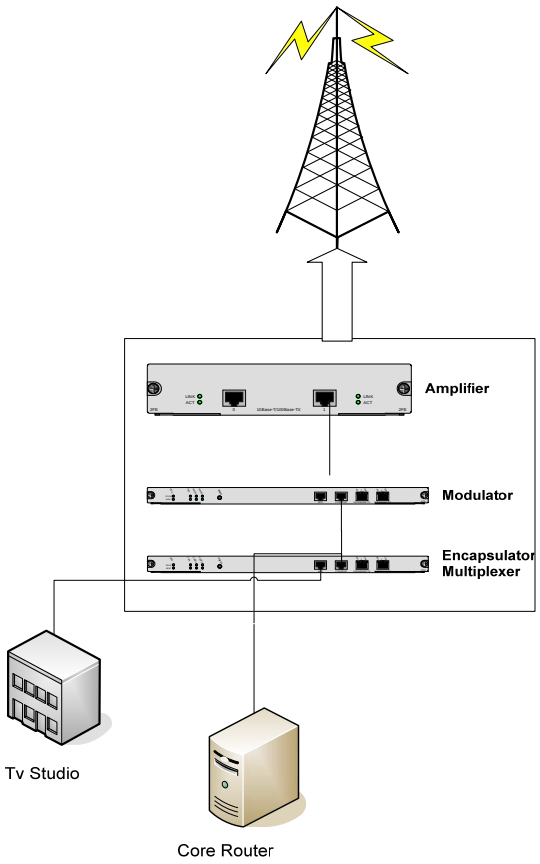
- Το κεντρικό σημείο εκπομπής (πλατφόρμα DVB - T)
- Τους ενδιάμεσους κόμβους διανομής (Cell Main Nodes CMN)
- Τους τελικούς χρήστες

6.2. Κεντρικό Σημείο Εκπομπής

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

Ο DiffServ Core Router (DCR) αναλαμβάνει να ελέγξει και να κατηγοριοποιήσει την κίνηση που φτάνει από τους CMN βάσει των script των Διαφοροποιημένων Υπηρεσιών (DiffServ) που τρέχουν σε αυτόν. Στη συνέχεια η κατηγοριοποιημένη κίνηση προωθείτε στον ενθυλακωτή-πολυπλέκτη (Encapsulator-Multiplexer) για να ενθυλακωθούν τα πακέτα στο MPEG2-TS μαζί με τα τηλεοπτικά προγράμματα που στέλνονται 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 διαμορφωτής 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 . Τέλος η τελική ροή μεταφοράς MPEG-2 δηλαδή τα IP πακέτα ενθυλακωμένα μαζί με τα τηλεοπτικά προγράμματα περνάνε μέσα από τον ενισχυτή και εκπέμπονται τελικά σε όλη τη περιοχή κάλυψης του DVB-T.



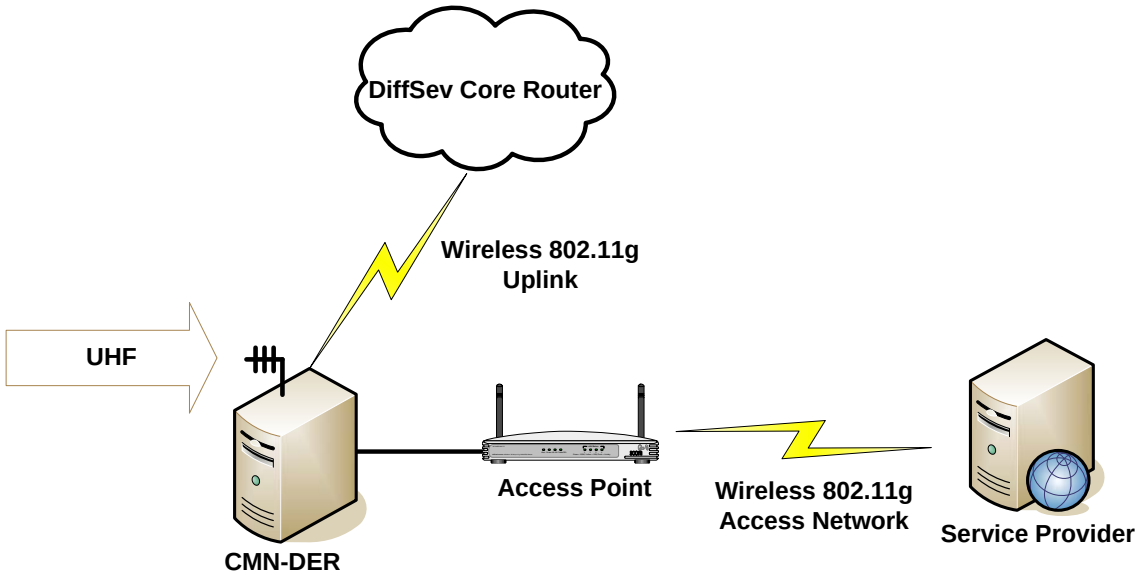
Σχήμα 26. Κεντρικό σημείο εκπομπής

6.3. Ενδιάμεσος Κόμβος Διανομής (Cell Main Node)

Ο ενδιάμεσος κόμβος διανομής (CMN) χρησιμοποιείτε για την επικοινωνία των χρηστών αλλά και για την ταξινόμηση, αστυνόμευση και διαμόρφωση των πακέτων ανάλογα με την εκάστοτε υπηρεσία.

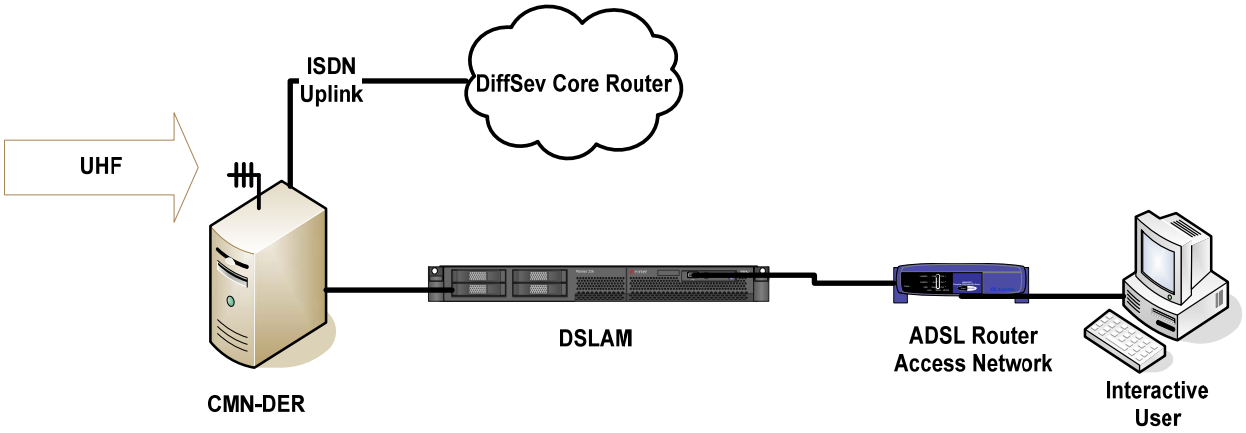
Στο πρώτο CMN-DER (DiffServ Edge Router) έχουμε εγκαταστημένο ένα λειτουργικό σύστημα Linux-Debian στον οποίο συνδέεται ο Service Provider μέσω ενός AP (Access Point). Το δίκτυο πρόσβασης του Service Provider με τον CMN1 αποτελείτε από ένα ασύρματο δίκτυο τύπου 802.11g. Πιο αναλυτικά, ο Service Provider έχει εγκατεστημένη μία ασύρματη κάρτα δικτύου για επικοινωνία με το AP το οποίο και συνδέεται με μια ενσύρματη κάρτα με τον CMN1. Ο CMN1 εκτός από την ενσύρματη κάρτα που έχει για την σύνδεση με το AP αποτελείτε επίσης από μια ασύρματη κάρτα δικτύου για την σύνδεση με το κεντρικό

σημείο εκπομπής, και τέλος από μία κάρτα ψηφιακής τηλεόρασης που χρησιμοποιείτε για την επικοινωνία από την πλατφόρμα DVB-T προς τον CMN1 σαν κανάλι καθόδου.



Σχήμα 27. Αρχιτεκτονική του Service Provider με τον CMN-DER

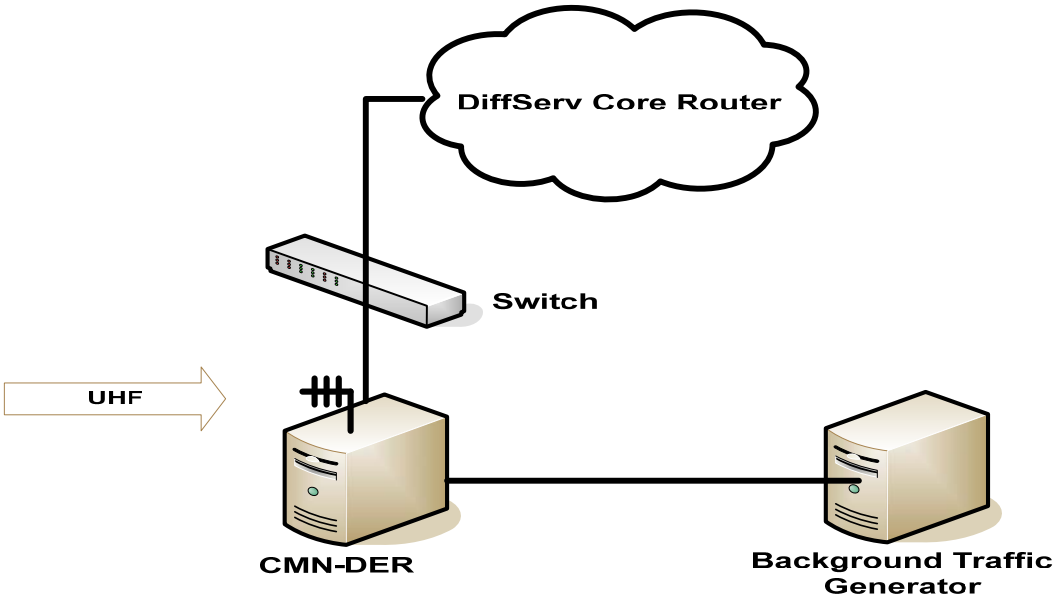
Στο δεύτερο CMN στον οποίο συνδέεται ο τελικός χρήστης έχουμε και εδώ εγκαταστημένο ένα λειτουργικό σύστημα Linux-Debian. Το δίκτυο πρόσβασης του τελικού χρήστη με τον CMN2-DER αποτελείτε από μία ADSL γραμμή. Συγκεκριμένα ο τελικός χρήστης συνδέεται μέσω ενός ADSL router με μια συσκευή DSLAM η οποία με την σειρά της συνδέεται στον CMN2. Η ADSL γραμμή που δημιουργείτε προσφέρει 8 Mbps Downlink και 512 Kbps Uplink. Ο CMN2 αποτελείτε από μία ενσύρματη κάρτα για την σύνδεση με τη συσκευή του DSLAM, μια ISDN γραμμή για την σύνδεση με το κεντρικό σημείο εκπομπής καθώς και από μία κάρτα ψηφιακής τηλεόρασης για την επικοινωνία από την πλατφόρμα DVB-T προς τον CMN2 σαν κανάλι καθόδου.



Σχήμα 28. Αρχιτεκτονική του Interactive User με τον CMN-DER

6.4. Γεννήτρια Κίνησης (Background Generator)

Για την υλοποίηση του Background Traffic Generator θα γίνει χρήση ενός υπολογιστή στον οποίο έχει εγκατασταθεί το ελεύθερο λειτουργικό σύστημα Linux (Debian Distribution) και επικοινωνεί με τον CMN – DER μέσω μιας κάρτας δικτύου Fast Ethernet. Από την πλευρά του CMN – DER χρησιμοποιείται ένας υπολογιστή με εγκατεστημένο το λειτουργικό σύστημα Linux (Debian Distribution), μια κάρτα δικτύου Fast Ethernet για την επικοινωνία με τον Background Traffic Generator, μια κάρτα δικτύου Fast Ethernet για την επικοινωνία με την DVB-T πλατφόρμα και ένα δέκτη DVB-T σήματος.

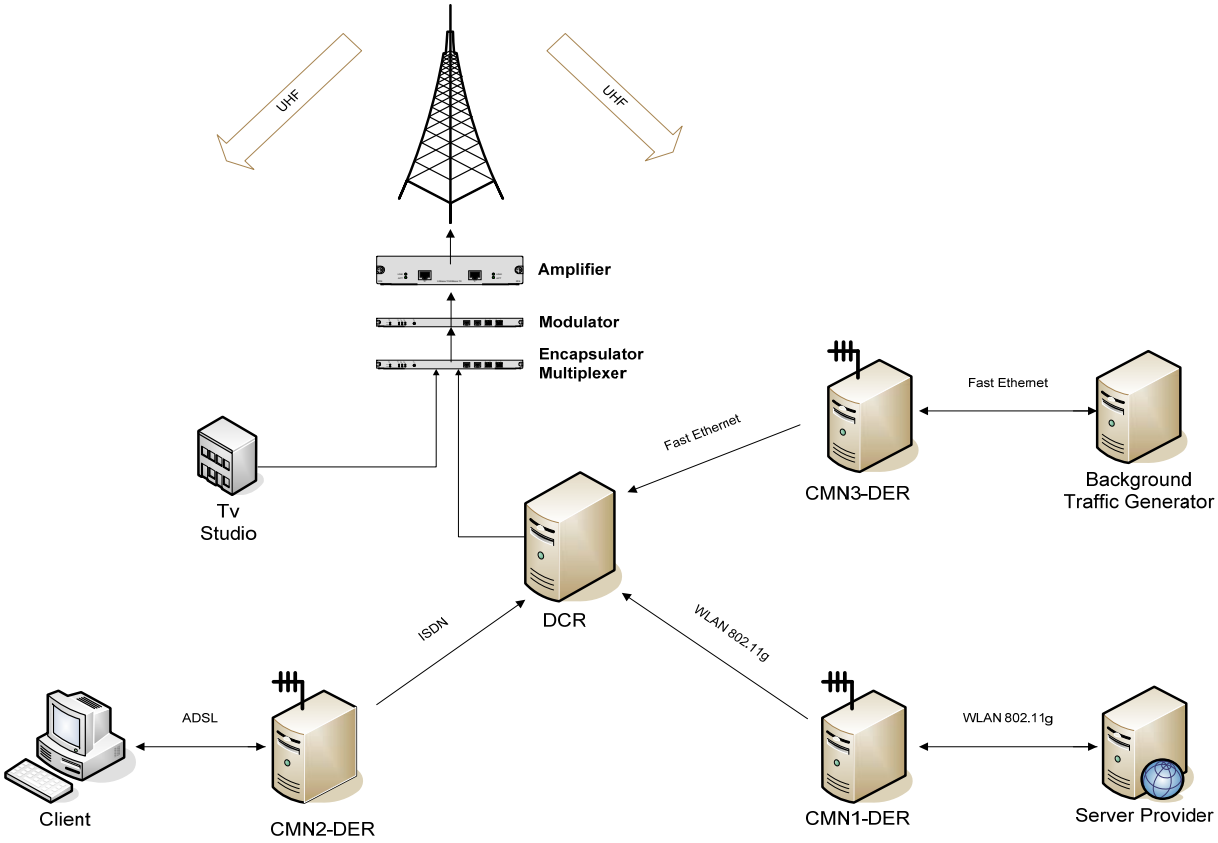


Σχήμα 29. Αρχιτεκτονική Background Traffic Generator με τον CMN-DER

7. ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ – ΣΕΝΑΡΙΑ

Παρακάτω θα γίνει μια περιγραφή της πειραματικής διαδικασίας καθώς και των σεναρίων που θα υλοποιηθούν. Κάθε ένα από τα σενάρια που θα υλοποιηθούν και που θα περιγραφούν παρακάτω καλύπτει και μία διαφορετική περίπτωση δικτυακής κίνησης. Σκοπός της πειραματικής αυτής διαδικασίας είναι να παρατηρηθεί η συμπεριφορά του δικτύου καθώς και να αξιολογηθεί η απόδοσή του κάτω από συγκεκριμένες συνθήκες.

Η ολοκληρωμένη αρχιτεκτονική πάνω στην οποία θα βασιστεί το πειραματικό μέρος αυτής της πτυχιακής εργασίας παρουσιάζεται στο παρακάτω σχήμα.



Σχήμα 30. Γενική Αρχιτεκτονική Του Δικτύου

Στα σενάρια που θα ακολουθήσουν κάθε μέτρηση έχει διάρκεια 180 δευτερολέπτων και τα μεγέθη που θα εξετάσουμε είναι:

- Η Μονόδρομη Καθυστέρηση (One Way Delay) και αναφέρεται στον χρόνο που χρειάζεται ένα πακέτο για να μεταδοθεί από ένα δικτυακό κόμβο σε έναν άλλο.

- Το Τρέμουλο (Jitter), που είναι η διακύμανση της καθυστέρησης, δηλαδή ο χρόνου μεταφοράς από άκρο σε άκρο.

- Οι Απώλειες (Losses), που ορίζεται ως ο λόγος του αριθμού των πακέτων που ελήφθησαν προς τον αριθμό των πακέτων που στάλθηκαν.

7.1. ΣΕΝΑΡΙΑ

- **Σενάριο 1ο:** Στο πρώτο αυτό σενάριο της πειραματικής διαδικασίας θα δημιουργηθεί κίνηση 4 ροών με ρυθμό μετάδοσης 2 Mbps η καθεμία, με εικονικά πακέτα δεδομένων από τον Server στο τελικό χρήστη χρησιμοποιώντας το UDP πρωτόκολλο και όλο το διαθέσιμο εύρος ζώνης του δικτύου.

- **Σενάριο 2ο:** Εδώ θα δημιουργηθεί κίνηση 4 ροών από τον Server στο τελικό χρήστη με εικονικά πακέτα δεδομένων και χρησιμοποιώντας το UDP πρωτόκολλο και όλο το διαθέσιμο εύρος ζώνης του δικτύου ενώ παράλληλα η γεννήτρια παραγωγής background κίνησης θα μεταδίδει με ρυθμό 7 Mbps. Αυτό το κάνουμε έτσι ώστε να προκαλέσουμε κατάσταση συμφόρησης στο δίκτυό μας.

- **Σενάριο 3ο:** Στο σενάριο αυτό ενεργοποιούμε το μηχανισμό DiffServ και δημιουργούμε κίνηση 4 ροών με ρυθμό μετάδοσης και εδώ 2 Mbps η καθεμία, με εικονικά πακέτα δεδομένων από τον Server στο τελικό χρήστη χρησιμοποιώντας πάντα το UDP πρωτόκολλο, ενώ παράλληλα η γεννήτρια παραγωγής background κίνησης θα μεταδίδει με ρυθμό 7 Mbps.

- **Σενάριο 4ο:**

  Μέρος Α: Θα δημιουργηθεί κίνηση 4 ροών με Background κίνηση χωρίς ενεργοποιημένο το μηχανισμό DiffServ από τον Server στο τελικό χρήστη όπως ακριβώς και στα παραπάνω σενάρια με τη διαφορά όμως ότι τα εικονικά πακέτα θα αντικατασταθούν από πραγματικά βίντεο.

  Μέρος Β: Θα επαναληφθεί το Α' μέρος του σεναρίου αυτού με το μηχανισμό DiffServ όμως αυτή τη φορά ενεργοποιημένο.

7.2. Προγράμματα που Χρησιμοποιήθηκαν Για Την Υλοποίηση Και Ανάλυση Των Πειραματικών Μετρήσεων

Η δημιουργία της κίνησης έγινε με τα εξής προγράμματα :

- *MEGN*: 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*: 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

Η σύλληψη της δικτυακής κίνησης με :

- *Tcpdump*: 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*: 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*: 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 [21]*: 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.

7.3. Παρουσίαση Των Πειραματικών Μετρήσεων

Στο κεφάλαιο αυτό θα παρουσιάσουμε και θα αναλύσουμε τις μετρήσεις της πειραματικής διαδικασίας, το οποίο θα γίνει μέσα από πίνακες και γραφήματα καθώς και θα σχολιάσουμε τα αποτελέσματα των πειραματικών αυτών μετρήσεων.

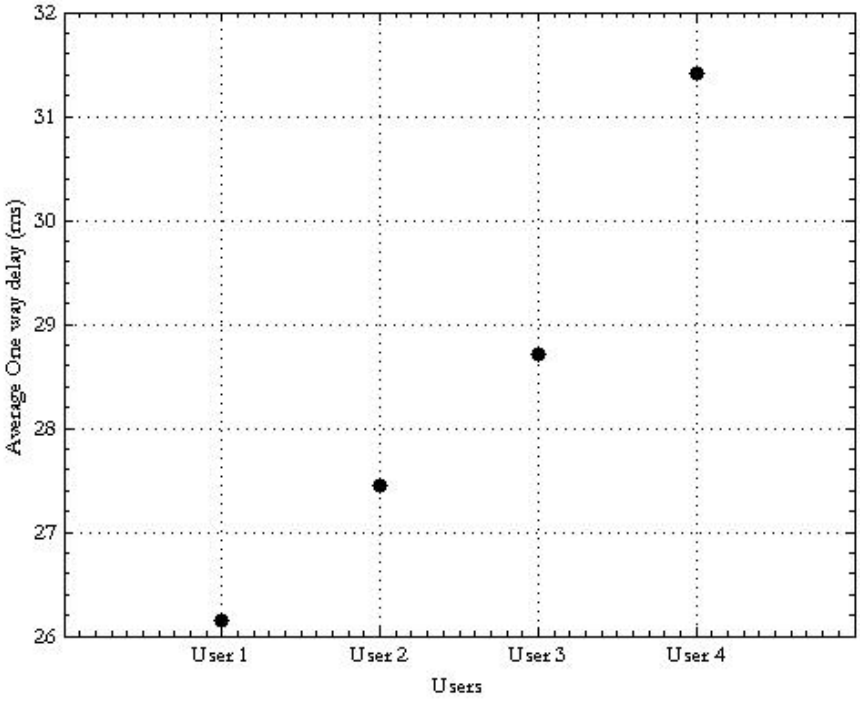
7.3.1. *Σενάριο 1ο:*

Στην πειραματική μέτρηση του πρώτου αυτού σεναρίου παρατηρούμε τα εξής αποτελέσματα:

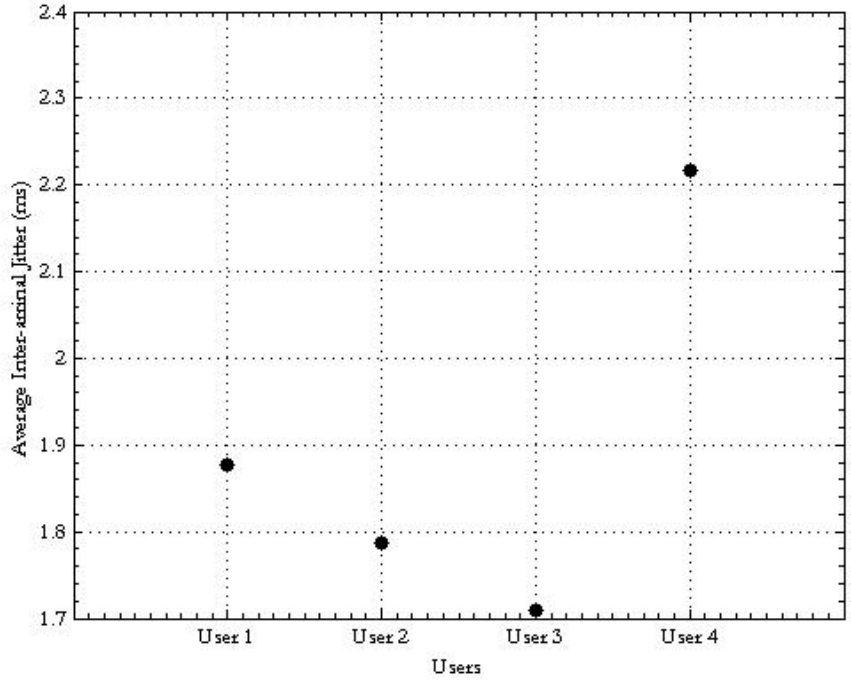
	One way delay average (ms)	**Smooth jitter average (ms)**	**Total losses (%)**
User 1	26.167	1.877	0
User 2	27.449	1.787	0
User 3	28.712	1.710	0
User 4	31.414	2.217	0

Σχήμα 31. Πίνακας Αποτελεσμάτων Πρώτου Σεναρίου

Από τα αποτελέσματα της πειραματικής μέτρησης του πρώτου σεναρίου προέκυψε ότι και οι τέσσερις ροές κάνανε χρήση όλου του διαθέσιμου bandwidth του δικτύου. Πιο αναλυτικά, στα αποτελέσματα που παραθέτονται στο παραπάνω πίνακα βλέπουμε ότι το One Way Delay κυμαίνεται σε πολύ ικανοποιητικές τιμές και για τις τέσσερις ροές, η διακύμανση της καθυστέρησης (Jitter) επίσης είναι πολύ καλή ενώ και οι απώλειες είναι μηδενικές. Ακολουθούν οι γραφικές παραστάσεις των παραπάνω αποτελεσμάτων.



Σχήμα 32. Μέσο One Way Delay πρώτου σεναρίου



Σχήμα 33. Μέσο Inter-arrival Jitter πρώτου σεναρίου

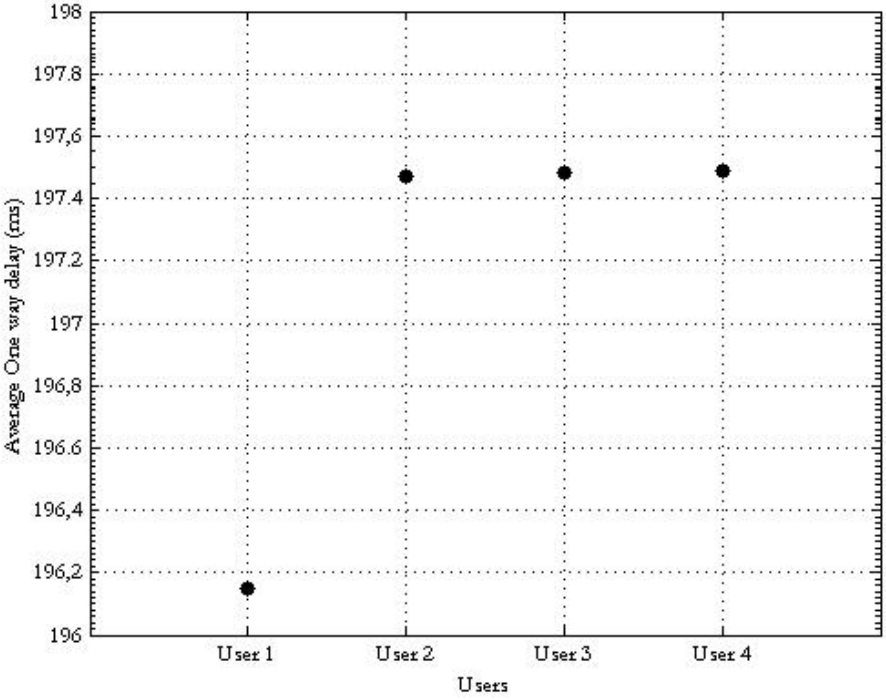
7.3.2. *Σενάριο 2ο:*

Από την ανάλυση των πειραματικών μετρήσεων του δεύτερου σεναρίου εξάγαμε τα παρακάτω αποτελέσματα:

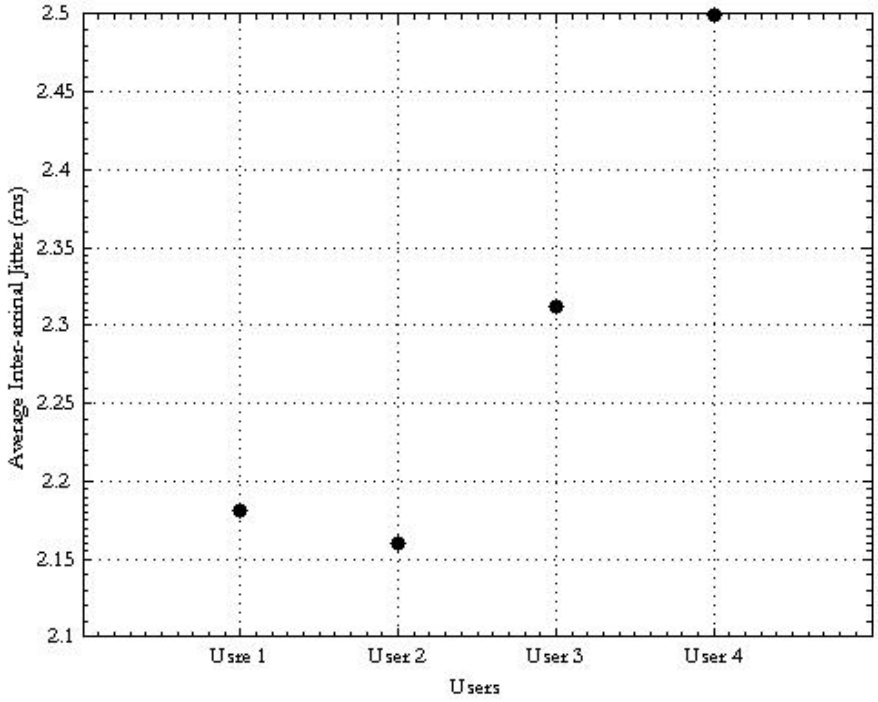
	One way delay average (ms)	**Smooth jitter average (ms)**	**Total losses (%)**
User 1	196.152	2.181	90.8
User 2	197.468	2.160	92.2
User 3	198.589	2.312	94.7
User 4	199.324	2.499	95.3

Σχήμα 34. Πίνακας Αποτελσμάτων Δεύτερου σεναρίου

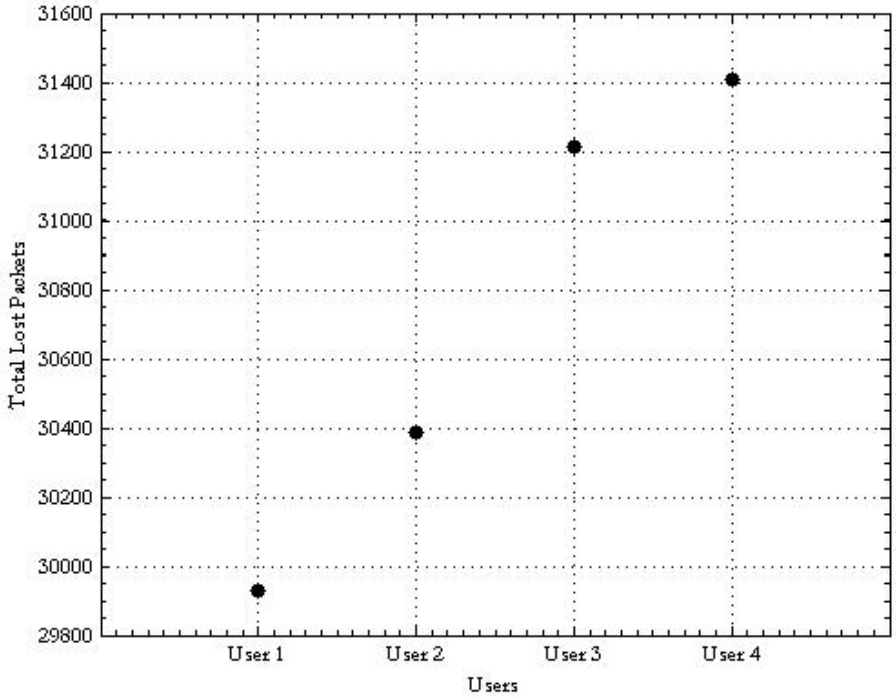
Παρατηρούμε λοιπόν ότι σε συνθήκες συμφόρησης του δικτύου όπως είναι αυτή του σεναρίου 2 το μέσο One way delay καθώς και τα χαμένα πακέτα κυμαίνονται σε μη αποδεκτές τιμές καθώς το μεν One way delay θα έπρεπε να μην κυμαίνεται πάνω από 150 ms, όπως και το ποσοστό των χαμένων πακέτων το οποίο δεν θα έπρεπε να ξεπερνά το 20%. Αντίθετα οι τιμές του Jitter βρίσκονται σε ικανοποιητικά επίπεδα. Παρακάτω παραθέτουμε τα γραφήματα των παραπάνω αποτελεσμάτων



Σχήμα 35. Μέσο One Way Delay Δεύτερου σεναρίου



Σχήμα 36. Μέσο Inter-arrival Jitter Δεύτερου σεναρίου



Σχήμα 37. . Συνολικά Χαμένα Πακέτα Δεύτερου σεναρίου

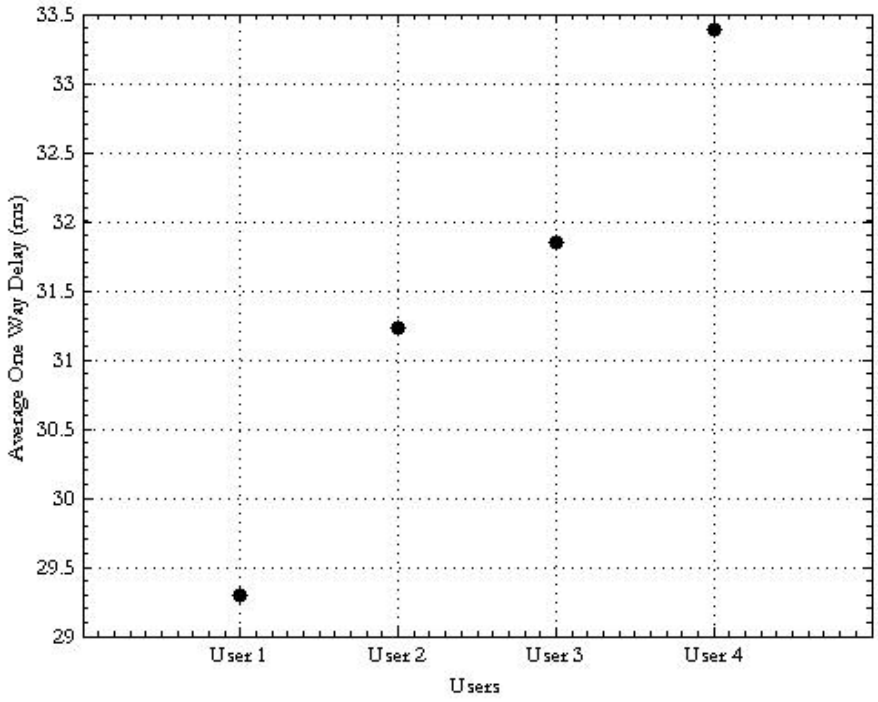
7.3.3. *Σενάριο 3ο:*

Στο 3ο αυτό σενάριο ενεργοποιούμε το μηχανισμό του DiffServ παράλληλα με την Background κίνηση, κατηγοριοποιώντας έτσι με αυτό τον τρόπο τους 4 χρήστες μας. Τα αποτελέσματα που πήραμε από την ανάλυση των μετρήσεων παρουσιάζονται στον παρακάτω πίνακα.

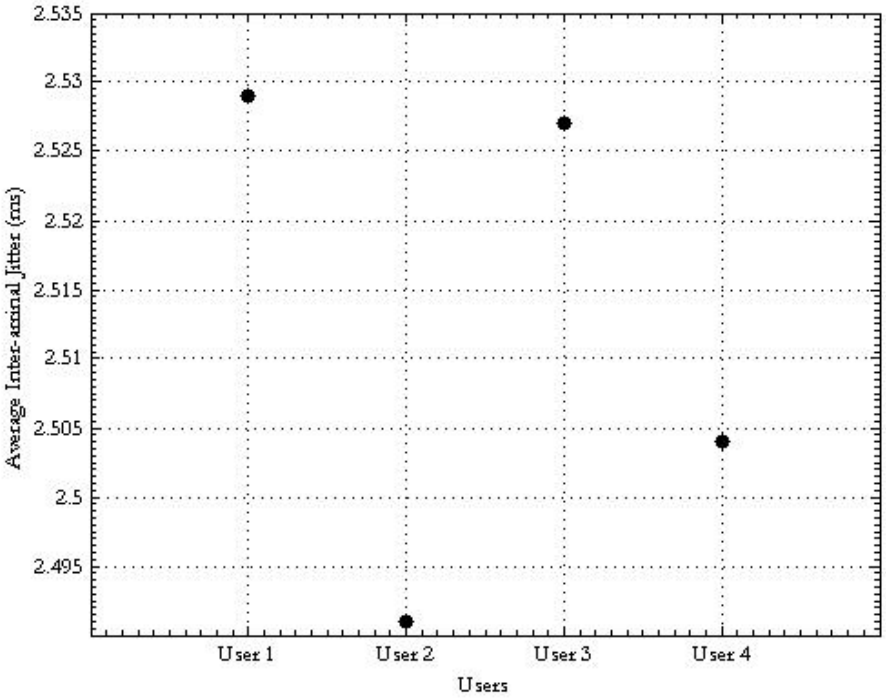
	One way delay average (ms)	**Smooth jitter average (ms)**	**Total losses (%)**
User 1	29.300	2.529	0
User 2	31.231	2.491	0
User 3	31.847	2.527	1.18
User 4	33.392	2.504	2.45

Σχήμα 38. Πίνακας Αποτελεσμάτων Τρίτου σεναρίου

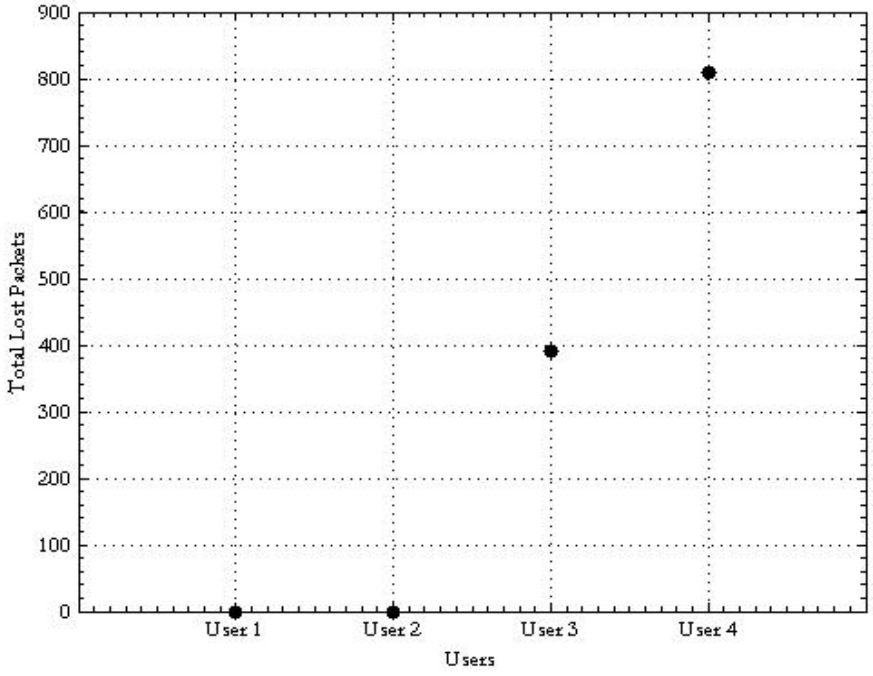
Από τα παραπάνω αποτελέσματα παρατηρούμε την βελτίωση που επιφέρει ο μηχανισμός του DiffServ σε σχέση με το πείραμα του σεναρίου 2, διασφαλίζοντας την ποιότητα και επιτυγχάνοντας ικανοποιητικές τιμές στα χαρακτηριστικά που καθορίζουν την ποιότητα ενός βίντεο. Πιο συγκεκριμένα οι δύο πρώτες ροές δεν είχαν απώλειες σε αντίθεση με τη τρίτη και τη τέταρτη ροή που παρουσιάζουν απώλειες, οι οποίες όμως κυμαίνονται κάτω από το 2.5% και δεν αποτελούν πρόβλημα στη ποιότητα των βίντεο. Οι τιμές του One Way Delay και του Jitter κυμαίνονται σε άκρως ικανοποιητικές τιμές όπως βλέπουμε στο πίνακα των αποτελεσμάτων. Ακολουθούν οι γραφικές παραστάσεις των παραπάνω αποτελεσμάτων



Σχήμα 39. Μέσο One Way Delay Τρίτου Σεναρίου



Σχήμα 40. Μέσο Inter-arrival Jitter Τρίτου σεναρίου

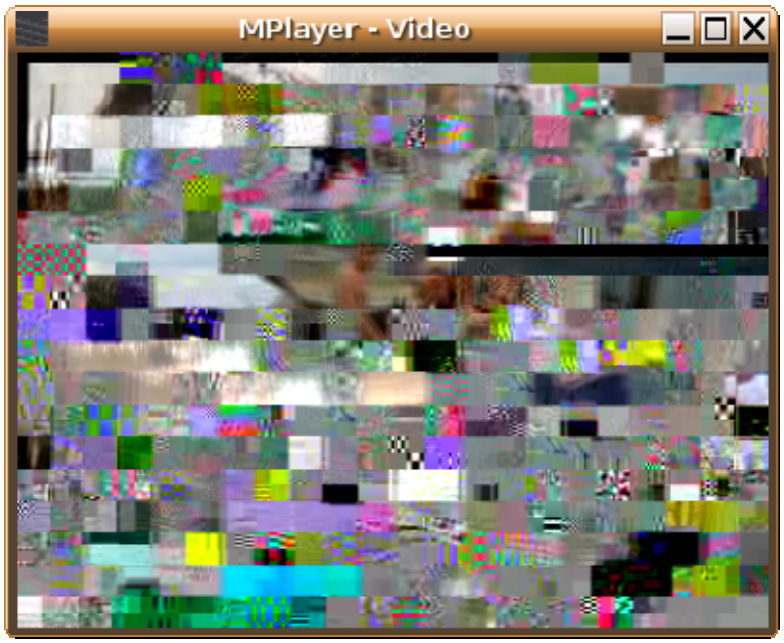


Σχήμα 41. Συνολικά Χαμένα Πακέτα Τρίτου σεναρίου

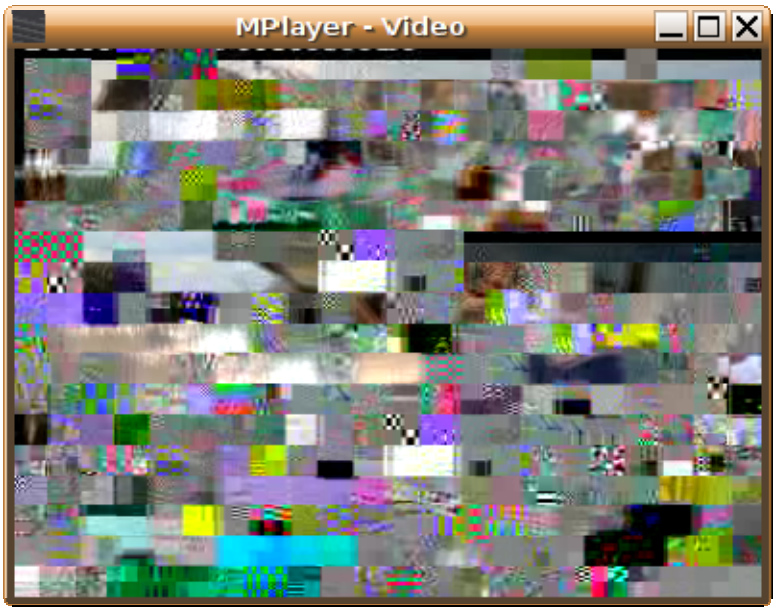
7.3.4. *Σενάριο 4ο:*

Α' Μέρος

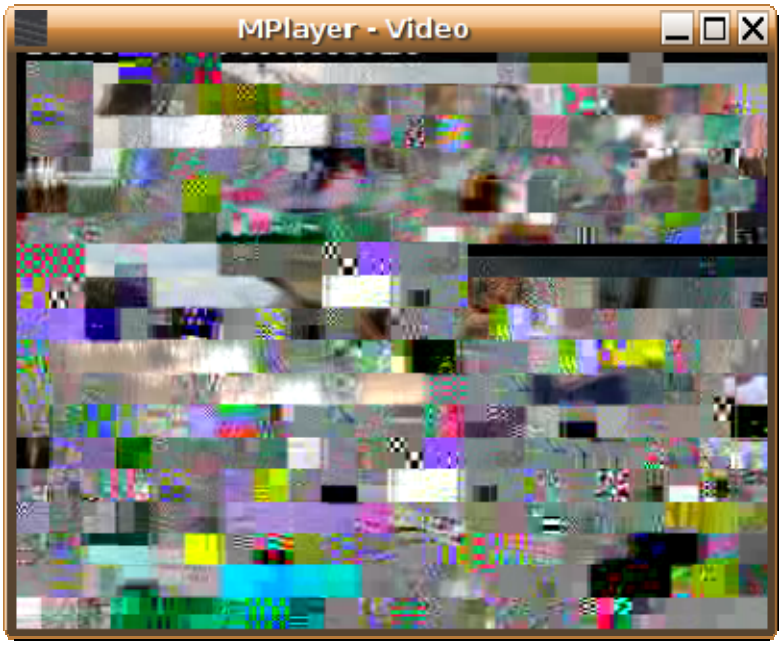
Στο Α' μέρος του τέταρτου και τελευταίου σεναρίου έχουμε διατηρήσει την Background κίνηση αλλά έχουμε το μηχανισμό του DiffServ ανενεργό. Έχοντας λοιπόν δημιουργήσει συνθήκες συμφόρησης στο δίκτυο μας, στέλνουμε τέσσερα πραγματικά βίντεο από τον Server στον Client. Τα αποτελέσματα φαίνονται παρακάτω



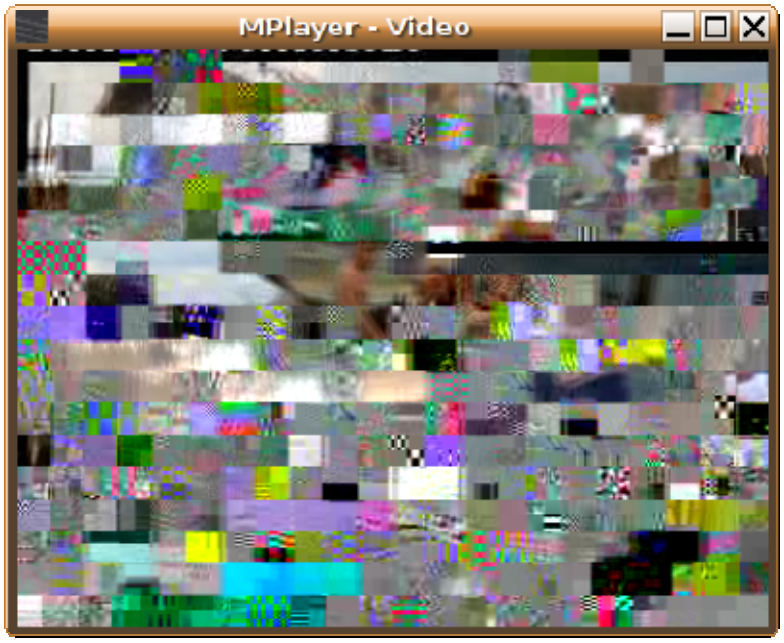
Σχήμα 42. DiffServ Απενεργοποιημένο-Πρώτος χρήστης



Σχήμα 43. DiffServ Απενεργοποιημένο-Δεύτερος χρήστης



Σχήμα 44. DiffServ Απενεργοποιημένο-Τρίτος χρήστης



Σχήμα 45. DiffServ Απενεργοποιημένο-Τέταρτος χρήστης

Από τα παραπάνω screenshot των βίντεο παρατηρούμε ότι όλα τα βίντεο παρουσιάζουν κάποια παραμόρφωση στην εικόνα τους και δεν είναι ευδιάκριτα. Η παραμόρφωση αυτή της ποιότητας της εικόνας των βίντεο οφείλεται στη συμφόρηση που έχουμε προκαλέσει στο δίκτυο. Βλέπουμε λοιπόν ότι τα αποτελέσματα του πρώτου μέρους του πειράματος αυτού έρχονται να επιβεβαιώσουν τα αποτελέσματα του δεύτερου σεναρίου της πειραματικής μας διαδικασίας όπου είχε πραγματοποιηθεί η ίδια μέτρηση αλλά με εικονικά πακέτα δεδομένων.

Β' Μέρος

Στο Β' μέρος του τέταρτου σεναρίου κρατώντας την Background κίνηση, ενεργοποιούμε τον μηχανισμό του DiffServ και στέλνουμε ξανά τα 4 βίντεο από τον Server στον Client. Τα αποτελέσματα που πήραμε φαίνονται παρακάτω

Σχήμα 46. DiffServ Ενεργοποιημένο-Πρώτος χρήστης

Σχήμα 47. DiffServ Ενεργοποιημένο-Δεύτερος χρήστης

Σχήμα 48. DiffServ Ενεργοποιημένο-Τρίτος χρήστης

Σχήμα 49. DiffServ Ενεργοποιημένο-Τέταρτος χρήστης

Από τα παραπάνω παρατηρούμε την βελτίωση η οποία επέρχεται στην ποιότητα των βίντεο. Παρατηρούμε λοιπόν ότι η ποιότητα της εικόνας και των τεσσάρων ροών είναι πάρα πολύ καλή έως άριστη παρότι υπάρχει και ο θόρυβος (background κίνηση), ο οποίος χάρις το μηχανισμό των διαφοροποιημένων υπηρεσιών (DiffServ) έχει εξαλυφθεί.

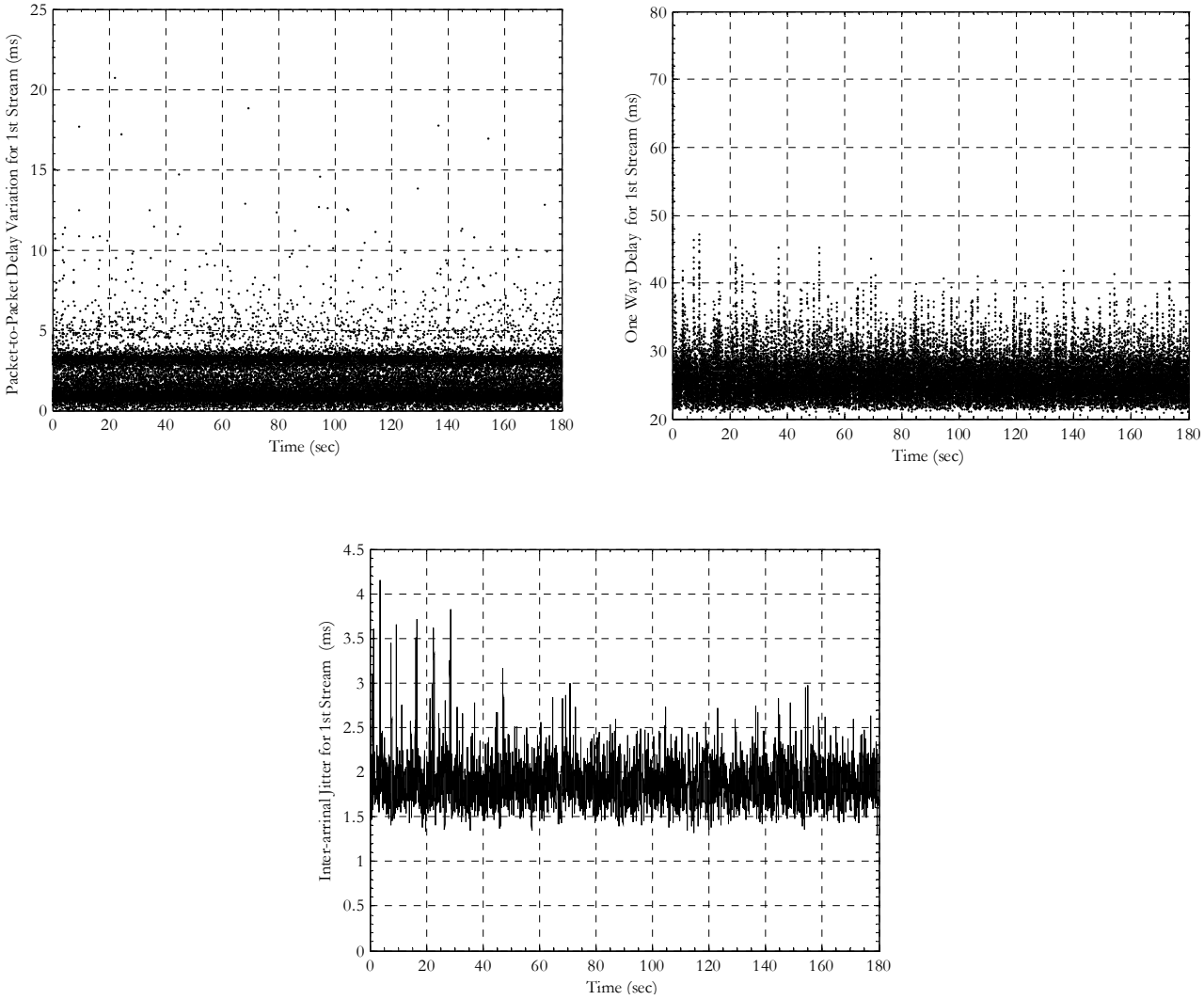
8. Συμπεράσματα

Από την ανάλυση όλων των αποτελεσμάτων της πειραματικής διαδικασίας που παρουσιάστηκαν παραπάνω καταλήγουμε στο συμπέρασμα ότι με την ενεργοποίηση του μηχανισμού διασφάλισης της ποιότητας των υπηρεσιών (Differentiation Service) η απόδοση του δικτύου έγινε καλύτερη καθώς δεσμευτικέ το απαιτούμενο εύρος ζώνης για κάθε μία από τις ροές εξασφαλίζοντας με αυτόν τον τρόπο την αξιόπιστη μετάδοση όλων των δεδομένων. Έτσι μας δίνεται η δυνατότητα να κατηγοριοποιούμε τις παρεχόμενες υπηρεσίες ανάλογα με το είδος της υπηρεσίας ή ανάλογα με τον χρήστη της υπηρεσίας .

9. Σχήματα Ανάλυσης Πειραματικών Μετρήσεων

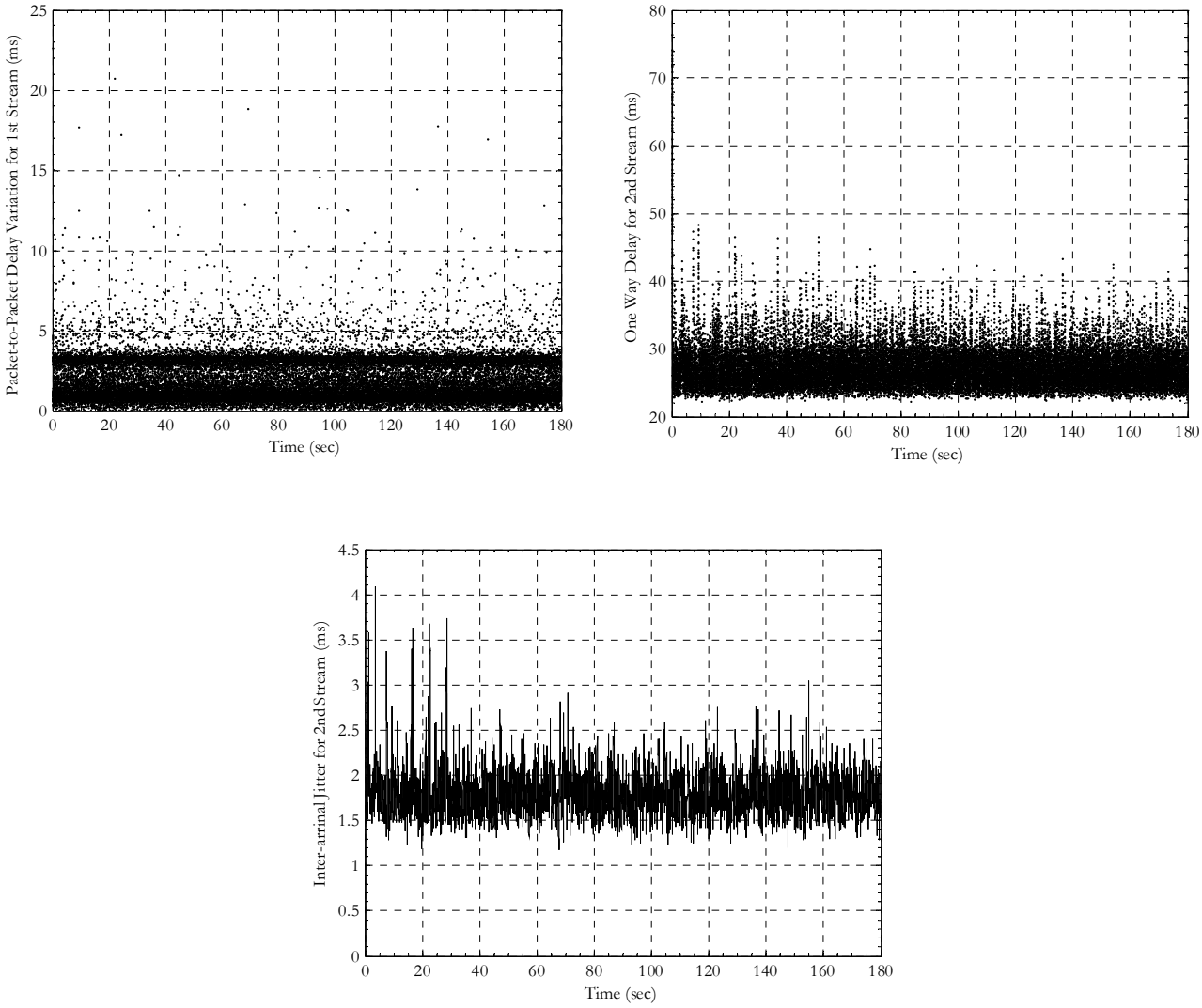
9.1. Σενάριο 1ο

9.1.1. *Πρώτη Ροή*



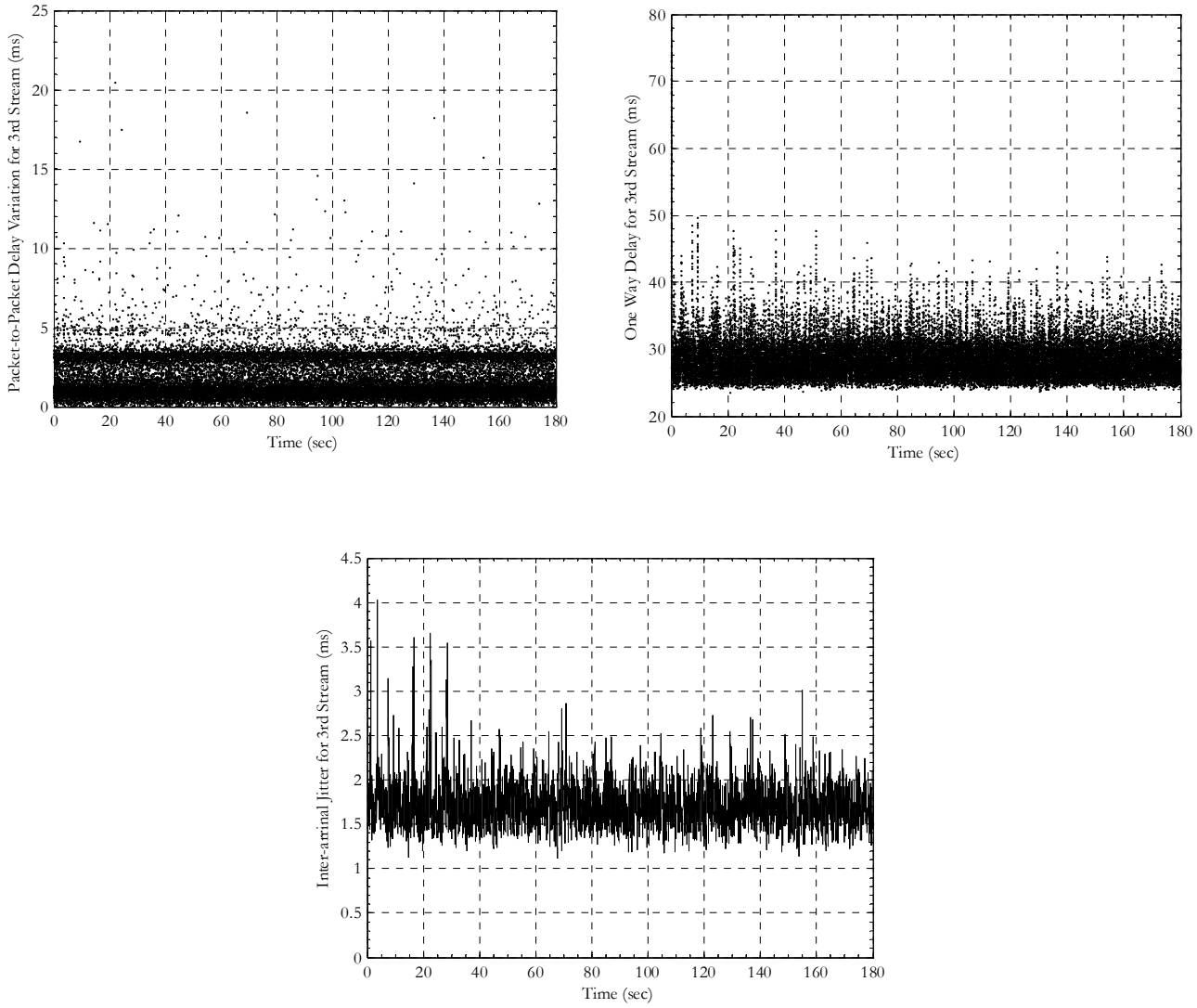
Σχήμα 50. Γραφικές Παραστάσεις Σεναρίου 1 –Ροή 1

9.1.2. *Δεύτερη Ροή*



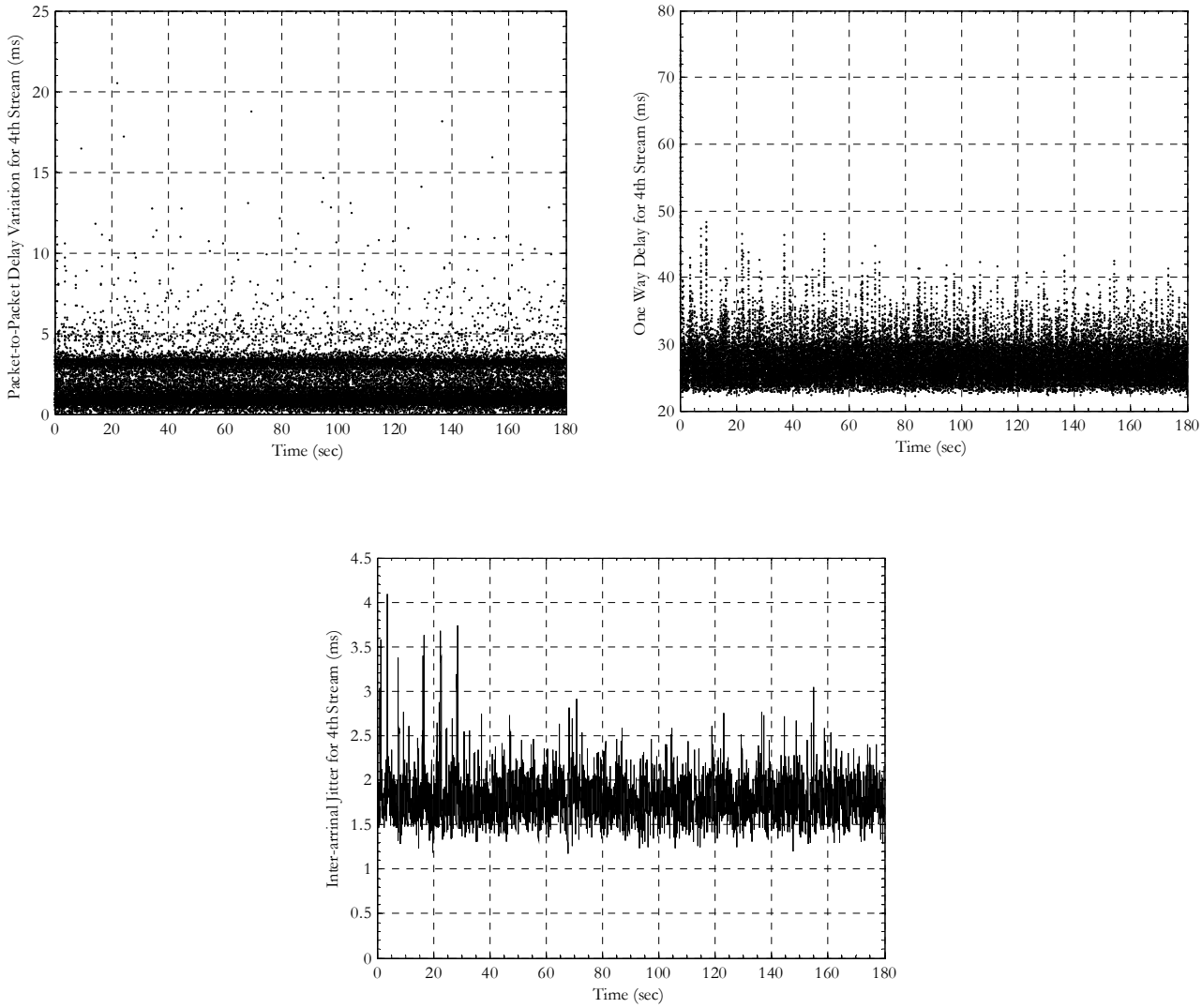
Σχήμα 51. Γραφικές Παραστάσεις Σεναρίου 1 –Ροή 2

9.1.3. *Τρίτη Ροή*



Σχήμα 52. Γραφικές Παραστάσεις Σεναρίου 1 –Ροή 3

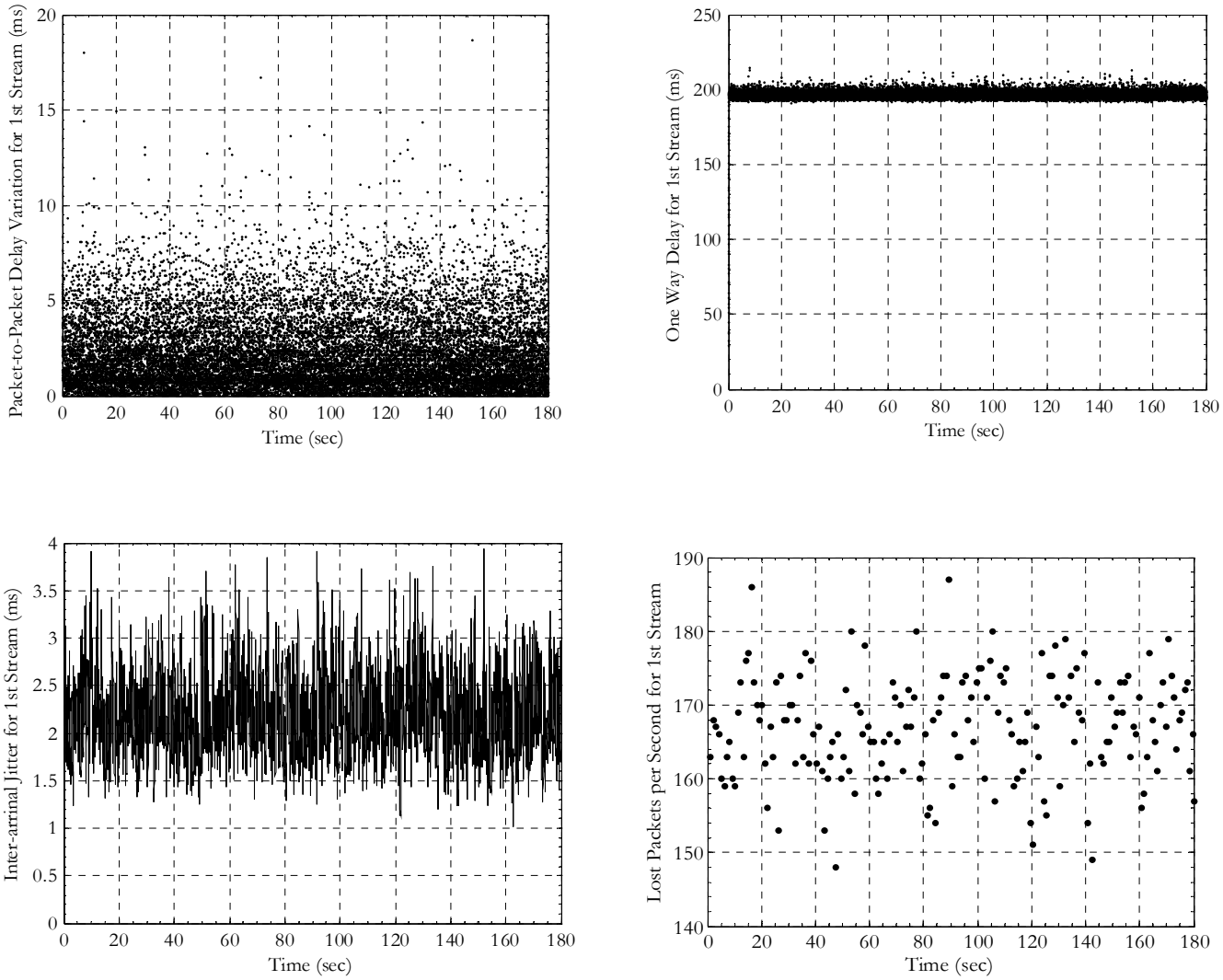
9.1.4. *Τέταρτη Ροή*



Σχήμα 53. Γραφικές Παραστάσεις Σεναρίου 1 –Ροή 4

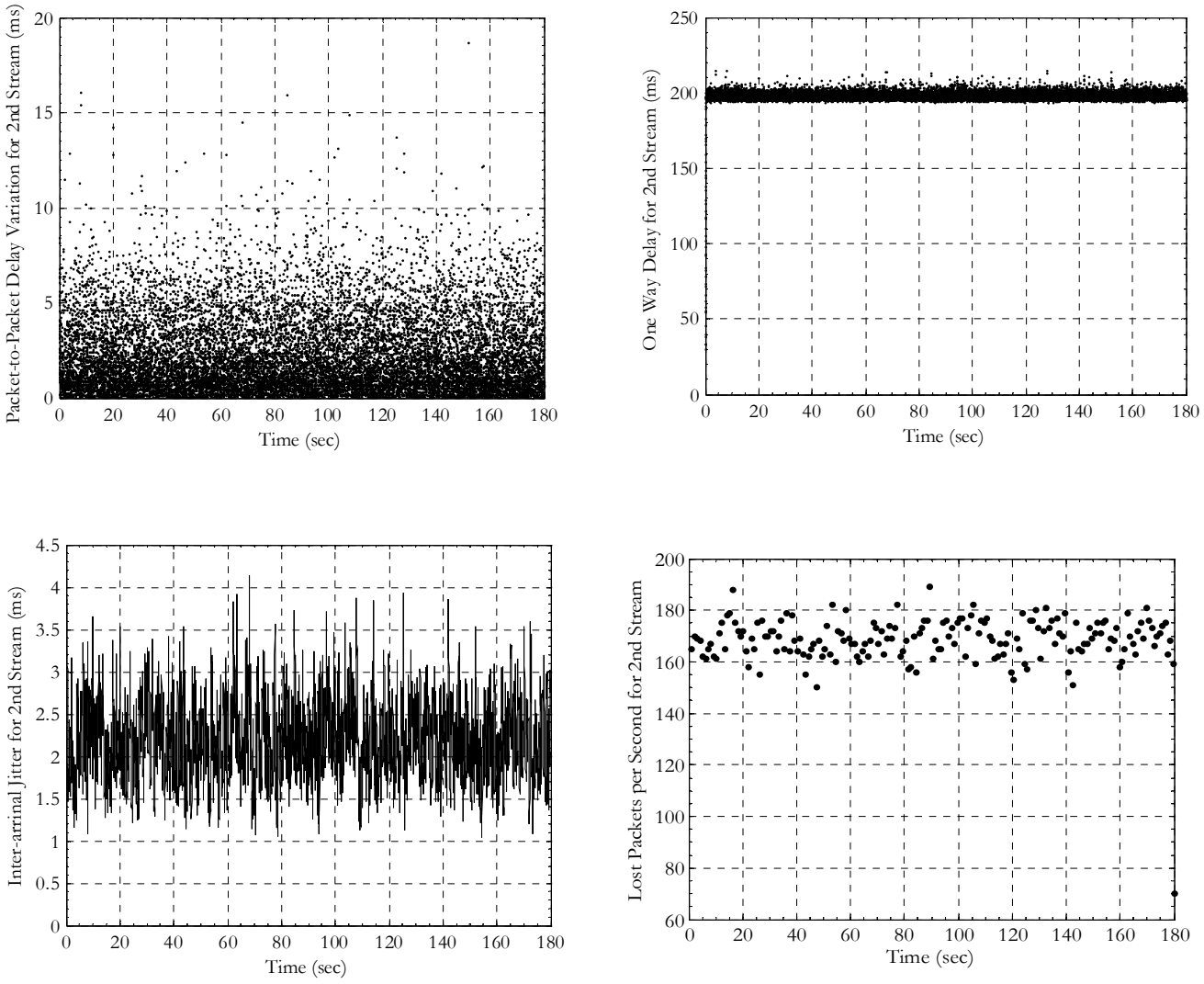
9.2. Σενάριο 2ο

9.2.1. *Πρώτη Ροή*



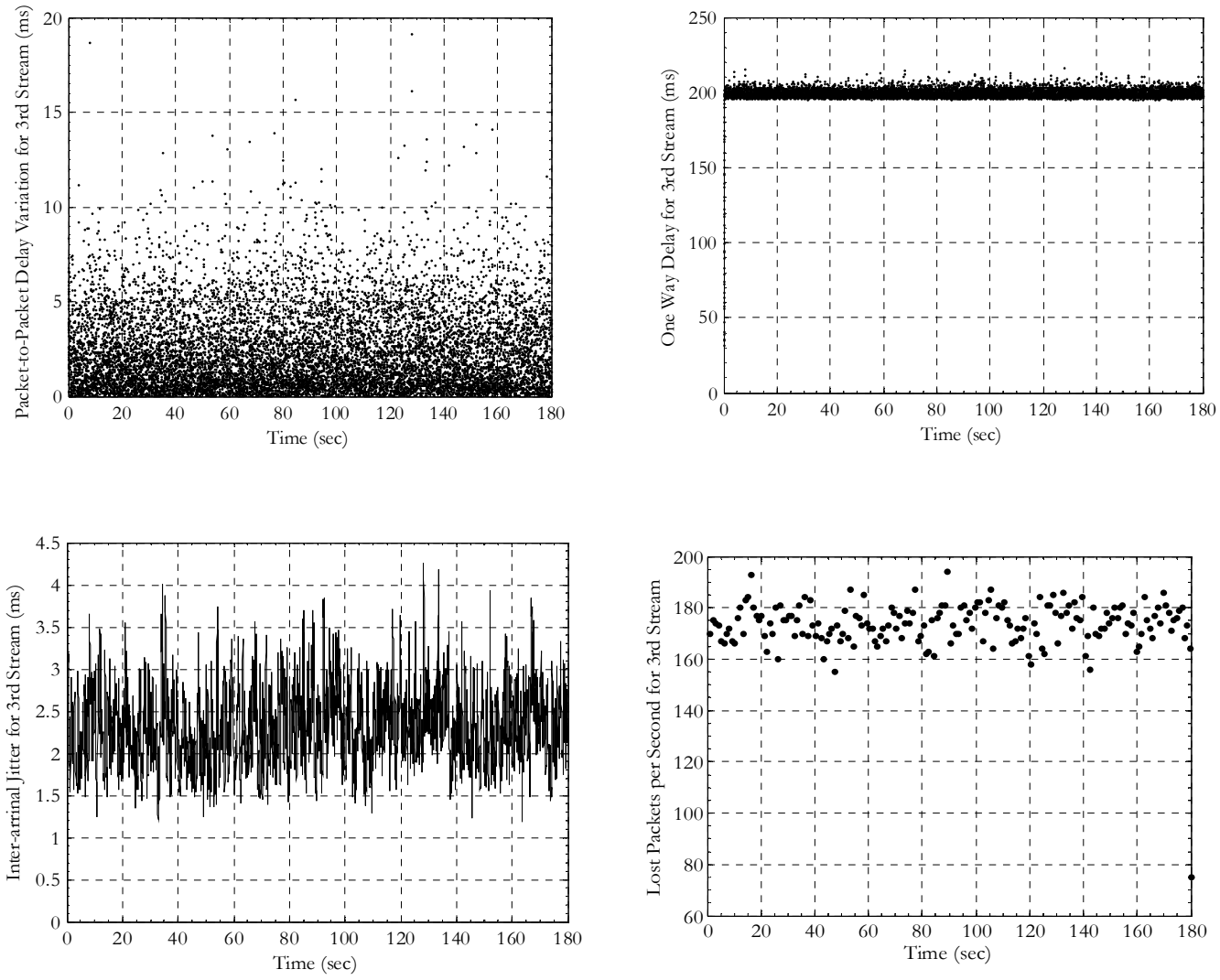
Σχήμα 54. Γραφικές Παραστάσεις Σεναρίου 2 –Ροή 1

9.2.2. *Δεύτερη Ροή*



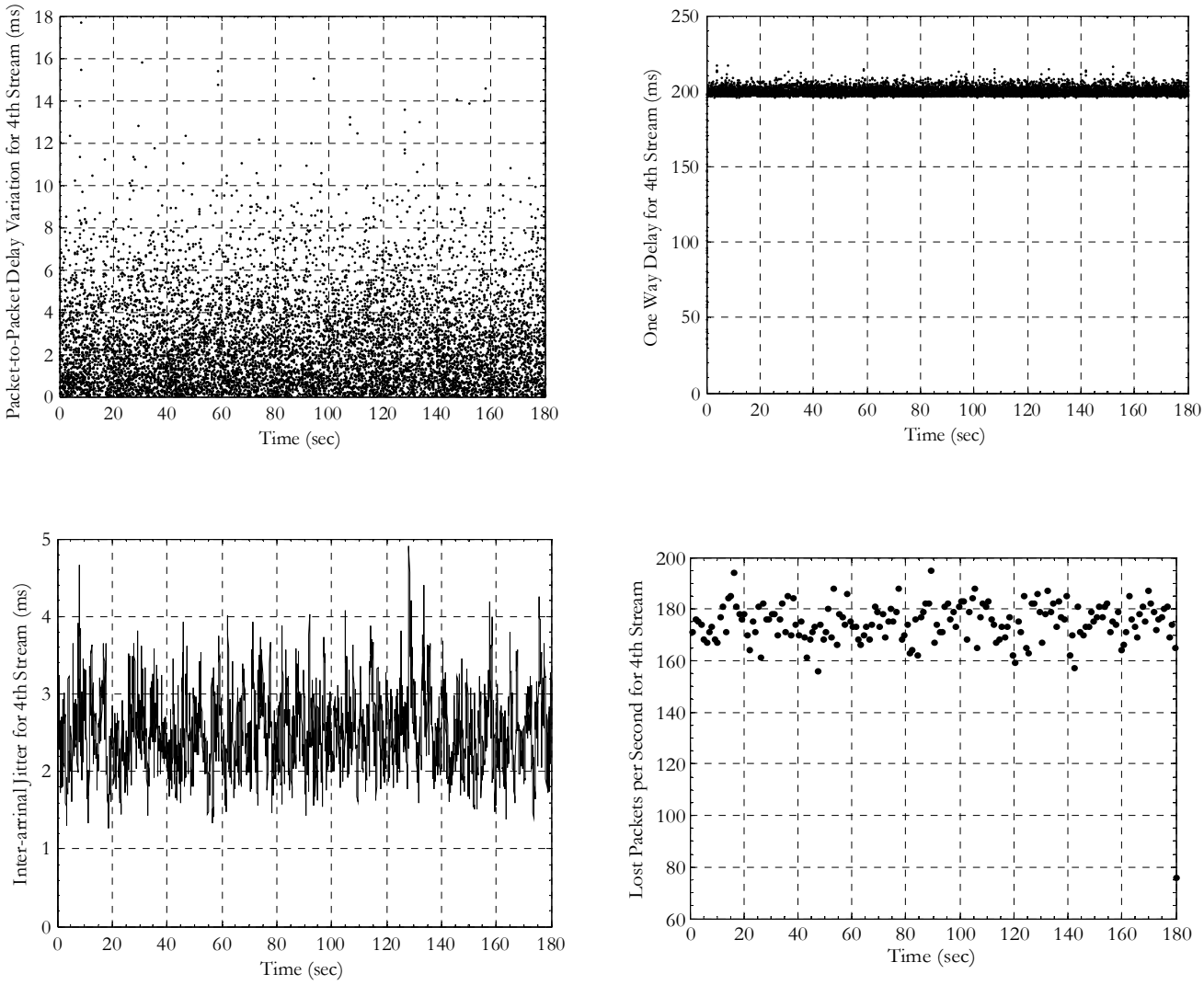
Σχήμα 55. Γραφικές Παραστάσεις Σεναρίου 2 –Ροή 2

9.2.3. *Τρίτη Ροή*



Σχήμα 56. Γραφικές Παραστάσεις Σεναρίου 2 –Ροή 3

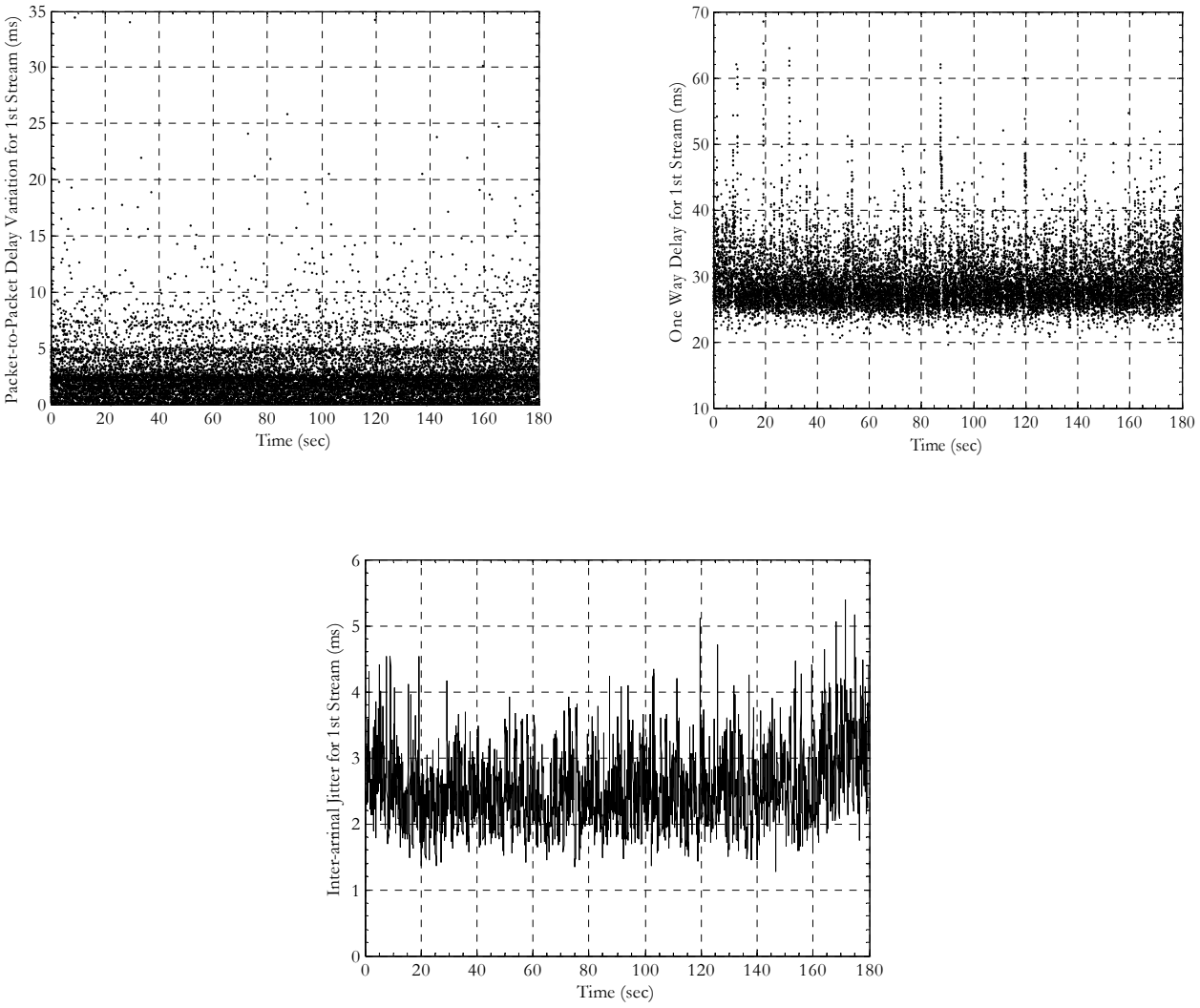
9.2.4. *Τέταρτη Ροή*



Σχήμα 57. Γραφικές Παραστάσεις Σεναρίου 2 –Ροή 4

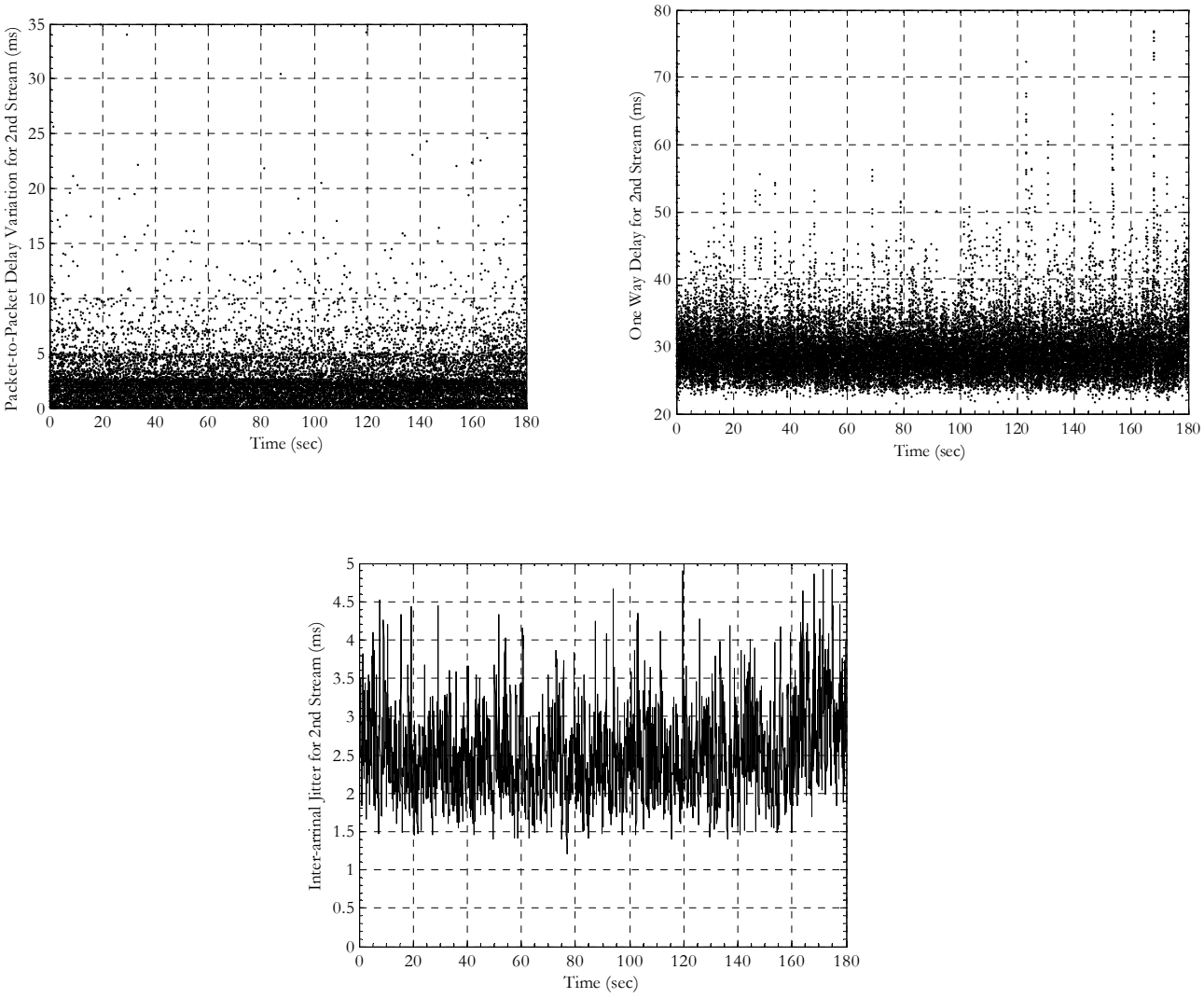
9.3. Σενάριο 3ο

9.3.1. *Πρώτη Ροή*



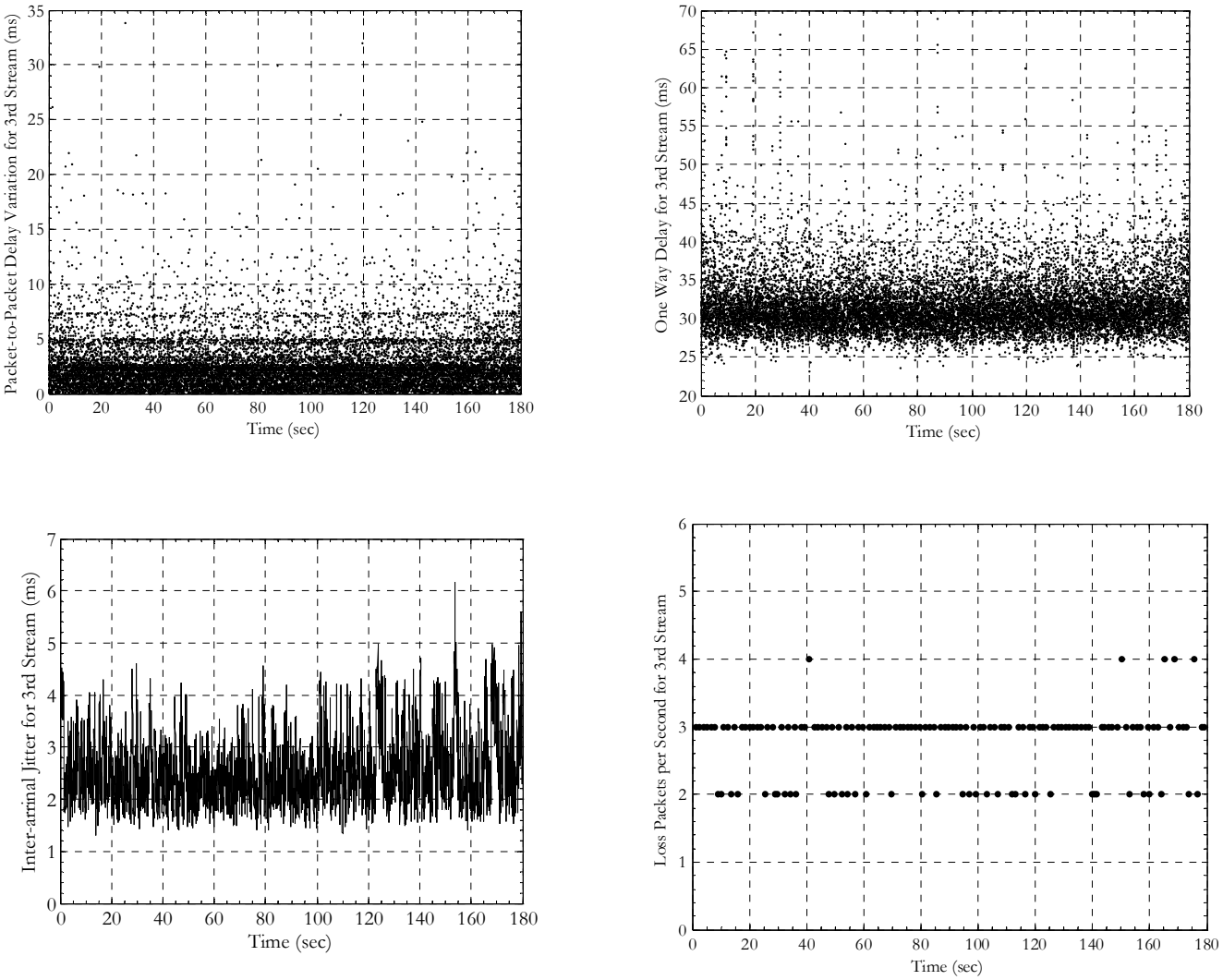
Σχήμα 58. Γραφικές Παραστάσεις Σεναρίου 3 –Ροή 1

9.3.2. *Δεύτερη Ροή*



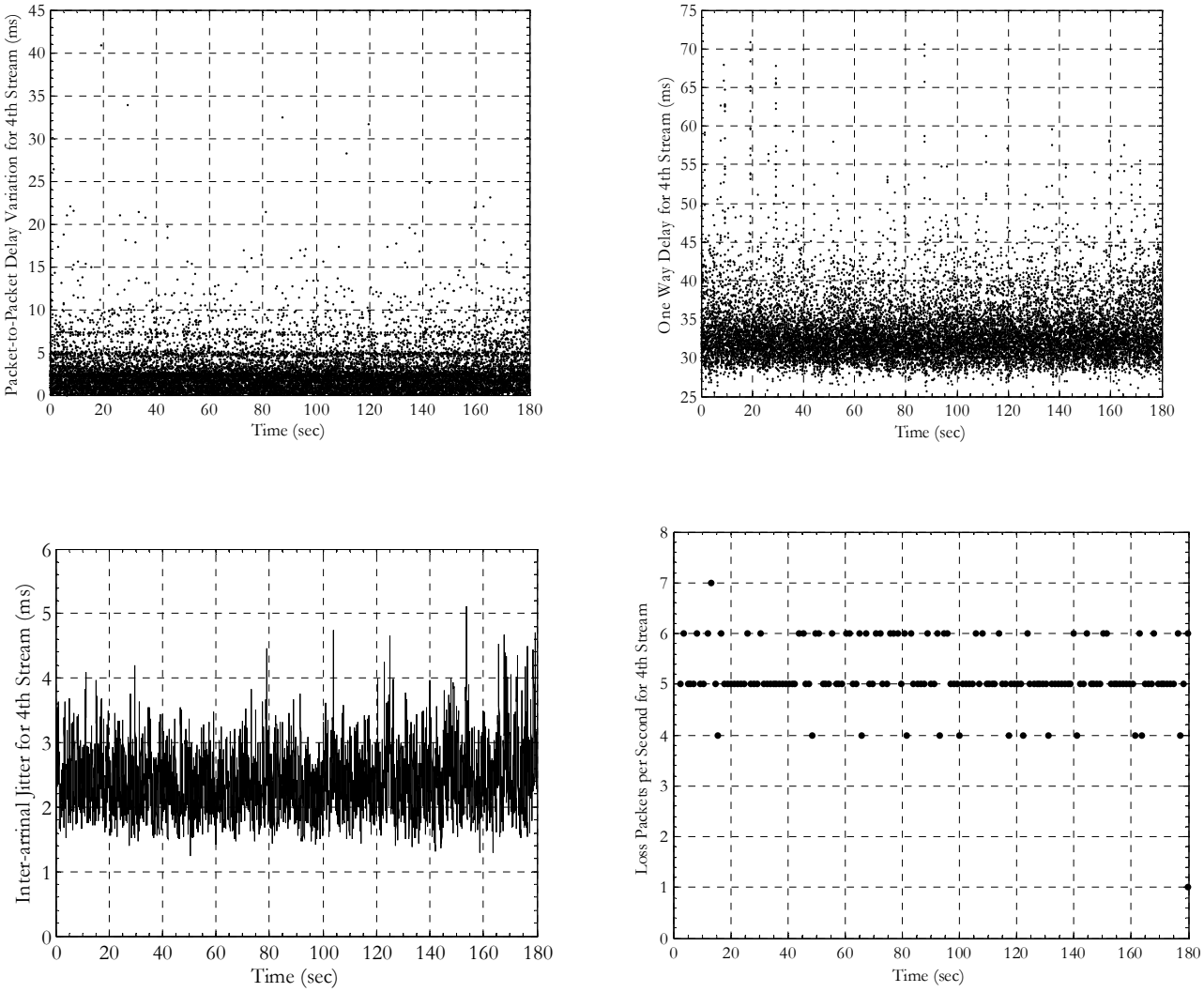
Σχήμα 59. Γραφικές Παραστάσεις Σεναρίου 3 –Ροή 2

9.3.3. *Τρίτη Ροή*



Σχήμα 60. Γραφικές Παραστάσεις Σεναρίου 3 –Ροή 3

9.3.4. *Τέταρτη Ροή*



Σχήμα 61. Γραφικές Παραστάσεις Σεναρίου 3 –Ροή 4

10. Βιβλιογραφία

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

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

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

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

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

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

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

[8]. 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11. 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

12. Παράρτημα

12.1. 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

12.2. Παράδειγμα Σύλληψης Κίνησης

- Παράδειγμα tcpdump

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

12.3. Προγράμματα Ανάλυσης Κίνησης

UDP κίνηση

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);
```

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);
```

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";
}
```

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";
```

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);
```

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";
```

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";
```

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";
```

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";
```

12.4. Matlab scripts

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');
```

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');
```

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');
```

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');
```