ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΚΡΗΤΗΣ

Πτυχιακή Εργασία

Μοντέλο προσομοίωσης της ULE ενθυλάκωσης

Σπουδαστής:
Νικόλαος Βορνιωτάκης

Εισηγητής:
Δρ. Ευάγγελος Πάλλης

31 Οκτωβρίου 2007

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μου εργασίας, η οποία υλοποιήθηκε στο Εργαστήριο Τηλεπικοινωνιών και Δικτύων (ΠΑΣΙΦΑΗ) του Α.Τ.Ε.Ι Κρήτης, θα ήθελα να ευχαριστήσω τους ανθρώπους οι οποίοι βοήθησαν στην περάτωση αυτής της εργασίας. Θα ήταν παράλειψη να μην αναφερθώ σε όλους εκείνους που συμπαραστάθηκαν σε αυτήν την προσπάθεια.

Κατά κύριο λόγο, οφείλω να ευχαριστήσω τον επιβλέποντά μου καθηγητή από το Α.Τ.Ε.Ι Κρήτης κ. Πάλλη Ευάγγελο για την, καθ'όλη την διάρκεια της πτυχιακής εργασίας, υποστήριξη και παροχή της υλικοτεχνικής υποδομής, απαραίτητης για μια μελέτη τέτοιου είδους.

Ιδιαίτερες ευχαριστίες οφείλονται στους εργαστηριακούς συνεργάτες του εργαστηρίου Τηλεπικοινωνιών και Δικτύων (ΠΑΣΙΦΑΗ) του Α.Τ.Ε.Ι. Κρήτης κ. Μαρκάκη Ευάγγελο, κ. Σιδέρη Ανάργυρο καθώς και στους υποψήφιους διδάκτορες κ. Μαστοράκη Γεώργιο και κ. Ξυλούρη Γεώργιο. Η συμβολή όλων ήταν καθοριστικής σημασίας για την επιτυχή έκβαση της εργασίας αυτής, καθώς οι πολύτιμες συμβουλές τους και η στήριξη που μου προσέφεραν όποτε χρειάστηκε, κατάφεραν να περιορίσουν σημαντικά τις όποιες δυσχέρειες μιας τέτοιας έρευνας.

Τέλος, ευχαριστώ όλους εκείνους που ήταν δίπλα μου σε όλη αυτή την προσπάθεια παρέχοντας απεριόριστη ψυχολογική υποστήριξη και κατανόηση.

Ηράκλειο, Σεπτέμβριος 2007

Βορνιωτάκης Νικόλαος

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Γενική περιγραφή

Το θέμα με το οποίο ασχολείται η συγκεκριμένη πτυχιακή εργασία, αφορά τη μελέτη, τη σχεδίαση, την υλοποίηση και την αξιολόγηση ενός μοντέλου ULE ενθυλάκωσης με την χρήση πλατφόρμας επίγειας ψηφιακής τηλεόρασης. Στα πλαίσια αυτά ο σχεδιασμός και η μελέτη ενός τέτοιου μοντέλου στηρίζεται στην τεχνολογία του Digital Video Broadcasting - Terrestrial (DVB-T) και στις σύγχρονες τεχνικές ενθυλάκωσης που έχουν προταθεί (ULE). Για τον σκοπό αυτό θα υλοποιηθεί μία δικτυακή εφαρμογή η οποία θα αναλάβει την επέκταση ενός κλασσικού ethernet δικτύου στα όρια της περιοχής κάλυψης της ψηφιακής τηλεόρασης κάνοντας χρήση των τεχνικών της ULE ενθυλάκωσης.

1.2 Στόχοι

Βασικός στόχος της πτυχιακής εργασίας αυτής είναι η ανάπτυξη της εφαρμογής η οποία, όπως ήδη αναφέρθηκε, θα διασυνδέει ένα κλασικό δίκτυο με ένα δίκτυο εκπομπής όπως αυτό της ψηφιακής τηλεόρασης. Στα πλαίσια αυτού του στόχου θα δημιουργηθεί και ένα πρότυπο δίκτυο, που θα προσφέρει αυτήν την λειτουργικότητα, ώστε να είναι δυνατή η αξιολόγηση της προτεινόμενης αρχιτεκτονικής.

1.3 Δομή

Σε μια προσπάθεια για την σφαιρική κάλυψη των αρχών που διέπουν ένα τέτοιο σύστημα έχουν εισαχθεί τα κεφάλαια 2, 3 και 4.

Στο κεφάλαιο 2 γίνεται η παρουσίαση του προτύπου MPEG το οποίο αποτελεί σημαντικό κομμάτι της έρευνας αφού όλα τα δεδομένα που διακινούνται σε ένα κανάλι ψηφιακής τηλεόρασης είναι στην μορφή MPEG2-TS συρμών δεδομένων.

Στο κεφάλαιο 3 γίνεται η παρουσίαση του προτύπου DVB-T.

Στο κεφάλαιο 4 παρουσιάζεται και αναλύεται η μέθοδος ενθυλάκωσης ULE. Σε αυτήν στηρίζεται η μελέτη αυτή.

Στο κεφάλαιο 5 γίνεται ο σχεδιασμός του συστήματος και αναλύονται οι απαιτήσεις από πλευράς υλικού για τα επιμέρους στοιχεία του. Το πρόβλημα βολιδοσκοπείται από την πλευρά της εφαρμογής, την πλευρά του υλικού και την πλευρά του δικτύου.

Στο κεφάλαιο 6 αναλύεται η διαδικασία για την υλοποίηση των επιμέρους στοιχείων του συστήματος.

Στο κεφάλαιο 7 περιγράφεται η λογική και η διαδικασία των μετρήσεων για την αξιολόγηση του δικτύου που υλοποιήθηκε.

Στο κεφάλαιο 8 περιγράφεται η διαδικασία που ακολουθήθηκε για να στηθεί το σύστημα σε ένα Linux μηχάνημα.

Στο κεφάλαιο 9 παρουσιάζονται και σχολιάζονται τα αποτελέσματα των μετρήσεων.

Στο κεφάλαιο 10 αναλύονται τα συμπεράσματα και γίνονται προτάσεις για μελλοντικές βελτιώσεις.

Κεφάλαιο 2

Το πρότυπο MPEG

Σε αυτό το κεφάλαιο θα παρουσιαστεί το πρωτοκόλλο MPEG. Το πρωτόκολλο αυτό αποτελείται από πολλά ξεχωριστά κομμάτια και καθορίζει μεταξύ άλλων πρότυπα συμπίεσης εικόνας και ήχου, πρότυπα πολύπλεξης ρευμάτων δεδομένων πραγματικού χρόνου, πρότυπα επικοινωνίας εξυπηρετητή και πελατών κ.α. Ιδιαίτερη έμφαση θα δοθεί στο πρότυπο MPEG-TS, το οποίο χρησιμοποιείται για την πολύπλεξη και την πακετοποίηση των δεδομένων που πρόκειται να μεταφερθούν. Πρόκειται για ένα πρότυπο ειδικά σχεδιασμένο για δικτυακή χρήση, το οποίο είναι αρκετά ανθεκτικό σε σφάλματα μετάδοσης και χρησιμοποιείται ευρέως στις μεταδόσεις δορυφορικής ψηφιακής τηλεόρασης.

2.1 Moving Picture Experts Group

Το ακρωνύμιο MPEG (Moving Picture Experts Group) είναι τα αρχικά μιας ομάδας εργασίας του International Organization for Standardization (ISO) και του International Electrotechnical Commission (IEC), η οποία είναι επιφορτισμένη με την ανάπτυξη προτύπων κωδικοποίησης για ψηφιακό βίντεο και ήχο. Η ομάδα ξεκίνησε τις εργασίες της το 1988 και σήμερα αριθμεί 350 μέλη τόσο από το χώρο της βιομηχανίας όσο και από τον ακαδημαϊκό χώρο. Πρόκειται για μία ευρέως αποδεκτή ομάδα εργασίας και τα πρωτόκολλα που έχει μέχρι στιγμής εκδώσει χρησιμοποιούνται ευρέως από τη βιομηχανία. Συγκεκριμένα, μέχρι σήμερα η ομάδα έχει εκδώσει 6 πρότυπα:

1. MPEG-1: Το πρώτο πρωτόκολλο που εκδόθηκε από την επιτροπή ήταν το MPEG-1 και εκδόθηκε το 1992 στο Λονδίνο. Πρόκειται για την πρώτη έκδοση της ομάδας και αφορά την ανάπτυξη προτύπων συμπίεσης ήχου και εικόνας για την αποθήκευση μέσων με ρυθμό μέχρι 1.5 Mbit/sec και ποιότητα που αντιστοιχεί στο πρότυπο VHS. Το πρωτόκολλο είναι χωρισμένο σε πέντε μέρη. Στο πρώτο μέρος παρουσιάζει ένα πρότυπο, το Program Stream(PS), το οποίο επιτρέπει το συγχρονισμό και την πολύπλεξη ρευμάτων ήχου και εικόνας. Στο δεύτερο μέρος το πρότυπο καθορίζει έναν αλγόριθμο συμπίεσης βίντεο σε ποιότητα VHS και απόδοση 1,15 Mbit/s χρησιμοποιώντας των αλγόριθμο συμπίεσης jpeg για το κάθε frame του video. Στο τρίτο μέρος του πρωτοκόλλου περιγράφεται μια νέα τεχνική συμπίεσης ήχου, το MP3 το οποίο είναι από τα πιο διαδεδομένα πρότυπα συμπίεσης ήχου σήμερα. Το πρότυπο αυτό βασίζεται τόσο στη τεχνική συμπίεση δεδομένων PCM όσο και σε κάποιες παρατηρήσεις της επιστήμης της ψυχοακουστικής. Στο τέταρτο μέρος του πρωτοκόλλου περιγράφονται κάποιες μέθοδοι ελέγχου συμβατότητα ενός συστήματος με το πρωτόκολλο MPEG- 1. Τέλος στο πέμπτο μέρος του πρωτόκολλο παρουσιάζεται ένα σύστημα MPEG-1 υλοποιημένο στη γλώσσα προγραμματισμού C. Το πρωτόκολλο MPEG–1 αν και σήμερα θεωρείται ξεπερασμένο για την κωδικοποίηση βίντεο, κάποια μέρη του χρησιμοποιούνται σε μεγάλο βαθμό σε καθημερινές εφαρμογές όπως στη δημιουργία video-cd και στη συμπίεση αρχείων ήχου.

2. MPEG-2: Το δεύτερο πρωτόκολλο που επικύρωσε η επιτροπή ήταν το MPEG-2. Η πρώτη επίσημη έκδοσή του ήταν το 1994. Το πρωτόκολλο αυτό μας ενδιαφέρει καθώς χρησιμοποιείται σε μεγάλο βαθμό στην εφαρμογή μας. Το MPEG–2 συνολικά αποτελείται από 11 μέρη, με τα πέντε πρώτα να έχουν ίδιο θέμα με το MPEG-1. Οι κυριότερες διαφορές μεταξύ των δύο πρωτοκόλλων εντοπίζονται σε 2 κυρίως σημεία στα 5 πρώτα κεφάλαια. Στο πρώτο μέρος του, το πρωτόκολλο MPEG-2 παρουσιάζει ένα νέο πρότυπο πολύπλεξης και πακετοποίησης των διαφόρων ρευμάτων ενός μέσου. Το πρότυπο αυτό ονομάζεται Transport Stream (TS) και στόχο έχει την αποδοτικότερη αποστολή μέσων μέσω του διαδικτύου,

ενώ παράλληλα επεκτείνεται το υπάρχον πρότυπο Program Stream (PS) επιτρέποντας την αποτελεσματικότερη αποθήκευση ρευμάτων. Στο δεύτερο κεφάλαιο του MPEG–2 που αφορά τα πρότυπα συμπίεσης βίντεο, το πρωτόκολλο καθορίζει μια προέκταση του υπάρχοντος προτύπου που επιτρέπει την υποστήριξη διαπλεκόμενων εικόνων (interlaced images) δυνατότητα η οποία επιτρέπει την αποδοτικότερη συμπίεση. Επίσης σε αυτήν την έκδοση η ερευνητική ομάδα παρουσίασε κάποια νέα κεφάλαια στο πρωτόκολλο. Στο έκτο και δέκατο κεφάλαιο παρουσιάζεται το πρωτόκολλο DSM-CC (Digital Storage Media Command and Control), το οποίο επιτρέπει τη δημιουργία συνεδριών μεταξύ ενός πελάτη και ενός εξυπηρετητή μέσων, καθώς και τον απομακρυσμένο έλεγχο των εξυπηρετητών. Στο έβδομο κεφάλαιο παρουσιάστηκε ένα νέο πρότυπο κωδικοποίησης ήχου, το ACC, το οποίο επιτρέπει την κωδικοποίηση πολυκάναλων ρευμάτων ήχου. Τέλος, στο όγδοο κεφάλαιο το πρωτόκολλο καθορίζει μεθόδους διαχείρισης εικόνων βίντεο για τις οποίες το χρώμα αναπαρίσταται με περισσότερα από 8 bits ενώ στο ένατο κεφάλαιο παρουσιάζει μια διεπαφή ανάμεσα στο Transport Stream και στον αποκωδικοποιητή, που στόχο έχει να καλύψει τις ανάγκες του DSM-CC για τη δημιουργία συνεδριών.

3. MPEG-4: Η τρίτη έκδοση πρωτοκόλλου από την ομάδα MPEG ήταν το MPEG-4. Το πρωτόκολλο αυτό έχει συνολικά 21 κεφάλαια, διατηρώντας βέβαια τους τίτλους των 6 πρώτων ίδιους με αυτούς τους MPEG-2. Το πρωτόκολλο αυτό παρουσιάστηκε επίσημα το 1998 και είχε ως τίτλο "Κωδικοποίηση οπτικοακουστικών αντικειμένων". Σε γενικές γραμμές, η έκδοση αυτή του πρωτοκόλλου είχε ως πρωταρχικό στόχο την προσθήκη της δυνατότητας κωδικοποίησης μεμονωμένων αντικειμένων σε μία παρουσίαση στα ήδη υπάρχοντα πρωτόκολλα. Μερικές από τις επεκτάσεις που υποστηρίζονται είναι η δυνατότητα διαχείρισης πολλαπλών ρευμάτων ήχου με διαφορετική κωδικοποίηση του καθενός και διαφορετικής χρηστικότητα καθώς και η δυνατότητα διαχείρισης 3D αντικειμένων. Επίσης προστέθηκε ένας μηχανισμός διαχείρισης και προστασίας δικαιωμάτων, που προκύπτουν από μεμονωμένα αντικείμενα ενός αρχείου. Τέλος, το πρωτόκολλο παρουσίασε και κάποιους νεότερους και αποδοτικότερους αλγορίθμους

συμπίεσης ήχου και βίντεο (AVC). Το MPEG-4 μπορεί ποια να καλύψει ρυθμούς μετάδοσης δεδομένων μέχρι 1 Gbit/s.

4. MPEG-7: Το πρωτόκολλο αυτό είναι το τέταρτο πρωτόκολλο που εκδόθηκε από την ομάδα. Η μελέτη του ξεκίνησε τον Απρίλιο του 1997 και ακόμα βρίσκεται υπό ανάπτυξη. Αποτελείται συνολικά από 11 κεφάλαια. Το MPEG-7 καθορίζει κάποιες νέες δυνατότητες για τα παλιότερα πρωτόκολλα που στόχο έχουν τον εμπλουτισμό τους με δυνατότητες αποθήκευσης μεταπληροφοριών που περιγράφουν τα δεδομένα του αρχείου. Τα δεδομένα αυτά υπακούν σε κάποιο δεδομένο σχήμα και μπορούν να αναφέρονται σε εικόνες, σε ήχο και σε απλό κείμενο.

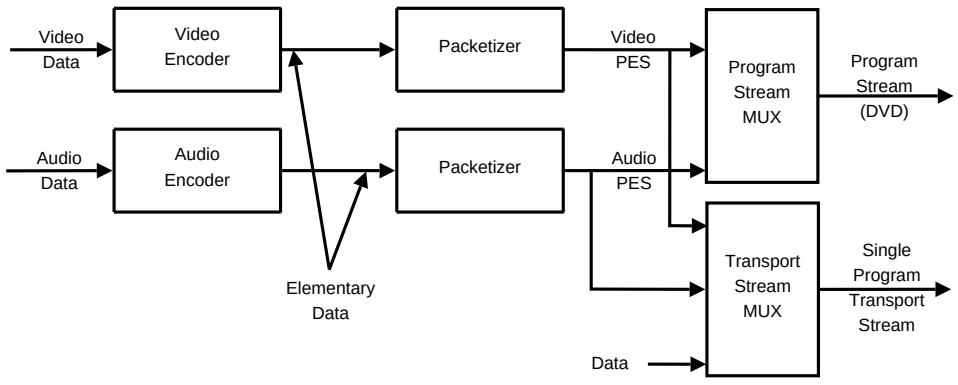
5. MPEG-21: Το πρωτόκολλο MPEG-21 είναι το όραμα της ομάδας για το μέλλον στον τομέα των πολυμέσων. Η ανάπτυξη του ξεκίνησε το Μάιο – Ιούνιο του 2000. Το MPEG-21 είναι ένα ανοιχτό πρωτόκολλο που στόχο έχει την ανάπτυξη ενός πλαισίου για τον έλεγχο των δικαιωμάτων σε εφαρμογές πολυμέσων. Ποιο συγκεκριμένα το πρωτόκολλο καθορίζει ένα πρότυπο για μια "Γλώσσα έκφρασης δικαιωμάτων" (Rights Expression Language) με την οποία μπορεί να εμπλουτίζεται κάθε αρχείο πολυμέσων και η οποία καθορίζει την άδεια, τα δικαιώματα και τους περιορισμούς που επιβάλει ο δημιουργός του περιεχομένου στους χρήστες. Σε γενικές γραμμές το MPEG-21 προσπαθεί να εισάγει τις έννοιες της δημιουργίας, πρόσβασης, της ανταλλαγής και διαχείρισης ηλεκτρονικών δικαιωμάτων στο σημερινό πλαίσιο ανάπτυξης πολυμέσων με τρόπο ο οποίος να μη γίνεται αντιληπτός άμεσα από τους χρήστες. Η ανάπτυξη του προτύπου αυτού έχει ως κύριο στόχο τη καταπολέμηση της πειρατείας η οποία αναπτύσσεται σε μεγάλο βαθμό από τα δίκτυα Peer to Peer. Το πρωτόκολλο αυτό βρίσκεται ακόμα υπό ανάπτυξη και στοχεύει να αλλάξει κατά πολύ τον τρόπο που αντιλαμβανόμαστε τις εφαρμογές πολυμέσων.

Η ομάδα ανάπτυξης του MPEG αποτελείται τόσο από ακαδημαϊκούς εγνωσμένης αξίας όσο και από εκπροσώπους της βιομηχανίας. Η ανάπτυξη των πρωτοκόλλων γίνεται μέσα από ανοιχτές προσκλήσεις για προτάσεις, οι οποίες γίνονται μόνο προς τα μέλη της ομάδας. Με βάση τις προτάσεις, που γίνονται, οι

ομάδες που έχουν αναλάβει την ανάπτυξη κάποιου προτύπου προτείνουν κάποιο αρχικό κείμενο το οποίο μέσα από συζητήσεις και ψηφοφορίες καταλήγει στην τελική δημοσίευση. Η ανάπτυξη βέβαια των προτύπων του MPEG δέχεται από πολλούς κριτική, καθώς είναι αρκετά κλειστή διαδικασία και τα τελικά πρότυπα δεν είναι ελεύθερα σε όλους τους χρήστες, αναγκάζοντας τους ενδιαφερόμενους να καταβάλουν κάποιο ποσό στον ISO για να λάβουν αντίγραφα των προτύπων.

2.2 MPEG-2

Το MPEG–2 όπως είδαμε και παραπάνω καθορίζει ένα πρότυπο για την πολύπλεξη ενός ή περισσοτέρων στοιχειωδών ρευμάτων ήχου και βίντεο, καθώς και άλλων δεδομένων, μιας παρουσίασης σε ένα ή περισσότερα ρεύματα κατάλληλα για αποθήκευση ή μετάδοση [4]. Οι λεπτομέρειες για τον τρόπο λειτουργίας της πολύπλεξης του MPEG–2 καθορίζονται στο πρώτο κεφάλαιο του πρωτοκόλλου. Η χρήση του προτύπου MPEG–2 για την πολύπλεξης δε απαιτεί η κωδικοποίηση ή η συμπίεση των δεδομένων των ρευμάτων να έχει γίνει σύμφωνα με το πρότυπο του MPEG–2 αλλά μπορεί να έχει χρησιμοποιηθεί οποιοσδήποτε συμβατός αλγόριθμος συμπίεσης. Για παράδειμα, η συμπίεση του ρεύματος βίντεο μπορεί να γίνει με βάση το πρότυπο H.264 το οποίο είναι νεότερο του MPEG–2 και μας δίνει καλύτερη ποιότητα εικόνας σε χαμηλότερο ρυθμό μετάδοσης, με μεγαλύτερες όμως από την άλλη πλευρά απαιτήσεις σε CPU.



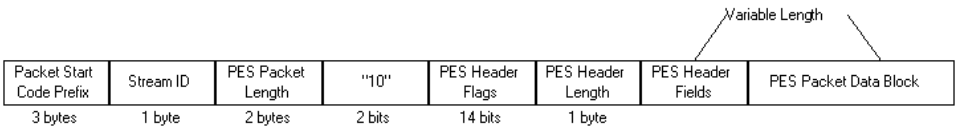
Σχήμα 2.1: Ροές βίντεο, ήχου και δεδομένων συνδυάζονται για να σχηματίσουν συρμούς PS και TS.

Η διαδικασία πολύπλεξης των επί μέρους ρευμάτων ακολουθεί μια στρωματική προσέγγιση δύο επιπέδων και παρουσιάζεται διαγραμματικά στην εικόνα 2.1.

2.2.1 Πρώτη φάση πολύπλεξης

Στην πρώτη φάση της διαδικασίας πολύπλεξης του MPEG–2, το σύστημα παράγει ένα ρεύμα πακέτων που ονομάζεται συσκευασμένο στοιχειώδες ρεύμα (Packetized Elementary Stream) για κάθε στοιχειώδες ρεύμα δεδομένων (Elementary Stream). Σε γενικές γραμμές ένα στοιχειώδες ρεύμα είναι ένα ρεύμα συμπιεσμένου ήχου, εικόνας ή και απλών δεδομένων. Συνήθως, προκύπτει από την έξοδο κάποιου κωδικοποιητή και τα δεδομένα είναι αποκλειστικά ενός τύπου. Τα Στοιχειώδη ρεύματα που αφορούν ήχο ή εικόνα γενικά είναι οργανωμένα σε μονάδες πρόσβασης (access units). Μία μονάδα πρόσβασης αναπαριστά μία θεμελιώδη μονάδα κωδικοποίησης. Κάτι τέτοιο για το ρεύμα βίντεο μπορεί να είναι ένα πλαίσιο, ενώ για το ρεύμα ήχου μπορεί να είναι κάποια κομμάτι δεδομένης διάρκειας.

Κάθε Elementary stream καταλήγει σε ένα MPEG–2 επεξεργαστή ο οποίος συγκεντρώνει τα δεδομένα και δημιουργεί το συσκευασμένο στοιχειώδες ρεύμα (Packetized Elementary Stream). Το ρεύμα PES είναι μία λογική κατασκευή που είναι χρήσιμη μόνο στις υλοποιήσεις του πρωτοκόλλου και δεν μπορεί να χρησιμοποιηθεί για τη μετάδοση ή αποθήκευση ενός μέσου. Το κάθε πακέτο PES, που προκύπτει από τον επεξεργαστή, μπορεί να ποικίλει ως προς το μέγεθος αλλά δε μπορεί να είναι μεγαλύτερο από 65 KB, ενώ περιλαμβάνει και μία επικεφαλίδα 8 byte τουλάχιστον. Η δομή ενός PES πακέτου εμφανίζεται στην εικόνα 2.2.



Σχήμα 2.2: Η δομή ενός PES πακέτου.

2.2.2 Δεύτερη φάση πολύπλεξης

Στη δεύτερη φάση της διαδικασίας πολύπλεξης τα πακέτα PES συνδυάζονται σε ένα ή περισσότερα ρεύματα προγράμματος (Program Stream) ή σε ένα ρεύμα μεταφοράς (Transport Stream). Τα δύο αυτά ρεύματα αναπτύχθηκαν για να καλύψουν αντίστοιχα τις ανάγκες δύο μεγάλων ομάδων γνωστών ή αναπτυσσόμενων εφαρμογών. Έτσι ενσωματώνουν ένα μεγάλο βαθμό ευελιξίας ενώ ταυτόχρονα επιβεβαιώνουν ότι υπάρχει διαλειτουργικότητα ανάμεσα σε διάφορες υλοποιήσεις συσκευών.

2.2.2.1 Program Stream

Το ρεύμα προγράμματος (Program Stream) του MPEG-2 είναι ανάλογο του στρώματος συστήματος (system layer) του MPEG–1. Το αποτέλεσμά του είναι ο συνδυασμός ενός ή περισσοτέρων ρευμάτων PES πακέτων με κοινό μετρητή χρόνο σε ένα και μοναδικό ρεύμα. Αυτό το ρεύμα δεδομένων χρησιμοποιείται συνήθως σε περιβάλλοντα σχετικά με πολύ μικρή πιθανότητα σφάλματος και είναι κατάλληλο για εφαρμογές που ασχολούνται με επεξεργασία των πληροφοριών του ρεύματος όπως διαδραστικές πολυμεσικές εφαρμογές.

Τα πακέτα ρεύματος προγράμματος είναι μεταβλητού μεγέθους. Σε αυτά συνήθως περιλαμβάνονται πληροφορίες χρονισμού οι οποίες χρησιμοποιούνται για να καθορίσουν από το σύστημα μια σταθερή καθυστέρηση κατή τη μεταφορά των πακέτων από την είσοδο του κωδικοποιητή στην έξοδο του αποκωδικοποιητή. Ένα πακέτο ρεύματος προγράμματος αποτελείται από πολλά επιμέρους πακέτα PES τα οποία έχουν κοινό χρόνο. Γενικά η δομή ενός πακέτου του ρεύματος προγράμματος εμφανίζεται και στην εικόνα 2.2. Βλέπουμε ότι κάθε πακέτο αποτελείται από μία επικεφαλίδα και διάφορα PES πακέτα. Η επικεφαλίδα έχει μέγεθος 15 βψτες και αποτελείται από τα εξής πεδία:

2.2.2.2 Transport Stream

Το ρεύμα μετάδοσης (Transport Stream) συνδυάζει ένα η περισσότερα επί μέρους ρεύματα δεδομένων που βασίζονται σε ένα ή περισσότερους μετρητές

χρόνου σε ένα και μοναδικό ρεύμα. Το ρεύμα μετάδοσης σχεδιάστηκε για χρήση σε περιβάλλοντα με μεγάλη πιθανότητα λάθους, όπως αποθήκευση ή μετάδοση σε περιβάλλοντα με θόρυβο. Γενικά το ρεύμα μεταφοράς μπορεί να χρησιμοποιηθεί σε πολλά επίπεδα σε μία εφαρμογή πολυμέσων που χρησιμοποιεί στρώμα, και έχει σχεδιαστεί για αποδοτική και εύκολη υλοποίηση του σε εφαρμογές που χρησιμοποιούν δίκτυα υψηλών ταχυτήτων.

Γενικά το ρεύμα μετάδοσης σχεδιάστηκε για να μπορεί μια εφαρμογή να εκτελεί εύκολα κάποιες λειτουργίες. Κάποιες από αυτές τις λειτουργίες είναι:

- Ανάκτηση ενός μόνο συγκεκριμένου συμπιεσμένου ρεύματος δεδομένων, από τα ρεύματα που περιλαμβάνει το ρεύμα μεταφοράς.
- Εξαγωγή ενός μόνο συγκεκριμένου συμπιεσμένου ρεύματος δεδομένων, από τα ρεύματα που περιλαμβάνει το ρεύμα μεταφοράς, και δημιουργία νέου ρεύματος που θα περιλαμβάνει μόνο το ρεύμα αυτό.
- Δημιουργία ενός ρεύματος μετάδοσης από ρεύματα προγράμματος για τη μετάδοση των πληροφοριών σε κάποιο περιβάλλον με σφάλματα και δημιουργία μετά από το ρεύμα μετάδοσης των αρχικών ρευμάτων προγράμματος.

Ο συρμός δεδομένων αποτελείται από πακέτα. Το μέγεθος τον πακέτων αυτών είναι σταθερό και ίσο με 188 bytes. Τα πρώτα 4 bytes του πακέτου είναι η επικεφαλίδα. Αυτή χωρίζεται σε επιμέρους πεδία το κάθε ένα από τα οποία προσδίδει ένα χαρακτηριστικό στο πακέτο. Τα πεδία αυτά είναι τα εξής:

Sync Byte: Το πεδίο αυτό σηματοδοτεί την αρχή του πακέτου. Είναι μεγέθους 8 bit και έχει πάντα την τιμή 0x47[1] ώστε να είναι δυνατό σε έναν δέκτη να συγχρονιστεί με τον συρμό δεδομένων.

Transport Error Indicator: Αυτό το πεδίο, μήκους ενός bit, είναι μια σημαία που παίρνει τιμές 0 ή 1 ανάλογα εάν το πακέτο περιέχει λάθη ή όχι. Το πεδίο παίρνει τιμή κατά την διάρκεια της λήψης των δεδομένων ώστε να γίνεται γνωστό ότι έχει προκύψει σφάλμα κατά την μετάδοση.

[1] Ο συμβολισμός 0x, μπροστά από κάποιον αριθμό, δηλώνει ότι ο αριθμός που ακολουθεί είναι δεκαεξαδικός. Έτσι, για παράδειγμα, το 0x47 είναι ο αριθμός 47 με βάση το 16, ή ο αριθμός 71 στο δεκαδικό σύστημα.

Payload Start Indicator: Άλλη μία σημαία, η οποία παίρνει τιμή 1 όταν το πακέτο περιέχει την αρχή ενός πακέτου PSI ή PES. Όταν χρησιμοποιούνται τεχνικές ενθυλάκωσης, αυτή η σημαία έχει άλλη αντιμετώπιση που θα περιγραφεί παρακάτω.

Transport Priority: Αυτό το πεδίο είναι μία σημαία ενός bit και λαμβάνει την τιμή 1, για να δείχνει ότι τα δεδομένα αυτού του πακέτου, έχουν μεγαλύτερη προτεραιότητα από τα δεδομένα των υπόλοιπων μέσα στον συρμό.

PID: Αυτό το πεδίο έχει μήκος 13 bit. Είναι ένα από τα σημαντικότερα πεδία, καθώς η τιμή του ορίζει το είδος του πακέτου. Τα πακέτα ελέγχου του συρμού έχουν συγκεκριμένες τιμές σε αυτό. Επίσης, όλα τα πακέτα που περιέχουν δεδομένα ενός PES, πρέπει να έχουν την ίδια τιμή, ώστε να είναι εφικτή η απομόνωσή τους από τον συρμό. Έτσι, είναι δυνατόν μέσα στον ίδιο συρμό, να περιέχονται πολλά προγράμματα, όπως είναι δυνατή και η μεταφορά ενός προγράμματος από τον ένα συρμό στον άλλο.

Transport Scrambling Control: Αυτό το πεδίο έχει μέγεθος 2 bit και αναφέρει εάν το πακέτο έχει υποστεί τεχνικές scrambling[2].

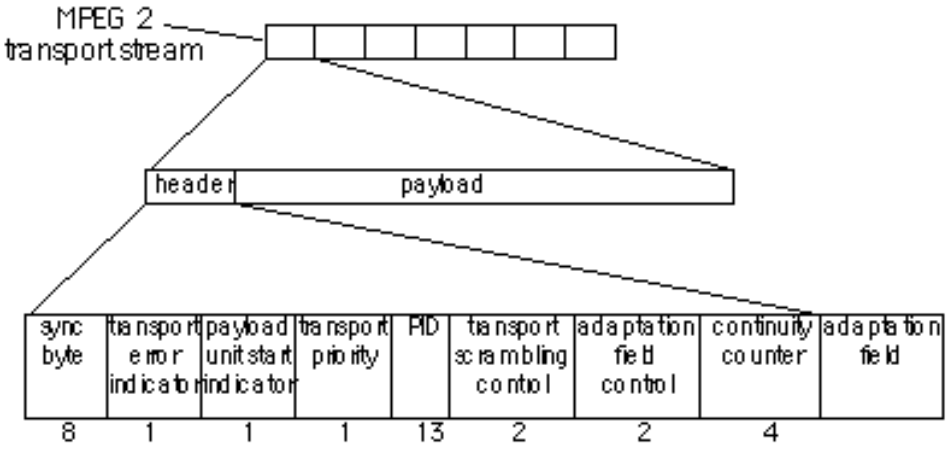
Adaptation Field Contol: Αυτό το πεδίο είναι μήκους 2 bit και ορίζει αν μετά την κεφαλίδα ακολουθούν περισσότερα δεδομένα ελέγχου. Αν το πρώτο bit έχει τιμή 1, τότε ακολουθεί πεδίο adaptation. Μια από τις σημαντικές πληροφορίες που κρατάει το πεδίο adaptation, είναι το PCR (Program Clock Reference) βάσει του οποίου συγχρονίζονται οι διάφορες επιμέρους ροές του συρμού μεταξύ τους. Το επόμενο bit καθορίζει αν ακολουθούν δεδομένα ή αν το πακέτο είναι κενό.

Continuity Counter: Το πεδίο αυτό έχει μήκος 4 bit. Στην ουσία είναι ένας μετρητής που αυξάνεται σε κάθε επόμενο πακέτο. Αυτός είναι χρήσιμος

[2]Στις τηλεπικοινωνίες είναι σύνηθες, πριν μεταδοθεί μία ροή δεδομένων, να εφαρμόζονται σε αυτήν αλγόριθμοι scrambling ώστε να αλλάξει η σειρά των δεδομένων. Αυτή η τεχνική δεν έχει να κάνει με κρυπτογράφηση, αφού ο σκοπός δεν είναι να καταστήσει τα δεδομένα μη προσβάσιμα, αλλά να τους προσδώσει χρήσιμες τεχνικές ιδιότητες. Αυτό επιτυγχάνεται αντικαθιστώντας ανεπιθύμητες αλληλουχίες (πολλοί άσσοι, πολλά μηδενικά, περιοδικές αλληλουχίες...) με άλλες 'καλύτερης' συμπεριφοράς.

για την ανίχνευση σφαλμάτων μετάδοσης στην σειρά των πακέτων.

Το ρεύμα προγράμματος και το ρεύμα μετάδοσης έχουν σχεδιαστεί για διαφορετικά είδη εφαρμογών. Είναι δυνατόν και λογικό να μπορούμε να μετατρέψουμε ένα είδος ρεύματος σε κάποιο άλλο καθώς το ένα ρεύμα δεν είναι υποσύνολο του άλλου. Για τη μετατροπή των ρευμάτων μια εφαρμογή μπορεί να χρησιμοποιήσει την κοινή δομή των PES πακέτων. Βέβαια, οι πληροφορίες που περιέχονται σε ένα πακέτο του ρεύματος προγράμματος δεν είναι δυνατόν να αποκτηθούν άμεσα από τις πληροφορίες των επικεφαλίδων των πακέτων ρεύματος μετάδοσης, αλλά μπορούν να δημιουργηθούν από κάποια περαιτέρω επεξεργασία των πακέτων.



Σχήμα 2.3: Η δομή ενός TS πακέτου.

Στο πρώτο κεφάλαιο του πρωτοκόλλου MPEG-2 ιδιαίτερη μνεία γίνεται και για τον συγχρονισμό των διαφόρων ρευμάτων. Η λειτουργία αυτή είναι αρκετά σημαντική αφενός για τη σωστή αναπαραγωγή των παρουσιάσεων, αλλά

παράλληλα η εφαρμογή μας χρησιμοποιεί τις πληροφορίες χρονισμού που μεταδίδονται για τον υπολογισμό των χρονικών στιγμών κατά τις οποίες οι πελάτες θα πρέπει να να ζητήσουν το επόμενο κλειδί κρυπτογράφησης. Για το συγχρονισμό των επιμέρους ρευμάτων κάθε υλοποίηση του πρωτοκόλλου θέτει κάποια μέγιστα κατώφλια για τη μετάδοση και ανάκτηση πακέτων, τόσο του ρεύματος του προγράμματος, όσο και του ρεύματος μεταφοράς. Συγκεκριμένα, το κατώφλι θα πρέπει να καθορίζει το μέγιστο επιτρεπτό χρόνο που απαιτείται για τη συμπίεση του ρεύματος, την τοπική αποθήκευση, την πολύπλεξη, τη μετάδοση ή την αποθήκευση, την αποπολύπλεξη, την αποκωδικοποίηση και την αναπαραγωγή του ρεύματος. Για το λόγο αυτό λοιπόν το πρωτόκολλο καθορίζει ότι σε κάθε τύπου πακέτου θα πρέπει περιλαμβάνονται πληροφορίες που αφορούν χρόνο και οι οποίες χρησιμοποιούνται για τον υπολογισμό της καθυστέρησης από άκρη σε άκρη. Ο υπολογισμός των χρόνων γίνεται με τη βοήθεια ενός κοινού συστήματος μέτρησης χρόνου, το οποίο ονομάζεται Ρολόι Συστήματος (System Time Clock). Με βάση αυτό το ρολόι το πρωτόκολλο εισάγει σε κάθε πακέτο PES ένα πεδίο Σφραγίδας Χρόνου Παρουσίασης (Presentation Time Stamps). Οι σφραγίδες αυτές καθορίζουν το χρόνο αναπαραγωγής ενός πακέτου με βάσει ένα ρολόι παλμών 90 kHz. Όπως είδαμε και παραπάνω το πρωτόκολλο αναπτύσσει περαιτέρω τα όρια μετάδοσης δεδομένων εισάγοντας κάποια extra πεδία στα πακέτα των ρευμάτων, τα οποία καθορίζουν επακριβώς τους χρόνους που θα πρέπει να φτάσουν τα δεδομένα στο κάθε επίπεδο του πρωτοκόλλου. Έτσι λοιπόν έχουμε το πεδίο αναφοράς Ρολογιού προγράμματος (Program Clock Reference) και το προαιρετικό πεδίο Αναφοράς Ρολογιού Στοιχειώδους Ρεύματος (Elementary Stream Clock Reference) για το ρεύμα προγράμματος και το πεδίο Αναφοράς Ρολογιού Συστήματος (System Clock Reference) για το ρεύμα μεταφοράς. Τα πεδία αυτά καθορίζουν το μέγιστο επιτρεπτό χρόνο που μπορεί να κάνει το πακέτο μέχρι την είσοδό του στον αποκωδικοποιητή.

Κεφάλαιο 3

Το πρότυπο DVB

Τα τελευταία δέκα χρόνια του 20ου αιώνα η Αμερική (ATSC) και η Ιαπωνία (ISDB) αναπτύσσουν τη δική τους τεχνολογία πάνω στον τομέα της ψηφιακής τηλεόρασης. Παράλληλα οριοθετείται το 1991 η δημιουργία μίας Ευρωπαϊκής ομάδας ανάπτυξης, η οποία θα καταλήξει το 1992, στην δημιουργία ενός προτύπου το οποίο θα αποτελέσει τη βάση για την ανάπτυξη της ψηφιακής τηλεόρασης στην Ευρώπη. Το 1993 ολοκληρώνεται η ιδέα, με την ίδρυση της οικογένειας προτύπων DVB (Digital Video Broadcasting). Η οικογένεια αυτή, είχε ως στόχο της την παροχή υψηλού επιπέδου υπηρεσιών ψηφιακής τηλεόρασης χρησιμοποιώντας δορυφορικά, καλωδιακά και επίγεια μέσα μετάδοσης. Υιοθετώντας την κωδικοποίηση κατά MPEG-2 σε όλα τα πρότυπα, έχει επιτευχθεί η ψηφιοποίηση προγραμμάτων εικόνας και ήχου, η μεταφορά δεδομένων με υψηλή ταχύτητα, καθώς και η πολυπλεξία τους.

Τα πιο γνωστά πρότυπα που απαρτίζουν την οικογένεια DVB, μέχρι σήμερα, είναι το DVB-S (Satellite) που υποστηρίζει τη δορυφορική μετάδοση, το DVB-C (Cable) το οποίο υποστηρίζει την καλωδιακή μετάδοση, το DVB-T (Terrestrial) το οποίο αναφέρεται στην αμφίδρομη επίγεια μετάδοση και τέλος ένα νέο πρότυπο, το DVB-H (Handhelds) το οποίο προσθέτει την έννοια του κινητού χρήστη στην επίγεια ψηφιακή μετάδοση.

Το δορυφορικό σύστημα DVB-S είναι το παλαιότερο και πιο διαδεδομένο από την οικογένεια προτύπων DVB και έχει αδιαμφισβήτητα τύχει παγκόσμιας αποδοχής. Το DVB-S σχεδιάστηκε για να εκμεταλλεύεται πλήρως το

εύρος ζώνης των δορυφορικών τηλεοπτικών αναμεταδοτών. Χρησιμοποιεί ρυθμό μεταφοράς των 54Mbps με διαμόρφωση QPSK σε συνδυασμό με ένα σχήμα διπλής κωδικοποίησης και διεμπλοκής (coding/interleaving). Είναι το μόνο πρότυπο από την οικογένεια DVB που έχει τύχει εμπορικής εφαρμογής και στη χώρα μας.

Το καλωδιακό σύστημα DVB-C έχει τεχνικά, αρκετές ομοιότητες με το DVB-S. Η διαφορά του έγκειται στο ότι χρησιμοποιεί την αποδοτικότερη (ως προς το εύρος ζώνης) αλλά και πιο ευαίσθητη σε παρεμβολές διαμόρφωση 64-QAM αντί για την QPSK. Έτσι, ένα καλωδιακό κανάλι των 8 MHz μπορεί να μεταφέρει 38.5Mbps. Εναλλακτικά, μπορεί να χρησιμοποιηθεί QAM λιγότερων ή περισσότερων επιπέδων. Σε κάθε περίπτωση, υπάρχει μία αντιστάθμιση μεταξύ ταχύτητας και αξιοπιστίας.

Το επίγειο σύστημα DVB-T αποτελεί το τελευταίο χρονικά μέλος της οικογένειας DVB. Επιτυγχάνει ψηφιακή μετάδοση υψηλών ταχυτήτων πάνω από το 'δύσκολο' επίγειο κανάλι, χρησιμοποιώντας διαμόρφωση πολλαπλών φερόντων στο σχήμα της πολυπλεξίας με ορθογωνική διαίρεση συχνότητας (Orthogonal Frequency Division Multiplexing - OFDM). Το σχήμα OFDM του DVB-T χρησιμοποιεί ένα μεγάλο αριθμό φερόντων (6817 ή 1704 για μετάδοση 8K και 2K αντίστοιχα), κάθε ένα από τα οποία διαμορφώνεται κατά QPSK, 16QAM ή 64QAM. Έτσι, η πληροφορία κατανέμεται ομοιόμορφα στο φάσμα και σε συνδυασμό με κωδικοποίηση και διεμπλοκή δύο στρωμάτων, το σήμα αποκτά μεγάλη ευρωστία ακόμη και σε περιβάλλοντα με ισχυρές διαλήψεις και φαινόμενα πολυδιαδρομικής μετάδοσης (multipath).

Το επίγειο σύστημα DVB-H, αποτελεί συνέχεια του DVB-T με τη διαφορά όμως ότι αναφέρεται σε κινητούς χρήστες. Το πρότυπο αυτό έχει προσαρμοστεί στις ιδιότητες που έχουν οι συσκευές κινητής πρόσβασης, όπως μέγεθος συσκευής και χωρητικότητα μνήμης, ανάγκη εύκολης πρόσβασης στο δίκτυο και εξοικονόμηση ενέργειας. Στην ουσία δημιουργείται μία γέφυρα που ενώνει τα δίκτυα ευρυζωνικής εκπομπής με τον κόσμο των κυψελωτών ασυρμάτων δικτύων.

3.1 Λειτουργία και εφαρμογή του προτύπου DVB-T

Το πρότυπο DVB-T είναι το πιο πρόσφατο και πιο εξελιγμένο τεχνολογικά πρότυπο. Η χρήση του στην επίγεια ψηφιακή μετάδοση, υλοποιείται σε όλο και περισσότερες ευρωπαϊκές χώρες. Αναπτύχθηκε, όπως και τα υπόλοιπα προαναφερθέντα standards, από το DVB Forum και εγκρίθηκε από τον οργανισμό ETSI ως το βασικό πανευρωπαϊκό πρότυπο το 1997. Το πρώτο επίγειο πρόγραμμα σε ψηφιακή μορφή, ξεκίνησε στην Αγγλία ένα χρόνο αργότερα. Σήμερα, συστήματα DVB-T υλοποιούνται στην Ευρωπαϊκή Ένωση, τη Ρωσία, την Ανατολική Ευρώπη, την Ινδία, την Σιγκαπούρη και την Αυστραλία. Το DVB-T προπορεύεται του αντίστοιχου αμερικάνικου προτύπου ATSC, το οποίο και αναπτύχθηκε στις Ηνωμένες Πολιτείες, ως προς κάποια χαρακτηριστικά (κινητικότητα χρήστη, αντοχή σε multipath, δημιουργία SFN), ενώ το Ιαπωνικό πρότυπο ISDB-T βασίστηκε ουσιαστικά στη φιλοσοφία του DVB-T.

Οι προοπτικές χρήσης του για μετάδοση δεδομένων IP είναι πολυάριθμες. Σημαντικό πλεονέκτημα του επίγειου συστήματος είναι ότι δεν απαιτεί ιδιαίτερο εξοπλισμό από πλευράς χρήστη (π.χ. δορυφορικό δέκτη ή καλωδιακή υποδομή) ενώ από την πλευρά του παροχέα, αποτελεί την πιο προσιτή και πιο ευέλικτη λύση σε σχέση με την ανάγκη δορυφορικής μετάδοσης ή τη χρήση καλωδιακού δικτύου. Ένα πολύ σημαντικό χαρακτηριστικό του συγκεκριμένου προτύπου, είναι η δυνατότητα προσθήκης ενός επίγειου ψηφιακού συστήματος σε περιοχές κάλυψης με κυψελωτή δομή. Προσφέρει υπηρεσίες και σε κινούμενους χρήστες, μία δυνατότητα που οι υπόλοιπες τεχνολογίες δεν είναι σε θέση να προσφέρουν ακόμα.

Καθώς το DVB-T ορίζει από μόνο του ένα σύστημα εκπομπής ευρείας κάλυψης (broadcasting) παραλείποντας τον ορισμό της τεχνολογίας επιστροφής (reverse path) και δεδομένου ότι η μορφή του σήματος βασικής ζώνης και ο αλγόριθμος συμπίεσης της εικόνας περιγράφεται στην προδιαγραφή MPEG-2, το πρότυπο DVB-T περιορίζεται αποκλειστικά στην περιγραφή των λειτουργιών του διαμορφωτή. Ο διαμορφωτής, δέχεται στην είσοδο του από τον πολυπλέκτη, ένα συρμό μεταφοράς MPEG-2, που περιέχει πολυπλεγμένες τις υπηρεσίες εικόνας, ήχου και δεδομένων υπό μορφή σήματος βασικής ζώνης και παράγει το

RF σήμα το οποίο είναι έτοιμο προς αποστολή. Το τελευταίο έχει εύρος ζώνης 8MHz και τοποθετείται σε ένα από τα κανάλια 21-69 της μπάντας των UHF, όπως ακριβώς ένα αναλογικό τηλεοπτικό κανάλι.

Ο διαμορφωτής DVB-T χρησιμοποιεί σχήμα OFDM για να αντιμετωπίσει στη μετάδοση διαλήψεις επιλεκτικές ως προς τη συχνότητα. Αυτό το χαρακτηριστικό του OFDM διευκολύνει πολύ τη λήψη δεδομένων ευρείας ζώνης και από κινητούς χρήστες. Στο χώρο μάλιστα των τεχνολογιών εκπομπής (broadcasting), το OFDM παρέχει τη δυνατότητα κατασκευής δικτύων με διεσπαρμένους πομπούς που εκπέμπουν συγχρονισμένα τα ίδια δεδομένα στην ίδια συχνότητα χωρίς η συμβολή των σημάτων τους να επηρεάζει σημαντικά το δέκτη. Τέτοια δίκτυα ονομάζονται Single Frequency Networks (SFN) και μπορούν να αυξήσουν την συνολική χωρητικότητα του δικτύου σε ολόκληρη την γεωγραφική επικράτεια μέχρι και 45 φορές. Καθώς εξαλείφεται η ανάγκη να υπάρχουν ελεύθερα κανάλια σε ορισμένες περιοχές, αποφεύγονται παρεμβολές με γειτονικούς πομπούς.

Το πρότυπο DVB-T συνδυάζει το OFDM με σύνθετες τεχνικές ισοστάθμισης και κωδικοποίησης, εισάγοντας την τεχνολογία του κωδικοποιημένου OFDM (Coded OFDM- COFDM). Συνδυάζοντας κωδικοποίηση και διεμπλοκή δύο επιπέδων, η διαδικασία διαμόρφωσης καθιστά το σήμα ιδιαίτερα ανθεκτικό σε πολυδιαδρομική διάδοση και παρεμβολές. Η λειτουργία του διαμορφωτή είναι σχετικά σύνθετη. Προκειμένου να γίνει κατανοητή, θα αναφερθούν και θα αναλυθούν τα στάδιά της ξεχωριστά. Οι λειτουργίες αυτές, με τη σειρά που εφαρμόζονται στο Ρεύμα Μεταφοράς, είναι οι εξής:

- Προσαρμογή MPEG-2 πακέτων και τυχαιοποίηση (randomization)
- Εξωτερική κωδικοποίηση (προστασία έναντι λαθών με κώδικα Reed-Solomon)
- Εξωτερική συνελικτική διεμπλοκή (convolutional interleaving)
- Εσωτερική κωδικοποίηση με διάτρητο συνελικτικό κώδικα (punctured convolutional code)
- Εσωτερική διεμπλοκή (inner interleaving) στον χρόνο και στη συχνότητα

- Αντιστοίχηση και διαμόρφωση των φερόντων
- Πολυπλεξία κατά OFDM με αντίστροφο ταχύ μετασχηματισμό Fourier (IFFT) και διαμόρφωση του φέροντος IF
- Άνω μετατροπή (up-conversion) στην τελική RF συχνότητα.

Η τελευταία λειτουργία δεν υποστηρίζεται εγγενώς από αρκετούς διαμορφωτές. Θα πρέπει να εισαγάγουμε μία πρόσθετη μονάδα για άνω μετατροπή.

Στην περίπτωση της ιεραρχικής διαμόρφωσης, το σήμα βασικής ζώνης προϋπάρχει διαιρεμένο σε δύο Ρεύματα Μεταφοράς: ένα υψηλής προτεραιότητας (high priority TS) και ένα χαμηλής (low priority TS). Τα δύο σήματα διαμορφώνονται ταυτόχρονα σε ένα ιεραρχικό QAM σήμα. Ως αποτέλεσμα, ένας δέκτης με κακές συνθήκες λήψης λαμβάνει μόνο τα δεδομένα υψηλής προτεραιότητας, ενώ ένας με καλύτερες λαμβάνει το σύνολο. Η λειτουργία ιεραρχικής διαμόρφωσης παρέχει σημαντική ευελιξία στο σύστημα, ιδίως όταν συνοδεύεται από κλιμακωτή κωδικοποίηση της κινούμενης εικόνας κατά MPEG-2 (scalable MPEG-2 encoding).

Προκειμένου να γίνει κατανοητός ο τρόπος μετάδοσης των υπηρεσιών, περιγράφονται παρακάτω τα βήματα που ακολουθεί αυτή η ψηφιακή μετάδοση.

Για την κωδικοποίηση των τηλεοπτικών αλλά και των ηχητικών σημάτων καθώς και για την πολυπλεξία χρησιμοποιείται το MPEG-2. Απόλυτη ευκρίνεια και άριστος ήχος. Ένας συρμός μεταφοράς MPEG-2, μπορεί πρακτικά να μεταφέρει οτιδήποτε είναι δυνατό να ψηφιοποιηθεί.

3.2 Στάδια διαμόρφωσης DVB-T

3.2.1 Προσαρμογή MPEG-2 πακέτων και τυχαιοποίηση

Το σήμα βασικής ζώνης που εισέρχεται στον διαμορφωτή είναι σταθερού ρυθμού (constant bit rate - CBR) και οργανωμένο σε πακέτα σταθερού μήκους των 188 bytes. Κάθε πακέτο ξεκινά με το byte συγχρονισμού, που είναι πάντα ίσο με 0x47. Προκειμένου να περιοριστεί το ενδεχόμενο να υπάρχουν μεγάλα

διαστήματα χωρίς δυαδική μεταβολή (μακριές ακολουθίες '0' ή '1' - κάτι που συμβαίνει π.χ. σε πακέτα κενού περιεχομένου που χρησιμοποιούνται μόνο για stuffing), ακολουθείται μια διαδικασία τυχαιοποίησης, όπως φαίνεται και στο παρακάτω σχήμα. Το πολυώνυμο για την γεννήτρια ψευδοτυχαίας ακολουθίας είναι:

$$1 + X^{14} + X^{15}$$

3.2.2 Εξωτερική κωδικοποίηση και συνελικτική διεμπλοκή

Στη συνέχεια, πραγματοποιείται εξωτερική κωδικοποίηση (outer coding) η οποία καλείται κωδικοποίηση Reed Solomon. Κατά τη διάρκεια της εξωτερικής κωδικοποίησης γίνεται χρήση μιας τεχνικής που ονομάζεται Εμπρόσθια Διόρθωση Σφαλμάτων, FEC (Forward Error Correction). Σύμφωνα με την τεχνική αυτή, σε κάθε πακέτο μεταφοράς των 188 bytes, προστίθενται 16 bytes πλεονασμού, οπότε προκύπτει ένα πακέτο προστατευμένο από σφάλματα, μήκους 204 bytes.

Ακολουθεί η τεχνική της συνελικτικής διεμπλοκής, σκοπός της οποίας είναι η αύξηση της απόδοσης της κωδικοποίησης Reed-Solomon. Ο λόγος για τον οποίο απαιτείται αποδοτικότερη προστασία έναντι στα σφάλματα, είναι ότι στα κανάλια μετάδοσης η ποιότητα των σημάτων μεταβάλλεται και είναι δυνατόν ένας μεγάλος αριθμός από δυαδικά ψηφία, ο οποίος υπερβαίνει την δυνατότητα διόρθωσης της κωδικοποίησης RS, να αλλοιωθεί (από πιθανό χτύπημα κεραυνού ή από παρεμβολές ηλεκτρικών συσκευών). Η προστασία κάθε πακέτου μετάδοσης από τέτοιου είδους αλλοιώσεις δεν είναι ιδιαίτερα οικονομική. Σε αυτήν την περίπτωση εφαρμόζεται συνελικτική διεμπλοκή, σύμφωνα με την οποία τα δεδομένα αφού κωδικοποιηθούν με την Εμπρόσθια Διόρθωση Σφαλμάτων, τροφοδοτούνται σε μια μνήμη RAM και μεταδίδονται αναδιατεταγμένα (μια πιθανή εκδοχή είναι τα δεδομένα να εισέρχονται στη RAM σε γραμμές και να εξέρχονται σε στήλες). Με τη χρήση μιας δεύτερης RAM κατά τη λήψη, τα δεδομένα τοποθετούνται στην αρχική τους δομή. Το αποτέλεσμα της διαδικασίας της διεμπλοκής είναι ότι το συσσωμάτωμα των δυαδικών ψηφίων που έχουν υποστεί σφάλμα μετατρέπεται σε ένα μεγάλο αριθμό ενιαίων εσφαλμένων συμβόλων, τα

οποία είναι εύκολα διορθώσιμα.

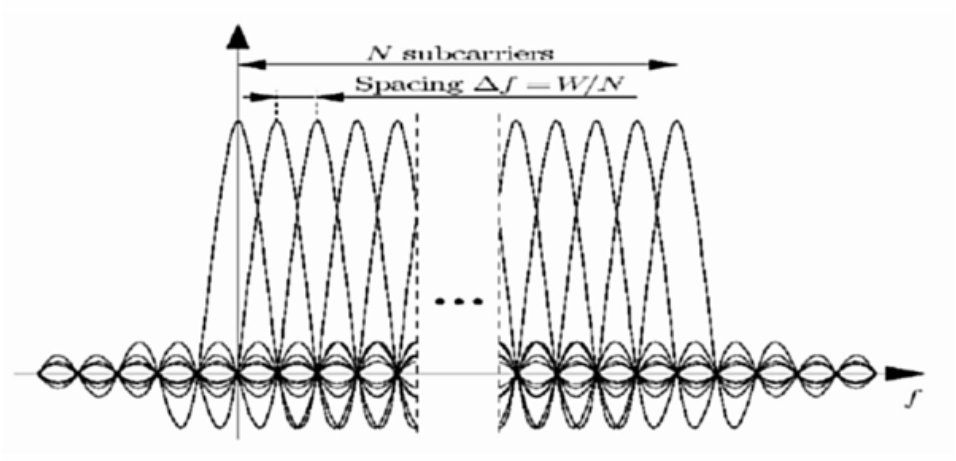
3.2.3 Εσωτερική κωδικοποίηση και διεμπλοκή

Η εσωτερική κωδικοποίηση ακολουθεί τη συνελικτική διεμπλοκή και τη συμπληρώνει αποδοτικά καθώς διορθώνει άλλου είδους σφάλματα. Ο ισχυρός πλεονασμός που εισάγεται από αυτή (100%, καθώς ο συνελικτικός κωδικοποιητής παράγει δύο ροές εξόδου, καθεμιά με τον ίδιο ρυθμό μετάδοσης όπως η ροή εισόδου) επιτρέπει μια πολύ ισχυρή διόρθωση λαθών. Αυτό μπορεί να είναι αναγκαίο για σήματα με πολύ χαμηλό λόγο σήματος-προς-θόρυβο (SNR, signal-to-noise ratio) στην είσοδο του δέκτη, αλλά έχει ως αποτέλεσμα τον υποδιπλασιασμό της φασματικής απόδοσης του καναλιού. Ωστόσο, αυτού του τύπου η συνελικτική κωδικοποίηση, επιτρέπει ο πλεονασμός που εισάγεται, να μειωθεί διαμέσου της διάτρησης (puncturing) της εξόδου του συνελικτικού κωδικοποιητή. Αυτός καθιστά δυνατή τη μη λήψη όλων των διαδοχικών bits των ακολουθιών εξόδου, αλλά μόνο ένα από τα δύο ταυτόχρονα βιτς με ένα συγκεκριμένο λόγο διάτρησης (puncturing ratio). Με αυτό τον τρόπο, είναι πιθανό να επιτευχθούν οι ρυθμοί κώδικα διάτρησης (punctured code rates) που καθορίζονται από το στάνταρ DVB (2/3, 3/4, 5/6 ή 7/8), που εκφράζουν το λόγο της εισόδου προς τον εκπεμπόμενο ρυθμό (στην έξοδο). Δεδομένης της ισχύος του αναμεταδότη και του μεγέθους της κεραίας λήψης, ο ρυθμός κώδικα που επιλέγεται από το σταθμό εκπομπής θα αποτελεί ένα συμβιβασμό μεταξύ ενός επιθυμητού ρυθμού μετάδοσης και του τύπου της παρεχόμενης υπηρεσίας.

3.2.4 Διαμόρφωση και μετάδοση

Η διαδικασία της εσωτερικής διεμπλοκής παράγει μια ακολουθία από bits ήδη οργανωμένη σε σύμβολα QAM. Όπως προαναφέρθηκε, τα δυνατά σχήματα διαμόρφωσης είναι: QPSK (2 bits/symbol), 16QAM (4 bits/symbol) και 64QAM (6 bits/symbol). Τα διαγράμματα αστερισμού (constellation maps) για κάθε τύπο διαμόρφωσης περιγράφονται αναλυτικά στην προδιαγραφή. Τα σύμβολα ομαδοποιούνται και μεταδίδονται ταυτόχρονα με τη χρήση πολυπλεξίας OFDM. Κάθε σύμβολο OFDM αποτελείται από ένα σύνολο N=6817 ('8k mode') ή 1705 ('2k mode') φερόντων και μεταδίδεται με διάρκεια TS=896μsec και

TS=224μsec αντίστοιχα. Η απόσταση μεταξύ δύο γειτονικών φερόντων είναι Δf=1116Hz και Δf=4464Hz για τις δύο καταστάσεις λειτουργίας αντίστοιχα. Η ορθογωνιότητα μεταξύ των φερόντων εξασφαλίζεται από το γεγονός ότι Δf=1/TS πάντα. Με τη συνθήκη αυτή, η διάταξη των φερόντων αποκτά τη μορφή του παρακάτω σχήματος.:



Σχήμα 3.1: DVB διαμορφομένο σήμα.

Το συνολικό εύρος ζώνης του σήματος DVB-T ανέρχεται στα 7.61MHz για ονομαστική κατάσταση λειτουργίας 8 MHz, ενώ προβλέπονται και καταστάσεις λειτουργίας των 7 και 6 MHz. Το κάθε σύμβολο OFDM αποτελείται από δύο μέρη: ένα ωφέλιμο τμήμα με διάρκεια TU και ένα διάστημα φρούρησης (guard interval), με διάρκεια D. Το διάστημα φρούρησης αποτελείται από μια κυκλική επανάληψη του ωφέλιμου τμήματος, και εισάγεται πριν από αυτό. 68 διαδοχικά σύμβολα OFDM αποτελούν ένα πλαίσιο OFDM (OFDM frame), ενώ τέσσερα διαδοχικά πλαίσια αποτελούν ένα υπέρ-πλαίσιο (OFDM super-frame). Μέσα σε κάθε σύμβολο OFDM, οι πληροφορίες που μεταφέρονται από τα φέροντα μπορεί να είναι είτε δεδομένα είτε πληροφορίες συγχρονισμού και γενικώς δεδομένα χρήσιμα για την καλή λειτουργία του δέκτη. Τα φέροντα που δεν φέρουν πληροφορία διακρίνονται σε:

- Διεσπαρμένα φέροντα-πιλότους (pilot carriers). Αυτά εκπέμπουν εκ περιτροπής ένα δεδομένο σήμα, το οποίο γνωρίζει ο δέκτης. Μετρώντας την ισχύ των φερόντων αυτών, ο δέκτης μπορεί ανά πάσα στιγμή να σχηματίσει μια εκτίμηση της απόκρισης συχνότητας (frequency response) του καναλιού.

- Σταθερά φέροντα-πιλότους. Αυτά κατέχουν σταθερή θέση μέσα στο σύμβολο.

- Φέροντα σηματοδοσίας παραμέτρων μετάδοσης (TPS -Transmission Parameter Signalling carriers). Αυτά κατέχουν επίσης σταθερή θέση μέσα στο σύμβολο, και πληροφορούν το δέκτη για τις παραμέτρους που χρησιμοποιούνται στη μετάδοση (ρυθμός κωδικοποίησης, διάρκεια διαστήματος φρούρησης, τύπος διαμόρφωσης), ούτως ώστε ο δέκτης να μπορεί να προσαρμόζεται αυτόματα.

3.2.5 Ωφέλιμο bit rate

Η τιμή που περισσότερο ίσως ενδιαφέρει τον παροχέα DVB-T είναι το ωφέλιμο bit rate που μπορεί να μεταφερθεί από το ψηφιακό σήμα, δηλαδή ο ρυθμός του Ρεύματος Μεταφοράς MPEG-2 που μεταδίδεται. Η τιμή αυτή εξαρτάται από τη διάρκεια του διαστήματος φρούρησης, τον ρυθμό κωδικοποίησης και τον τύπο της διαμόρφωσης, όπως φαίνεται και στον ακόλουθο πίνακα (τιμές σε Mbps).

Ο συνδυασμός των παραμέτρων που θα χρησιμοποιηθούν βρίσκεται στην επιλογή του χρήστη. Γενικά πάντως ισχύει ότι όσο αυξάνει ο ωφέλιμος ρυθμός, τόσο πιο ευάλωτο γίνεται το σήμα σε φαινόμενα διαλείψεων και πολυδιαδρομικής μετάδοσης (multipath). Απαιτείται δηλαδή να γίνει ένας συμβιβασμός (trade-off) από την πλευρά του παροχέα μεταξύ χωρητικότητας και ανθεκτικότητας του σήματος.

3.3 Μετάδοση IP δεδομένων πάνω από το κανάλι DVB-T

Όπως αναφέρθηκε ένα χαρακτηριστικό του πρότυπου DVB-T είναι η ενθυλάκωση των πακέτων IP στα πακέτα μεταφοράς. Τα πακέτα έχουν μήκος 188 bytes χρησιμοποιώντας τέσσερα από αυτά ως header(επικεφαλίδα). Για την ενσωμάτωση των πακέτων IP στο TS και για να διακριθούν από τα ψηφιακά πακέτα τηλεοπτικών προγραμμάτων χρειαζόμαστε μια διαδικασία που θα εκτελέσει τη λειτουργία της χαρτογράφησης, την προσαρμογή και την κατάτμηση, αυτές οι λειτουργίες ορίζονται από το πρότυπο ETSI EN 301 192 (Digital Video Broadcasting (DVB): DVB προδιαγραφή για εκπομπή δεδομένων) [7]. Αυτό το πρότυπο καθορίζει τέσσερις τεχνικές ενθυλάκωσης.

- Τεχνική σωλήνωσης στοιχείων (Data piping technique). Αυτή η τεχνική επιτρέπει στα πακέτα IP δεδομένων να ενσωματωθούν ως ωφέλιμο φορτίο στα πακέτα δεδομένων MPEG-2 TS.

- Ροή δεδομένων (Data Streaming) όπου το ρεύμα δεδομένων διαμορφώνεται με ένα συμβατό MPEG-2 στοιχειώδες ρεύμα το οποίο από κάθε στροφή οργανώνεται σε πακέτα ακολουθώντας την δομή Πακεταρισμένο Στοιχειώδες Ρεύμα (PES). Τελικά τα PES πακέτα τεμαχίζονται και διανέμονται στο MPEG-2 ωφέλιμο φορτίο πακέτων μεταφοράς.

- Ενθυλάκωση πολυπρωτοκόλλων (Multiprotocol Encapsulation). Αυτή η τεχνική επιτρέπει τη μεταφορά διάφορων πρωτοκόλλων (π.χ. TCP/IP) μέσω του καναλιού DVB έχοντας ενσωματώσει τα πακέτα δεδομένων σε τμήματα δεδομένων όπως καθορίζονται στο πρότυπο MPEG-2 DSM-CC. Αυτά τα τμήματα δεδομένων είναι συμβατά στο πριατε_σεστιον που καθορίζεται στο ατ ISO/IEC 13818-1 και μπορούν να ενσωματωθούν απευθείας στο TS. Για τη διάκριση αυτών των πακέτων χρησιμοποιούμε έναν αριθμό ταυτότητας πακέτου (PID) , το πεδίο MAC και την IP διεύθυνση.

- Ιπποδρόμιο δεδομένων (Data Carousel). Αυτή η τεχνική είναι κατάλληλη για μη-αμφίδρομη μετάδοση δεδομένων ενώ τα μεταδιδόμενα δεδομένα εί-

ναι συνήθως στόχος σε μεγάλη ομάδα χρηστών. Τα δεδομένα ιπποδρομίου μεταδίδονται κατά μια περιοδική αναλογία.

3.4 Αμφιδρομικότητα

Από το 2001 και μετά υπήρχε διάχυτη μία τάση στην επιστημονική κοινότητα να υπάρξει δυνατότητα, εκτός από την εκπομπή οπτικοακουστικού υλικού, εκπομπής δεδομένων μεσω του DVB και παροχής αμφίδρομών διαδραστικών υπηρεσιών. Έτσι αναπτύχθηκαν πρότυπα για δημιουργία αμφιδρομότητας στο DVB (DVB-RCS, DVB-RCT).

Οι βασικοί στόχοι της Αμφίδρομης Διανομής είναι η ενοποίηση των τεχνολογιών (GSM, DVB, LMDS, UMTS κλπ) και η επέκταση της παροχής υπηρεσιών πολυμέσων σε κινούμενους χρήστες, δεδομένης της ευρείας περιοχής κάλυψης που παρέχει η ψηφιακή τηλεόραση. Ακόμα ένας από τους σημαντικότερους στόχους είναι η καθολική χρήση IP για όλες τις υπηρεσίες, καθώς είναι πολλά τα οφέλη μιας τέτοιας ομοιογένειας στην μετάδοση.

Κεφάλαιο 4

Η μέθοδος ενθυλάκωσης ULE

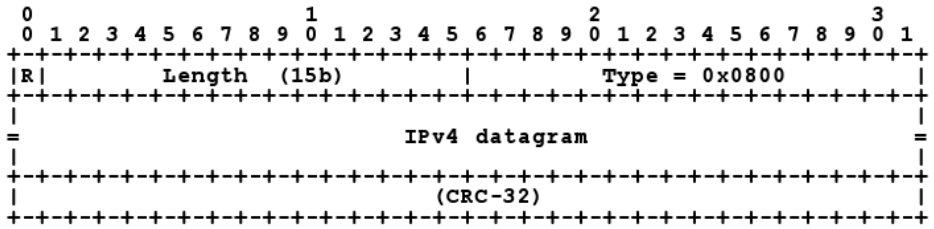
Το ULE (Unidirectional Lightweight Encapsulation) είναι ένα πρωτόκολλο ενθυλάκωσης στο επίπεδο διασύνδεσης το οποίο προτυποποιεί την μεταφορά πακέτων επιπέδου δικτύου, όπως για παράδειγμα IP datagrams, μέσω MPEG-2 συρμών δεδομένων. Εξ αιτίας του πολύ μικρού overhead που απαιτεί είναι ιδιαίτερα κατάλληλο για εφαρμογές IP over Satellite όπου έχει μεγάλη σημασία η εξοικονόμηση εύρους ζώνης. Ένας ακόμη λόγος που μπορεί να προτιμηθεί αυτή η μέθοδος ενθυλάκωσης, είναι το γεγονός ότι είναι σημαντικά ευκολότερο να αναλυθεί και να υποστεί επεξεργασία από άλλες εναλλακτικές (π.χ. MPE).

Επειδή αυτή η μέθοδος ενθυλάκωσης έχει ως βάση το MPEG-2, είναι κατάλληλη για να ενσωματωθεί στην αρχιτεκτονική του DVB προσφέροντας έτσι ένα μονόδρομο (simplex) κανάλι επικοινωνίας. Το ULE έχει υποστήριξη για επίγεια (DVB-T), δορυφορική (DVB-S) και καλωδιακή (DVB-C) τηλεόραση. Μέσω αυτών των προτύπων μπορούν επίσης να δημιουργηθούν δικατευθηντικα (duplex) κανάλια αφού στο DVB προτείνονται πολλά είδη καναλιών επιστροφής όπως δυο δρόμων satellite links (ETSI-RCS) και dial-up modem links (RFC3077).

4.1 Χαρακτηριστικά

Το ULE περιγράφεται στο RFC 4326 [3] της IETF. Μέχρι σήμερα (2007) δεν έχει γίνει standard. Το πρότυπο περιγράφει έναν μηχανισμό ενθυλάκωσης

ο οποίος καθιστά δυνατή την μεταφορά IPv4 και IPv6 δεδομενογραφημάτων καθώς και πακέτων άλλων δικτυακών πρωτοκόλλων κατευθείαν πάνω στον MPEG-2 συρμό μεταφοράς με την μορφή TS Private Data. Το RFC 4326 προδιαγράφει την βασική μορφή την ενθυλάκωσης η οποία υποστηρίζει μια μορφή επέκτασης που επιτρέπει, μέσω επιπρόσθετης πληροφορίας στην κεφαλίδα, επαυξημένη λειτουργικότητα.



Σχήμα 4.1: Μια τυπική μορφή ενός SNDU.

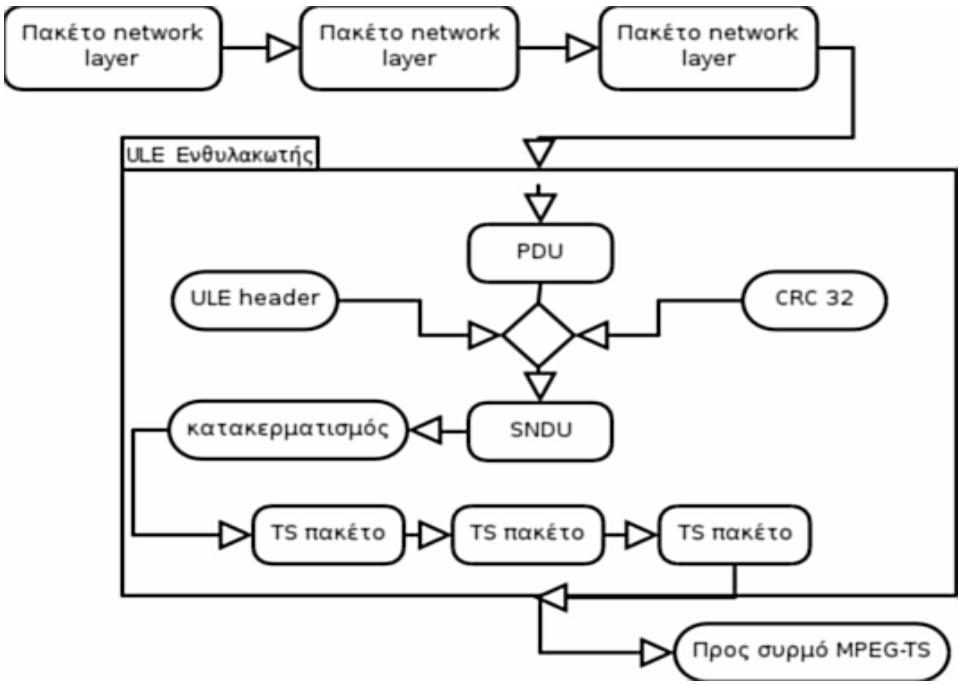
Η μέθοδος της ULE ενθυλάκωσης μπορεί να αναλυθεί σε τρία βασικά στάδια. Αρχικά πρέπει να δημιουργήσουμε ή με κάποιον τρόπο να ανακτήσουμε την πληροφορία που θα επεξεργαστούμε. Αυτή υπάρχει στην μορφή πακέτων επιπέδου δικτύου, όπως για παράδειγμα πλαίσια Ethernet και IP δεδομενογραφήματα. Τα πακέτα αυτά, τα οποία ονομάζουμε PDUs (Protocol Data Units), οδηγούνται στην είσοδο του ενθυλακωτή.

Στην συνέχεια ο ενθυλακωτής μορφοποιεί κάθε PDU σε ένα πακέτο SNDU (SubNetwork Data Unit) το οποίο και αποτελεί ένα ULE πακέτο. Η μορφοποίηση κάθε πακέτου περιλαμβάνει την προσθήκη της κεφαλίδας ενθυλάκωσης πριν το PDU και στο τέλος μια ακολουθία ελέγχου ακεραιότητας τεσσάρων Bytes. Ο έλεγχος ακεραιότητας επιτυγχάνεται με έναν αλγόριθμο CRC32 με τα χαρακτηριστικά που περιγράφονται στο πρότυπο.

Αφού έχει κατασκευαστεί και το SNDU δεν μένει παρά να κατακερματισθεί ώστε να μοιραστεί στο ωφέλιμο φορτίο ενός ή περισσοτέρων MPEG-TS πακέτων. Ο συρμός που δημιουργείται έτσι μπορεί να ενσωματωθεί σε ένα λογικό κανάλι ενός TS συρμού και να τον ακολουθήσει σε μια αλυσίδα εκπομπής DVB έτσι ώστε να φτάσει στον τελικό χρήστη. Εκεί, ακολουθώντας την

αντίστροφη διαδικασία, μπορούμε να ανακτήσουμε την ροή πακέτων που με την μέθοδο της ULE ενθυλάκωσης είχαμε συμπεριλάβει στο εκπεμπόμενο DVB σήμα.

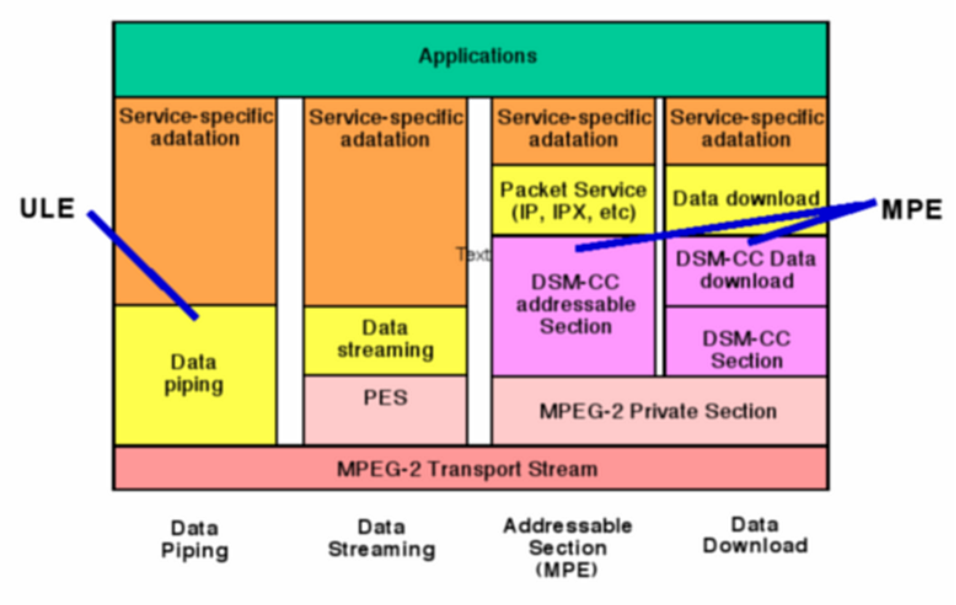
Με αυτήν την τεχνική κατά νου θα πρέπει να είναι προφανές ότι με αυτήν την απλή διαδικασία θα μπορούσε κανείς να περάσει πάνω από ένα οποιοδήποτε δίκτυο DVB δεδομένα με την μορφή δικτυακής κίνησης. Αν τώρα γίνει χρήση και ενός καναλιού επιστροφής, τότε μπορούμε να προσφέρουμε στον τελικό χρήστη μια απεριόριστη γκάμα αμφίδρομων, πραγματικά διαδραστικών υπηρεσιών. Ακόμα μπορούμε να χρησιμοποιήσουμε το δίκτυο του DVB σαν ένα backbone δίκτυο αφού για παράδειγμα σε μια πόλη όπου έχω εκπομπή DVB-T ο ρυθμός δεδομένων που εκπέμπεται μέσω των UHF είναι της τάξης των 2 Gbps, μια διόλου ευκαταφρόνητη ποσότητα για τα δεδομένα μιας μέσης πόλης.



Σχήμα 4.2: Το διάγραμμα λειτουργίας ενός ULE ενθυλακωτή.

4.2 Σύγκριση

Στις μέρες μας η κύρια μέθοδος ενθυλάκωσης δεδομένων IP σε έναν MPEG2-TS συρμό, ώστε να μπορεί να εκπεμφθεί μέσω μιας πλατφόρμας ψηφιακής τηλεόρασης, είναι η MPE (Multi Protocol Encapsulation). Παρολαυτά η σχεδίαση του MPE αποδεικνύεται ότι δεν είναι η ιδανική για εφαρμογές IP over DVB καθώς χρησιμοποιεί πολλή πληροφορία η οποία δεν χρειάζεται με αποτέλεσμα να έχουμε σπατάλη εύρους ζώνης.



Σχήμα 4.3: Το ULE και το MPE στα επίπεδα του DVB.

Τώρα το ULE προτείνεται σαν νέα εναλλακτική μέθοδος έναντι του MPE, προσφέροντάς μας απλούστευση, αυξημένη απόδοση και παραμετροποίηση. Αυτό έχει επιβεβαιωθεί και με απευθείας μετρήσεις σε πλατφόρμα DVB ώστε να επιβεβαιωθεί η θεωρία [2, 3, 5].

Το ULE σε σύγκριση με το MPE μπορεί να επιτύχει μεγαλύτερες ταχύτητες μετάδοσης ωφέλιμης πληροφορίας για ορισμένες εφαρμογές. Αυτό οφείλεται στο σημαντικά μικρότερο overhead που απαιτεί για την μετάδοση και είναι φανερό

RTP/AMR Overhead	
MPE Padded	71%
MPE Packed	23%
MPE/LLC	31%
ULE+D	21%
ULE	13%
Bridged MPE	41%
Bridged ULE	29%

Πίνακας 4.1: Το overhead που απαιτείται για ορισμένα είδη μεταδόσεων.

και από τον πίνακα 4.1. Έτσι σε πολλές υπηρεσίες όπου η εξοικονόμηση εύρους ζώνης είναι μείζονος σημασίας το ULE είναι προτιμότερο από το MPE.

4.3 Υποστήριξη

Αν και γύρω από το ULE υπάρχει σημαντική έρευνα σε εξέλιξη και αρκετό ενδιαφέρον, ακόμα δεν έχει γίνει standard λόγω του ότι η εμφάνισή του στην επιστημονική κοινότητα (RFC 4342) είναι σχετικά πρόσφατη. Αυτό έχει ως αποτέλεσμα να μην υπάρχει η απαραίτητη τεχνογνωσία και ποικιλία έτοιμων λύσεων υλικού ακόμα. Παρόλα αυτά ο πυρήνας του Linux ήδη από το 2002 και μετά υποστηρίζει out-of-the-box το ULE. Έτσι στον τομέα των δεκτών υπάρχει σχεδόν καθολική υποστήριξη αφού οποιαδήποτε πλατφόρμα μπορεί να τρέξει Linux.

Κεφάλαιο 5

Σχεδιασμός

5.1 Απαιτήσεις υλικού

Σε μια πρώτη προσέγγιση του προβλήματος μπορεί κανείς να καταλήξει σε μία σκιαγράφηση των απαιτήσεων από πλευράς υλικού. Ας δούμε πως αναλύεται το πρόβλημα στις συνιστώσες του.

Κατ΄ αρχήν ένας χρήστης ο οποίος επιθυμεί να στείλει δικτυακά δεδομένα μέσω ψηφιακής τηλεόρασης πρέπει τα δεδομένα του να αποκτήσουν πρόσβαση στην αλυσίδα εκπομπής ώστε να μπορέσουν να μεταδοθούν. Αυτό μας οδηγεί στην απαίτηση ύπαρξης ενός κόμβου ο οποίος έχει δυνατότητα επικοινωνίας και με τον χρήστη αλλά και με την αλυσίδα εκπομπής. Επίσης αφού ο χρήστης στέλνει δικτυακά δεδομένα τα οποία στο επίπεδο μεταφοράς χρησιμοποιούν το πρωτόκολλο IP, πρέπει η διασύνδεση του χρήστη με τον κόμβο να χρησιμοποιεί ένα πρωτόκολλο το οποίο να μπορεί να μεταφέρει IP δεδομένα. Τα πιο κατάλληλα θα μπορούσαν να θεωρηθούν το Ethernet (ενσύρματο ή ασύρματο) ή το PPP.

Από την μεριά του ο κόμβος για να επικοινωνήσει με την αλυσίδα εκπομπής χρειάζεται να κάνει χρήση ενός από τα πρωτόκολλα που χρησιμοποιούν συσκευές αυτής της κλάσης για να ανταλλάξουν δεδομένα μεταξύ τους. Τα δύο ευρύτερα χρησιμοποιούμενα είναι το ASI και το SDI. Έτσι προκύπτει η απαίτηση ο κόμβος να διαθέτει ένα τέτοιο interface.

Τέλος, για να μπορέσει να γίνει κατανοητό αυτό που στέλνει ο κόμβος στην

πλατφόρμα πρέπει να μετατραπεί από IP πακέτα σε MPEG συρμούς δεδομένων. Έτσι ο κόμβος που θα υλοποιηθεί – ο οποίος από τώρα και στο εξής θα αποκαλείται ULE Encapsulator – θα έχει και την υποχρέωση της μετατροπής αυτής η οποία πραγματοποιείται με την μέθοδο της ULE ενθυλάκωσης.

Έχοντας αυτά κατά νου είμαστε σε θέση να προσδιορίσουμε με ακρίβεια τις απαιτήσεις που έχει ο κόμβος που χρειαζόμαστε από πλευράς υλικού.

5.2 Σχεδιασμός δικτύου

Στο δίκτυο που πρέπει να υλοποιηθεί, κεντρικό ρόλο έχει ο ULE ενθυλακωτής. Αυτός είναι ο συνδετικός κρίκος ανάμεσα στην αλυσίδα εκπομπής DVB-T και στον χρήστη. Ο χρήστης πρέπει, με κάποιον τρόπο, να αλληλεπιδρά με τον ενθυλακωτή, ώστε να μπορεί να του μεταφέρει τα δεδομένα που πρέπει να διασχίσουν το δίκτυο της ψηφιακής τηλεόρασης. Η διασύνδεση αυτή μπορεί να γίνει με ένα δικτυακό interface.

Αφού, τώρα, ο ενθυλακωτής λάβει τα δικτυακά δεδομένα και τα μετατρέψει σε MPEG2-TS πρέπει να τα μεταφέρει στην αλυσίδα εκπομπής, μέσω ενός ASI interface, ώστε να διαμορφωθούν και να εκπεμφθούν.

Από την μεριά του χρήστη ο οποίος λαμβάνει την ροή του DVB, τώρα, πρέπει να υπάρχει ένας DVB-T receiver ώστε να λάβει το σήμα. Ακόμα, το σύστημα του χρήστη, πρέπει να έχει ένα λειτουργικό το οποίο να μπορεί να αποκωδικοποιήσει μία ροή ULE μέσα από ένα MPEG-2 TS συρμό δεδομένων και να το μεταφράσει σε δικτυακή κίνηση. Αυτό αποτελεί μία απλή διαδικασία, αν το λειτουργικό του χρήστη ανήκει στην οικογένεια των Linux.

Ένα τελευταίο βήμα που πρέπει να εφαρμοστεί, ώστε να έχουμε ένα πραγματικό δίκτυο πάνω από το DVB-T, είναι να βρούμε έναν τρόπο, ώστε δεδομένα από τον δεύτερο χρήστη, να είναι δυνατόν να φτάσουν στον πρώτο. Αυτό είναι αναγκαίο, δεδομένης της μονόδρομης φύσης του DVB-T πρωτοκόλλου. Για τον σκοπό της έρευνας αυτής της εργασίας, η λύση σε αυτό το πρόβλημα είναι τετριμμένη και μπορεί να εξομοιωθεί με την χρήση ενός κοινού δικτυακού interface.

5.3 Σχεδιασμός εφαρμογής

Τα δεδομένα προς ενθυλάκωση αποστέλνονταν στον ενθυλακωτή μέσω ενός Ethernet interface. Έτσι η εφαρμογή ενθυλάκωσης όφειλε να τα λάβει διαβάζοντας τα πακέτα που κατευθύνονταν προς το μηχάνημα.

Ακόμη, ο ενθυλακωτής οφείλει να τελεί ταυτόχρονα αρκετές διαδικασίες. Πρέπει να μπορεί να λαμβάνει ταυτόχρονα δεδομένα από πολλούς χρήστες και να τα πολυπλέκει στον ίδιο συρμό δεδομένων. Ακόμη είναι υποχρεωτικό, ανά τακτά χρονικά διαστήματα, να δημιουργεί και να ενσωματώνει πακέτα ελέγχου στον συρμό δεδομένων.

Άλλη μία υποχρέωση είναι να μπορεί να αλληλεπιδράσει με το ASI interface, που υπάρχει στο μηχάνημα, ώστε να είναι δυνατή η αποστολή του συρμού που δημιουργείται μέσω αυτού.

Επίσης, μια πολύ σημαντική απαίτηση είναι η υποστήριξη των packing και padding τεχνικών για την ενθυλάκωση.

Όλα τα δεδομένα που λαμβάνονται και εκπέμπονται από τον ενθυλακωτή, πρέπει να αποθηκεύονται προσωρινά στην μνήμη με την βοήθεια κάποιου είδους δομής δεδομένων. Αυτό είναι απαραίτητο για να είναι δυνατή η αποστολή του σωστού αριθμού null packets ώστε να παραμένει σταθερό το bit rate όπως και για να είναι εφικτή η μέθοδος ενθυλάκωσης packing.

Κεφάλαιο 6

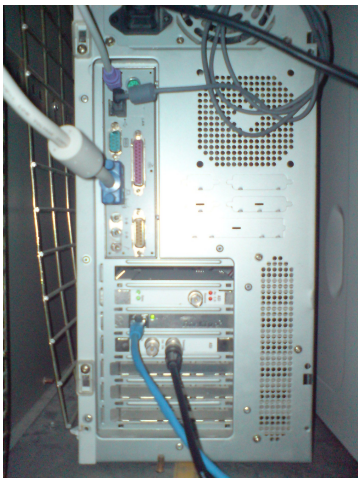
Υλοποίηση

6.1 Υλοποίηση κόμβου

Ο υπολογιστής που χρησιμοποιήθηκε για τον ρόλο του ULE Encapsulator ήταν ένας Pentium III με 512 MBytes SD RAM, εφοδιασμένος με μια Ethernet κάρτα δικτύου και μια DVB Master III Tx κάρτας με ASI interface ώστε να είναι δυνατή η επικοινωνία με την αλυσίδα εκπομπής.

Το λειτουργικό που έτρεχε ο ULE Encapsulator ήταν το Ubuntu Linux στην έκδοση 'Dapper Drake' χωρίς να χρειαστεί καμία ιδιαίτερη παραμετροποίηση. Το μόνο αναγκαίο ήταν η εγκατάσταση των οδηγών της κάρτας DVB Master III Tx, οι οποίοι παρέχονται από την κατασκευάστρια εταιρία.

Πριν μπορέσει, βέβαια, κανείς να στείλει κάτι από την κάρτα DVB Master III Tx, πρέπει να γίνουν οι απαραίτητες ρυθμίσεις. Οι σημαντικότερες ρυθμίσεις είναι ο προσδιορισμός του μεγέθους και του πλήθους των buffers, οι οποίοι θα χρησιμοποιηθούν από την κάρτα. Οι μεταβολή των τιμών αυτών επιδρά στην επίδοση του συστήματος. Έτσι για να υπάρξει βέλτιστη επίδοση πρέπει να βρεθούν οι βέλτιστες τιμές. Έπειτα από μερικές πειραματικές μετρήσεις προσδιορίστηκε το πλήθος και το μέγεθος των buffers να είναι 6 και 1182 αντίστοιχα.

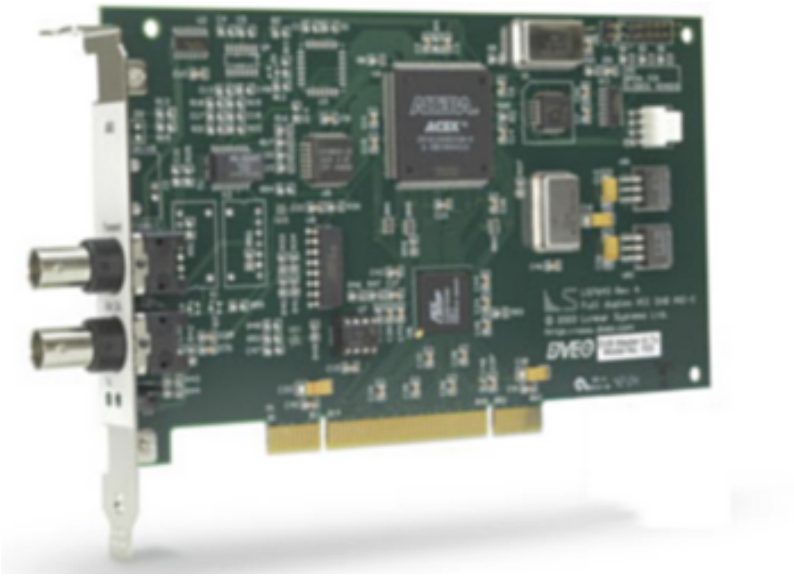


Σχήμα 6.1: Η είσοδος δικτύου και η ASI έξοδος του ULE ενθυλακωτή.

6.2 Υλοποίηση δικτύου

Προκειμένου να γίνουν μετρήσεις και να αξιολογηθεί η απόδοση του ULE Encapsulator σχεδιάστηκε και υλοποιήθηκε ένα πρότυπο δίκτυο. Αυτό αποτελούνταν από δύο χρήστες, τον ULE ενθυλακωτή και την αλυσίδα εκπομπής ψηφιακής τηλεόρασης.

Ο υπολογιστής που ανέλαβε τον ρόλο των χρηστών, και από τώρα θα αναφέρεται ως client, ήταν ένας Pentium 4/ 3GHz με 1GB RAM. Το λειτουργικό που έτρεχε ήταν Debian testing. Για να είναι δυνατή η αποστολή δεδομένων στον ενθυλακωτή, χρησιμοποιήθηκε ένα δικτυακό interface στον client (eth0) με IP διεύθυνση 172.16.0.1. Αυτό συνδέθηκε στο ίδιο switch με το δικτυακό interface του ενθυλακωτή (eth0) το οποίο είχε IP διεύθυνση 172.16.0.236. Ακόμη για να προωθηθεί η κίνηση από τον ενθυλακωτή σε ένα άλλο interface του client, υπήρχε μια κάρτα λήψης DVB-T σήματος. Με την βοήθεια του εργαλείου dvbnet η ενθυλακωμένη IP πληροφορία, που βρισκόταν σε ένα προκαθορισμένο PID, αποχωρίστηκε από τον λαμβανόμενο συρμό δεδομένων και προωθήθηκε σε ένα ιδεατό δικτυακό interface, το dvb0_0. Με αυτόν τον τρόπο επιτεύχθηκε η προώθησή των δεδομένων σε ένα άλλο interface μέσω μιας διεργασίας διάφανης προς τον χρήστη και το λειτουργικό.



Σχήμα 6.2: Η κάρτα DVB Master III Tx.

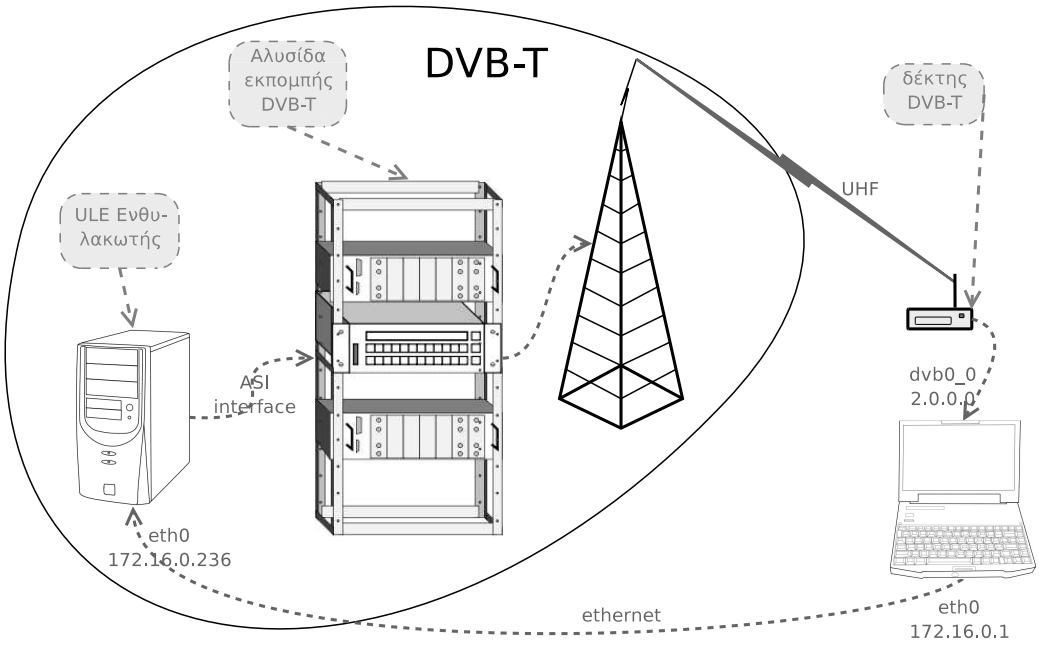
Για να φτάσουν όμως τα δεδομένα από τον ενθυλακωτή στην κάρτα λήψης του client, πρέπει να μετατρέψει τα δεδομένα που έλαβε από τον client από το interface eth0 σε έναν συρμό δεδομένων με την μέθοδο της ULE ενθυλάκωσης. Αυτό είναι αναγκαίο ώστε να είναι εφικτή η προώθησή του στην αλυσίδα εκπομπής η οποία περιγράφεται αναλυτικότερα παρακάτω. Αφού τα δεδομένα φτάσουν στην αλυσίδα εκπομπής, αυτή αναλαμβάνει να διαμορφώσει τον συρμό δεδομένων ώστε να παραχθεί το DVB-T σήμα προς εκπομπή.

Έπειτα το διαμορφωμένο σήμα προωθείται στην κεραία για την εκπομπή του στην μπάντα των UHF. Μετά από αυτό το στάδιο οποιοσδήποτε χρήστης, ο οποίος βρίσκεται εντός της περιοχής κάλυψης, μπορεί να έχει πρόσβαση στα δεδομένα, αρκεί φυσικά να διαθέτει ένα σύστημα λήψης σήματος επίγειας ψηφιακής τηλεόρασης, ένα σύστημα αποενθυλάκωσης των ULE δεδομένων σε IP πακέτα. Αυτά τα απαιτούμενα χαρακτηριστικά είναι υπαρκτά στον client όπως, έχουμε αναφέρει, άρα ο client είναι σε θέση να λάβει τα εκπεμπόμενα δεδομένα.

Σύμφωνα με τα παραπάνω είναι δυνατόν με τον διαθέσιμο εξοπλισμό να κατασκευαστεί το δίκτυο της εικόνας 6.3.

Η ιδέα για την υλοποίηση αυτής της αρχιτεκτονικής συνοψίζεται στα εξής:

- ο client στέλνει στον ενθυλακωτή μέσω του interface eth0 τα δικτυακά



Σχήμα 6.3: Η αρχιτεκτονική του δικτύου που χρησιμοποιήθηκε για τις μετρήσεις.

πακέτα (με άλλα λόγια, την πληροφορία που επιθυμεί να εκπέμψει).

- ο ενθυλακωτής συλλέγει αυτά τα δεδομένα, τα ενθυλακώνει σε έναν MPEG2-TS συρμό δεδομένων και τα προωθεί, μέσω του ASI interface που διαθέτει στην πλατφόρμα εκπομπής ψηφιακής τηλεόρασης.

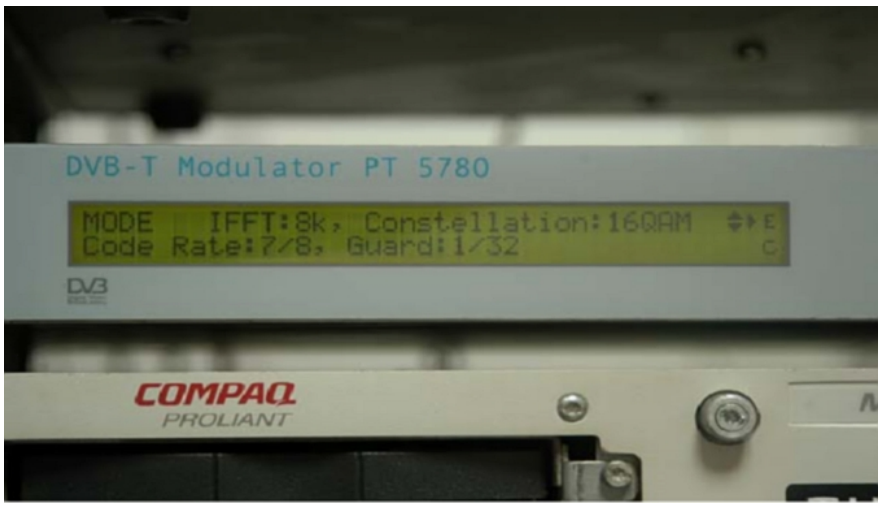
- ο συρμός δεδομένων φτάνει στον διαμορφωτή όπου και μετατρέπεται σε σήμα DVB-T. Το σήμα, αφού ενισχυθεί, προωθείται στην κεραία και εκπέμπεται στα UHF.

- ο client, έχοντας στην διάθεσή μία κοινή κάρτα λήψης DVB-T, λαμβάνει το σήμα και, μέσω ειδικού software[1], δημιουργείται ένα εικονικό δικτυακό interface (dvb0_0) όπου και αναδημιουργούνται τα δικτυακά πακέτα που είχαν αρχικά εκπεμφθεί.

[1] η διαδικασία περιγράφεται στο κεφάλαιο 8.1.3.

6.2.1 Αλυσίδα εκπομπής

Από την αλυσίδα εκπομπής ψηφιακής τηλεόρασης που διαθέτει το εργαστήριο Έρευνας και Ανάπτυξης Τηλεπικοινωνιακών Συστημάτων ΠΑΣΙΦΑΗ, χρησιμοποιήθηκε ένας DVB-T modulator και ένας ενισχυτής. Έτσι έγινε δυνατή η εκπομπή σήματος ψηφιακής τηλεόρασης το οποίο περιείχε τα IP δεδομένα πού έλαβε σαν είσοδο ο DVB-T modulator.



Σχήμα 6.4: Ο DVB-T διαμορφωτής.

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 21.11Mbps. Επίσης ως φέροντα σήματα δηλώνονται 8K . Η συχνότητα του σήματος εκπομπής είναι στα 538MHz (κανάλι 29) με επίπεδο δύναμης (power level) 0dBm και το συγκεκριμένο κανάλι έχει εύρος 8MHz.

6.3 Υλοποίηση εφαρμογής

6.3.1 Buffers

Ένα πολύ σημαντικό μέρος της εφαρμογής, είναι η δημιουργία buffers όπου θα γίνεται η προσωρινή αποθήκευση των πακέτων ώστε να είναι δυνατή η ασύγχρονη επεξεργασία τους. Αυτό είναι απαραίτητο ώστε η εφαρμογή να μπορεί να λειτουργήσει σωστά και να επιτευχθεί η μέγιστη δυνατή απόδοση.

Στην ουσία οι buffers που υλοποιήθηκαν είναι κυκλικές ουρές. Πριν χρησιμοποιηθούν αρχικοποιούνται έχοντας ως ορίσματα το μέγιστο μέγεθος δεδομένων για κάθε θέση και το μέγιστο πλήθος θέσεων που μπορεί να υποστηρίξει ο buffer.

6.3.2 Επικοινωνία με την κάρτα

Προκειμένου να προωθηθούν τα πακέτα στην κάρτα το πρόγραμμα χρησιμοποιεί κώδικα ο οποίος παρέχεται με τους οδηγούς της κάρτας και φροντίζουν για την αρχικοποίηση της.

6.3.2.1 Αρχικοποίηση

Για την αρχικοποίηση της κάρτας τροποποιήθηκε το αρχείο txtest.c ώστε αντί να στέλνει την κίνηση που λαμβάνει σαν είσοδο, απλά να αρχικοποιεί την κάρτα με τα ορίσματα που παρέχονται κατά την κλήση της εφαρμογής. Τα ορίσματα αυτά είναι η κάρτα η οποία θα χρησιμοποιηθεί (συνήθως /dev/asitx0) και ρυθμίσεις για να επιτευχθεί το επιθυμητό bit rate.

6.3.2.2 Αποστολή δεδομένων

Η αποστολή ενός TS πακέτου γίνεται από την συνάρτηση send_tp η οποία φαίνεται παρακάτω.

```
// tp_send sends the transport packet to output
void send_tp(unsigned char *tp) {
  fd_set writeset;
  int result, bytes;

  do {
    FD_ZERO(&writeset);
    FD_SET(fd, &writeset);
    result = select(fd + 1, NULL, &writeset, NULL, NULL);
  } while (result == -1 );
```

```
  if (result > 0) {
    if (FD_ISSET(fd, &writeset)){
      if ( (bytes= write(fd, tp, 188)) < 0){
        printf("\nError writing.\n");
      } else {
//          printf("wrote %d bytes \n", bytes);
      }
    }
  }
  return;
}
```

Μετά την αρχικοποίηση της κάρτας, η οποία εμφανίζεται σαν αρχείο στο λειτουργικό, μπορεί πλέον να προσπελαστεί παρόμοια με ένα κοινό αρχείο. Έτσι είναι δυνατόν να μεταφερθούν δεδομένα στην κάρτα κάνοντας χρήση κλασικών συναρτήσεων διαχείρισης αρχείων της C.

6.3.3 Συλλογή IP πακέτων

Βασικό μέρος της λειτουργίας του ενθυλακωτή είναι η ανάκτηση των IP πακέτων από το δίκτυο ώστε αυτά να υποστούν την διαδικασία της ενθυλάκωσης. Ένα αυτόνομο thread αναλαμβάνει αυτήν την διεργασία η οποία αναλύεται παρακάτω.

6.3.3.1 Βιβλιοθήκη libpcap

Για την ανάκτηση πακέτων από ένα δίκτυο υπάρχει ένα API το οποίο αναφέρεται ως pcap. Η υλοποίηση αυτού του API για Unix συστήματα είναι γνωστή ως libpcap και είναι σχεδιασμένη για χρήση από τις γλώσσες προγραμματισμού C και C++. Έτσι κάνοντας χρήση του libpcap η εφαρμογή που αναπτύχθηκε είναι σε θέση να συλλέξει από το δίκτυο τα προς ενθυλάκωση IP πακέτα.

Για να αποφευχθεί η σύγχυση με άλλα πακέτα του δικτύου, η συλλογή των πακέτων περιορίστηκε με την χρήση ενός φίλτρου. Το φίλτρο αυτό επέτρεπε την συλλογή μόνο IP πακέτων από ένα συγκεκριμένο αποστολέα.

6.3.3.2 IP buffer

Καθώς, για καλύτερα αποτελέσματα και για να είναι δυνατή η μέθοδος ενθυλάκωσης packing, τα IP πακέτα που συλλέγονται χρησιμοποιούνται ανά συγκεκριμένα χρονικά διαστήματα, πρέπει κάπου να γίνεται η προσωρινή αποθήκευση των πακέτων. Γι' αυτόν τον σκοπό κατασκευάστηκαν οι buffers που περιγράφονται στο κεφάλαιο 6.3.1.

Η αρχικοποίηση του buffer γίνεται με την παρακάτω εντολή. Οι τιμές σημαίνουν ότι το μέγιστο μέγεθος πακέτου είναι 1540 και υπάρχει δυνατότητα για αποθήκευση ως και χιλίων πακέτων. Δεδομένου ότι ο buffer αυτός αποθηκεύει τα πακέτα που φτάνουν σε διάστημα 5ms οι τιμές αυτές έχουν υπολογιστεί ώστε να δίνουν αρκετό χώρο για το εύρος της κίνησης του δικτύου.

```
init_buffer(&ip_buff, 1000, 1560);
```

6.3.3.3 Το thread συλλογής πακέτων

Όπως προαναφέρθηκε, η συλλογή πακέτων γίνεται σε ένα αυτόνομο thread. Η υποχρέωση αυτού του thread είναι η συλλογή των επιθυμητών πακέτων και η προώθησή τους στον IP buffer. Έτσι τα πακέτα που συλλέγονται παραμένουν στην μνήμη μέχρι να έρθει η ώρα να χρησιμοποιηθούν.

6.3.4 Ενθυλάκωση

Η ενθυλάκωση είναι η κύρια λειτουργία του συστήματος που αναπτύχθηκε. Στην ουσία είναι η διαδικασία της μετατροπής IP πακέτων σε πακέτα MPEG-TS ώστε να είναι δυνατή η προώθησή τους μέσω της ψηφιακής τηλεόρασης.

6.3.4.1 Αλγόριθμος ενθυλάκωσης

Εδώ παρουσιάζεται ο αλγόριθμος της διαδικασίας της ενθυλάκωσης.

Ενθυλάκωση SNDU Αν ο ενθυλακωτής δεν έχει στείλει προηγουμένως ένα TS πακέτο σε ένα συγκεκριμένο πρόγραμμα ή μετά από ένα διάσημε αδράνειας, αρχίζει να στέλνει ένα SNDU στο πρώτο διαθέσιμο TS πακέτο. Αυτό το πρώτο πακέτο ΠΡΕΠΕΙ να φέρει την τιμή 1 στο πεδίο PUSI. ΠΡΕΠΕΙ επίσης να φέρει τιμή 0 στο πεδίο Payload Pointer, ώστε να είναι φανερό ότι το SNDU αρχίζει αμέσως μετά τον Payload Pointer.

Ο ενθυλακωτής ΠΡΕΠΕΙ να εξασφαλίσει ότι ο Continuity Counter που υπάρχει στην κεφαλίδα του TS πακέτου έχει καθοριστεί σύμφωνα με το πρότυπο ISO-MPEG2. Αυτή η τιμή πρέπει να αυξάνεται κατά ένα (modulo 16) για κάθε επακόλουθο πακέτο που περιέχει ένα μέρος ή ολόκληρο το SNDU που στέλνεται σε ένα πρόγραμμα.

Ο ενθυλακωτής ΜΠΟΡΕΙ να αποφασίσει να μην στείλει άλλο SNDU αμέσως, ακόμη και αν υπάρχει διαθέσιμος χώρος σε ένα μη πλήρες TS πακέτο. Αυτή η διαδικασία είναι γνωστή ως padding. Ο End Indicator πληροφορεί ότι δεν υπάρχουν άλλα SNDU σε αυτό το TS πακέτο. Ο End Indicator ακολουθείται από μηδέν η περισσότερα μη χρησιμοποιούμενα bytes μέχρι το τέλος του TS πακέτου. Όλα τα μη χρησιμοποιούμενα bytes ΠΡΕΠΕΙ να πάρουν την τιμή 0xFF, ακολουθώντας την τρέχουσα πρακτική του MPEG-2. Η διαδικασία του padding ανταλλάσσει μειωμένη αποδοτικότητα με βελτιωμένη καθυστέρηση.

Εναλλακτικά, αν υπάρχουν περισσότερα πακέτα εν αναμονή στον ενθυλακωτή, και ένα πακέτο έχει αρκετό υπολειπόμενο χώρο μεταφοράς, ο ενθυλακωτής μπορεί να συνεχίσει το προηγούμενο SNDU με ένα επόμενο χρησιμοποιώντας το επόμενο διαθέσιμο byte στο TS πακέτο. Αυτή η διαδικασία αποκαλείται packing.

Διαδικασία για packing και padding. Μόλις ο ενθυλακωτής ολοκληρώσει την ενθυλάκωση ενός SNDU υπάρχουν πέντε διαφορετικές περιπτώσεις που μπορεί να συμβούν:

1. Αν το πακέτο TS δεν διαθέτει άλλο χώρο, ο ενθυλακωτής μεταδίδει αυτό το πακέτο. Έπειτα ξεκινάει η μετάδοση ενός νέου SNDU σε ένα νέο TS πακέτο.

2. Αν το TS πακέτο που περιέχει το τελευταίο κομμάτι ενός SNDU έχει ένα byte αχρησιμοποίητου χώρου, ο ενθυλακωτής ΠΡΕΠΕΙ να τοποθετήσει την τιμή 0xFF σε αυτό το τελευταίο byte και να μεταδώσει αυτό το TS πακέτο.

3. Αν το TS πακέτο που περιέχει το τελευταίο κομμάτι ενός SNDU έχει ακριβώς δύο byte αχρησιμοποίητου χώρου, και το πεδίο PUSI ΔΕΝ έχει ήδη τιμή 1, ο ενθυλακωτής ΠΡΕΠΕΙ να τοποθετήσει την τιμή 0xFFFF σε αυτά τα τελευταία byte, ώστε να ορίσει έναν δείκτη τέλους, και να μεταδώσει αυτό το TS πακέτο. Έπειτα ο ενθυλακωτής ΠΡΕΠΕΙ να ξεκινήσει την μετάδοση του επόμενου SNDU σε ένα νέο TS πακέτο.

4. Αν το TS πακέτο έχει πάνω από δυο byte αχρησιμοποίητου χώρου, ο ενθυλακωτής ΜΠΟΡΕΙ να μεταδώσει αυτό το μη πλήρες TS πακέτο αλλά ΠΡΕΠΕΙ πρώτα να τοποθετήσει την τιμή 0xFF σε όλα τα εναπομείναντα αχρησιμοποίητα byte. Έπειτα ο ενθυλακωτής ΠΡΕΠΕΙ να ξεκινήσει την μετάδοση του επόμενου SNDU σε ένα νέο TS πακέτο.

5. Αν τουλάχιστον δύο byte είναι διαθέσιμα (με άλλα λογία, τρία byte αν το PUSI ΔΕΝ έχει πάρει τιμή 1 ήδη, και δύο byte αν έχει πάρει τιμή 1 ήδη), ο ενθυλακωτής ΜΠΟΡΕΙ να ενθυλακώσει επιπλέον PDU ευρισκόμενα εν αναμονή, με το να αρχίσει το επόμενο SNDU στο επόμενο διαθέσιμο byte του TS πακέτου. Όταν ο ενθυλακωτής συνεχίζει να πακετάρει S-NDU σε ένα TS πακέτο όπου το PUSI δεν έχει πάρει τιμή 1, το PUSI ΠΡΕΠΕΙ να πάρει τιμή 1, και ένας 8-bit Payload Pointer ΠΡΕΠΕΙ να εισαχθεί στο πρώτο byte που ακολουθεί την κεφαλίδα του TS πακέτου. Αν δεν υπάρχουν άλλα PDU σε αναμονή, ο ενθυλακωτής ΜΠΟΡΕΙ να περιμένει για επόμενα PDU ώστε να γεμίσει το TS πακέτο. Η μέγιστη χρονική περίοδος που μπορεί να περιμένει ένας ενθυλακωτής, γνωστή ως Packing Threshold, ΠΡΕΠΕΙ να οριοθετηθεί και ΟΦΕΙΛΕΙ να είναι παραμετροποιήσιμη από τον ενθυλακωτή. Αν ΔΕΝ ληφθούν πακέτα μέσα στα όρια του Packing Threshold, ο ενθυλακωτής ΠΡΕΠΕΙ να εισάγει έ-ναν δείκτη τέλους.

6.3.4.2 Υλοποίηση ενθυλακωτή

Στο σχήμα 6.5 φαίνεται το διάγραμμα ροής της διαδικασίας της ενθυλάκωσης χρησιμοποιώντας την μέθοδο packing. Η ενθυλάκωση λαμβάνει χώρα σε ένα αυτόνομο thread.

Βασικός βρόγχος Η ενθυλάκωση γίνεται σε έναν βρόγχο ο οποίος επαναλαμ-βάνεται κάθε 5 ms. Αρχικά για την είσοδο στον βρόγχο ενθυλάκωσης ελέγχε-ται αν ο buffer που κρατάει τα συλλεχθέντα IP πακέτα είναι άδειος – οπότε δεν χρειάζεται να γίνει ενθυλάκωση – ή αν από πριν έχει μείνει ένα SNDU που δεν έχει ολοκληρωθεί η ενθυλάκωσή του οπότε και χρειάζεται να συνεχιστεί η διαδικασία.

Όταν γίνει η είσοδος στον βρόγχο, αν υπάρχει διαθέσιμη αρκετή πληροφορία, αυτή προωθείται σε ένα TS πακέτο, αλλιώς ανακτάται το επόμενο SNDU από τον IP buffer.

Στο επόμενο στάδιο, αν και όσο υπάρχουν δεδομένα πάνω από 184 bytes προς ενθυλάκωση τα προωθεί σε επόμενα TS πακέτα.

Έπειτα, αν έχει επιλεγεί η μέθοδος ενθυλάκωσης padding, γράφονται όσα bytes έχουν απομείνει σε ένα TS πακέτο και το υπόλοιπο γεμίζεται με null bytes (διαδικασία stuffing). Αν η επιλεγμένη μέθοδος ενθυλάκωσης είναι η packing, τότε, αν υπάρχουν άλλα διαθέσιμα SNDU διαβάζονται από τον IP buffer μέχρι να συμπληρωθεί το TS πακέτο. Αν μείνουν bytes που δεν χώρεσαν στο πακέτο, τότε στην επόμενη επανάληψη η ενθυλάκωση ξεκινάει από αυτά. Αν δεν υπάρχουν αρκετά διαθέσιμα SNDU ώστε να γεμίσει το TS πακέτο, τότε ακολουθεί η διαδικασία του stuffing, ώστε να γεμίσει το πακέτο και να προωθηθεί στον TS buffer.

Αφού γίνει αυτή η διαδικασία, γίνεται πάλι είσοδος στον βρόγχο ενθυλάκωσης και επαναλαμβάνεται αυτή η διαδικασία, μέχρι να αδειάσει ο IP buffer.

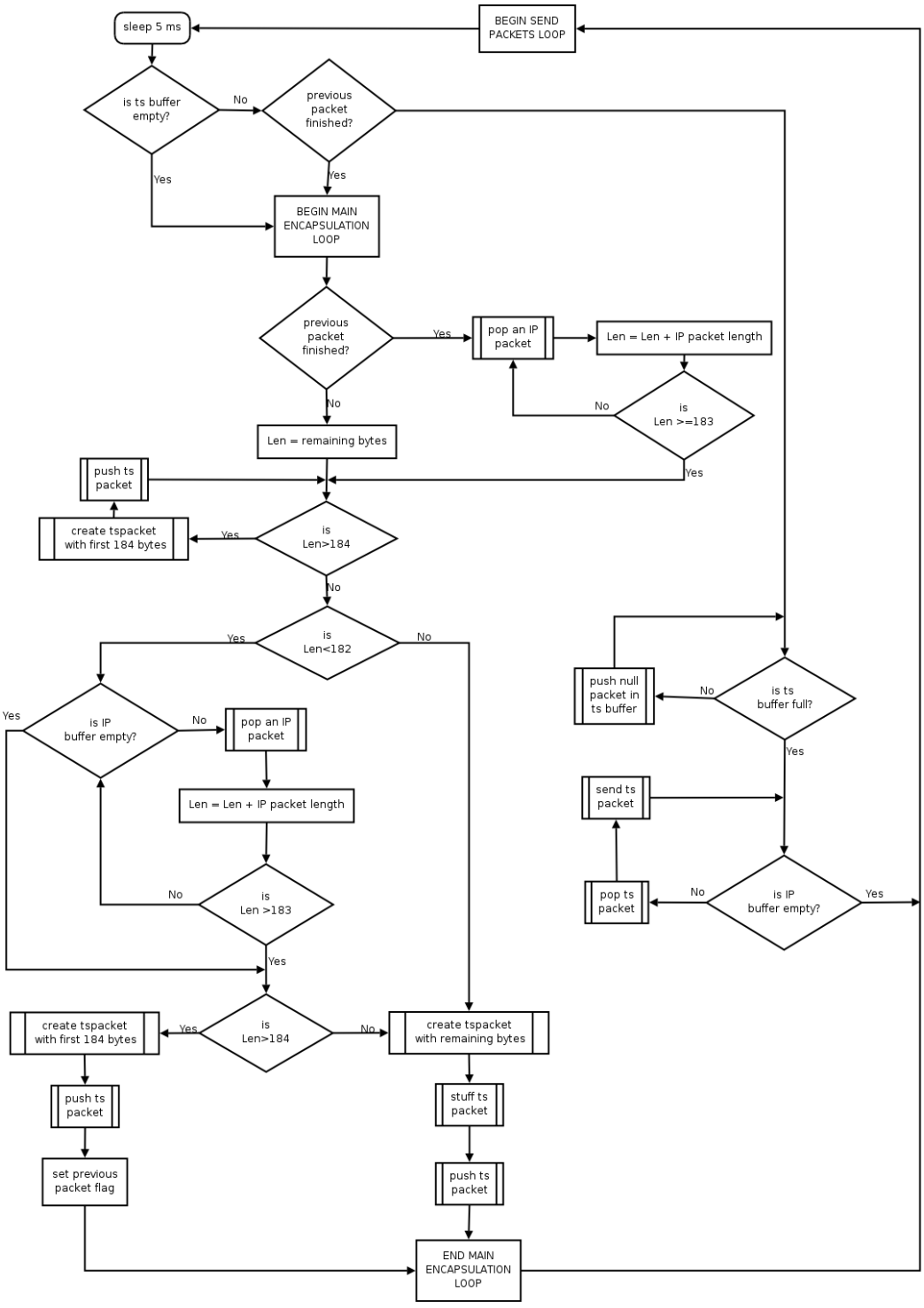
Μόλις τελειώσει ο βρόγχος ενθυλάκωσης, πρέπει να διαπιστωθεί εάν ο TS buffer είναι πλήρης όπως οφείλει να είναι για να διατηρηθεί σταθερό το bit rate. Αν δεν είναι τότε ακολουθεί ένας βρόγχος ο οποίος προωθεί null πακέτα στον TS buffer μέχρι να γεμίσει.

Τέλος, αφού ο TS buffer είναι πλήρης, πρέπει να προωθήσουμε τα πακέτα στην κάρτα. Αυτό γίνεται σε έναν βρόγχο όπου διαβάζονται ένα ένα τα πακέτα από τον TS buffer και προωθούνται στην κάρτα.

Αφού γίνει και αυτό υπάρχει μια αναμονή 5 ms και η ροή του προγράμματος επιστρέφει στον κύριο βρόγχο.

6.3.4.3 Crc checksum

Κάθε πακέτο SNDU περιέχει στα τελευταία 4 Bytes τα οποία δημιουργούνται από έναν αλγόριθμο κυκλικού ελέγχου πλεονασμού (crc checksum) 32 bit. Αυτά τα bit χρησιμοποιούνται για να πιστοποιηθεί στον δέκτη η ακεραιότητα του δεδομενογραφήματος που έλαβε. Το κλειδί που γενεσιουργό πολυώνυμο για τον αλγόριθμο είναι το 0x104C11DB7 και η αρχική τιμή είναι 0xFFFFFFFF.



Σχήμα 6.5: Διάγραμμα ροής για την μέθοδο του packing.

6.3.4.4 TS buffer

Κάθε πακέτο που TS που παράγεται κατά την διαδικασία της ενθυλάκωσης αποθηκεύεται σε έναν buffer ο οποίος έχει αρχικοποιηθεί κατάλληλα για υποστηρίξει την ροή των πακέτων.

6.3.4.5 Αποστολή

Ανά τακτά χρονικά διαστήματα γίνεται η προώθηση των TS πακέτων από τον buffer της εφαρμογής στους buffers της κάρτας. Το bit rate πρέπει να διατηρείται σταθερό και ο buffer της εφαρμογής έχει αρχικοποιηθεί κατάλληλα ώστε, αν αδειάζει ολόκληρος στα προβλεπόμενα χρονικά διαστήματα, να δίνει το επιθυμητό bit rate. Έτσι, πριν προωθηθούν τα πακέτα, έχει προβλεφθεί να γεμίζει ο buffer με null πακέτα και συνεπώς προωθούνται τόσα πακέτα όσα χρειάζεται ακριβώς ώστε να διατηρηθεί το bit rate σταθερό.

Η αποστολή των δεδομένων, τώρα, γίνεται με την συνάρτηση tp_send που περιγράφηκε στο κεφάλαιο 6.3.2.2.

6.3.5 Δημιουργία πακέτων ελέγχου

Προκειμένου να δημιουργηθεί ένας έγκυρος συρμός δεδομένων που θα μπορεί να αναγνωριστεί πρέπει ανά διαστήματα να εκπέμπονται πακέτα ελέγχου. Κάθε πακέτο έχει ειδική σημασία μέσα σε έναν MPEG2 συρμό δεδομένων. Τα πακέτα που χρησιμοποιήθηκαν από την εφαρμογή περιγράφονται παρακάτω.

6.3.5.1 PAT

Το PAT (Program Association Table) περιέχει μία λίστα από τα PID όλων τών PMT πακέτων που περιέχονται στον συρμό δεδομένων. Τα PAT πακέτα έχουν πάντα PID ίσο με 0.

Στην εφαρμογή που αναπτύχθηκε το PAT περιέχει μόνο το PID 48 που είναι το PID του PMT που περιγράφει το μοναδικό πρόγραμμα μέσα στον συρ-

μό δεδομένων που δημιουργείται. Το πρόγραμμα αυτό είναι τα ενθυλακωμένα δεδομένα.

Παρακάτω φαίνεται η μορφή του PAT πακέτου.

```
0000:  47 40 00 12 00 00 b0 09  02 22 c1 00 00 00 05 e0   G@.......".....
0010:  30 b1 22 84 db ff ff ff  ff ff ff ff ff ff ff ff   0.".............
0020:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0030:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0040:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0050:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0060:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0070:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0080:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0090:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
00a0:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
00b0:  ff ff ff ff ff ff ff ff  ff ff ff ff               ............
```

6.3.5.2 PMT

Το PMT (Program Map Table) περιέχει πληροφορίες για προγράμματα. Για κάθε πρόγραμμα υπάρχει ένα PMT το οποίο με την σειρά του εμφανίζεται σε ένα συγκεκριμένο PID. Το PMT περιγράφει ποια PID περιέχουν δεδομένα ή μεταδεδομένα για το συγκεκριμένο πρόγραμμα. Για παράδειγμα για ένα πρόγραμμα το PMT οφείλει να περιέχει το PID που έχει η ροή εικόνας, το PID που έχει η ροή του ήχου κτλ.

Στην εφαρμογή που αναπτύχθηκε το PMT δείχνει το PID όπου υπάρχουν τα ULE ενθυλακωμένα δεδομένα· δηλαδή το PID 0x150.

Παρακάτω φαίνεται η μορφή του PAT πακέτου.

```
0000:  47 40 00 12 00 00 b0 09  02 22 c1 00 00 00 05 e0   G@.......".....
0010:  30 b1 22 84 db ff ff ff  ff ff ff ff ff ff ff ff   0.".............
0020:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0030:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0040:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0050:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0060:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0070:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0080:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0090:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
00a0:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
00b0:  ff ff ff ff ff ff ff ff  ff ff ff ff               ............
```

6.3.5.3 SDT

Το SDT (Service Description Table) περιέχει πληροφορίες όπως το όνομα του προγράμματος και τον παροχέα. Το PID του είναι πάντα 17.

Παρακάτω φαίνεται η μορφή του SDT πακέτου.

```
0000:  47 40 11 17 00 42 b0 26  02 22 c1 00 00 12 34 ff   G@...B.&."....4.
0010:  00 05 fc 80 15 48 13 0c  06 6e 6b 76 6f 72 6e 0a   .....H...nkvorn.
0020:  44 56 42 20 68 61 63 6b  65 72 09 27 80 e2 ff ff   DVB hacker.'....
0030:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0040:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0050:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0060:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0070:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0080:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
0090:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
00a0:  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff   ................
00b0:  ff ff ff ff ff ff ff ff  ff ff ff ff               ............
```

6.3.5.4 Αποστολή

Τα πακέτα αυτά οφείλουν να εκπέμπονται μέσω του MPEG2 συρμού δεδομένων το πολύ κάθε 100ms ώστε να γίνεται η σωστή αναγνώρισή του. Γι΄ αυτόν τον σκοπό υπάρχει ένας βρόγχος ο οποίος αναλαμβάνει αυτή την υποχρέωση. Μέρος της υλοποίησης του βρόγχου από το τμήμα της αποστολής παρατίθεται.

```
...
        patPacket[3] = 0x10 | counter;
        pmtPacket[3] = 0x10 | counter;
        sdtPacket[3] = 0x10 | counter;

        pthread_mutex_lock(&mutexfd);
        send_tp(patPacket);
        send_tp(pmtPacket);
        send_tp(sdtPacket);
        pthread_mutex_unlock(&mutexfd);
        usleep(100000);
        counter = ++counter % 16;
...
```

Κεφάλαιο 7

Έλεγχος

7.1 Εργαλεία μετρήσεων

Για την πραγματοποίηση μετρήσεων, ώστε να γίνει δυνατή η αξιολόγηση του δικτύου, χρειάστηκαν ορισμένα προγράμματα τα οποία μπορούν να δώσουν την δυνατότητα να δημιουργηθεί κίνηση στο δίκτυο, να μετρηθεί και να αναλυθεί. Τα προγράμματα που χρησιμοποιήθηκαν θα περιγραφούν παρακάτω.

7.1.1 tcpdump

Το tcpdump είναι ένα πρόγραμμα γραμμής εντολών το οποίο επιτρέπει την ανίχνευση και καταγραφή πακέτων που εκπέμπονται ή λαμβάνονται σε μια κάρτα δικτύου. Είναι ιδιαίτερα χρήσιμο για τις μετρήσεις που οφείλουν να γίνουν, καθώς δίνει την δυνατότητα να συλληφθούν όλα τα πακέτα ή συγκεκριμένα πακέτα, βάσει περιορισμών, που κινούνται σε ένα δίκτυο και να αποθηκευτούν σε ένα αρχείο. Τα αρχεία αυτά, που δημιουργούνται για κάθε μέτρηση, χρησιμοποιούνται στην μετέπειτα ανάλυση.

7.1.2 mgen

Το mgen είναι ένα πρόγραμμα ανοικτού λογισμικού το οποίο παρέχει την δυνατότητα να πραγματοποιηθούν τεστ δικτυακής απόδοσης και μετρήσεις χρησιμοποιώντας UDP/IP κίνηση.

Αυτό το εργαλείο δημιουργεί μορφώματα κίνησης πραγματικού χρόνου, ούτως ώστε να μπορεί το δίκτυο να φορτωθεί με κίνηση με μία ποικιλία συνδυασμών. Χρησιμοποιεί script αρχεία ώστε να είναι δυνατή η χρονική κατανομή των μορφωμάτων κίνησης.

Αυτό το εργαλείο είναι σχεδιασμένο για συστήματα unix αλλά και windows συστήματα.

7.1.3 dvbnet

Το dvβνετ είναι ένα εργαλείο που περιέχεται στο υποσύστημα οδηγών για DVB συσκευές του πυρήνα του Linux. Αυτό το υποσύστημα οδηγών αναπτύσσεται και συντηρείται από το πρότζεκτ Linux TV.

Αυτό το εργαλείο είναι σε θέση να δημιουργήσει ένα εικονικό interface δικτύου το οποίο αντιστοιχίζεται σε ένα συγκεκριμένο PID μίας ροής δεδομένων. Αυτή η ροή υπάρχει σε ένα κανάλι ψηφιακής τηλεόρασης στο οποίο πρώτα πρέπει να έχει συντονιστεί χρησιμοποιώντας μια κάρτα λήψης σήματος ψηφιακής τηλεόρασης.

Αφού γίνει αυτό, αν εντοπιστεί ένα ενθυλακωμένο πακέτο στην ροή δεδομένων, το dvbnet αναλαμβάνει να το αποενθυλακώσει και να το προωθήσει στο εικονικό δικτυακό interface που έχει κατασκευάσει. Αυτή η διαδικασία είναι διάφανη στο λειτουργικό σύστημα, με αποτέλεσμα τα δεδομένα που δρομολογούνται μέσω των UHF να φτάνουν αμέσως στον τελικό χρήστη με την μορφή δικτυακής κίνησης.

Ένα μεγάλο ατού του dvbnet είναι το γεγονός ότι υποστηρίζει εδώ και καιρό τις δύο βασικές μορφές ενθυλάκωσης, την MPE και την ULE. Η χρήση του είναι απαραίτητη για την υλοποίηση του δικτύου καθώς περιθωριοποιεί το πρόβλημα της από-ενθυλάκωσης.

7.1.4 scripts

Για την ανάλυση των δεδομένων που παρήγαγαν οι μετρήσεις χρησιμοποιήθηκαν μερικά scripts σε γλώσσα προγραμματισμού Perl και Unix shell script. Αυτά ήταν σε θέση, δίνοντάς τους σαν είσοδο ένα αρχείο με τα πακέτα που στάλθηκαν

και ένα αρχείο με τα πακέτα που λήφθηκαν, να υπολογίσουν την καθυστέρηση, την μεταβολή της καθυστέρησης και τις απώλειες του δικτύου.

7.1.5 gnuplot

Το gnuplot είναι ένα λογισμικό ανοιχτού κώδικα κατάλληλο για την δημιουργία γραφημάτων από αρχεία δεδομένων όπως τα αρχεία που προέκυψαν από την ανάλυση των μετρήσεων.

7.2 Σενάρια μετρήσεων

Για να μετρηθεί η απόδοση του δικτύου στάλθηκαν ροές UDP δεδομένων από τον client στον ULE ενθυλακωτή. Το UDP πρωτόκολλο ήταν η προφανής επιλογή αφού το δίκτυο που στήθηκε για την αξιολόγηση του συστήματος είναι από την φύση του μονόδρομο. Οι ροές που στάλθηκαν με το mgen ήταν διάρκειας 30 δευτερολέπτων και εύρους 1500 Kbps. Το μέγεθος του πακέτου και η μέθοδος ενθυλάκωσης (packing ή padding) ήταν οι εξαρτημένες μεταβλητές σε κάθε μέτρηση.

7.3 Πραγματοποίηση μετρήσεων

7.3.1 Παραμετροποίηση ενθυλακωτή

Προκειμένου να αποδώσει τα βέλτιστα το πρόγραμμα που υλοποιήθηκε είναι απαραίτητη η παραμετροποίηση της DVB κάρτας. Αυτό, στην ουσία, σημαίνει ότι χρειάζεται να δοθούν οι κατάλληλες τιμές αριθμού και μεγέθους των buffers της κάρτας. Για να ανεβρεθούν οι πιο κατάλληλες τιμές, οι τιμές δηλαδή με τις οποίες το σύστημα συμπεριφέρεται με τον καλύτερο τρόπο, χρειάστηκε να πραγματοποιηθούν μετρήσεις, στις οποίες μεταβαλλόταν ο αριθμός και το μέγεθος των buffers.

Το δίκτυο που στήθηκε για την διεξαγωγή των πειραματικών μετρήσεων αυτών φαίνεται στην εικόνα 6.3. Είναι μια απλή αρχιτεκτονική καθώς αποτελείται από:

- έναν χρήστη ο οποίος στέλνει δεδομένα με την μορφή πακέτων ethernet στον ULE ενθυλακωτή
- τον ULE ενθυλακωτή, ο οποίος λαμβάνει τα δεδομένα, τα ενθυλακώνει σε έναν συρμό δεδομένων και τα προωθεί, μέσω της ASI κάρτας που διαθέτει, στην DVB-T αλυσίδα εκπομπής.
- Την DVB-T αλυσίδα εκπομπής, η οποία λαμβάνοντας τον συρμό δεδομένων αναλαμβάνει για την εκπομπή του.[1]
- Έναν δεύτερο χρήστη, ο οποίος διαθέτει μία κάρτα λήψης DVB-T και μέσω κατάλληλου λογισμικού τα μετατρέπει ξανά σε δικτυακή κίνηση υπό την μορφή IP πακέτων.

Για την μέτρηση χρησιμοποιήθηκε μία UDP ροή δεδομένων 1,5 Mbps η οποία δημιουργήθηκε με το προγράμματα mgen.

Για την εξαγωγή αποτελεσμάτων είναι απαραίτητο να ανακτήσουμε, με κάποιον τρόπο, όλα τα πακέτα που μεταφέρθηκαν στο δίκτυο κατά την διάρκεια της μέτρησης. Αυτό επιτυγχάνεται με την χρήση του εργαλείου tcpdump το οποίο χρησιμοποιείται για να πιάσει πακέτα τα οποία φτάνουν σε ένα δικτυακό interface. Χρησιμοποιήθηκε με τα κατάλληλα ορίσματα για να αποθηκευτούν τα πακέτα σε ένα αρχείο το οποίο μετά χρησιμοποιήθηκε για την ανάλυση. Δύο τέτοια αρχεία είναι απαραίτητα για την ανάλυση· ένα με πακέτα που εστάλησαν από τον χρήστη 1 μέσω του δικτυακού interface και ένα με τα πακέτα που λήφθηκαν από τον χρήστη 2 [2]. Έχοντας αυτά τα αρχεία και με τα κατάλληλα scripts εξήχθησαν τα αποτελέσματα [3].

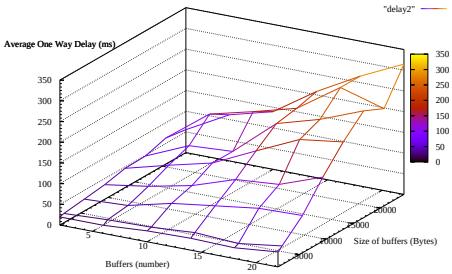
Για κάθε ζεύγος μεταβλητών παραγόταν 4 αποτελέσματα προς σύγκριση· η τιμή για την καθυστέρηση, για την μεταβολή της καθυστέρησης, την εξομαλυμένη μεταβολή της καθυστέρησης και τα πακέτα που χάθηκαν σε κάθε μέτρηση. Τα αποτελέσματα παρουσιάζονται στην εικόνα 7.1. Είναι φανερό ότι το σύστημα έχει καλύτερη συμπεριφορά, όσον αφορά τα χαμένα πακέτα και την

[1] λεπτομέρειες για την σύσταση της πλατφόρμας στο κεφάλαιο 6.2.1

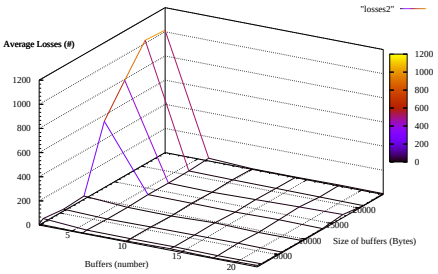
[2] Όπως εξηγείται στο κεφάλαιο 6.2, στην πραγματικότητα οι χρήστες 1 και 2, είναι το ίδιο μηχάνημα αλλά η εκπομπή και η λήψη γίνονται από διαφορετικά interface.

[3] τα scripts που χρησιμοποιήθηκαν υπάρχουν στο παράρτημα B.2 και η διαδικασία αναλύεται στο κεφάλαιο 7.3.2.

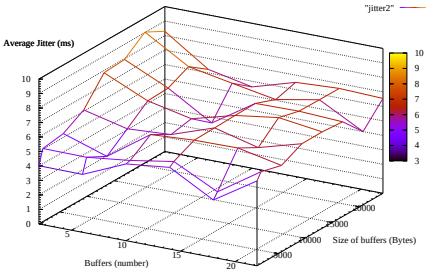
καθυστέρηση, όσο το μέγεθος και ο αριθμός των buffers μειώνονται. Επίσης φαίνεται ότι, για τις μετρήσεις που πραγματοποιήθηκαν, οι τιμές της μεταβολής της καθυστέρησης δεν επηρεάζονται σημαντικά.



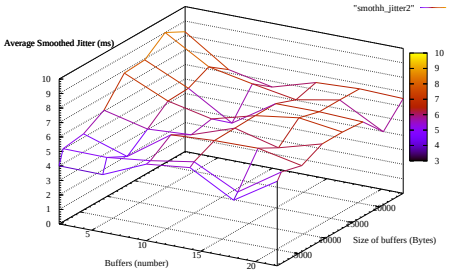
(α′) Η μεταβολή της καθυστέρησης.



(β′) Οι απώλειες.



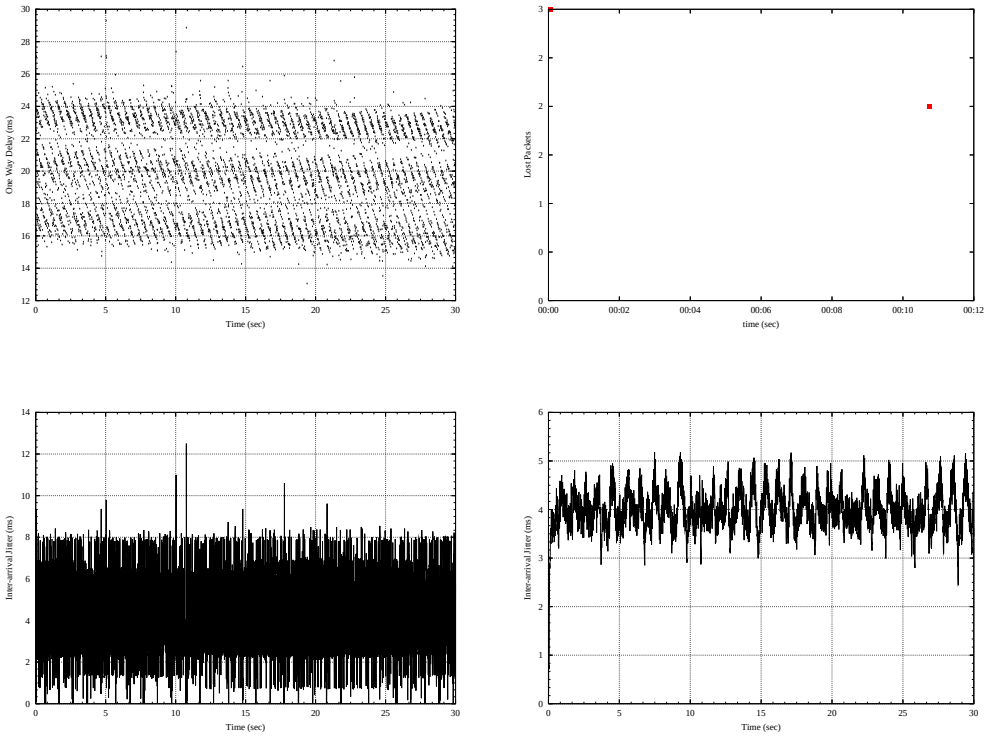
(γ′) Η μεταβολή της καθυστέρησης.



(δ′) Η εξομαλυμένη μεταβολή της καθυστέρησης.

Σχήμα 7.1: Τα αποτελέσματα των μετρήσεων για την εύρεση του βέλτιστου buffer.

Έχοντας αυτά κατά νου αποφασίστηκε ότι το καταλληλότερο ζεύγος τιμών για τους buffers της κάρτας είναι 2 buffers με μέγεθος 1880 Bytes. Γι′ αυτόν τον συνδυασμό τιμών η καθυστέρηση βρίσκεται στα 27,58 ms, τα χαμένα πακέτα περιορίζονται στο 0,16% και οι τιμές για τη μεταβολή της καθυστέρησης βρίσκονται κοντά στα 5 ms. Οι γραφικές γι' αυτές τις μεταβλητές παρουσιάζονται στο σχήμα 7.2.



Σχήμα 7.2: Οι γραφικές παραστάσεις της καθυστέρησης, των απωλειών, της μεταβολής της καθυστέρησης και της εξομαλυμένης μεταβολής της καθυστέρησης συναρτήσει του χρόνου.

7.3.2 Πραγματοποίηση μετρήσεων αξιολόγησης δικτύου

Από το eth1 interface του client στάλθηκε κίνηση UDP εύρους ζώνης 1.5 Mbps με χρήση του εργαλείου mgen. Έγιναν δύο σετ μετρήσεων· ένα χρησιμοποιώντας την μέθοδο ενθυλάκωσης padding και ένα χρησιμοποιώντας την μέθοδο packing. Για κάθε μία μέθοδο μεταβαλλόταν το μέγεθος πακέτου από 75 μέχρι 1425 bytes.

Η λήψη της κίνησης αυτής έγινε επίσης από τον client, αλλά σε άλλο interface, το dvb0_0. Προκειμένου να γίνει η ανάλυση και ο υπολογισμός καθυστέρησης, απωλειών κ.α. χρειάστηκε να πιαστούν όλα τα πακέτα που εκπέμ-

φθηκαν και λήφθηκαν. Αυτό έγινε με χρήση του εργαλείου tcpdump με τις εντολές που παρατίθενται παρακάτω.

```
% su -
# tcpdump -vv -w tx eth0 udp 7500 -n
# tcpdump -vv -w rx dvb0_0 udp 9000 -n
```

Έπειτα ενεργοποιήθηκε το mgen και με το τέλος της μέτρησης δημιουργήθηκαν τα αρχεία rx και tx που περιείχαν τα πακέτα που μεταδόθηκαν και λήφθηκαν. Αυτά τα δύο αρχεία δημιουργήθηκαν για κάθε μέτρηση και καλώντας για κάθε ζεύγος το bash script ipv4_udp, το οποίο παρατίθεται στο παράρτημα Β΄.2.1.1, εξήχθησαν τα αποτελέσματα της ανάλυσης.

Τα αποτέλεσμα των μετρήσεων παρουσιάζονται στο κεφάλαιο 9.1, και αναλυτικά τα διαγράμματα για κάθε μέτρηση υπάρχουν στο παράρτημα Α΄.

Κεφάλαιο 8

Λειτουργία

8.1 Εγκατάσταση

8.1.1 Εγκατάσταση οδηγών κάρτας

Η εγκατάσταση των οδηγών της κάρτας είναι μια απλή διαδικασία.

Στον κατάλογο που υπάρχουν οι πηγαίοι κώδικες των οδηγών πρέπει να εκτελεστούν οι ακόλουθες εντολές:

```
$ make
# make install
# ./mkdev.asi
```

Ο σκοπός αυτών των εντολών είναι η μεταγλώττιση του πηγαίου κώδικα σε γλώσσα μηχανής, η εγκατάσταση των οδηγών και τέλος η δημιουργία των device nodes[1] στα οποία είναι αντιστοιχισμένη κάθε κάρτα που υπάρχει στο σύστημα.

Αφού, τώρα, έχουν εγκατασταθεί οι οδηγοί της κάρτας, μπορούν να φορτωθούν στο σύστημα με την εντολή:

```
# modprobe dvbm
```

[1] Στα Unix συστήματα, κάθε συσκευή υλικού αντιστοιχίζεται σε ένα ειδικό αρχείο το οποίο βοηθάει στην διαφανή επικοινωνία των εφαρμογών επιπέδου χρήστη με το υλικό. Αυτού του είδους τα αρχεία αποκαλούνται device nodes.

Στα περισσότερα συστήματα οι οδηγοί θα φορτώνονται αυτόματα με το που ανιχνεύεται η ύπαρξη της κάρτας. Αν αυτό δεν γίνεται αυτόματα σε κάποιο σύστημα, είναι δυνατόν να αναγκαστεί το σύστημα να φορτώνει τους οδηγούς κατά την διάρκεια της εκκίνησης με την παρακάτω εντολή.

```
# echo modprobe dvbm > /etc/rc.d/rc.local
```

Μετά από αυτήν την διαδικασία η κάρτα πρέπει να είναι αντιστοιχισμένη με το device node /dev/asitx0. Για να τροποποιηθούν οι ρυθμίσεις της κάρτας με διαδραστικό τρόπο υπάρχει η εξής εντολή:

```
# txcfg /dev/asitx0
```

Τέλος για να διαπιστωθεί η σωστή λειτουργία της κάρτας μπορεί να εκτελεστεί η εντολή:

```
# mknull | txtest /dev/asitx0
```

Με αυτόν τον τρόπο αποστέλλεται ένας συρμός δεδομένων από κενά πακέτα στην κάρτα και έτσι είναι δυνατόν να διαπιστωθεί η λειτουργία της.

8.1.2 Συντονισμός

Με την εντολή scan εντοπίζονται όλα τα τηλεοπτικά κανάλια που εκπέμπονται από την πλατφόρμα DVB-T, όπως φαίνεται και στο παρακάτω screenshot . Παρατηρείται ότι ανιχνεύονται 5 τηλεοπτικά προγράμματα τα οποία αποθηκεύονται στη συνέχεια σε ένα αρχείο που το ονομάζουμε mychannels.

Τέλος, με την εντολή tzap, παρέχεται η δυνατότητα για συντονισμό σε κάποιο από τα ανιχνεύσιμα προγράμματα σε μια συγκεκριμένη συχνότητα, όπως παρατηρείται και από το screenshot .

8.1.3 dvbnet

Για να γίνεται αυτόματα η αποενθυλάκωση στην μεριά του χρήστη και η αναδρομολόγηση των IP πακέτων που ανακτήθηκαν, γίνεται χρήση του εργαλείου dvbnet. Αυτό δημιουργεί ένα εικονικό δικτυακό interface όπου και δρομολογεί τα IP πακέτα ώστε να επεξεργαστούν από το λειτουργικό.

Αυτό μπορεί να επιτευχθεί με την παρακάτω εντολή.

```
# dvbnet –U –p 336
```

8.2 Λειτουργία

Η εφαρμογή μπορεί να κληθεί από την γραμμή εντολών με την ακόλουθη εντολή:

```
# ./ipcapx /dev/asitx0 26 24 0 0 0 0
```

Το πρώτο όρισμα είναι το device node για την κάρτα που θα χρησιμοποιηθεί. Τα επόμενα τέσσερα ορίσματα είναι οι απαραίτητες τιμές που πρέπει να οριστούν, ώστε η κάρτα να στέλνει σε ένα συγκεκριμένο bit rate. Αυτές οι τιμές μπορούν να παραχθούν δίνοντας ως όρισμα το επιθυμητό bit rate στο πρόγραμμα calcstuff το οποίο παρέχεται μαζί με τους οδηγούς της κάρτας.

8.3 Παραμετροποίηση

Για να υπάρχει σταθερός ρυθμός δεδομένων, πράγμα που απαιτεί το πρωτόκολλο ASI, πρέπει να ρυθμιστεί η κάρτα ώστε να εισάγει αυτόματα null packets όταν τα διαθέσιμα δεδομένα δεν επαρκούν. Αυτό μπορεί να γίνει με τις ακόλουθες εντολές.

```
$ su –
# echo 1 > /sys/class/asi/asitx0/nullpackets
```

Ακόμα πριν σταλούν δεδομένα στην κάρτα, πρέπει να βρεθούν οι κατάλληλοι buffers. Η διαδικασία που ακολουθήθηκε στα πλαίσια αυτής της έρευνας περιγράφεται στο κεφάλαιο 7.3.1. Αν, για παράδειγμα, ο κατάλληλος αριθμός buffers είναι 6 και το μέγεθος των buffer είναι 1128, τότε οι buffer μπορούν να οριστούν με τις ακόλουθες εντολές.

```
$ su –
# echo 6 > /sys/class/asi/asitx0/buffers
# echo 1128 > /sys/class/asi/asitx0/bufsize
```

Κεφάλαιο 9

Παρουσίαση μετρήσεων

9.1 Παρουσίαση αποτελεσμάτων

Για την αξιολόγηση του δικτύου έγιναν οι μετρήσεις που αναφέρθηκαν. Από τα αποτελέσματα δημιουργήθηκαν τα διαγράμματα για την απόδοση του δικτύου, τα οποία φαίνονται παρακάτω.

9.1.1 Απόδοση

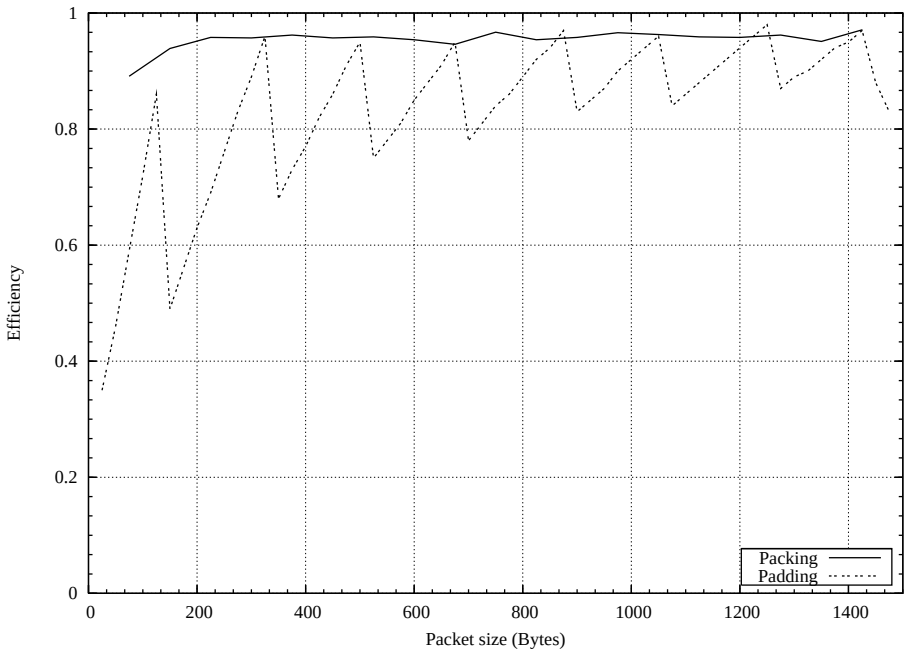
Από τα αποτελέσματα δημιουργήθηκαν τα διαγράμματα για την απόδοση του δικτύου, τα οποία φαίνονται παρακάτω.

Είναι προφανές ότι η απόδοση είναι καλύτερη όταν χρησιμοποιείται η μέθοδος του packing. Αυτό συμβαίνει επειδή η μέθοδος του padding όταν τελειώνει ένα ULE SNDU συμπληρώνει το υπόλοιπο του πακέτου με null bytes. Επειδή τα sndus μοιράζονται ανά 184 bytes για να ενθυλακωθούν στα TS πακέτα, αν το υπόλοιπο της διαίρεσης του μήκους του SNDU με το 184 είναι μικρό, τότε αυξάνεται το πλήθος των stuffing bytes με αποτέλεσμα να μειώνεται η απόδοση του δικτύου. Αντίθετα, αν το υπόλοιπο της διαίρεσης πλησιάζει το 184, τότε τα stuffing bytes είναι λιγότερα και η απόδοση του δικτύου μεγαλύτερη. Αυτό εξηγεί τις αυξομειώσεις στο διάγραμμα της απόδοσης όσο μεταβάλλεται το μέγεθος του IP πακέτου.

Το packing, αν υπάρχει η δυνατότητα, συνεχίζει να τοποθετεί τα δεδομένα του επόμενου SNDU. Με αυτόν τον τρόπο αξιοποιείται χώρος στα TS πακέτα

που αλλιώς θα πήγαινε χαμένος. Αυτό το γεγονός κάνει την μέθοδο packing να προσφέρει πολύ καλύτερη απόδοση σε σχέση με το padding.

Στην εικόνα 9.1 φαίνεται η απόδοση του δικτύου χρησιμοποιώντας την μέθοδο packing και την μέθοδο padding. Σαν απόδοση ορίζεται ο λόγος των εισερχόμενων bytes προς τα εξερχόμενα bytes. Αφού η διαφορά αυτών των δύο υπάρχει επειδή προστίθεται η πληροφορία από το overhead της ενθυλάκωσης, ο λόγος αυτός είναι λογικό να μας δίνει την εικόνα της απόδοσης της ULE ενθυλάκωσης.



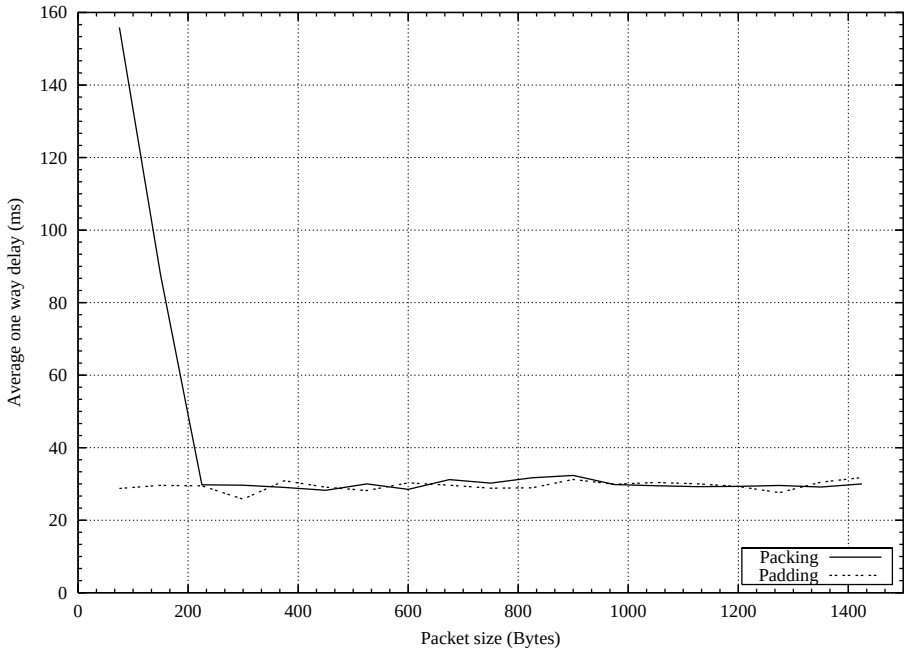
Σχήμα 9.1: Απόδοση του δικτύου.

Η απόδοση αυτή είναι σύμφωνη με άλλες πρόσφατες έρευνες για την απόδοση της ULE ενθυλάκωσης [5].

9.1.2 Καθυστέρηση

Η καθυστέρηση είναι ο χρόνος που απαιτείται για το κάθε πακέτο για να φτάσει στον προορισμό του από την ώρα που ξεκίνησε από τον αποστολέα.

Οι μέσοι όροι της καθυστέρησης για κάθε σετ μετρήσεων φαίνεται στο διάγραμμα 9.2.



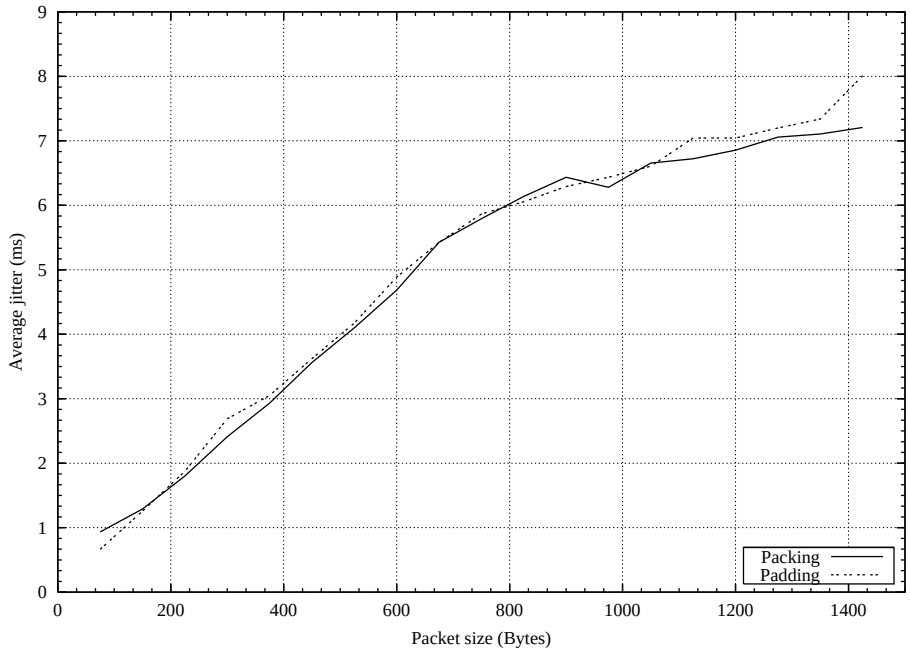
Σχήμα 9.2: Το διάγραμμα καθυστέρησης.

Όπως φαίνεται, με εξαίρεση την περίπτωση μικρού μεγέθους πακέτων για την μέθοδο του packing, ο μέσος όρος της καθυστέρησης του δικτύου διατηρείται σχεδόν σταθερή στα 29 ms ανεξάρτητα από το μέγεθος πακέτου. Άρα σε σχέση με την καθυστέρηση δεν φαίνεται κάποια μέθοδος να υπερτερεί σε σχέση με μια άλλη.

9.1.3 Μεταβολή της καθυστέρησης

Εδώ παρουσιάζονται τα αποτελέσματα της ανάλυσης για την μεταβολή της καθυστέρησης. Αυτή η τιμή απεικονίζει την μεταβολή της καθυστέρησης ανάμεσα σε δύο πακέτα. Για πολλές δικτυακές υπηρεσίες είναι σημαντικό η τιμή αυτή να παραμένει σε χαμηλά επίπεδα.

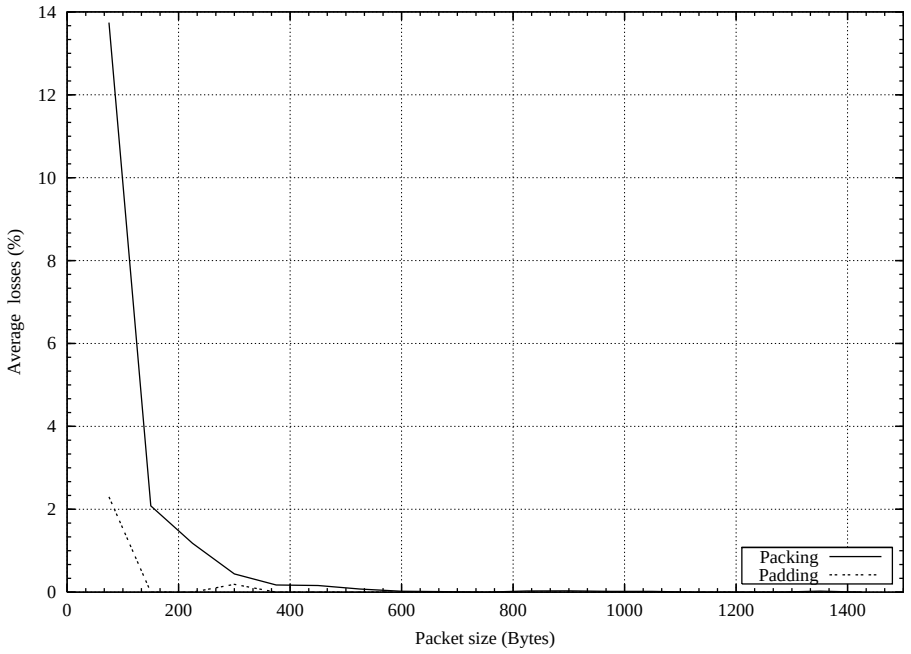
Στην εικόνα 9.3 παρουσιάζονται οι μέσες τιμές της μεταβολής της καθυστέρησης σε σχέση με το μέγεθος πακέτου για τις δύο μεθόδους ενθυλάκωσης. Απ' ότι φαίνεται από το διάγραμμα η μεταβολή της καθυστέρησης αυξάνεται αναλογικά και σχεδόν γραμμικά σε σχέση με το μέγεθος του πακέτου.



Σχήμα 9.3: Η μεταβολή της καθυστέρησης.

9.1.4 Απώλειες

Οι απώλειες του δικτύου παρουσιάζονται στο διάγραμμα 9.4. Σαν απώλεια υπολογίζεται ένα πακέτο το οποίο δεν έφτασε στον προορισμό του. Τα αποτελέσματα εμφανίζουν το ποσοστό των πακέτων που χάθηκαν σε σχέση με με τα πακέτα που εστάλησαν.



Σχήμα 9.4: Οι απώλειες του συστήματος.

Είναι φανερό ότι το ποσοστό των χαμένων πακέτων αυξάνεται εκθετικά όσο μειώνεται το μέγεθος του πακέτου. Αυτό, όπως, αναφέρθηκε και πιο πάνω, οφείλεται στο ότι οι buffer είναι ρυθμισμένοι για ένα συγκεκριμένο ρυθμό μετάδοσης. Αν εξαιρεθεί η περίπτωση των πολύ μικρών μεγεθών πακέτων, παρατηρείται ότι το ποσοστό των απωλειών πλησιάζει το 0 και για τους δύο τύπους ενθυλάκωσης.

9.2 Σχολιασμός αποτελεσμάτων

Η απόδοση του δικτύου είναι πολύ καλύτερη με την μέθοδο packing.

Η καθυστέρηση διατηρείται σε πολύ καλά επίπεδα σε όλα τα μεγέθη πακέτων και με τις δύο μεθόδους. Η μόνη εξαίρεση είναι όταν υπάρχουν ροές με πολύ μικρό μέγεθος πακέτων όπου η καθυστέρηση αυξάνεται δραματικά, αλλά παραμένει σε ανεκτά επίπεδα.

Η μεταβολή της καθυστέρησης διατηρήθηκε σε χαμηλά επίπεδα και ήταν ανάλογη με το μέγεθος του πακέτου. Οι τιμές της κυμάνθηκαν από 0.93 ms ως 7,20 ms. Δεδομένου ότι για τις περισσότερες υπηρεσίες ανεκτές τιμές για την μεταβολή της καθυστέρησης είναι από 100 ms και κάτω, συνεπάγεται ότι το δίκτυο που αναπτύχθηκε δεν είναι ευπαθές στις μεταβολές της καθυστέρησης.

Παρατηρούνται μεγάλες απώλειες για μικρό μέγεθος πακέτου. Αυτό οφείλεται στο ότι οι buffers της κάρτας έχουν ρυθμιστεί για βέλτιστη λειτουργία σε ένα συγκεκριμένο ρυθμό μετάδοσης δεδομένων. Όσο μειώνεται το μέγεθος του πακέτου, όμως, τόσο αυξάνεται ο ρυθμός μετάδοσης, καθώς αυξάνεται το overhead ανά χρόνο ανά πακέτο.

Κεφάλαιο 10

Συμπεράσματα

Σκοπός αυτής της πτυχιακής εργασίας ήταν η μεταφορά δικτυακών πακέτων πάνω από ένα κανάλι UHF με την τεχνολογία της ψηφιακής τηλεόρασης. Για τον σκοπό αυτό κατασκευάστηκε ένας ULE ενθυλακωτής σύμφωνα με το RFC 4326 και υλοποιήθηκε ένα πρότυπο μονόδρομο δίκτυο στο οποίο μεταδιδόταν τα δεδομένα σε μία UHF συχνότητα και η λήψη τους γινόταν από έναν υπολογιστή με τον κατάλληλο εξοπλισμό. Η αξιολόγηση της συμπεριφοράς του δικτύου στην μετάδοση από σημείο σε σημείο έγινε κάνοντας χρήση των πρωτοκόλλων UDP/IP. Τα αποτελέσματα των μετρήσεων κρίνονται αναμενόμενα και είναι συναφή με προηγούμενες έρευνες.

Η χρηστικότητα του δικτύου που δημιουργήθηκε μπορεί αυξηθεί δραματικά με την χρήση ενός καναλιού επιστροφής. Τέτοιες αρχιτεκτονικές έχουν ήδη προταθεί σε πρόσφατες έρευνες.

Η τεχνολογία η οποία μελετήθηκε και αξιολογήθηκε στην παρούσα πτυχιακή εργασία μπορεί να επεκταθεί και σε συγγενικά πρωτόκολλα της επίγειας ψηφιακής τηλεόρασης, όπως το DVB-H, ώστε να καλύψει και κινητούς χρήστες που διαθέτουν φορητές συσκευές και επιθυμούν να έχουν δικτυακές υπηρεσίες. Δεδομένης της ευρείας περιοχής κάλυψης της ψηφιακής τηλεόρασης, είναι εύλογο και αναμενόμενο τέτοιου είδους υπηρεσίες να γίνουν αντικείμενο εντατικότερης έρευνας.

Βιβλιογραφία

[1] EN 301 192 "Digital Video Broadcasting (DVB); Specification for Data Broadcasting"

[2] G. Fairhurst, B. Collini-Nocker, H.D. Clausen, H. Linder, "A Framework for transmission of IP datagrams over MPEG-2 networks", IETF Work in Progress, http://www.ietf.org/internet-drafts/draft-fair-ipdvb-req-05.txt

[3] G. Fairhurst , B. Collini-Nocker. "Ultra Lightweight Encapsulation (ULE) for transmission of IP datagrams over MPEG-2/DVB networks", IETF Work in Progress, http://www.ietf.org/internet-drafts/draft-ietf-ipdvb-ule-01.txt

[4] TR 101 211 "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)"

[5] G. Xilouris, G. Gardikis, H. Koumaras and A. Kourtis, "Unidirectional Lightweight Encapsulation: Performance Evaluation and Application Perspectives", IEEE Transaction on Broadcasting, vol. 52, Issue 3, Sept 2006, pp. 374-380.

[6] TR 101 162 "Digital Video Broadcasting (DVB); Allocation of Service Information (SI) Codes for Digital Video Broadcasting (DVB) Systems"

[7] B. Collini-Nocker, H. Linder, G. Fairhurst, "Ultra Lightweight Encapsulation (ULE) Extension Header" IETF Work in Progress, http://www.ietf.org/internet-drafts/draft- collini-ipdvb-xule-00.txt

[8] http://www.ist-athena.org/Deliverables/ATHENA%20D3.1%20Final.pdf Ημερομηνία τελευταίας επίσκεψης: 06/09/2006

[9] E Pallis, C. Mantakas, G. Mastorakis, V. Zacharopoulos, "Digital Switchover in UHF: the ATHENA concept for Broadband access", 14th IST Mobile & Wireless Communications Summit 2005, Dresden, 19-23 June, 2005

[10] http://dast.nlanr.net/Projects/Iperf/ Ημερομηνία τελευταίας επίσκεψης: 08/10/2007

[11] http://pf.itd.nrl.navy.mil/mgen/ Ημερομηνία τελευταίας επίσκεψης: 08/10/2007

[12] http://jarok.cs.ohiou.edu/software/tcptrace/ Ημερομηνία τελευταίας επίσκεψης: 08/10/2007

[13] ISO/IEC 13818-2, "Generic Coding of Moving Pictures and Associated Audio Information (MPEG-2) Part 2: Video", ISO, 1996.

[14] ISO/IEC 13818-1, "Generic Coding of Moving Pictures and Associated Audio Information (MPEG-2) Part 1: Systems", ISO, 1996.

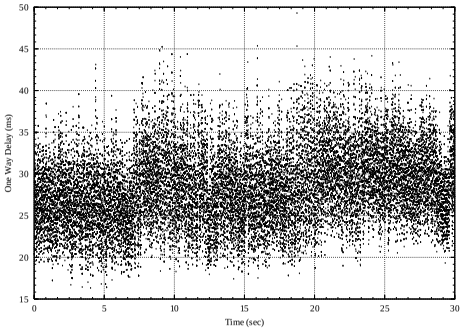
[15] ETSI EN 300 744, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television", v1.5.1, ETSI, 2004.

[16] ETS 300 744, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital Terrestrial television (DVB-T)", ETSI, 1997.

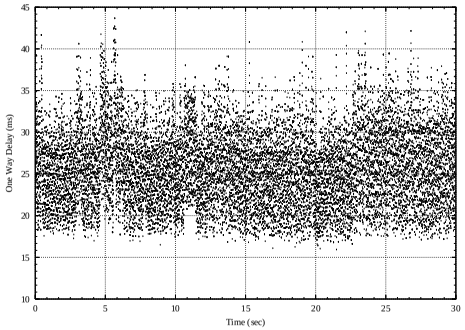
Index

Παράρτημα Α΄

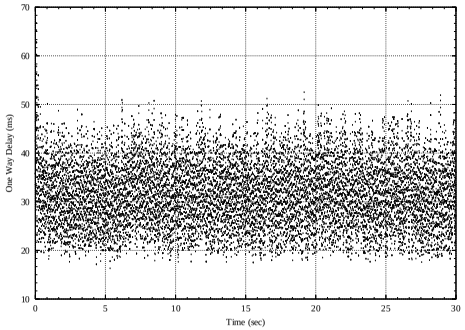
Διαγράμματα



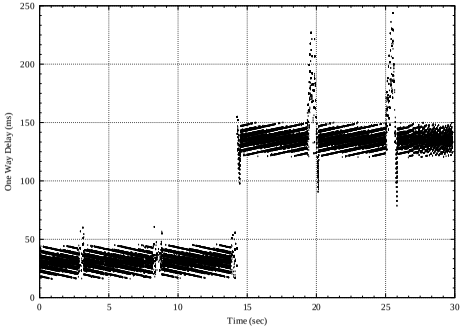
(a) padding 150



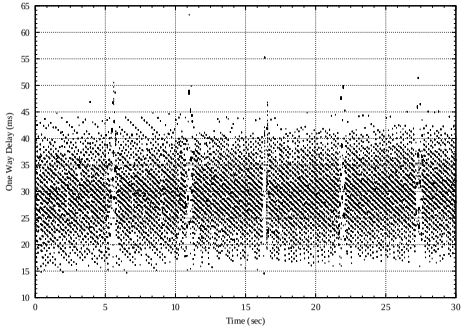
(b) padding 225



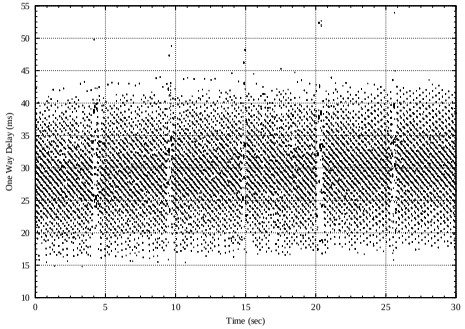
(c) paddind 300



(d) packing 150

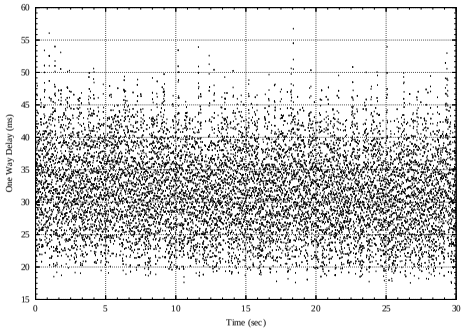


(e) packing 225

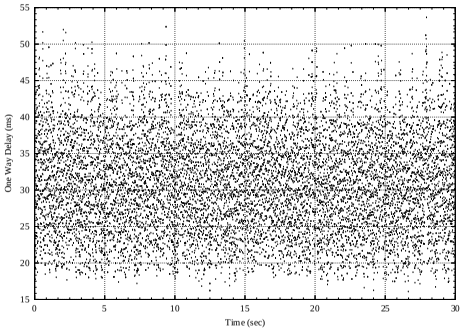


(f) packing 300

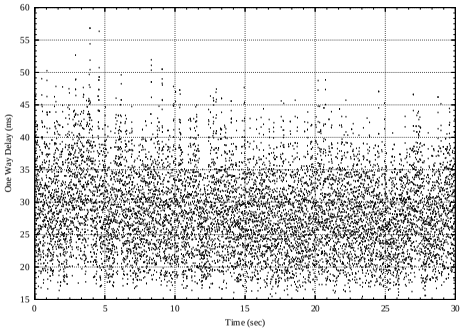
Σχήμα Α΄.1: Καθυστέρηση



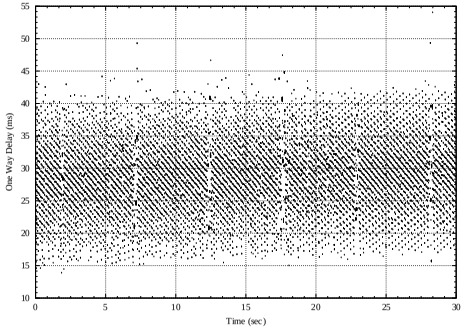
(a) padding 375



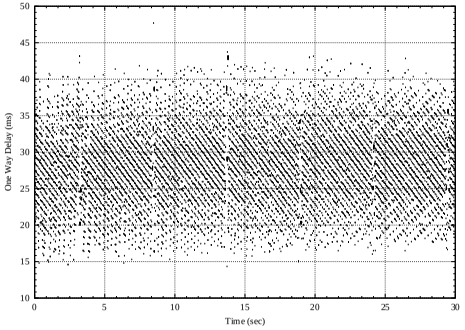
(b) padding 450



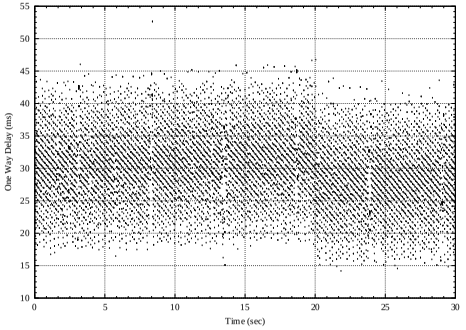
(c) paddind 525



(d) packing 375

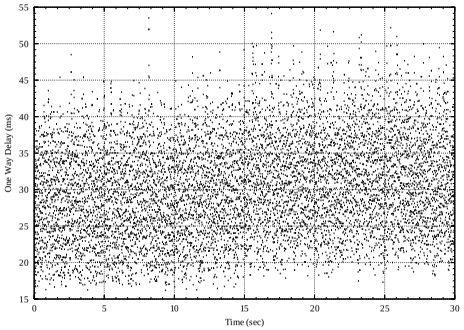


(e) packing 450

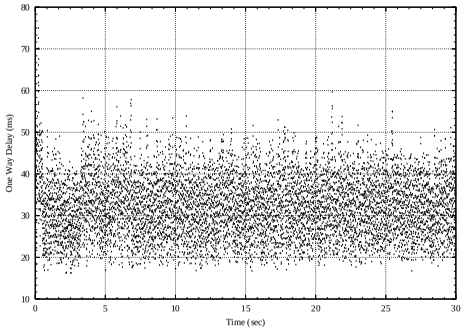


(f) packing 525

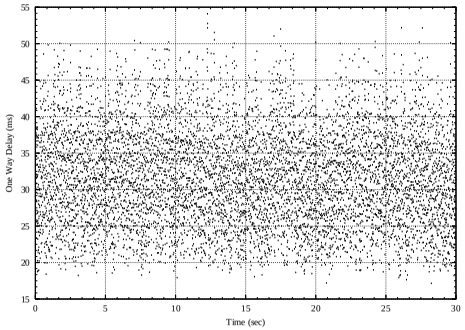
Σχήμα Α΄.2: Καθυστέρηση



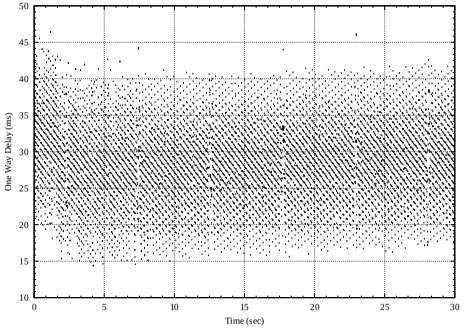
(a) padding 600



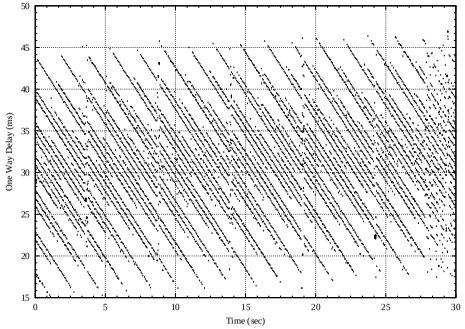
(b) padding 675



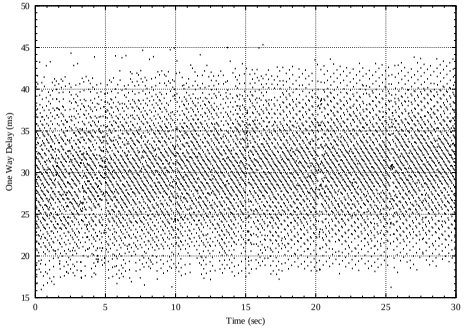
(c) paddind 750



(d) packing 600

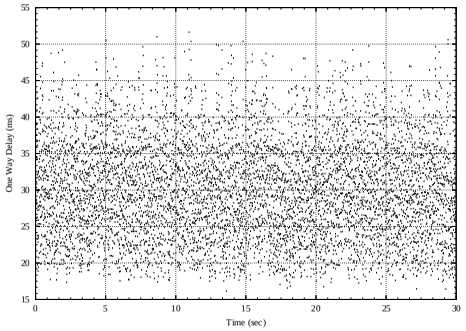


(e) packing 675

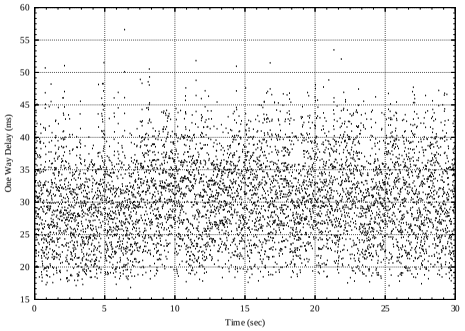


(f) packing 750

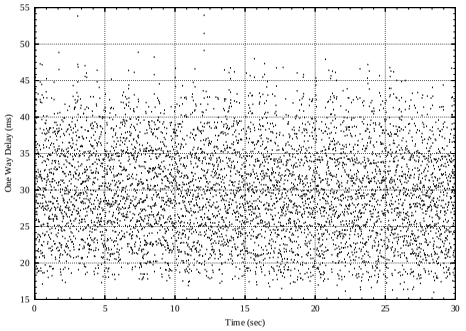
Σχήμα Α΄.3: Καθυστέρηση



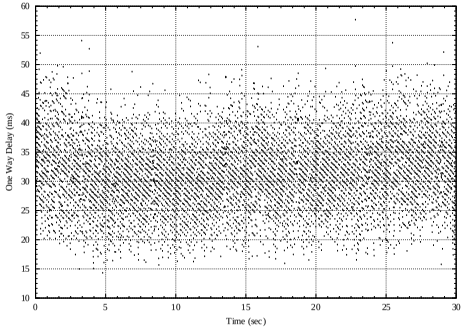
(a) padding 825



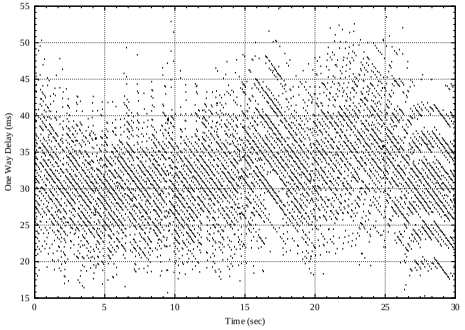
(b) padding 900



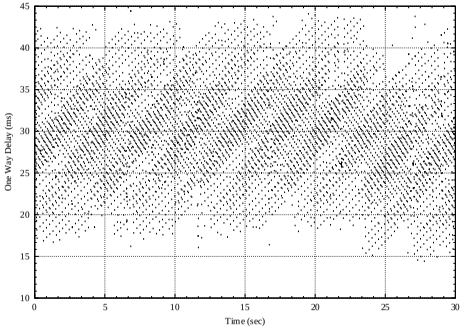
(c) paddind 975



(d) packing 825

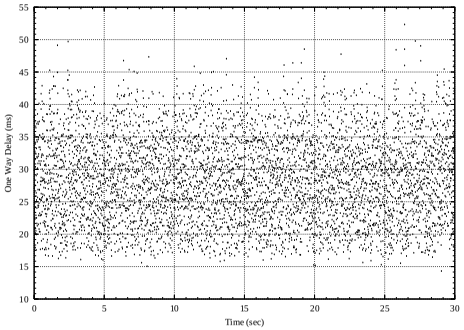


(e) packing 900

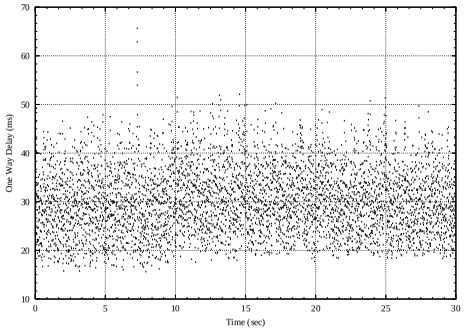


(f) packing 975

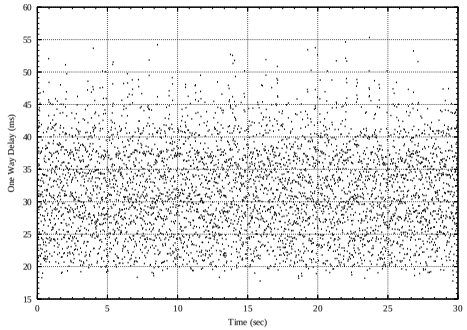
Σχήμα Α΄.4: Καθυστέρηση



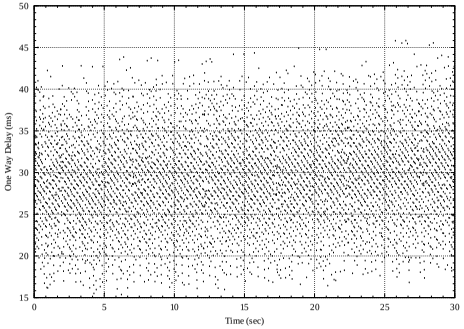
(a) padding 1050



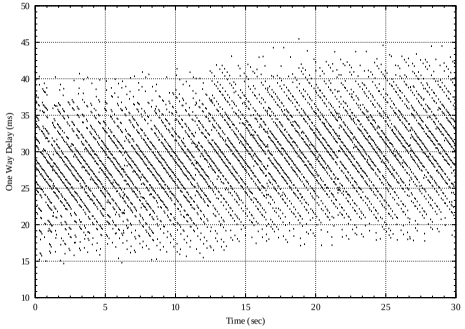
(b) padding 1125



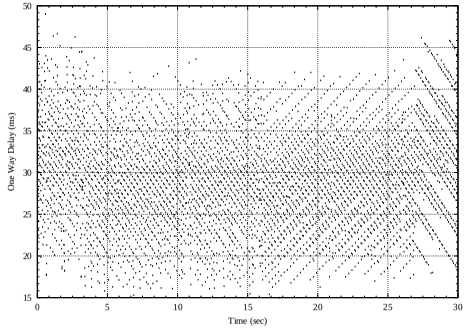
(c) paddind 1200



(d) packing 1050

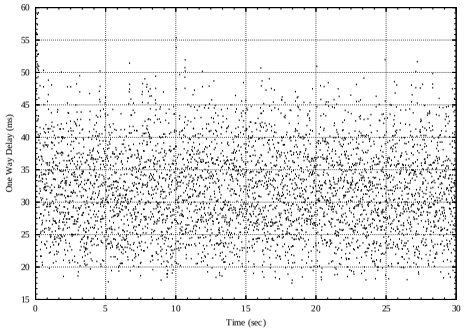


(e) packing 1125

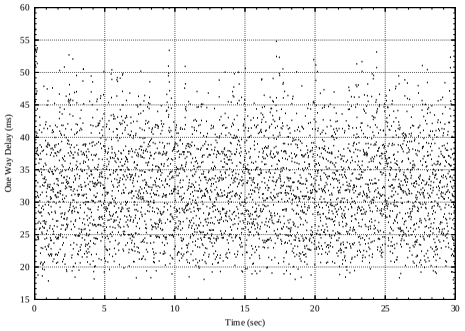


(f) packing 1200

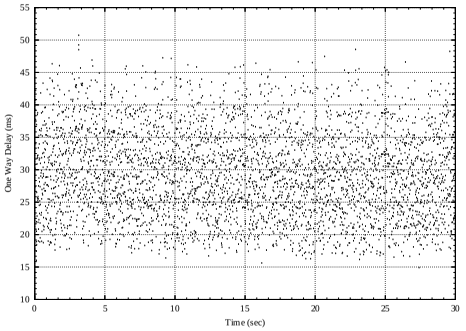
Σχήμα Α΄.5: Καθυστέρηση



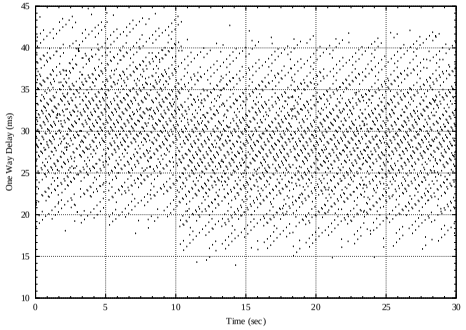
(a) padding 1275



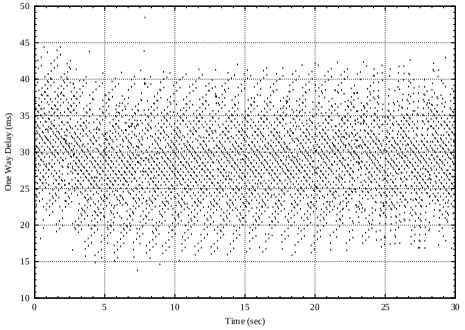
(b) padding 1350



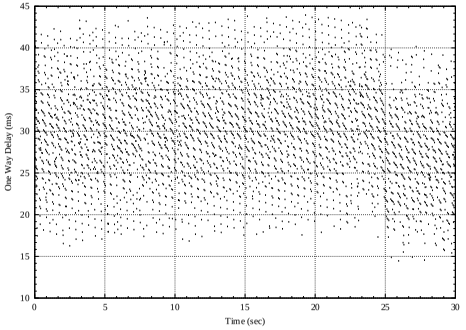
(c) paddind 1425



(d) packing 1275

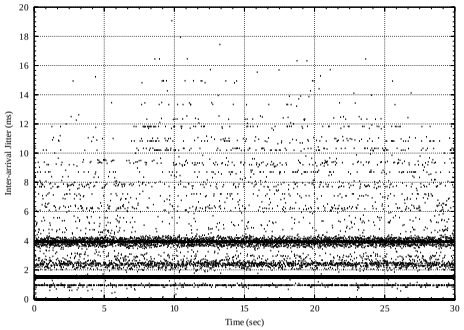


(e) packing 1350

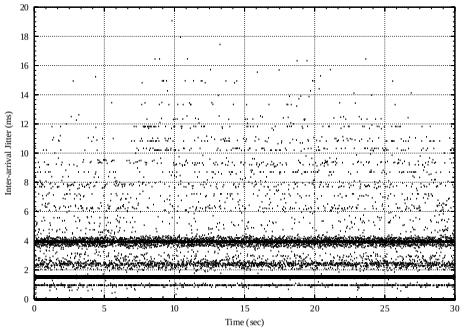


(f) packing 1425

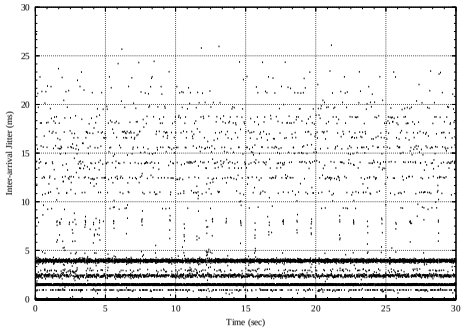
Σχήμα Α΄.6: Καθυστέρηση



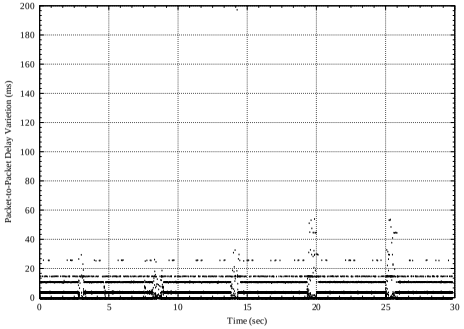
(a) padding 150



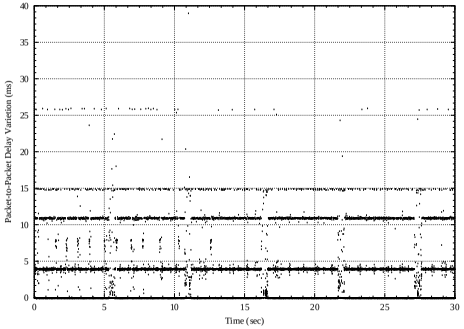
(b) padding 225



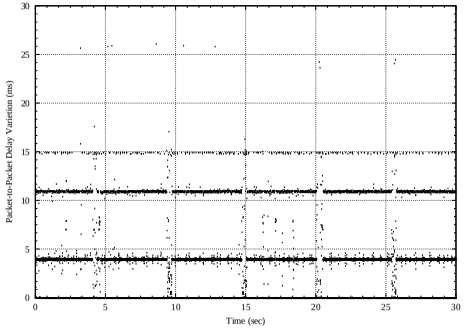
(c) paddind 300



(d) packing 150

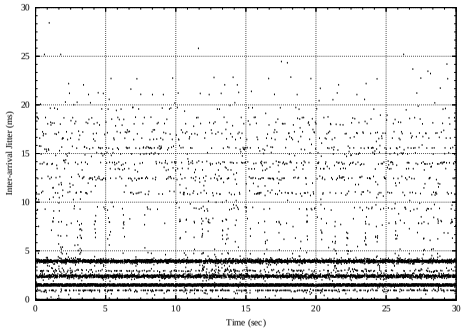


(e) packing 225

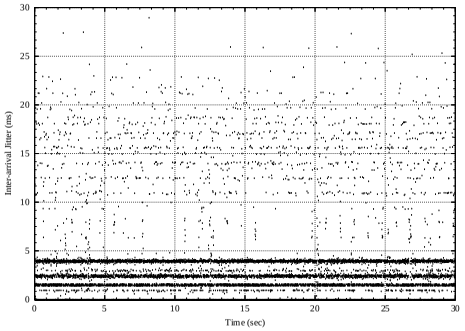


(f) packing 300

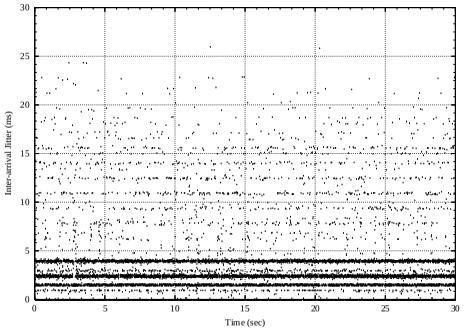
Σχήμα Α΄.7: Μεταβολή καθυστέρησης



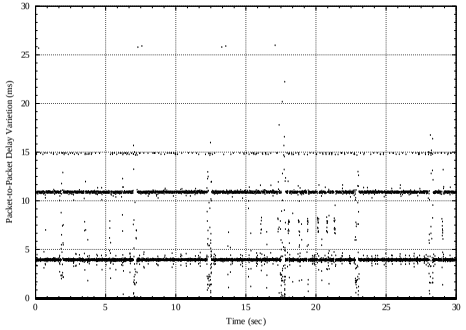
(a) padding 375



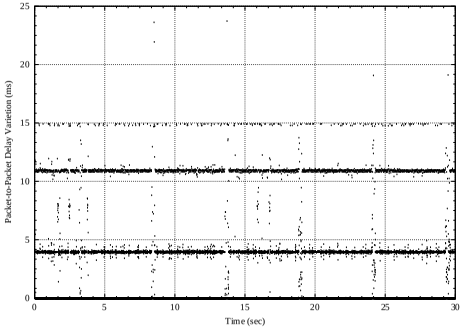
(b) padding 450



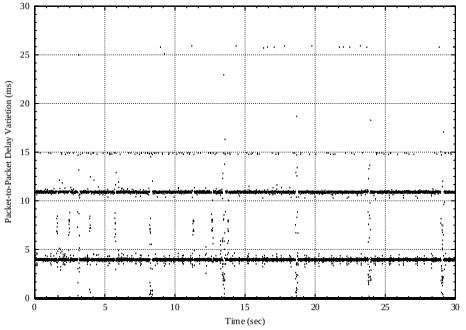
(c) paddind 525



(d) packing 375



(e) packing 450



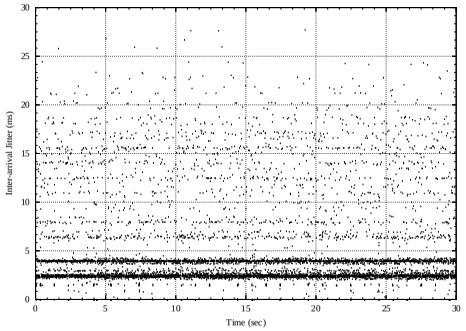
(f) packing 525

Σχήμα Α′.8: Μεταβολή καθυστέρησης

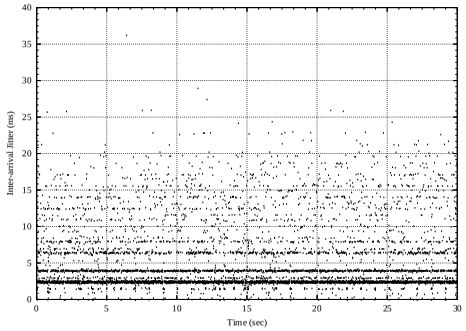
(a) padding 600 (b) padding 675 (c) padding 750

(d) packing 600 (e) packing 675 (f) packing 750

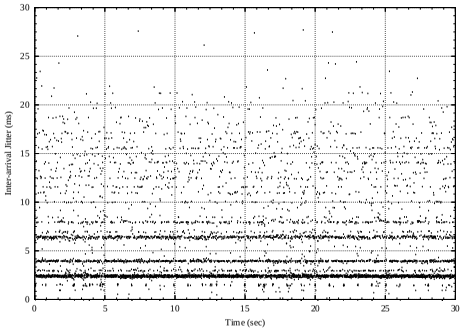
Σχήμα Α΄.9: Μεταβολή καθυστερήσεων



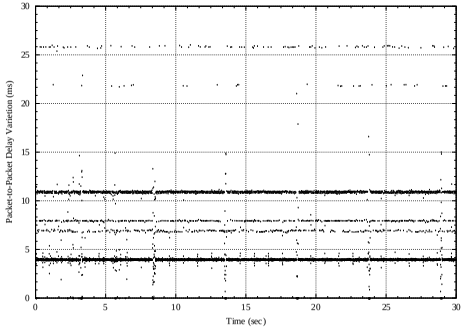
(a) padding 825



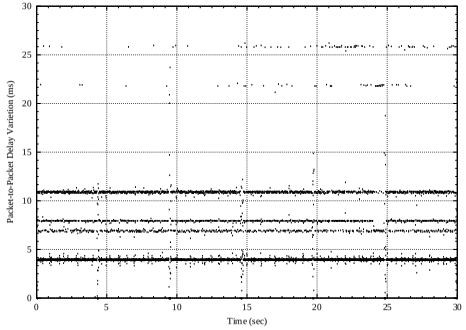
(b) padding 900



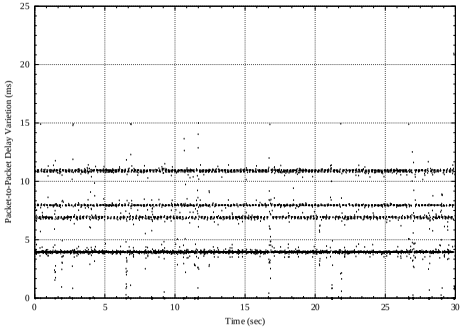
(c) paddind 975



(d) packing 825

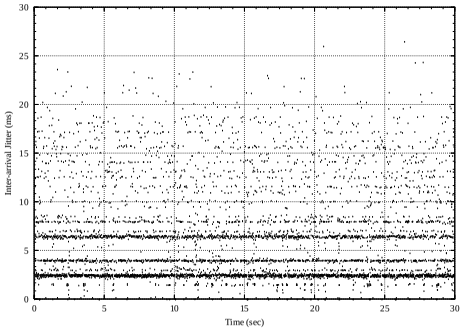


(e) packing 900

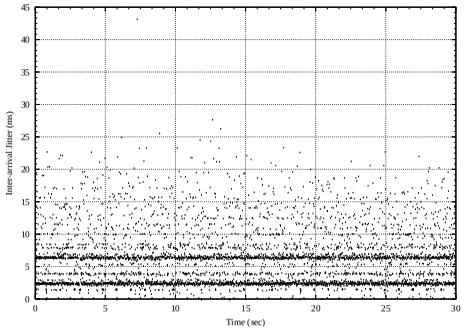


(f) packing 975

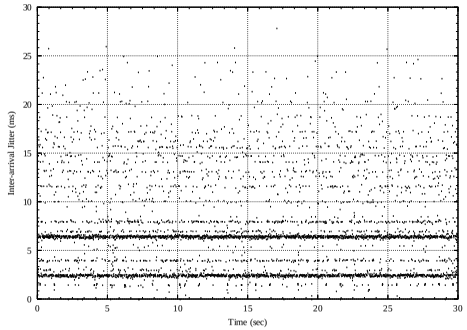
Σχήμα Α΄.10: Μεταβολή καθυστέρησης



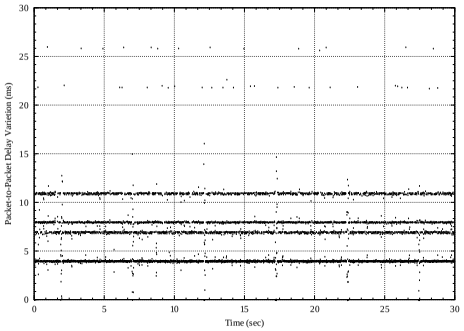
(a) padding 1050



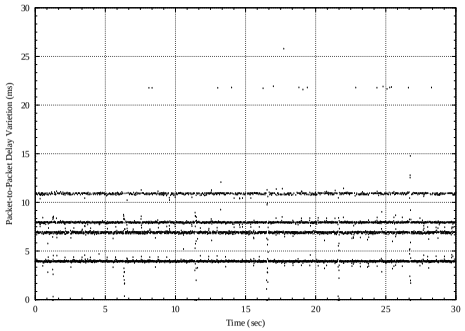
(b) padding 1125



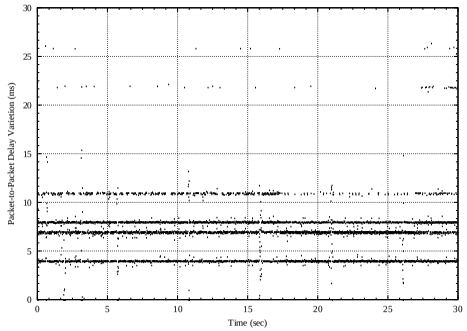
(c) paddind 1200



(d) packing 1050

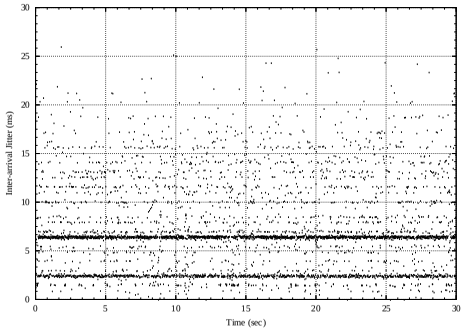


(e) packing 1125

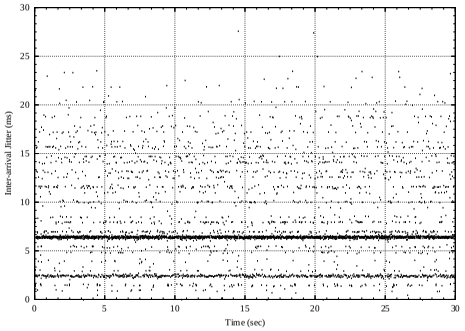


(f) packing 1200

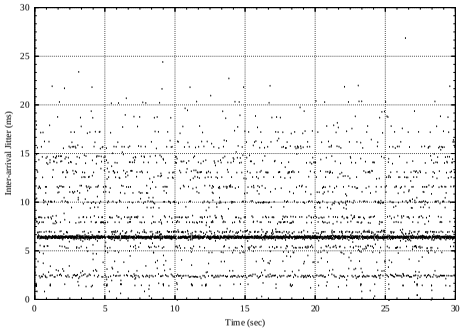
Σχήμα Α΄.11: Μεταβολή καθυστέρησης



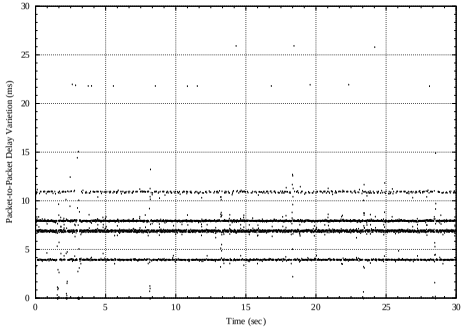
(a) padding 1275



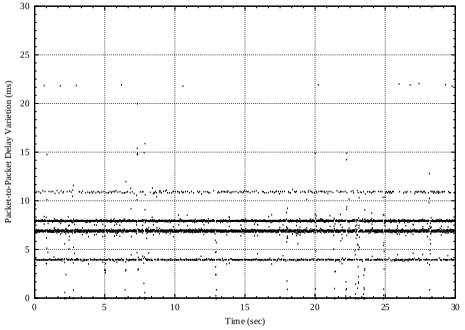
(b) padding 1350



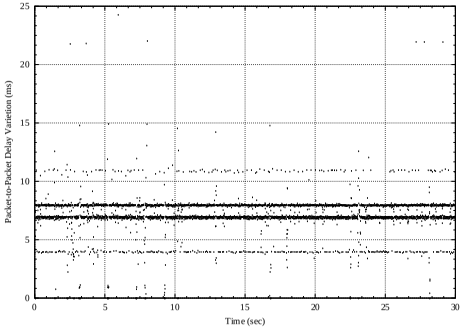
(c) paddind 1425



(d) packing 1275

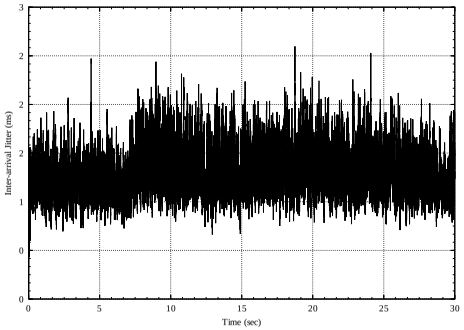


(e) packing 1350

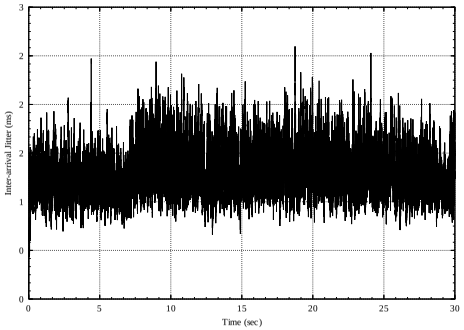


(f) packing 1425

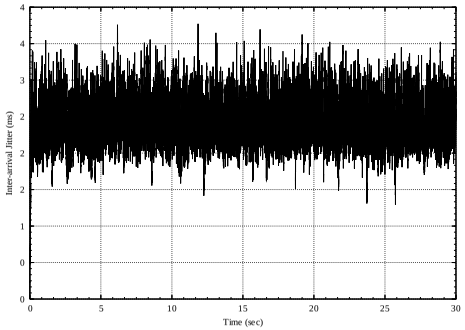
Σχήμα Α΄.12: Μεταβολή καθυστέρησης



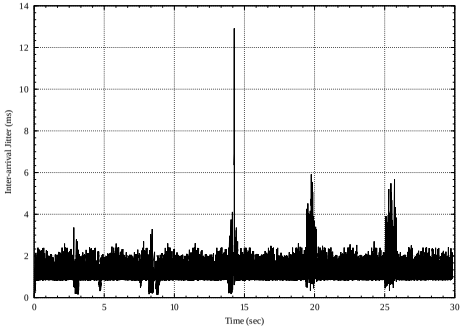
(a) padding 150



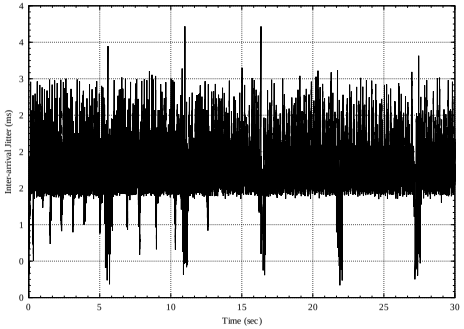
(b) padding 225



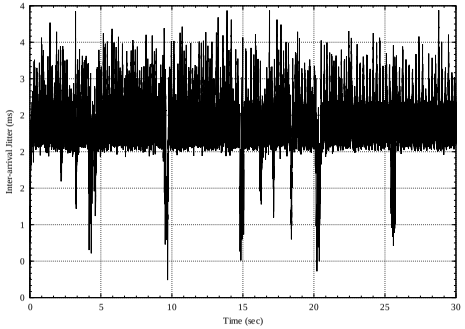
(c) paddind 300



(d) packing 150

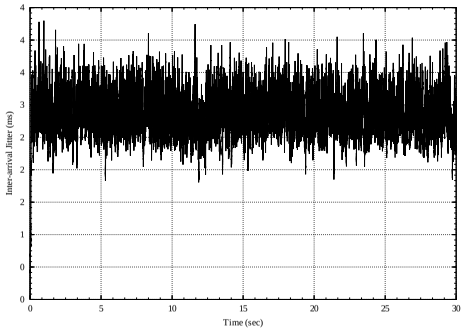


(e) packing 225

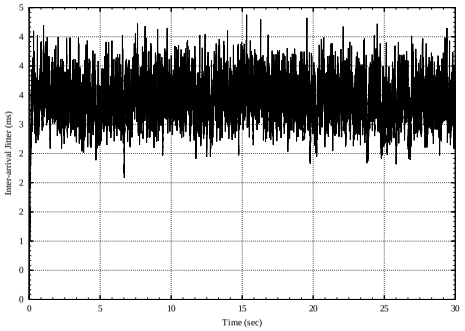


(f) packing 300

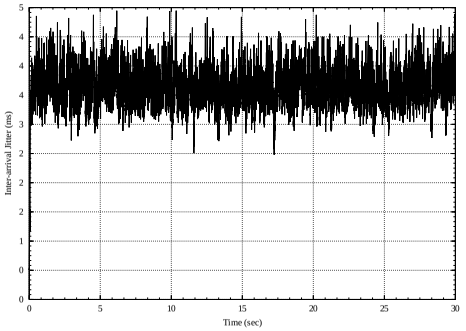
Σχήμα Α΄.13: Εξομαλυμένη μεταβολή της καθυστέρησης



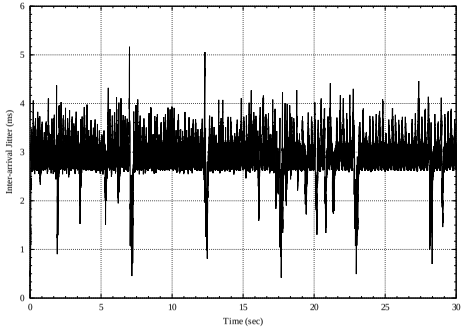
(a) padding 375



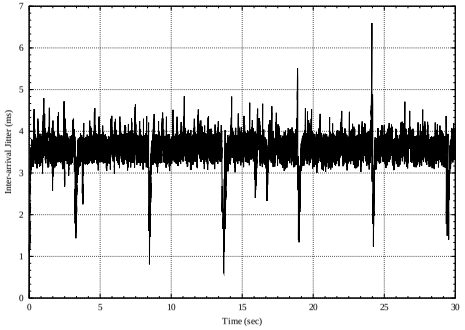
(b) padding 450



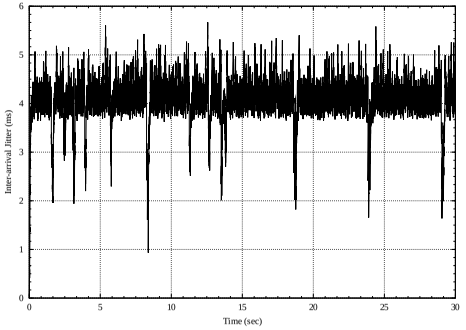
(c) paddind 525



(d) packing 375

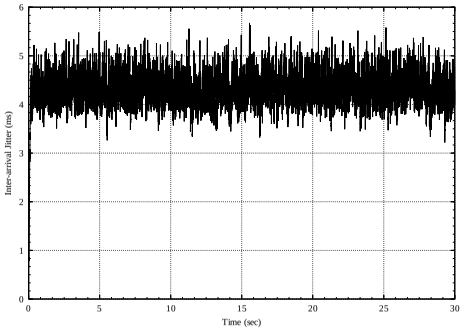


(e) packing 450

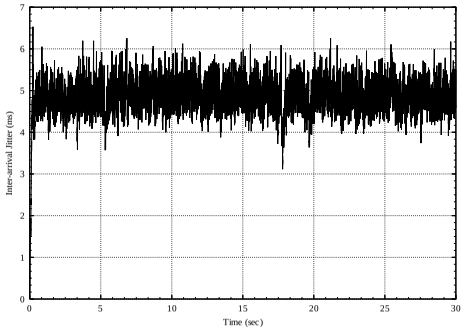


(f) packing 525

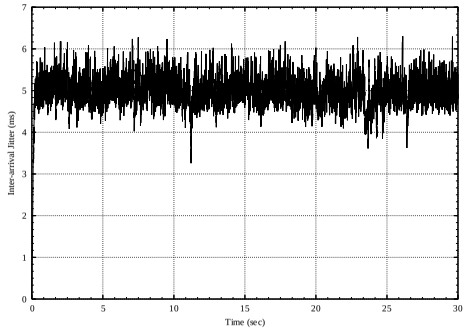
Σχήμα Α΄.14: Εξομαλυμένη μεταβολή της καθυστέρησης



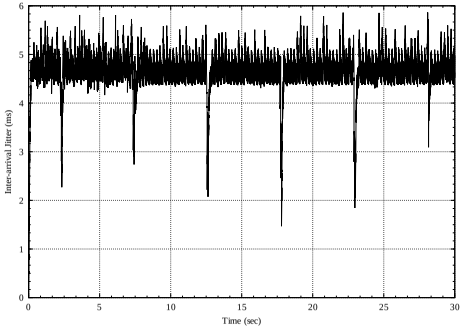
(a) padding 600



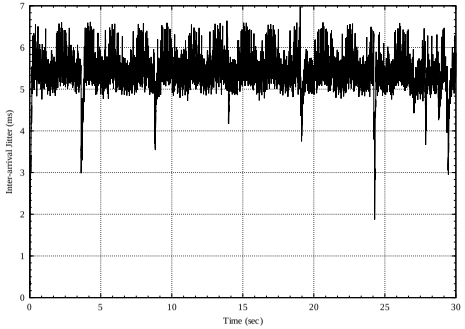
(b) padding 675



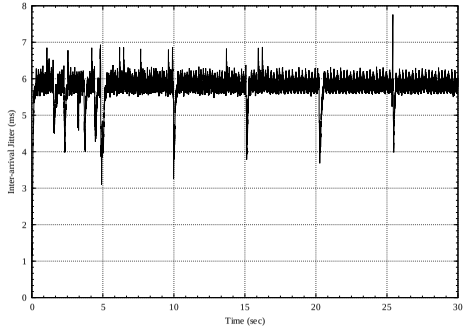
(c) paddind 750



(d) packing 600

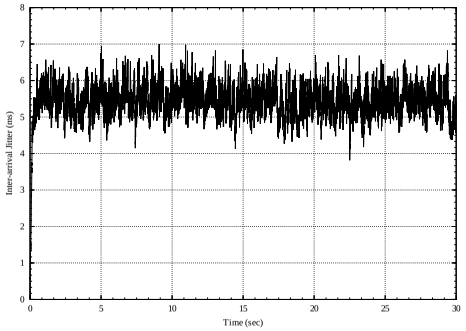


(e) packing 675

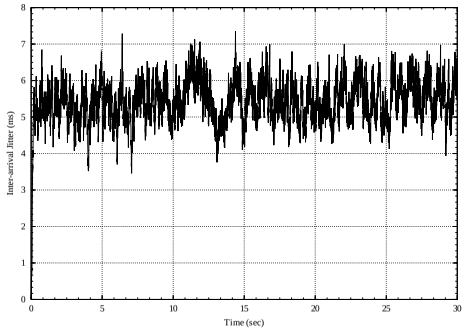


(f) packing 750

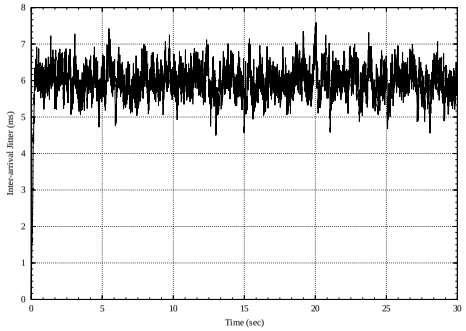
Σχήμα Α΄.15: Εξομαλυμένη μεταβολή της καθυστέρησης



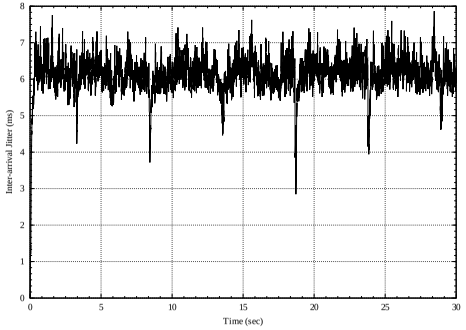
(a) padding 825



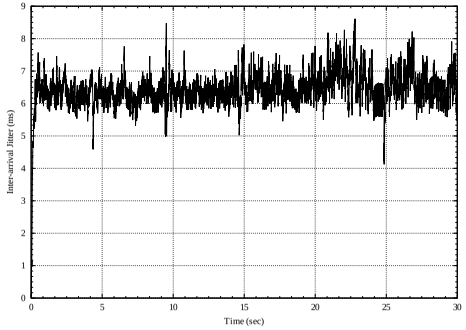
(b) padding 900



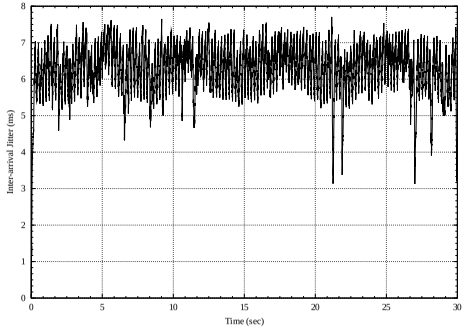
(c) paddind 975



(d) packing 825

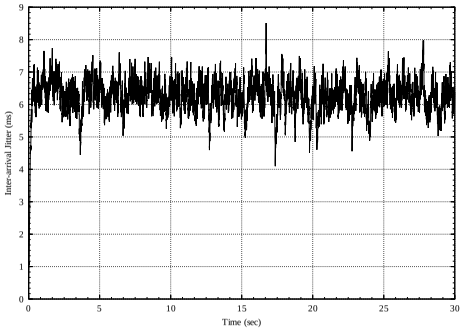


(e) packing 900

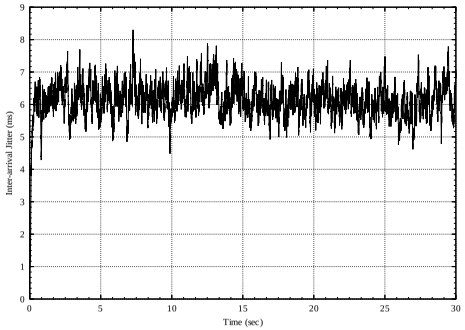


(f) packing 975

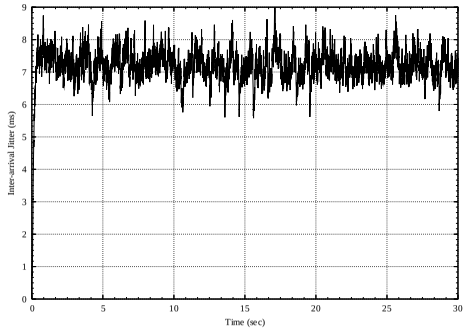
Σχήμα Α΄.16: Εξομαλυμένη μεταβολή της καθυστέρησης



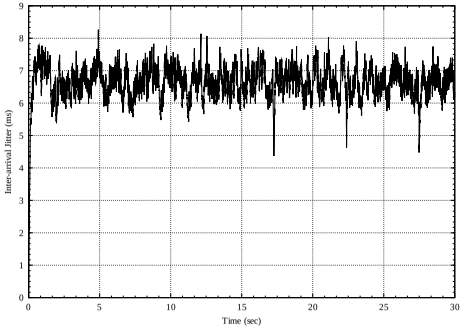
(a) padding 1050



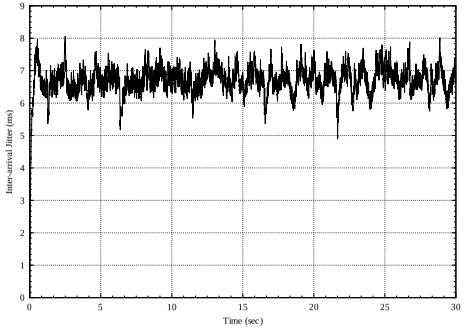
(b) padding 1125



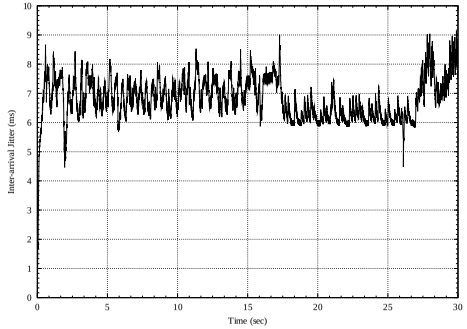
(c) paddind 1200



(d) packing 1050

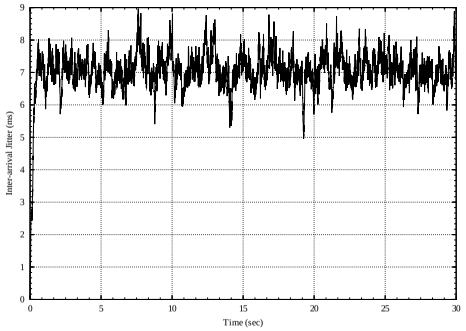


(e) packing 1125

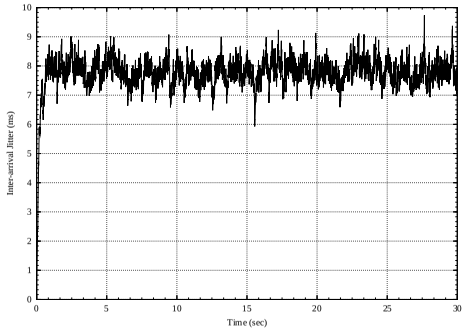


(f) packing 1200

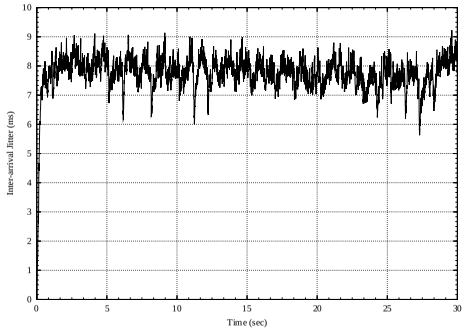
Σχήμα Α΄.17: Εξομαλυμένη μεταβολή της καθυστέρησης



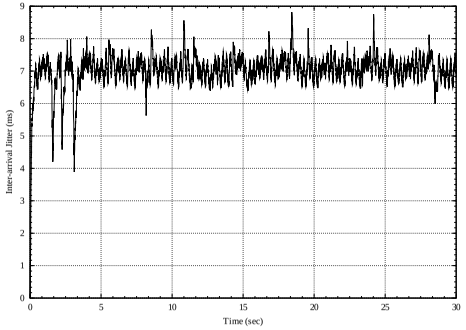
(a) padding 1275



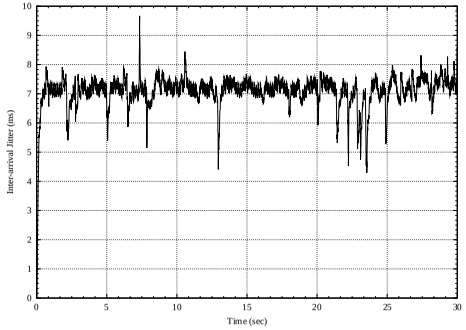
(b) padding 1350



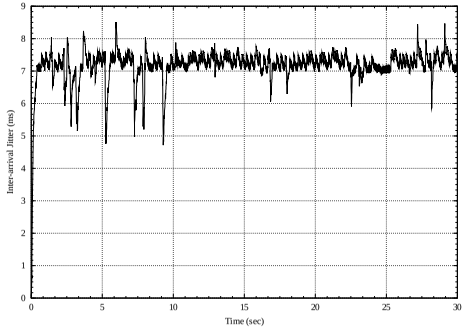
(c) paddind 1425



(d) packing 1275



(e) packing 1350



(f) packing 1425

Σχήμα Α΄.18: Εξομαλυμένη μεταβολή της καθυστέρησης

Παράρτημα Β′

Κώδικες

B.1 ULE Ενθυλακωτής

B.1.1 ulencap.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pcap.h> /* if this gives you an error try pcap/pcap.h */
4  #include <errno.h>
5  #include <sys/socket.h>
6  #include <netinet/in.h>
7  #include <arpa/inet.h>
8  #include <netinet/if_ether.h> /* includes net/ethernet.h */
9  #include <time.h> /* strftime */
10
11 #include <pthread.h>
12
13 #include "ulenc.h"
14 #include "stuff_tx.h"
15
16 #include "buffer.h"
17 #include "captureThread.h"
18
19
20 char errbuf[PCAP_ERRBUF_SIZE];
21 char *dev;
22
23 pthread_mutex_t mutexfd;
24 pthread_mutex_t mutexpush;
25
26 int main(int argc, char **argv) {
27     pthread_t captIPs[MAX_IPS];
28     char                ips[MAX_IPS][80];
29     int                 streams=0;
30     FILE      *fpc;
31
32     init_buffer(&ip_buff, 1000, 1560);
33     init_buffer(&ts_buff, 53, 188);
```

```
34
35      if(configure_tx(argc, argv) !=0 ){
36          printf("Error␣configuring␣the␣card␣\n");
37          return 1;
38      }
39
40      pthread_mutex_init(&mutexfd, NULL);
41      pthread_mutex_init(&mutexpush, NULL);
42
43      init_transmitter();
44
45
46  //              printf("\naaok");
47      fpc = fopen("ule.conf", "r");
48
49      if (fpc!=NULL) {
50          int i=0;
51          char ch;
52          while ( (ch=fgetc(fpc)) != EOF ) {
53              if ((ch=='\n')) {
54                  ips[streams][i]= '\0';
55                  pthread_create(&captIPs[streams], NULL,
56                          captureIP, (void*)ips[streams]);  // TODO: pass a
                                struct instead with char** streams and buffer*
57  //                              pthread_join(&captIPs[streams++], NULL);
58  //                printf("ip: %s stream %d\n"
59  //                          , ips[streams-1], streams);
60                  i=0;
61              } else {
62                  ips[streams][i++] = ch;
63              }
64          }
65          fclose(fpc);
66      } else {
67          printf("can't␣find␣configuration␣file...\n\n");
68          exit(2);
69      }
70
71      pthread_t thr1;
72      pthread_create( &thr1, NULL, sendPAT, (void*)streams);
73  //      pthread_join( &thr1, NULL);
74
75
76      pthread_t thr2;
77      pthread_create( &thr2, NULL, dequeue, (void*)NULL);
78
79      while (1) {
80          usleep(100000);
81      }
82
83      return 0;
84  }
```

B.1.2 buffer.c

```
1
2   #include "buffer.h"
3
4
5   void
```

```
6  push(_buffer* b, unsigned char *d, int size) {
7  //      printf("addind packet\n");
8      int i;
9      static int pkt;
10      if(buffer_full(b))
11           return;
12
13      b->head->length = size>b->nsize?b->nsize:size;
14
15 /*
16          printf("before %d:\n", ++pkt);
                                                  // DEBUG
17          for (i = 0; i < b->head->length; i++)                          //
                DEBUG
18              printf("%02x%s", d[i],(i+1)%32?" ":"\n");      // DEBUG
19          printf("\n");                                        // DEBUG
20 */
21      memcpy(b->head->data, d, b->head->length);
22
23 /*
24          printf("after %d:\n", pkt);                                    //
                DEBUG
25          for (i = 0; i < b->head->length; i++)                          //
                DEBUG
26              printf("%02x%s", b->head->data[i],(i+1)%32?" ":"\n");      //
                    DEBUG
27          printf("\n");                                          // DEBUG
28 */
29      b->head = b->head->next;
30 //     printf("packet added!\n");
31 }
32
33
34 bNode *
35 pop(_buffer* b) {
36      if(buffer_empty(b)) return (bNode*)NULL;
37
38      bNode* tmp = b->tail;
39      b->tail = b->tail->next;
40      return tmp;
41 }
42
43
44 int
45 buffer_full(_buffer* b) {
46      return b->head->next==b->tail;
47 }
48
49
50 int
51 buffer_empty(_buffer* b) {
52      return b->head==b->tail;
53 }
54
55
56 void
57 init_buffer(_buffer** b, int length, int nsize) {
58      int i;
59
60      *b = malloc(sizeof(_buffer));
61
62      bNode* tmp = malloc(sizeof(bNode));
```

```
63      tmp->data = malloc(nsize*sizeof(unsigned char));
64      (*b)->head = tmp;
65      (*b)->tail = tmp;
66
67      for (i=1; i<length; i++) {
68           bNode* tmp2 = malloc(sizeof(bNode));
69           tmp2->data = malloc(nsize*sizeof(unsigned char));
70           tmp->next = tmp2;
71           tmp = tmp->next;
72      }
73
74      tmp->next = (*b)->head;
75
76      (*b)->length = length;
77      (*b)->nsize  = nsize;
78  }
```

B.1.3 captureThread.c

```
1   #include "captureThread.h"
2   #include "crc.h"
3   #include "buffer.c"
4   #include "tspacket.h"
5
6   #define WIDTH    (8 * sizeof(crc))
7   #define TOPBIT   (1 << (WIDTH - 1))
8
9   #if (REFLECT_DATA == TRUE)
10  #undef  REFLECT_DATA
11  #define REFLECT_DATA(X)                 ((unsigned char) reflect((X), 8))
12  #else
13  #undef  REFLECT_DATA
14  #define REFLECT_DATA(X)                 (X)
15  #endif
16
17  #if (REFLECT_REMAINDER == TRUE)
18  #undef  REFLECT_REMAINDER
19  #define REFLECT_REMAINDER(X)    ((crc) reflect((X), WIDTH))
20  #else
21  #undef  REFLECT_REMAINDER
22  #define REFLECT_REMAINDER(X)    (X)
23  #endif
24
25
26
27  extern int fd;
28  extern pthread_mutex_t mutexfd;
29  extern pthread_mutex_t mutexpush;
30
31  void* captureIP(void *salami) {
32      const char     *str = (char*) salami;
33      unsigned char  *sndu;
34      int i;
35
36      printf("starting␣thread␣%s\n", str);
37      usleep(100000);
38
39      char* errbuf;
40      char* dev;
41      pcap_t* descr;
```

```
42      const u_char *packet;
43      struct pcap_pkthdr hdr;     /* pcap.h */
44      struct ether_header *eptr;  /* net/ethernet.h */
45      struct bpf_program filter;
46      bpf_u_int32 netp;
47
48      u_char *ptr; /* printing out hardware header info */
49
50      dev = pcap_lookupdev(errbuf);
51
52      if(dev == NULL) {
53          printf("%s\n", errbuf);
54          exit(1);
55      }
56      printf("listening on %s\n", dev);
57
58  //    printf("ok 0\n");
59      descr = pcap_open_live(dev, BUFSIZ, 0, -1, errbuf);
60  //    printf("ok 1 \n");
61
62      if(descr == NULL) {
63          printf("pcap_open_live(): %s\n", errbuf);
64          exit(1);
65      }
66
67      char tmp[80];
68      strcat(tmp, "src host ");
69      strcat(tmp, str);
70      if(pcap_compile( descr, &filter, tmp , 0, netp) == -1) {
71          printf("pcap_compile() error\n"); exit(1);
72      }
73
74      if( pcap_setfilter( descr, &filter) == -1)
75      { printf("pcap_setfilter: %s\n", pcap_geterr(descr) ); exit(1); }
76
77
78      while(1){
79
80          packet = pcap_next(descr, &hdr);
81
82          if(packet == NULL) {
83              printf("Warning: Didn't grab packet\n");
84              continue;
85  //            exit(1);
86          }
87          eptr = (struct ether_header *) packet;
88
89
90          // TODO: implement my make_sndu()
91          if (ntohs(eptr->ether_type) == ETHERTYPE_IP) {
92  //                send_packet(packet+14, hdr.len - 14);
93  //                push(ip_buff, packet+14, hdr.len - 14); // enqueue ip
    packet
94              push(ip_buff, make_sndu(packet+14, hdr.len - 14, 0x0800)
95              , (hdr.len-14)+8); // enqueue ip packet
96          }else  if (ntohs(eptr->ether_type) == ETHERTYPE_ARP) {
97              printf("Ethernet type hex:%x dec:%d is an ARP packet\n",
98                      ntohs(eptr->ether_type),
99                      ntohs(eptr->ether_type));
100         }else {
101             printf("Ethernet type %x not IP", ntohs(eptr->ether_type));
102             exit(1);
```

```
103         }
104     }
105 }
106
107 void* sendPAT(void *ptr) {
108     FILE*               ff;
109     int             i, j, tmp, counter=0;
110     unsigned long   crcc;
111     int             nStr = (int)ptr;
112     unsigned char       patPacket[188];
113     unsigned char       pmtPacket[188];
114     unsigned char       sdtPacket[188];
115     unsigned char       temp[188];
116
117     /***************************************
118      *           PAT Packet                *
119      ***************************************/
120
121     patPacket[0] = 0x47;                // ts header
122     patPacket[1] = 0x40;
123     patPacket[2] = 0x00;
124     patPacket[3] = 0x10;
125     // PAT header
126     patPacket[4] = 0x00;                // PID
127     patPacket[5] = 0x00;                // table-id
128     tmp = 5 + nStr*4 + 4;               // Section_length
129     patPacket[6] = 0xb0 | (tmp>>8);
130     patPacket[7] = tmp & 0xff;
131     tmp = 0x0222;                       // Transport_Stream_ID
132     patPacket[8] = tmp>>8;
133     patPacket[9] = tmp&0xff;
134     patPacket[10] = 0xc1;
135     patPacket[11] = 0x00;
136     patPacket[12] = 0x00;
137     tmp = 0x0005;                    // progam_number
138     patPacket[13]   = tmp>>8;
139     patPacket[14] = tmp&0xff;
140     tmp = 0xe000 | (0x30);      // Program_map_PID
141     patPacket[15] = tmp>>8;
142     patPacket[16] = tmp&0xff;
143     for (i=5; i<17; i++) {
144         temp[i-5] = patPacket[i];
145     }
146     crcc = dvb_crc32_calc(&patPacket[5], 17-5);
147     patPacket[17] = crcc>>24;
148     patPacket[18] = (crcc>>16)&0xff;
149     patPacket[19] = (crcc>>8)&0xff;
150     patPacket[20] = crcc&0xff;          // ba e2 3e 29
151     /*      patPacket[17] = 0xba;
152      * patPacket[18] = 0xe2;
153      * patPacket[19] = 0x3e;
154      * patPacket[20] = 0x29;*/
155     /*      tmp = 0xe000 | (0x48+((j-13)/4));     // Program_map_PID
156      * for (j=13; j<=(nStr*4)+13; j+=4) {
157      * tmp = 0x0005+((j-13)/4);          // progam_number
158      * patPacket[j]   = tmp>>8;
159      * patPacket[j+1] = tmp&0xff;
160      * tmp = 0xe000 | (0x48+((j-13)/4));         // Program_map_PID
161      * patPacket[j+2] = tmp>>8;
162      * patPacket[j+3] = tmp&0xff;
163      * }
164      * patPacket[j-4] = 0x11;
```

```
165       * patPacket[j-3] = 0x11;
166       * patPacket[j-2] = 0x11;
167       * patPacket[j-1] = 0x11;*/
168      for (i=21; i<188; i++)
169          patPacket[i] = 0xff;
170
171      /*************************************
172       *           PMT Packet                      *
173       *************************************/
174
175      pmtPacket[0] = 0x47;                 // ts header
176      pmtPacket[1] = 0x40;
177      pmtPacket[2] = 0x30;
178      pmtPacket[3] = 0x10;
179
180      pmtPacket[4] = 0x00;
181      pmtPacket[5] = 0x02;
182      pmtPacket[6] = 0xb0;                 // fix
183      pmtPacket[7] = 0x00;                 // section_length later
184      tmp = 0x0005;                        // program_Number
185      pmtPacket[8] = tmp>>8;
186      pmtPacket[9] = tmp&0xff;
187      pmtPacket[10] = 0xff;
188      pmtPacket[11] = 0x00;
189      pmtPacket[12] = 0x00;
190      tmp = 0x1fff;                        // PCR_PID
191      pmtPacket[13] = 0xe5 | (tmp>>8);
192      pmtPacket[14] = tmp&0xff;
193      tmp = 0x0000;                        // program_info_length
194      pmtPacket[15] = 0xf0 | (tmp>>8);
195      pmtPacket[16] = tmp&0xff;    // replace with for for file data
196
197      for (j=17, i=0; i<nStr; i++) {
198          pmtPacket[j++] = 0x0c;
199          tmp = 0x150 + i;         // elementary_PID
200          pmtPacket[j++] = 0xe0 | (tmp>>8);
201          pmtPacket[j++] = tmp&0xff;
202          tmp = 0x0000;            // ES_info_length (rep with for)
203          pmtPacket[j++] = 0xf0 | (tmp>>8);
204          pmtPacket[j++] = tmp&0xff;
205      }
206      tmp = j-4;                   // section_length
207      pmtPacket[6] = 0xb0 | (tmp>>8);      // fix
208      pmtPacket[7] = tmp&0xff;
209      for (i=5; i<j; i++) {
210          temp[i-5] = pmtPacket[i];
211      }
212      crcc = dvb_crc32_calc(&pmtPacket[5], j-5);
213      pmtPacket[j++] = crcc>>24;
214      pmtPacket[j++] = (crcc>>16)&0xff;
215      pmtPacket[j++] = (crcc>>8)&0xff;
216      pmtPacket[j++] = crcc&0xff;            // df 1a ef c0
217      /*      pmtPacket[j++] = 0xdf;
218       * pmtPacket[j++] = 0x1a;
219       * pmtPacket[j++] = 0xef;
220       * pmtPacket[j++] = 0xc0; */
221      for (i=j; i<188; i++)
222          pmtPacket[i] = 0xff;
223
224      /*************************************
225       *           SDT Packet                      *
226       *************************************/
```

```
227
228     sdtPacket[0] = 0x47;                // ts header
229     sdtPacket[1] = 0x40;
230     sdtPacket[2] = 0x11;
231     sdtPacket[3] = 0x10;
232     // SDT header
233     sdtPacket[4] = 0x00;                //
234     sdtPacket[5] = 0x42;                // table-id
235     sdtPacket[6] = 0xb0;                // fix
236     sdtPacket[7] = 0x00;                // section_length later
237     tmp = 0x0222;                           // transport stream id
238     sdtPacket[8] = tmp>>8;
239     sdtPacket[9] = tmp&0xff;
240     sdtPacket[10] = 0xc1;
241     sdtPacket[11] = 0x00;
242     sdtPacket[12] = 0x00;
243     tmp = 0x1234;                           // original_network_id
244     sdtPacket[13] = tmp>>8;
245     sdtPacket[14] = tmp&0xff;
246     sdtPacket[15] = 0xff;
247     tmp = 0x0005;                           // service_id
248     sdtPacket[16] = tmp>>8;
249     sdtPacket[17] = tmp&0xff;
250     sdtPacket[18] = 0xfc;
251     sdtPacket[19] = 0x80;
252     sdtPacket[20] = 0x15;               // descriptors_loop_length
253     sdtPacket[21] = 0x48;               // dvb desc tag
254     sdtPacket[22] = 0x13;               // descriptor length
255     sdtPacket[23] = 0x0c;               // service_type
256     sdtPacket[24] = 0x06;               // service_provider_name_length
257     sdtPacket[25] = 'n'; sdtPacket[26] = 'k';
258     sdtPacket[27] = 'v'; sdtPacket[28] = 'o';
259     sdtPacket[29] = 'r'; sdtPacket[30] = 'n';
260     sdtPacket[31] = 0x0a;               // service_name_length
261     sdtPacket[32] = 'D'; sdtPacket[33] = 'V';
262     sdtPacket[34] = 'B'; sdtPacket[35] = ' ';
263     sdtPacket[36] = 'h'; sdtPacket[37] = 'a';
264     sdtPacket[38] = 'c'; sdtPacket[39] = 'k';
265     sdtPacket[40] = 'e'; sdtPacket[41] = 'r';
266     tmp = 46-8;                     // section_length
267     sdtPacket[6] = 0xb0 | (tmp>>8);     // fix
268     sdtPacket[7] = tmp&0xff;
269     crcc = dvb_crc32_calc(&sdtPacket[5], 42-5);
270     sdtPacket[42] = crcc>>24;
271     sdtPacket[43] = (crcc>>16)&0xff;
272     sdtPacket[44] = (crcc>>8)&0xff;
273     sdtPacket[45] = crcc&0xff;
274     for (i=46; i<188; i++)
275         sdtPacket[i] = 0xff;
276
277
278
279
280 //      ff = fopen("out","wb");
281     while (1) {
282         patPacket[3] = 0x10 | counter;
283         pmtPacket[3] = 0x10 | counter;
284         sdtPacket[3] = 0x10 | counter;
285 //      printf("crc %04x\n", calc_crc(patPacket, 16));
286 //      printf("crc %04x\n", calc_crc(pmtPacket, 16));
287 //      printf("crc %04x\n", calc_crc(sdtPacket, 16));
288
```

```
289         /*              for (i=0; i<188; i++)
290          * printf("%02x%s", patPacket[i], (i+1)%16?" ":"\n");
291          * printf("\n");
292          * for (i=0; i<188; i++)
293          * printf("%02x%s", pmtPacket[i], (i+1)%16?" ":"\n");
294          * printf("\n");
295          * for (i=0; i<188; i++)
296          * printf("%02x%s", sdtPacket[i], (i+1)%16?" ":"\n");
297          * printf("\n");
298          * fwrite (patPacket, sizeof(patPacket[0]), sizeof(patPacket), ff);
299          * fwrite (pmtPacket, sizeof(patPacket[0]), sizeof(patPacket), ff);
300          * fwrite (sdtPacket, sizeof(pmtPacket[0]), sizeof(pmtPacket), ff);
301          * fflush(ff); */
302
303         pthread_mutex_lock(&mutexfd);
304         send_tp(patPacket);
305         send_tp(pmtPacket);
306         send_tp(sdtPacket);
307         pthread_mutex_unlock(&mutexfd);
308         usleep(100000);
309         counter = ++counter % 16;
310     }
311
312 //      while(1) {
313 //              printf("Streams in PAT(%d): ", nStr);
314 //              for (i=0; i<nStr; i++) {
315 //                      printf("0x%03x ", 100+i);
316 //              }
317 //              printf("\n");
318 //              usleep(900000);
319 //      }
320 }
321
322
323 unsigned int dvb_crc_table[256] = {
324     0x00000000, 0x04c11db7, 0x09823b6e, 0x0d4326d9,
325     0x130476dc, 0x17c56b6b, 0x1a864db2, 0x1e475005,
326     0x2608edb8, 0x22c9f00f, 0x2f8ad6d6, 0x2b4bcb61,
327     0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbdbd,
328     0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9,
329     0x5f15adac, 0x5bd4b01b, 0x569796c2, 0x52568b75,
330     0x6a1936c8, 0x6ed82b7f, 0x639b0da6, 0x675a1011,
331     0x791d4014, 0x7ddc5da3, 0x709f7b7a, 0x745e66cd,
332     0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,
333     0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5,
334     0xbe2b5b58, 0xbaea46ef, 0xb7a96036, 0xb3687d81,
335     0xad2f2d84, 0xa9ee3033, 0xa4ad16ea, 0xa06c0b5d,
336     0xd4326d90, 0xd0f37027, 0xddb056fe, 0xd9714b49,
337     0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
338     0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1,
339     0xe13ef6f4, 0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d,
340     0x34867077, 0x30476dc0, 0x3d044b19, 0x39c556ae,
341     0x278206ab, 0x23431b1c, 0x2e003dc5, 0x2ac12072,
342     0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcdbb16,
343     0x018aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca,
344     0x7897ab07, 0x7c56b6b0, 0x71159069, 0x75d48dde,
345     0x6b93dddb, 0x6f52c06c, 0x6211e6b5, 0x66d0fb02,
346     0x5e9f46bf, 0x5a5e5b08, 0x571d7dd1, 0x53dc6066,
347     0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,
348     0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e6e04e,
349     0xbfa1b04b, 0xbb60adfc, 0xb6238b25, 0xb2e29692,
350     0x8aad2b2f, 0x8e6c3698, 0x832f1041, 0x87ee0df6,
```

```
351     0x99a95df3, 0x9d684044, 0x902b669d, 0x94ea7b2a,
352     0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e,
353     0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2,
354     0xc6bcf05f, 0xc27dede8, 0xcf3ecb31, 0xcbffd686,
355     0xd5b88683, 0xd1799b34, 0xdc3abded, 0xd8fba05a,
356     0x690ce0ee, 0x6dcdfd59, 0x608edb80, 0x644fc637,
357     0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,
358     0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f,
359     0x5c007b8a, 0x58c1663d, 0x558240e4, 0x51435d53,
360     0x251d3b9e, 0x21dc2629, 0x2c9f00f0, 0x285e1d47,
361     0x36194d42, 0x32d850f5, 0x3f9b762c, 0x3b5a6b9b,
362     0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,
363     0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623,
364     0xf12f560e, 0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7,
365     0xe22b20d2, 0xe6ea3d65, 0xeba91bbc, 0xef68060b,
366     0xd727bbb6, 0xd3e6a601, 0xdea580d8, 0xda649d6f,
367     0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,
368     0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7,
369     0xae3afba2, 0xaafbe615, 0xa7b8c0cc, 0xa379dd7b,
370     0x9b3660c6, 0x9ff77d71, 0x92b45ba8, 0x9675461f,
371     0x8832161a, 0x8cf30bad, 0x81b02d74, 0x857130c3,
372     0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640,
373     0x4e8ee645, 0x4a4ffbf2, 0x470cdd2b, 0x43cdc09c,
374     0x7b827d21, 0x7f436096, 0x7200464f, 0x76c15bf8,
375     0x68860bfd, 0x6c47164a, 0x61043093, 0x65c52d24,
376     0x119b4be9, 0x155a565e, 0x18197087, 0x1cd86d30,
377     0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,
378     0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088,
379     0x2497d08d, 0x2056cd3a, 0x2d15ebe3, 0x29d4f654,
380     0xc5a92679, 0xc1683bce, 0xcc2b1d17, 0xc8ea00a0,
381     0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb, 0xdbee767c,
382     0xe3a1cbc1, 0xe760d676, 0xea23f0af, 0xeee2ed18,
383     0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4,
384     0x89b8fd09, 0x8d79e0be, 0x803ac667, 0x84fbdbd0,
385     0x9abc8bd5, 0x9e7d9662, 0x933eb0bb, 0x97ffad0c,
386     0xafb010b1, 0xab710d06, 0xa6322bdf, 0xa2f33668,
387     0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4
388 };
389
390
391
392 unsigned long dvb_crc32_calc( const unsigned char *sectbuf, unsigned int
        size ) {
393     unsigned long crc32 = 0xffffffff;
394     unsigned int i = 0;
395
396     for (i = 0; i < size; i++)
397         crc32 = (crc32 << 8) ^ dvb_crc_table[(((crc32 >> 24) ^ sectbuf[i]) &
                0xff)];
398
399     return crc32;
400 }
401
402 void dequeue() {
403     bNode   *sndus[16];
404     ts_pkt  *tp;
405     static unsigned char pkt[188];
406     unsigned char tmp[188];
407     int     len=0, rem=0, i, k, j, f, prev=0, pos=0, snd=0;
408     int     packing = 1, count, firstPaddPack = 0;
409     static long bytes_in = 0;
410     static long bytes_out = 0;
```

```
411     int ippacks = 0, tspacks = 0;
412
413     pkt[0] = 0x47;
414     pkt[1] = 0x1f;
415     pkt[2] = 0xff;
416     pkt[3] = 0x10;
417     for (i = 4; i < 188; i++)
418         pkt[i] = 0xff;
419
420 //    printf("\t\t\t==== in dequeue() ====\n");                  // DEBUG
421     tp    = malloc(sizeof(ts_pkt*));
422     for (i=0; i<16; i++)
423         sndus[i] = malloc(sizeof(bNode*));
424
425 //    printf("\tdequeue 1\n");                 // DEBUG
426     tp->sync         = 0x47;
427     tp->tr_err_ind   = 0x0;
428     tp->tr_prio      = 0x0;
429     tp->pid          = 0x150;
430     tp->ad_field     = 0x0;
431     tp->tr_sc_ctrl   = 0x0;
432
433     while (1) {
434
435         usleep(10000);
436 //    printf("\tdequeue 3 em=%d !pr=%d\n", !buffer_empty(ip_buff),prev);
    // DEBUG
437     while (!buffer_empty(ip_buff) || prev) {
438         firstPaddPack = 1;
439         //printf("\tdequeue main loop\n");                 // DEBUG
440         pos = 0;
441        //printf("\tdequeue 3 prev=%d\n", prev);                 // DEBUG
442         if (!prev) {
443             sndus[0] = pop(ip_buff);
444             bytes_in += sndus[0]->length;
445             ++ippacks;
446             //printf("\n--- poped packet of %d bytes---\n",sndus[0]->length)
                ;  // DEBUG
447             //for (i = 0; i < sndus[0]->length; i++)                // DEBUG
448                //printf("%02x%s", sndus[0]->data[i],(i+1)%32?" ":"\n");
                              // DEBUG
449             //printf("\n");                                                //
                DEBUG
450             //printf("\tdequeue 3.0 %d\n", sndus[0]->length);
                            // DEBUG
451             len = sndus[0]->length;
452
453             //printf("\tdequeue 3.1\n");                 // DEBUG
454             if (len >= 183) {
455                 tp->data[0] = 0;
456                 tp->cc = ++count % 16;
457                 tp->pusi = 1;
458                 if (!packing)
459                     firstPaddPack = 0;
460                 //printf("\tdequeue 3.2 i=%d pos=%d\n", i, pos);
                              // DEBUG
461                 for (i = 1; i < 184; i++)
462                     tp->data[i] = sndus[0]->data[pos++];
463                     ts_data(tp,tmp);
464                     pthread_mutex_lock(&mutexpush);
465                     push(ts_buff,tmp,188);
466                     pthread_mutex_unlock(&mutexpush);
```

```
467             }
468         } else {
469             len = sndus[0]->length;
470             pos = prev;
471             //printf("\tdequeue 3p prev=%d len=%d\n", prev, len);
                                    // DEBUG
472             prev = 0;
473         }
474         while (len-pos>184) {
475             //printf("\tdequeue 4\n");                 // DEBUG
476 //            if (prev)
477 //                prev = 0;
478             tp->cc = ++count % 16;
479             tp->pusi = 0;
480             for (i = 0; i < 184; i++)
481                 tp->data[i] = sndus[0]->data[pos++];
482                 ts_data(tp,tmp);
483                 pthread_mutex_lock(&mutexpush);
484                 push(ts_buff,tmp,188);
485                 pthread_mutex_unlock(&mutexpush);
486         }
487         if (len-pos>0) {
488                 //printf("\tdequeue pack=%d len-pos=%d\n", packing,len-pos);
                        // DEBUG
489             if (packing && len-pos < 182) {
490                 //printf("\tdequeue 5 em=%d\n",!buffer_empty(ip_buff));
                                    // DEBUG
491                 snd = 0;
492                 int lng = len-pos;
493                 while (!buffer_empty(ip_buff)) {
494                     sndus[++snd] = pop(ip_buff);
495                     bytes_in += sndus[snd-1]->length;
496                     ++ippacks;
497             //printf("\n--- poped packet of %d bytes---\n",sndus[snd]->
                    length);  // DEBUG
498             //for (i = 0; i < sndus[snd]->length; i++)                  //
                    DEBUG
499                     //printf("%02x%s", sndus[snd]->data[i],(i+1)%32?" ":"\n");
                                    // DEBUG
500             //printf("\n");                                             //
                    DEBUG
501                     lng += sndus[snd]->length;
502             //printf("\t\t\tlng=%d\n", lng);                                 //
                    DEBUG
503                     if (lng > 183)
504                         break;
505                 }
506                 //printf("\tdequeue 5.1 snd=%d lng=%d\n",snd,lng);
                                    // DEBUG
507                 tp->data[0] = len-pos;
508                         //printf("--start------");       // DEBUG
509                 for (j = 1; j < 184; j++) {
510                     if (len-pos > 0) {
511                         tp->data[j] = sndus[0]->data[pos++];
512                         //printf("[%d,%d]%02x ",j,pos,sndus[0]->data[pos-1])
                                ;       // DEBUG
513                     } else
514                         break;
515                 }
516
517                 if (snd > 0) {
518                     k=0;
```

```
519                     while (++k<=snd) {
520                         f=0;
521                         while (f < sndus[k]->length) {
522                             //printf("%02x ",sndus[k]->data[f]);      //
                                    DEBUG
523                             //printf("(%d,%d) ",j,f);      // DEBUG
524                             tp->data[j++] = sndus[k]->data[f++];
525                             if (j>184) {
526                                 prev = f-1;
527                                 sndus[0] = sndus[k];
528                                 goto out;
529                             }
530                         }
531                     }
532                     prev = 0;
533 out:
534                     //printf("----end------\n");       // DEBUG
535                     for (i=j-1; i<184; i++)
536                         tp->data[i] = 0xFF;
537                 } else {
538                     for (i=j; i<184; i++)
539                         tp->data[i] = 0xFF;
540                 }
541                 tp->pusi = 1;
542             } else {                        // padding
543                 //printf("\t!padding\n");                 // DEBUG
544                 tp->pusi = firstPaddPack;
545                 for (j = 0; j < 184; j++) {
546                     if (len-pos >= 0)
547                         tp->data[j] = sndus[0]->data[pos++];
548                     else
549                         break;
550                 }
551                 for (i=j-1; i<=184; i++)
552                     tp->data[i] = 0xFF;
553             }
554             tp->cc = ++count % 16;
555                 ts_data(tp,tmp);
556                 pthread_mutex_lock(&mutexpush);
557                 push(ts_buff,tmp,188);
558                 pthread_mutex_unlock(&mutexpush);
559     }
560
561
562     while (!buffer_full(ts_buff)) {
563             pthread_mutex_lock(&mutexpush);
564             ++tspacks;
565             push(ts_buff, pkt,188);
566             pthread_mutex_unlock(&mutexpush);
567     }
568
569     bNode *ttmp;
570     ttmp = malloc(sizeof(bNode*));
571     while (!buffer_empty(ts_buff)) {
572         ttmp = pop(ts_buff);
573 /*
574         for (i = 0; i < 188; i++)                               // DEBUG
575             printf("%02x%s", ttmp->data[i],(i+1)%32?" ":"\n");      // DEBUG
576         printf("\n");                                           // DEBUG
577 */
578             pthread_mutex_lock(&mutexfd);
579             send_tp(ttmp->data);
```

```
580             pthread_mutex_unlock(&mutexfd);
581     }
582
583
584     // edo einai ta null
585 //    printf("\t --- buf em %d, pkt\n",
586 //             buffer_empty(ip_buff));
587
588     if (++count%100) {
589         printf("bytes in:\t%d\n", bytes_in);
590         printf("bytes out:\t%d\n", tspacks*188);
591     }
592     }
593
594 }
595
596 /*
597 unsigned char *null_pkt(unsigned char *pkt) {
598     int i;
599
600     pkt[0] = 0x47;
601     pkt[1] = 0x1f;
602     pkt[2] = 0xff;
603     pkt[3] = 0x10;
604     for (i = 4; i < 188; i++)
605         pkt[i] = 0xff;
606
607     return pkt;
608 }
609 */
610
611 void ts_data(ts_pkt* ts, unsigned char *tmp) {
612     int i;
613     static int cnt = 0;
614
615     //printf("\t in ts_data() packet # %07d\n", ++cnt);       // DEBUG
616
617     *tmp    = ts->sync;
618     *(tmp+1) = (ts->tr_err_ind & 0x1) << 7 |
619                (ts->pusi       & 0x1) << 6 |
620                (ts->tr_prio    & 0x1) << 5 |
621                (ts->pid        >> 8) & 0x1F;
622     *(tmp+2) = ts->pid         & 0xFF;
623     *(tmp+3) = (ts->tr_sc_ctrl & 0x3) << 6 |
624                (ts->ad_field   & 0x3) << 4 |
625                ts->cc          & 0xF;
626
627     for (i = 4; i < 188; i++) {
628         *(tmp+i) = ts->data[i-4];
629     }
630
631 /*
632     for (i = 0; i < 188; i++)                                 // DEBUG
633         printf("%02x%s", tmp[i],(i+1)%32?" ":"\n");           // DEBUG
634     printf("\n");                                             // DEBUG
635 */
636
637
638 }
```

B.1.4 stuff_tx.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <string.h>
6  #include <sys/ioctl.h>
7  #include <sys/stat.h>
8  #include <sys/poll.h>
9  #include <sys/time.h>
10
11 #include "asi.h"
12 #include "master.h"
13 #include "util.h"
14
15 #define BUFLEN 256
16
17 static const char progname[] = "txtest";
18
19
20
21 int configure_tx(int argc, char **argv)
22 {
23
24
25
26         const char fmt[] = "/sys/class/asi/asitx%i/%s";
27         int opt;
28         int fd, period, quiet, seconds, verbose, packetsize, num;
29         unsigned long int bufsize, mode, clksrc, timestamps, transport;
30         int read_ret, write_ret, val;
31         unsigned int bytes_per_sec;
32         struct stat buf;
33         unsigned char *data;
34         struct asi_txstuffing stuffing;
35         struct timeval tv;
36         unsigned int cap, bytes_written, bytes_read, bytes;
37         struct pollfd pfd;
38         double status_bytes, bitrate, time_sec, lasttime, dt;
39         char name[BUFLEN], str[BUFLEN], *endptr;
40
41         /* Parse the command line */
42         period = 0;
43         quiet = 0;
44         seconds = -1;
45         verbose = 0;
46         while ((opt = getopt (argc, argv, "hn:qs:vV")) != -1) {
47                 switch (opt) {
48                 case 'h':
49                         printf ("Usage: %s [OPTION]... DEVICE_FILE [IB IP\n"
50                                 "\t[NORMAL_IP BIG_IP [IL_NORMAL IL_BIG]]]\n"
                                        ,
51                                 argv[0]);
52                         printf ("Copy standard input to DEVICE_FILE "
53                                 "with interbyte stuffing IB,\n"
54                                 "interpacket stuffing "
55                                 "IP + (BIG_IP / (BIG_IP + NORMAL_IP)),\n"
56                                 "and interleaved finetuning parameters "
57                                 "IL_NORMAL and IL_BIG\n"
58                                 "while monitoring for "
```

```
59                              "DVB␣ASI␣transmitter␣events.\n\n");
60                      printf ("␣␣-h\t\tdisplay␣this␣help␣and␣exit\n");
61                      printf ("␣␣-n␣TIME\tstop␣transmitting␣"
62                              "after␣TIME␣seconds\n");
63                      printf ("␣␣-q\t\tquiet␣operation\n");
64                      printf ("␣␣-s␣PERIOD\tdisplay␣status␣"
65                              "every␣PERIOD␣seconds\n");
66                      printf ("␣␣-v\t\tverbose␣output\n");
67                      printf ("␣␣-V\t\toutput␣version␣information␣"
68                              "and␣exit\n");
69                      printf ("\nIf␣TIME␣<␣0,␣transmission␣never␣stops␣"
70                              "(default).\n");
71                      printf ("\nReport␣bugs␣to␣<support@linsys.ca>.\n");
72                      return 0;
73              case 'n':
74                      seconds = strtol (optarg, &endptr, 0);
75                      if (*endptr != '\0') {
76                              fprintf (stderr,
77                                      "%s:␣invalid␣timeout:␣%s\n",
78                                      argv[0], optarg);
79                              return -1;
80                      }
81                      break;
82              case 'q':
83                      quiet = 1;
84                      break;
85              case 's':
86                      period = strtol (optarg, &endptr, 0);
87                      if (*endptr != '\0') {
88                              fprintf (stderr,
89                                      "%s:␣invalid␣period:␣%s\n",
90                                      argv[0], optarg);
91                              return -1;
92                      }
93                      break;
94              case 'v':
95                      verbose = 1;
96                      break;
97              case 'V':
98                      printf ("%s␣from␣master-%s␣(%s)\n", progname,
99                              MASTER_DRIVER_VERSION,
100                             MASTER_DRIVER_DATE);
101                     printf ("\nCopyright␣(C)␣2000-2005␣"
102                             "Linear␣Systems␣Ltd.\n"
103                             "This␣is␣free␣software;␣"
104                             "see␣the␣source␣for␣copying␣conditions.␣␣"
105                             "There␣is␣NO\n"
106                             "warranty;␣not␣even␣for␣MERCHANTABILITY␣"
107                             "or␣FITNESS␣FOR␣A␣PARTICULAR␣PURPOSE.\n");
108                     return 0;
109             case '?':
110                     goto USAGE;
111             }
112     }
113
114     /* Check the number of arguments */
115     if (((argc - optind) == 2) ||
116             ((argc - optind) == 4) ||
117             ((argc - optind) == 6)) {
118             if (!quiet) {
119                     fprintf (stderr, "%s:␣missing␣arguments\n", argv[0])
                            ;
```

```
120                     goto USAGE;
121             }
122             return -1;
123     } else if ((argc - optind) > 7) {
124             if (!quiet) {
125                     fprintf (stderr, "%s: extra operand\n", argv[0]);
126                     goto USAGE;
127             }
128             return -1;
129     }
130
131     /* Read the stuffing parameters */
132     memset (&stuffing, 0, sizeof (stuffing));
133     if ((argc - optind) > 2) {
134             stuffing.ib = strtol (argv[optind + 1], &endptr, 0);
135             if (*endptr != '\0') {
136                     fprintf (stderr,
137                             "%s: invalid interbyte stuffing: %s\n",
138                             argv[0], argv[optind + 1]);
139                     return -1;
140             }
141             stuffing.ip = strtol (argv[optind + 2], &endptr, 0);
142             if (*endptr != '\0') {
143                     fprintf (stderr,
144                             "%s: invalid interpacket stuffing: %s\n",
145                             argv[0], argv[optind + 2]);
146                     return -1;
147             }
148     }
149     if ((argc - optind) > 4) {
150             stuffing.normal_ip = strtol (argv[optind + 3], &endptr, 0);
151             if (*endptr != '\0') {
152                     fprintf (stderr,
153                             "%s: invalid finetuning parameter: %s\n",
154                             argv[0], argv[optind + 3]);
155                     return -1;
156             }
157             stuffing.big_ip = strtol (argv [optind + 4], &endptr, 0);
158             if (*endptr != '\0') {
159                     fprintf (stderr,
160                             "%s: invalid finetuning parameter: %s\n",
161                             argv[0], argv[optind + 4]);
162                     return -1;
163             }
164             if (stuffing.normal_ip == 0) {
165                     stuffing.big_ip = 0;
166             }
167             if ((argc - optind) > 6) {
168                     stuffing.il_normal = strtol (argv[optind + 5],
169                             &endptr, 0);
170                     if (*endptr != '\0') {
171                             fprintf (stderr,
172                                     "%s: invalid interleaving parameter:
                                         "
173                                     "%s\n",
174                                     argv[0], argv[optind + 5]);
175                             return -1;
176                     }
177                     stuffing.il_big = strtol (argv [optind + 6],
178                             &endptr, 0);
179                     if (*endptr != '\0') {
180                             fprintf (stderr,
```

```
181                                     "%s: invalid interleaving parameter:
                                            "
182                                     "%s\n",
183                                     argv[0], argv[optind + 6]);
184                             return -1;
185                     }
186             }
187     }
188
189     /* Get the sysfs info */
190     memset (&buf, 0, sizeof (buf));
191     if (stat (argv[optind], &buf) < 0) {
192             if (!quiet) {
193                     fprintf (stderr, "%s: ", argv[0]);
194                     perror ("unable to get the file status");
195             }
196             return -1;
197     }
198     if (!S_ISCHR (buf.st_mode)) {
199             if (!quiet) {
200                     fprintf (stderr, "%s: not a character device\n",
201                             argv[0]);
202             }
203             return -1;
204     }
205     if (buf.st_rdev & 0x0080) {
206             if (!quiet) {
207                     fprintf (stderr, "%s: not a transmitter\n", argv[0])
                            ;
208             }
209             return -1;
210     }
211     num = buf.st_rdev & 0x007f;
212     snprintf (name, sizeof (name), fmt, num, "dev");
213     memset (str, 0, sizeof (str));
214     if (util_read (name, str, sizeof (str)) < 0) {
215             if (!quiet) {
216                     fprintf (stderr, "%s: ", argv[0]);
217                     perror ("unable to get the device number");
218             }
219             return -1;
220     }
221     if (strtoul (str, &endptr, 0) != (buf.st_rdev >> 8)) {
222             if (!quiet) {
223                     fprintf (stderr, "%s: not an ASI device\n", argv[0])
                            ;
224             }
225             return -1;
226     }
227     if (*endptr != ':') {
228             if (!quiet) {
229                     fprintf (stderr, "%s: error reading %s\n",
230                             argv[0], name);
231             }
232             return -1;
233     }
234
235     /* Open the file */
236     if (verbose && !quiet) {
237             printf ("Opening %s.\n", argv[optind]);
238     }
239     if ((fd = open (argv[optind], O_WRONLY, 0)) < 0) {
```

```
240                 if (!quiet) {
241                         fprintf (stderr, "%s: ", argv[0]);
242                         perror ("unable to open file for writing");
243                 }
244                 return -1;
245         }
246
247         /* Get the transport type */
248         snprintf (name, sizeof (name), fmt, num, "transport");
249         if (util_strtoul (name, &transport) < 0) {
250                 if (!quiet) {
251                         fprintf (stderr, "%s: ", argv[0]);
252                         perror ("unable to get "
253                                 "the transmitter transport type");
254                 }
255                 close (fd);
256                 return -1;
257         }
258
259         /* Get the transmitter capabilities */
260         if (ioctl (fd, ASI_IOC_TXGETCAP, &cap) < 0) {
261                 if (!quiet) {
262                         fprintf (stderr, "%s: ", argv[0]);
263                         perror ("unable to get the transmitter capabilities"
                                    );
264                 }
265                 close (fd);
266                 return -1;
267         }
268
269         /* Get the buffer size */
270         snprintf (name, sizeof (name), fmt, num, "bufsize");
271         if (util_strtoul (name, &bufsize) < 0) {
272                 if (!quiet) {
273                         fprintf (stderr, "%s: ", argv[0]);
274                         perror ("unable to get "
275                                 "the transmitter buffer size");
276                 }
277                 close (fd);
278                 return -1;
279         }
280
281         /* Get the output packet size */
282         snprintf (name, sizeof (name), fmt, num, "mode");
283         if (util_strtoul (name, &mode) < 0) {
284                 if (!quiet) {
285                         fprintf (stderr, "%s: ", argv[0]);
286                         perror ("unable to get "
287                                 "the transmitter operating mode");
288                 }
289                 close (fd);
290                 return -1;
291         }
292         switch (mode) {
293         case ASI_CTL_TX_MODE_188:
294                 if (verbose && !quiet) {
295                         printf ("Assuming 188-byte packets.\n");
296                 }
297                 packetsize = 188;
298                 break;
299         case ASI_CTL_TX_MODE_204:
300                 if (verbose && !quiet) {
```

```
301                     printf ("Assuming 204-byte packets.\n");
302             }
303             packetsize = 204;
304             break;
305     case ASI_CTL_TX_MODE_MAKE204:
306             if (verbose && !quiet) {
307                     printf ("Appending sixteen 0x00 bytes to each "
308                             "188-byte packet.\n");
309             }
310             packetsize = 204;
311             break;
312     default:
313             if (!quiet) {
314                     fprintf (stderr, "%s: "
315                             "unknown transmitter operating mode\n",
316                             argv[0]);
317             }
318             close (fd);
319             return -1;
320     }
321
322     /* Get the clock source */
323     if (cap & ASI_CAP_TX_SETCLKSRC) {
324             snprintf (name, sizeof (name), fmt, num, "clock_source");
325             if (util_strtoul (name, &clksrc) < 0) {
326                     if (!quiet) {
327                             fprintf (stderr, "%s: ", argv[0]);
328                             perror ("unable to get "
329                                     "the clock source");
330                     }
331                     close (fd);
332                     return -1;
333             }
334     } else {
335             clksrc = 0;
336     }
337     if (verbose && !quiet) {
338             switch (clksrc) {
339             case ASI_CTL_TX_CLKSRC_ONBOARD:
340                     printf ("Using onboard oscillator.\n");
341                     break;
342             case ASI_CTL_TX_CLKSRC_EXT:
343                     printf ("Using external NTSC or 27 MHz reference.\n"
                                ");
344                     break;
345             case ASI_CTL_TX_CLKSRC_RX:
346                     printf ("Using recovered receive clock.\n");
347                     break;
348             case ASI_CTL_TX_CLKSRC_EXT_PAL:
349                     printf ("Using external PAL reference.\n");
350                     break;
351             default:
352                     printf ("Unknown clock source.\n");
353                     break;
354             }
355     }
356
357     /* Get the packet timestamping mode */
358     if (cap & ASI_CAP_TX_TIMESTAMPS) {
359             snprintf (name, sizeof (name), fmt, num, "timestamps");
360             if (util_strtoul (name, &timestamps) < 0) {
361                     if (!quiet) {
```

```
362                             fprintf (stderr, "%s: ", argv[0]);
363                             perror ("unable to get "
364                                     "the packet timestamping mode");
365                     }
366                     close (fd);
367                     return -1;
368             }
369     } else {
370             timestamps = ASI_CTL_TSTAMP_NONE;
371     }
372     if (verbose && !quiet) {
373             switch (timestamps) {
374             case ASI_CTL_TSTAMP_NONE:
375                     break;
376             case ASI_CTL_TSTAMP_APPEND:
377                     printf ("Stripping eight bytes "
378                             "from the end of each packet.\n");
379                     break;
380             case ASI_CTL_TSTAMP_PREPEND:
381                     printf ("Releasing packets according to "
382                             "prepended timestamps.\n");
383                     break;
384             default:
385                     printf ("Unknown timestamping mode.\n");
386                     break;
387             }
388     }
389
390     switch (transport) {
391     default:
392     case ASI_CTL_TRANSPORT_DVB_ASI:
393             /* Set the stuffing parameters */
394             if (verbose && !quiet) {
395                     printf ("Setting %i K28.5 character(s) between bytes
                                ,\n",
396                             stuffing.ib);
397                     printf ("    and %i + 2 K28.5 characters between
                                packets.\n",
398                             stuffing.ip);
399                     if (cap & ASI_CAP_TX_FINETUNING) {
400                             printf ("Adding a K28.5 character to "
401                                     "%i / %i packets.\n",
402                                     stuffing.big_ip,
403                                     stuffing.normal_ip + stuffing.big_ip
                                            );
404                             if (cap & ASI_CAP_TX_INTERLEAVING) {
405                                     if (stuffing.il_normal && stuffing.
                                            il_big) {
406                                             int normal_mult, big_mult,
                                                    mult;
407
408                                             normal_mult = stuffing.
                                                    normal_ip /
409                                                     stuffing.il_normal;
410                                             big_mult = stuffing.big_ip /
411                                                     stuffing.il_big;
412                                             mult = (normal_mult >
                                                    big_mult) ?
413                                                     big_mult :
                                                        normal_mult;
414                                             printf ("Interleaving %i x %
                                                    i = %i "
```

```
415                                                     "K28.5 characters "
416                                                     "over %i x %i = %i
                                                            packets.\n",
417                                                     mult, stuffing.
                                                            il_big,
418                                                     mult * stuffing.
                                                            il_big,
419                                                     mult, stuffing.
                                                            il_normal +
420                                                     stuffing.il_big,
421                                                     mult * (stuffing.
                                                            il_normal +
422                                                     stuffing.il_big));
423                                     }
424                             } else {
425                                     printf ("Interleaved bitrate
                                                    finetuning "
426                                             "not supported.\n");
427                             }
428                     } else {
429                             printf ("Bitrate finetuning not supported.\n
                                    ");
430                     }
431             }
432             if (ioctl (fd, ASI_IOC_TXSETSTUFFING, &stuffing) < 0) {
433                     if (!quiet) {
434                             fprintf (stderr, "%s: ", argv[0]);
435                             perror ("unable to set the stuffing
                                    parameters");
436                     }
437                     close (fd);
438                     return -1;
439             }
440             /* Calculate the target interface bitrate */
441             if ((cap & ASI_CAP_TX_FINETUNING) &&
442                     (stuffing.normal_ip != 0) && (stuffing.big_ip != 0))
                             {
443                     bitrate = 270000000 * 0.8 * packetsize /
444                             (packetsize + (packetsize - 1) * stuffing.ib
                                    +
445                              stuffing.ip + (double)stuffing.big_ip /
446                              (stuffing.normal_ip + stuffing.big_ip) + 2)
                                     ;
447             } else {
448                     bitrate = 270000000 * 0.8 * (double)packetsize /
449                             (packetsize + (packetsize - 1) * stuffing.ib
                                    +
450                              stuffing.ip + 2);
451             }
452             break;
453         case ASI_CTL_TRANSPORT_SMPTE_310M:
454             bitrate = 19392658.46;
455             break;
456         }
457         if (verbose && !quiet) {
458             printf ("Target interface bitrate = %.0f bps.\n",
459                     bitrate);
460         }
461         bytes_per_sec = bitrate / 8;
462
463         /* Allocate some memory */
464         if (bufsize < BUFSIZ) {
```

```
465                 bufsize = BUFSIZ;
466         }
467         if (verbose && !quiet) {
468                 printf ("Allocating␣%lu␣bytes␣of␣memory.\n", bufsize);
469         }
470         if ((data = (unsigned char *)malloc (bufsize)) == NULL) {
471                 if (!quiet) {
472                         fprintf (stderr, "%s:␣unable␣to␣allocate␣memory\n",
473                                 argv[0]);
474                 }
475                 close (fd);
476                 return -1;
477         }
478
479         /* Repeatedly send the data and estimate the throughput */
480         lasttime = 0;
481         if (!quiet) {
482                 if (verbose) {
483                         printf ("Transmitting␣from␣standard␣input...\n");
484                 }
485                 if (gettimeofday (&tv, NULL) < 0) {
486                         fprintf (stderr, "%s:␣", argv[0]);
487                         perror ("unable␣to␣get␣time");
488                         free (data);
489                         close (fd);
490                         return -1;
491                 }
492                 lasttime = tv.tv_sec + (double)tv.tv_usec / 1000000;
493         }
494
495         return 0;
496
497
498 USAGE:
499         fprintf (stderr, "Try␣'%s␣-h'␣for␣more␣information.\n", argv[0]);
500         return -1;
501
502 }
```

B.2 Scripts ανάλυσης

B.2.1 bash

B′.2.1.1 udp_ipv4

```
1  #!/bin/bash
2
3  CUR_PATH=`pwd`;
4  DATA_PATH="$CUR_PATH/data";
5  PERL_PATH="$CUR_PATH/perl";
6
7  #echo $DATA_PATH;
8  #ls $DATA_PATH;
9
10 if [ -d $DATA_PATH ]
11 then
12         cd $DATA_PATH;
```

```
13
14          echo -e "\n\n________ Starting Analysis! _________________________\n
                ";
15
16          for i in $( ls );
17          do
18                  echo -e "\n\t___ Entering directory: $i ___";
19                  cd $i;
20                  if [ -f rx ] && [ -f tx ]
21                  then
22                  echo ok;
23                          echo -e "\n\t\t___ Converting files with tcpdump. 
                                ___\n";
24                          tcpdump -tt -vv -n -r tx > udp_sender.txt
25                          tcpdump -tt -vv -n -r rx > udp_receiver.txt
26                          #echo -e "|n|t|t___ Checking for duplicate packets.
                                ___|n";
27                          #perl $PERL_PATH/ipv4_replicid.pl;
28                          echo -e "\n\t\t___ Running ipv4_createendfiles.pl 
                                ___\n";
29                          perl $PERL_PATH/ipv4_createendfiles.pl;
30                          echo -e "\n\t\t___ Calculating losses. ___\n";
31                          perl $PERL_PATH/ipv_all_losses.pl;
32                          echo -e "\n\t\t___ Calculating data rate ___\n";
33                          perl $PERL_PATH/ipv_all_sender_receiver_rate.pl;
34                          echo -e "\n\t\t___ Alligning packets to calculate 
                                jitter ___\n";
35                          perl $PERL_PATH/ipv_all_align_for_delay_jitt.pl;
36                          echo -e "\n\t\t___ Creating time files. ___\n";
37                          perl $PERL_PATH/ipv_all_timestamp.pl;
38                          echo -e "\n\t\t___ Calculating smoothed jitter. ___\
                                n";
39                          perl $PERL_PATH/ipv_all_inter_arrival_jitter.pl;
40                          echo -e "\n\t\t___ Calculating one-way delay. ___\n"
                                ;
41                          perl $PERL_PATH/ipv_all_one_way_delay.pl;
42                          echo -e "\n\t\t___ Calculating jitter. ___\n";
43                          perl $PERL_PATH/ipv_all_jitter.pl;
44                          echo -e "\n\t\t___ Creating losses plot. ___";
45                          gnuplot $PERL_PATH/losses.gpl;
46                          echo -e "\n\t\t___ Creating one-way delay plot. ___"
                                ;
47                          gnuplot $PERL_PATH/one_way.gpl;
48                          echo -e "\n\t\t___ Creating jitter plot. ___";
49                          gnuplot $PERL_PATH/jitter.gpl;
50                          echo -e "\n\t\t___ Creating smooth jitter plot. ___"
                                ;
51                          gnuplot $PERL_PATH/jitter_smooth.gpl;
52                          echo -e "\n\t\t___ Calculating averages. ___";
53                          $PERL_PATH/calc_mean >report;
54                  else
55                          echo "WARNING: Cannot locate 'rx' and/or 'tx' files 
                                in directory $i!";
56                  fi
57                  echo -e "\n\t___ Leaving directory: $i ___\n";
58                  cd ..;
59          done
60
61          echo -e "\n\n________ Analysis finished! _________________________\n
                ";
62  else
63          echo "ERROR: Cannot find 'data' directory!";
```

```
64 fi
```

B.2.2 perl

B.2.2.1 ipv4_createendfiles.pl

```
1  # This program is free software; you can redistribute it and/or modify
2  # it under the terms of the GNU General Public License as published by
3  # the Free Software Foundation; either version 2 of the License, or
4  # (at your option) any later version.
5  #
6  # This program is distributed in the hope that it will be useful,
7  # but WITHOUT ANY WARRANTY; without even the implied warranty of
8  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9  # GNU General Public License for more details.
10 #
11 # You should have received a copy of the GNU General Public License
12 # along with this program; if not, write to the Free Software
13 # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14
15 #!/usr/bin/perl -w
16
17 open(INPUTFILE, "<udp_sender.txt") || die ("cannot␣open␣udp_sender␣file␣1\n"
       );
18
19 unless (open (OUTFILE, ">final.tx"))
20 {
21
22         die ("cannot␣open␣output␣file␣outfile\n");
23
24 }
25
26
27 $start = time;
28
29 my (@rec_lines) = <INPUTFILE>;
30
31 foreach $rec_line (@rec_lines)
32 {
33         chomp($rec_line);
34    @line=split(/[\t +]/,$rec_line);
35
36         for ($k=0;$k<@line;$k++)
37  {
38          $id=0;
39          if ($line[$k] eq "cid")
40                 {
41                         $id=$line[$k+1];
42                         last;
43                 }
44
45  }
46
47 for ($k=0;$k<@line;$k++)
48  {
49
50          if ($line[$k] eq "seq")
51                 {
52
```

```
53                          $id=$id.($line[$k+1]);
54                          last;
55                  }
56
57  }
58  for ($k=0;$k<@line;$k++)
59  {
60
61          if ($line[$k] eq "ser")
62                  {
63
64                          $id=$id.($line[$k+1]);
65                          last;
66                  }
67
68  }
69  for ($k=0;$k<@line;$k++)
70  {
71
72          if ($line[$k] eq "id")
73                  {
74
75                          $id=$id.($line[$k+1]);
76                          last;
77                  }
78
79  }
80
81
82  for ($k=0;$k<@line;$k++)
83   {
84
85          if ($line[$k] eq "length:")
86                  {
87
88                          $size=$line[$k+1];
89                          last;
90                  }
91
92   }
93     $time=$line[0];
94     #$id=substr($id,0,length($id)-1);
95     $size=substr($size,0,length($size)-1);
96  #print ("$id $size \n");
97   print OUTFILE ("$id\t$id\t$time\t$size\n");
98
99  }
100
101 close(INPUTFILE);
102 close(OUTPUTFILE);
103
104 #second file
105
106 open(INPUTFILE, "<udp_receiver.txt") || die ("cannot␣open␣udp_receiver␣file␣
        1\n");
107
108 unless (open (OUTFILE, ">final.rx"))
109 {
110
111         die ("cannot␣open␣output␣file␣outfile\n");
112
113 }
```

```
114
115
116 my (@rec_lines) = <INPUTFILE>;
117
118 foreach $rec_line (@rec_lines)
119 {
120         chomp($rec_line);
121    @line=split(/[\t +]/,$rec_line);
122         for ($k=0;$k<@line;$k++)
123  {
124          $id=0;
125          if ($line[$k] eq "cid")
126                 {
127                         $id=$line[$k+1];
128                         last;
129                 }
130
131  }
132
133 for ($k=0;$k<@line;$k++)
134  {
135
136          if ($line[$k] eq "seq")
137                 {
138
139                         $id=$id.($line[$k+1]);
140                         last;
141                 }
142
143  }
144  for ($k=0;$k<@line;$k++)
145  {
146
147          if ($line[$k] eq "ser")
148                 {
149
150                         $id=$id.($line[$k+1]);
151                         last;
152                 }
153
154  }
155  for ($k=0;$k<@line;$k++)
156  {
157
158          if ($line[$k] eq "id")
159                 {
160
161                         $id=$id.($line[$k+1]);
162                         last;
163                 }
164
165  }
166
167 for ($k=0;$k<@line;$k++)
168  {
169
170          if ($line[$k] eq "length:")
171                 {
172
173                         $size=$line[$k+1];
174                         last;
175                 }
```

```
176
177  }
178    $time=$line[0];
179    ##$id=substr($id,0,length($id)-1);
180    $size=substr($size,0,length($size)-1);
181
182  print OUTFILE ("$id\t$id\t$time\t$size\n");
183
184 }
185
186
187 $elapsed_sec = time - $start;
188
189 my $second = $elapsed_sec%60;
190 my $minute = ($elapsed_sec/60)%60;
191 my $hour = ($elapsed_sec/(60*60))%24;
192   print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
193
194 close(INPUTFILE);
195 close(OUTPUTFILE);
```

B.2.2.2 ipv_all_losses.pl

```
1  # This program is free software; you can redistribute it and/or modify
2  # it under the terms of the GNU General Public License as published by
3  # the Free Software Foundation; either version 2 of the License, or
4  # (at your option) any later version.
5  #
6  # This program is distributed in the hope that it will be useful,
7  # but WITHOUT ANY WARRANTY; without even the implied warranty of
8  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9  # GNU General Public License for more details.
10 #
11 # You should have received a copy of the GNU General Public License
12 # along with this program; if not, write to the Free Software
13 # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14 #Author irons
15 #Mail irons@pasiphae.teiher.gr
16 #This file calculates the losses in a udp transmission.
17
18 #!/usr/bin/perl -w
19
20 $start = time;
21
22 & calc_loss;
23
24 $elapsed_sec = time - $start;
25
26 my $second = $elapsed_sec%60;
27 my $minute = ($elapsed_sec/60)%60;
28 my $hour = ($elapsed_sec/(60*60))%24;
29   print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
30
31
32 sub calc_loss
33 # Simple loss calculation
34 {
35         my ($sender_packets);
36         my ($receiver_packets);
37         my ($loss_rate);
```

```
38
39          $sender_packets = 0;
40          $receiver_packets = 0;
41          $loss_rate = 0;
42
43  # Sender file
44          open (SENDER, "<final.tx") || die ("cannot␣open␣input␣file␣1\n");
45
46
47
48          while (<SENDER>)
49          {
50
51                  $sender_packets++;
52
53
54          }
55
56          close (SENDER);
57
58  # Receiver file
59          open (RECEIVER, "<final.rx") || die ("cannot␣open␣input␣file␣2\n");
60
61
62          while (<RECEIVER>)
63          {
64
65                  $receiver_packets++;
66
67
68          }
69
70
71          close (RECEIVER);
72
73  #calculation
74          $loss_rate =
75                  (($sender_packets -
76                    $receiver_packets) / $sender_packets) * 100;
77
78          print "sender␣packets␣$sender_packets,␣receiver␣packets␣
                $receiver_packets,␣losses␣$loss_rate%\n";
79
80         if (($sender_packets - $receiver_packets)!=0)
81  {
82    & lossvstime;
83  }
84
85  }
86
87  sub lossvstime
88  {
89
90  unless (open (OUTFILE, ">pack_num_seq_loss_vs_time"))
91  {
92
93          die ("cannot␣open␣output␣file␣outfile\n");
94
95  }
96
97
98
```

```
99          open (SENDER, "<final.tx");
100 open (RECEIVER, "<final.rx");
101 my (@sen_lines) = <SENDER>;
102 my ($sen_line);
103 my (@rec_lines) = <RECEIVER>;
104 my ($rec_line);
105 my ($temp);
106 my ($temp1);
107 my ($lock);
108 my ($lock_ref_time);
109 my ($send_time);
110 my ($start_ref_time);
111 my ($packet_counter);
112 close (SENDER);
113 close (RECEIVER);
114 $size=@rec_lines;
115 $size1=@sen_lines;
116 $packet_counter=0;
117 $lock_ref_time=0;
118
119 foreach $sen_line (@sen_lines)
120          {
121 $lock=0;
122 $packet_counter++;
123          chomp($sen_line);
124                  @line_sender=split(/[\t +]/,$sen_line);
125                  $temp1=$line_sender[0].$line_sender[1];
126
127           if ($lock_ref_time==0)
128 {
129  $start_ref_time=$line_sender[2];
130 $lock_ref_time=1;
131 #print ("\n$start_ref_time\n");
132
133 }
134 foreach $rec_line (@rec_lines)
135 {
136          chomp($rec_line);
137          @line_receiver=split(/[\t +]/,$rec_line);
138          $temp=$line_receiver[0].$line_receiver[1];
139 if ($temp eq $temp1)
140                  {
141
142                          #unless (open (OUTFILE, ">>aligned_sender"))
143                          #{
144
145                          #       die ("cannot open output file outfile\n");
146
147                          #}
148
149
150
151                          #close (OUTFILE);
152                         $lock=1;
153                          last;
154 #print " sender $sen_line receiver $rec_line\n";
155                  }
156
157
158          }
159 if ($lock==0)
160 {
```

```
161 $send_time=$line_sender[2]-$start_ref_time;
162 print OUTFILE "$send_time\t$packet_counter␣\n";
163
164 }
165 #if ($lock==1)
166 #{
167 ##$send_time=$line_sender[2]-$start_ref_time;
168 #print OUTFILE "$send_time\t0 \n";
169
170 #}
171 }
172 close (OUTFILE);
173 & avg_lossvstime;
174 }
175 sub avg_lossvstime{
176
177 unless (open (OUTFILE, ">lossvstime"))
178 {
179
180         die ("cannot␣open␣output␣file␣outfile\n");
181
182 }
183
184
185 open (LOSSES, "<pack_num_seq_loss_vs_time");
186
187 my (@sen_lines) = <LOSSES>;
188 my ($sen_line);
189 my ($temp1);
190 my ($temp);
191 my ($lock);
192 my ($lock1);
193 my ($send_time);
194 my ($packet_counter);
195 my ($line_counter);
196
197 close (LOSSES);
198
199 $size=@rec_lines;
200 $size1=@sen_lines;
201 $packet_counter=0;
202 $lock_ref_time=0;
203 $line_counter=0;
204 $lock=0;
205
206 $lock1=0;
207 foreach $sen_line (@sen_lines)
208         {
209
210 $packet_counter++;
211         chomp($sen_line);
212                 @line_sender=split(/[\t +]/,$sen_line);
213                 $temp=$line_sender[0];
214                 if ($lock==0)
215 {
216  $temp1=$line_sender[0];
217 #print "$temp1\n";
218 @line_sender_last=split(/[\t +]/,$sen_lines[(@sen_lines) -1]);
219 #print "$line_sender_last[0]\n";
220  ##$packet_counter=0;
221 $lock=1;
222 }
```

```
223
224 if (($temp1+1.0) < ($temp))
225 {
226    if ($lock1==0)
227     {
228     $pack_count=$packet_counter-1;
229   print OUTFILE "$send_time\t$pack_count \n";
230   #print "$send_time\t$packet_counter\t$temp \n";
231   $packet_counter=1;
232 $temp1=$temp;
233 }
234 }
235 if (($temp1+1) > ($line_sender_last[0]))
236 {
237  $lock1=1;
238     if ($temp==$line_sender_last[0])
239 {
240 $send_time=$temp;
241 # print "$send_time\t$packet_counter\t$temp \n";
242   print OUTFILE "$send_time\t$packet_counter \n";
243 }
244
245
246 }
247
248 $send_time=$temp;
249 $line_counter++;
250 }
251 close (OUTFILE);
252 #print "$pack_count\t$packet_counter\t$line_counter\n";
253 }
```

B.2.2.3 ipv_all_sender_receiver_rate.pl

```
1  # This program is free software; you can redistribute it and/or modify
2  # it under the terms of the GNU General Public License as published by
3  # the Free Software Foundation; either version 2 of the License, or
4  # (at your option) any later version.
5  #
6  # This program is distributed in the hope that it will be useful,
7  # but WITHOUT ANY WARRANTY; without even the implied warranty of
8  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9  # GNU General Public License for more details.
10 #
11 # You should have received a copy of the GNU General Public License
12 # along with this program; if not, write to the Free Software
13 # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14
15 #!/usr/bin/perl -w
16
17
18 my($sum_pack_size);
19 my($lock);
20 my($start_time);
21 my($end_time);
22 my($transffer_time);
23 my($data_rate);
24 $data_rate=0;
25 $transffer_time=0;
26 $sum_pack_size=0;
```

```
27 $lock=0;
28 $start_time=0;
29 $end_time=0;
30
31 open(SENDER, "<final.tx") || die ("cannot␣open␣input␣file␣1\n");
32
33 $start = time;
34
35 while (<SENDER>)
36 {
37   my($sen_line) = $_;
38   chomp($sen_line);
39
40   @line_sender=split(/[\t +]/,$sen_line);
41   $sum_pack_size+=$line_sender[3];
42    if ($lock==0)
43 {
44   $start_time=$line_sender[2];
45   $lock=1;
46 }
47
48 #print "$papa[3]\n";
49
50
51 }
52 $end_time=$line_sender[2];
53 #print "stime $start_time etime $end_time\n";
54 $transffer_time=$end_time-$start_time;
55 $data_rate=$sum_pack_size/$transffer_time;
56 print "SENDER␣RESULTS\n";
57 print "total␣bytes␣transferred␣$sum_pack_size␣in␣$transffer_time␣sec.␣Sender
       ␣output␣data␣rate␣is␣$data_rate␣bytes/sec␣\n";
58 close(SENDER);
59
60
61 open(RECEIVER, "<final.rx") || die ("cannot␣open␣input␣file␣2\n");
62 $data_rate=0;
63 $transffer_time=0;
64 $sum_pack_size=0;
65 $lock=0;
66 $start_time=0;
67 $end_time=0;
68 while (<RECEIVER>)
69 {
70   my($sen_line) = $_;
71   chomp($sen_line);
72
73   @line_receiver=split(/[\t +]/,$sen_line);
74   $sum_pack_size+=$line_receiver[3];
75    if ($lock==0)
76 {
77   $start_time=$line_receiver[2];
78   $lock=1;
79 }
80
81 #print "$papa[3]\n";
82
83
84 }
85 $end_time=$line_receiver[2];
86 print "stime␣$start_time␣etime␣$end_time\n";
87 $transffer_time=$end_time-$start_time;
```

```
88 $data_rate=$sum_pack_size/$transffer_time;
89 print "RECEIVER RESULTS\n";
90 print "total bytes transferred $sum_pack_size in $transffer_time sec.
       Receiver input data rate is $data_rate bytes/sec \n";
91
92 close(RECEIVER);
93
94 $elapsed_sec = time - $start;
95
96 my $second = $elapsed_sec%60;
97 my $minute = ($elapsed_sec/60)%60;
98 my $hour = ($elapsed_sec/(60*60))%24;
99   print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

B.2.2.4 ipv_all_align_for_delay_jitt.pl

```
1  # This program is free software; you can redistribute it and/or modify
2  # it under the terms of the GNU General Public License as published by
3  # the Free Software Foundation; either version 2 of the License, or
4  # (at your option) any later version.
5  #
6  # This program is distributed in the hope that it will be useful,
7  # but WITHOUT ANY WARRANTY; without even the implied warranty of
8  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9  # GNU General Public License for more details.
10 #
11 # You should have received a copy of the GNU General Public License
12 # along with this program; if not, write to the Free Software
13 # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14
15 #!/usr/bin/perl -w
16
17 unless (open (OUTFILE, ">aligned_sender"))
18 {
19
20         die ("cannot open output file outfile\n");
21
22 }
23
24
25
26
27 open (SENDER, "<final.tx");
28 open (RECEIVER, "<final.rx");
29 my (@sen_lines) = <SENDER>;
30 my ($sen_line);
31 my (@rec_lines) = <RECEIVER>;
32 my ($rec_line);
33 my ($temp);
34 my ($temp1);
35
36 close (SENDER);
37 close (RECEIVER);
38 $size=@rec_lines;
39 $size1=@sen_lines;
40 print "Receiver packets $size --- Sender packets $size1\n";
41
42  $start = time;
43
44
```

```
45 foreach $rec_line (@rec_lines)
46 {
47         chomp($rec_line);
48         @line_receiver=split(/[\t +]/,$rec_line);
49         $temp=$line_receiver[0].$line_receiver[1];
50         foreach $sen_line (@sen_lines)
51         {
52         chomp($sen_line);
53                 @line_sender=split(/[\t +]/,$sen_line);
54                 $temp1=$line_sender[0].$line_sender[1];
55                 if ($temp eq $temp1)
56                 {
57
58                         #unless (open (OUTFILE, ">>aligned_sender"))
59                         #{
60
61                         #       die ("cannot open output file outfile|n");
62
63                         #}
64
65                         print OUTFILE "$sen_line\n";
66
67                         #close (OUTFILE);
68
69                         last;
70 #print " sender $sen_line receiver $rec_line|n";
71                 }
72
73         }
74
75 }
76
77 $elapsed_sec = time - $start;
78
79 my $second = $elapsed_sec%60;
80 my $minute = ($elapsed_sec/60)%60;
81 my $hour = ($elapsed_sec/(60*60))%24;
82   print "Total␣Time␣elapsed:␣$hour␣hours:$minute␣min:$second␣sec\n";
83
84 close (OUTFILE);
```

B.2.2.5 ipv_all_timestamp.pl

```
1  #!/usr/bin/perl -w
2
3  unless (open (SENDER, "<aligned_sender"))
4          {
5
6                  die ("cannot␣open␣input␣file␣outfile\n");
7
8          }
9
10
11 unless (open (OUTFILE, ">sender_timestamp"))
12                         {
13
14                                 die ("cannot␣open␣output␣file␣outfile\n");
15
16                         }
17
```

```
18
19 $start = time;
20
21 while (<SENDER>)
22 {
23   my($sen_line) = $_;
24   chomp($sen_line);
25
26   @line_sender=split(/[\t +]/,$sen_line);
27
28
29   print OUTFILE "$line_sender[2]\n";
30
31
32
33
34
35 #print "$papa[3]\n";
36
37
38 }
39
40 close(OUTFILE);
41 close(SENDER);
42
43 unless (open (RECEIVER, "<final.rx"))
44         {
45
46                 die ("cannot␣open␣input␣file␣outfile\n");
47
48         }
49
50 unless (open (OUTFILE, ">receiver_timestamp"))
51                         {
52
53                                 die ("cannot␣open␣output␣file␣outfile\n");
54
55                         }
56
57 while (<RECEIVER>)
58 {
59   my($sen_line) = $_;
60   chomp($sen_line);
61
62   @line_receiver=split(/[\t +]/,$sen_line);
63
64
65   print OUTFILE "$line_receiver[2]\n";
66
67
68
69 #print "$papa[3]\n";
70
71
72 }
73
74
75 close(OUTFILE);
76
77 close(RECEIVER);
78
79 $elapsed_sec = time - $start;
```

```
80
81 my $second = $elapsed_sec%60;
82 my $minute = ($elapsed_sec/60)%60;
83 my $hour = ($elapsed_sec/(60*60))%24;
84   print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

B.2.2.6 ipv_all_inter_arrival_jitter.pl

```
1  # This program is free software; you can redistribute it and/or modify
2  # it under the terms of the GNU General Public License as published by
3  # the Free Software Foundation; either version 2 of the License, or
4  # (at your option) any later version.
5  #
6  # This program is distributed in the hope that it will be useful,
7  # but WITHOUT ANY WARRANTY; without even the implied warranty of
8  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9  # GNU General Public License for more details.
10 #
11 # You should have received a copy of the GNU General Public License
12 # along with this program; if not, write to the Free Software
13 # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14
15 #!/usr/bin/perl -w
16
17 unless (open (SENDER, "<sender_timestamp"))
18         {
19
20                 die ("cannot open input file outfile\n");
21
22         }
23 unless (open (RECEIVER, "<receiver_timestamp"))
24         {
25
26                 die ("cannot open input file outfile\n");
27
28         }
29
30 my (@sen_lines) = <SENDER>;
31 my ($sen_line);
32 my (@rec_lines) = <RECEIVER>;
33 my ($rec_line);
34 my($transit);
35 my($delta_transit);
36 my($last_transit);
37 my($jitter);
38 my($counter);
39 $counter=0;
40 $transit=0;
41 $delta_transit=0;
42 $last_transit=0;
43 $jitter=0;
44
45
46 close(SENDER);
47 close(RECEIVER);
48
49 unless (open (OUTFILE, ">final_jitter"))
50                         {
51
52                                 die ("cannot open output file outfile\n");
```

```
53
54                          }
55
56  unless (open (OUTFILE1, ">final_pack2packdelay"))
57                          {
58
59                                  die ("cannot␣open␣output␣file␣outfile\n");
60
61                          }
62
63  $start = time;
64
65  foreach $sen_line (@sen_lines)
66  {
67     chomp($sen_line);
68
69     $transit=$rec_lines[$counter]-$sen_line;
70  if ($last_transit!=0)
71  {
72  $delta_transit=$transit-$last_transit;
73  if ( $delta_transit < 0 ) {
74                      $delta_transit = -$delta_transit;
75                  }
76
77
78  $jitter+=($delta_transit-$jitter)/16.0;
79
80
81  }
82  $last_transit=$transit;
83     $result=$jitter*1000;
84  $timerec=$rec_lines[$counter];
85  chomp($timerec);
86  print OUTFILE "$timerec␣$result\n";
87  $pack_delay=$delta_transit*1000;
88  print OUTFILE1 "$timerec␣$pack_delay\n";
89     #print "$sen_line $rec_lines[$counter] $result\n";
90     $counter++;
91
92  }
93
94
95  close(OUTFILE);
96  close(OUTFILE1);
97
98  unless (open (INFILE, "<final_jitter"))
99          {
100
101                 die ("cannot␣open␣input␣file␣outfile\n");
102
103         }
104 $min=100000;
105 $max=0;
106 $counter=0;
107 $result=0;
108 $lock=0;
109 while (<INFILE>)
110 {
111
112 my($sen_line) = $_;
113   chomp($sen_line);
114 @values=split(/[\t +]/,$sen_line);
```

```
115 $value=$values[1];
116 $result+=$value;
117  if ($counter==1)
118    {
119 $min=$value;
120
121 }
122
123 if ($value>$max)
124 {
125   $max=$value;
126
127 }
128 if (($value<$min))
129 {
130   $min=$value;
131
132 }
133 $counter++;
134
135
136 }
137
138
139 close (INFILE);
140 $result=$result/$counter;
141
142 print "aver␣jitter␣is␣$result␣max␣is␣$max␣min␣$min\n";
143
144 unless (open (INFILE, "<final_pack2packdelay"))
145         {
146
147                 die ("cannot␣open␣input␣file␣outfile\n");
148
149         }
150 $min=100000;
151 $max=0;
152 $counter=0;
153 $result=0;
154 $lock=0;
155 while (<INFILE>)
156 {
157
158 my($sen_line) = $_;
159   chomp($sen_line);
160
161 @values=split(/[\t +]/,$sen_line);
162 $value=$values[1];
163 $result+=$value;
164  if ($counter==1)
165    {
166 $min=$value;
167
168 }
169
170 if ($value>$max)
171 {
172   $max=$value;
173
174 }
175 if (($value<$min))
176 {
```

```
177   $min=$value;
178
179 }
180 $counter++;
181
182
183 }
184
185
186 close (INFILE);
187 $result=$result/($counter);
188
189 print "aver␣pack2packdelay␣is␣$result␣max␣is␣$max␣min␣$min\n";
190
191
192
193 unless (open (INFILE, "<final_pack2packdelay"))
194         {
195
196                 die ("cannot␣open␣input␣file␣outfile\n");
197
198         }
199
200 unless (open (OUTFILE, ">timed_final_pack2packdelay"))
201         {
202
203                 die ("cannot␣open␣input␣file␣outfile\n");
204
205         }
206
207 my (@times) = <INFILE>;
208 close(INFILE);
209 $time=0;
210 $lock=0;
211 for ($i=0;$i<@times-1;$i++)
212 {
213
214  if ($lock==0)
215 {
216   @valuesplits=split(/[\t +]/,$times[$i]);
217   $valuesplit=$valuesplits[1];
218 chomp($valuesplit);
219   print OUTFILE "$time␣$valuesplit\n";
220 $lock=1;
221 }
222    #chomp($sen_line);
223  @timesplit=split(/[\t +]/,$times[$i+1]);
224  $temp_time1=$timesplit[0];
225 @timesplit=split(/[\t +]/,$times[$i]);
226 $temp_time2=$timesplit[0];
227 $time=($temp_time1-$temp_time2)+$time;
228  ##$time=$timesplit[0];
229  @valuesplits=split(/[\t +]/,$times[$i+1]);
230   $valuesplit=$valuesplits[1];
231   chomp($valuesplit);
232   print OUTFILE "$time␣$valuesplit\n";
233  #print "$time\n";
234
235 }
236
237 close(OUTFILE);
238
```

```
239
240 unless (open (INFILE, "<final_jitter"))
241         {
242
243                 die ("cannot␣open␣input␣file␣outfile\n");
244
245         }
246
247 unless (open (OUTFILE, ">timed_final_jitter"))
248         {
249
250                 die ("cannot␣open␣input␣file␣outfile\n");
251
252         }
253
254 my (@times) = <INFILE>;
255 close(INFILE);
256 $time=0;
257 $lock=0;
258 for ($i=0;$i<@times-1;$i++)
259 {
260
261  if ($lock==0)
262 {
263   @valuesplits=split(/[\t +]/,$times[$i]);
264   $valuesplit=$valuesplits[1];
265 chomp($valuesplit);
266   print OUTFILE "$time␣$valuesplit\n";
267 $lock=1;
268 }
269   #chomp($sen_line);
270  @timesplit=split(/[\t +]/,$times[$i+1]);
271  $temp_time1=$timesplit[0];
272 @timesplit=split(/[\t +]/,$times[$i]);
273 $temp_time2=$timesplit[0];
274 $time=($temp_time1-$temp_time2)+$time;
275  #$time=$timesplit[0];
276  @valuesplits=split(/[\t +]/,$times[$i+1]);
277   $valuesplit=$valuesplits[1];
278 chomp($valuesplit);
279   print OUTFILE "$time␣$valuesplit\n";
280  #print "$time\n";
281
282 }
283
284 close(OUTFILE);
285
286 $elapsed_sec = time - $start;
287
288 my $second = $elapsed_sec%60;
289 my $minute = ($elapsed_sec/60)%60;
290 my $hour = ($elapsed_sec/(60*60))%24;
291   print "Total␣Time␣elapsed:␣$hour␣hours:$minute␣min:$second␣sec\n";
```

B.2.2.7 ipv_all_one_way_delay.pl

```
1 # This program is free software; you can redistribute it and/or modify
2 # it under the terms of the GNU General Public License as published by
3 # the Free Software Foundation; either version 2 of the License, or
4 # (at your option) any later version.
```

```
5  #
6  # This program is distributed in the hope that it will be useful,
7  # but WITHOUT ANY WARRANTY; without even the implied warranty of
8  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9  # GNU General Public License for more details.
10 #
11 # You should have received a copy of the GNU General Public License
12 # along with this program; if not, write to the Free Software
13 # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14
15 #!/usr/bin/perl -w
16
17 unless (open (SENDER, "<sender_timestamp"))
18         {
19
20                 die ("cannot␣open␣input␣file␣outfile\n");
21
22         }
23 unless (open (RECEIVER, "<receiver_timestamp"))
24         {
25
26                 die ("cannot␣open␣input␣file␣outfile\n");
27
28         }
29
30 my (@sen_lines) = <SENDER>;
31 my (@rec_lines) = <RECEIVER>;
32 my($delay);
33 my($counter);
34 my($avg_delay);
35 my($sample_time);
36 my($min);
37 my($max);
38 $counter=0;
39 $delay=0;
40 $avg_delay=0;
41 $sample_time=0;
42 $max=-1;
43 $min=1000000;
44 close(SENDER);
45 close(RECEIVER);
46
47 unless (open (OUTFILE, ">one_way_delayvstime"))
48                         {
49
50                                 die ("cannot␣open␣output␣file␣jittervstime\n
                                        ");
51
52                         }
53
54
55 $start = time;
56
57
58 # One way delay formula is Di=abs(R_i-S_i)
59 #Avg One way Delay is Sum(Di)/n
60
61 for ($i=0;$i<@sen_lines;$i++)
62 {
63
64    #print("Sender line $sen_lines[$i]\n");
65  #print("Receiver line $rec_lines[$i]\n");
```

```
66    $delay=abs(($rec_lines[$i])-($sen_lines[$i]))*1000;
67   $avg_delay+=$delay;
68   if ($min>$delay)
69  {
70          $min=$delay;
71  }
72  if ($max<$delay)
73  {
74          $max=$delay;
75  }
76  $sample_time=$rec_lines[$i]-$rec_lines[0];
77  #print ("$sample_time $jitter\n");
78  print OUTFILE "$sample_time $delay\n";
79  $counter++;
80  }
81  $avg_delay=($avg_delay/$counter);
82  print ("Average One way Delay is $avg_delay ms. Max One way Delay is $max ms
        . Min One way Delay is $min ms\n");
83
84  close(OUTFILE);
85
86  $elapsed_sec = time - $start;
87
88  my $second = $elapsed_sec%60;
89  my $minute = ($elapsed_sec/60)%60;
90  my $hour = ($elapsed_sec/(60*60))%24;
91    print "Total Time elapsed: $hour hours:$minute min:$second sec\n";
```

B.2.2.8 ipv_all_jitter.pl

```
1   # This program is free software; you can redistribute it and/or modify
2   # it under the terms of the GNU General Public License as published by
3   # the Free Software Foundation; either version 2 of the License, or
4   # (at your option) any later version.
5   #
6   # This program is distributed in the hope that it will be useful,
7   # but WITHOUT ANY WARRANTY; without even the implied warranty of
8   # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
9   # GNU General Public License for more details.
10  #
11  # You should have received a copy of the GNU General Public License
12  # along with this program; if not, write to the Free Software
13  # Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
14
15  #!/usr/bin/perl -w
16
17  unless (open (SENDER, "<sender_timestamp"))
18          {
19
20                  die ("cannot open input file outfile\n");
21
22          }
23  unless (open (RECEIVER, "<receiver_timestamp"))
24          {
25
26                  die ("cannot open input file outfile\n");
27
28          }
29
30  my (@sen_lines) = <SENDER>;
```

```
31 my (@rec_lines) = <RECEIVER>;
32 my($jitter);
33 my($counter);
34 my($avg_jitter);
35 my($sample_time);
36 my($min);
37 my($max);
38 $counter=0;
39 $jitter=0;
40 $avg_jitter=0;
41 $sample_time=0;
42 $max=-1;
43 $min=1000000;
44 close(SENDER);
45 close(RECEIVER);
46
47 unless (open (OUTFILE, ">jittervstime"))
48                          {
49
50                                  die ("cannot␣open␣output␣file␣jittervstime\n
                                        ");
51
52                          }
53
54
55 $start = time;
56
57
58 # Jitter formula is Di=abs(R_(i)-R_(i-1))-(S_(i)-S_(i-1))
59 #Avg jitter is Sum(Di)/n
60
61 for ($i=0;$i<@sen_lines-1;$i++)
62 {
63
64    #print("Sender line $sen_lines[$i]\n");
65  #print("Receiver line $rec_lines[$i]\n");
66   $jitter=abs(($rec_lines[$i+1]-$rec_lines[$i])-($sen_lines[$i+1]-$sen_lines
         [$i]))*1000;
67  $avg_jitter+=$jitter;
68  if ($min>$jitter)
69 {
70         $min=$jitter;
71 }
72 if ($max<$jitter)
73 {
74         $max=$jitter;
75 }
76 $sample_time=$rec_lines[$i+1]-$rec_lines[0];
77 #print ("$sample_time $jitter\n");
78 print OUTFILE "$sample_time␣$jitter\n";
79 $counter++;
80 }
81 $avg_jitter=($avg_jitter/$counter);
82 print ("Average␣Jitter␣is␣$avg_jitter␣ms.␣Max␣jitter␣is␣$max␣ms.␣Min␣jitter␣
       is␣$min␣ms\n");
83
84 close(OUTFILE);
85
86 $elapsed_sec = time - $start;
87
88 my $second = $elapsed_sec%60;
89 my $minute = ($elapsed_sec/60)%60;
```

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

B.2.3 gnuplot

B.2.3.1 losses.gpl

```
1  set title "Losses␣vs␣Time"
2  set xlabel "time␣(sec)"
3  set ylabel "Lost␣Packets"
4  set format x "%.0f"
5  set format y "%.0f"
6  set yrange [0:*]
7  set xdata time
8  set nokey
9
10
11 set terminal postscript eps 22
12 set size 1,1; set term post landscape color "Times-Roman" 14
13 set output "losses.eps"
14 plot "lossvstime" using ($1-946684800.0):2 with points pointtype 5 pointsize
       1;
15
16 #load "a2b_tput.labels";
17 #plot "lossvstime" using ($1-946684800.0):2 with points pointtype 5
       pointsize 1;
18 #set terminal png color ;
19
20 #set output "losses.png"
21 #replot
22 #pause -1;
```

B.2.3.2 one_way.gpl

```
1
2  set xlabel "Time␣(sec)"
3  set ylabel "One␣Way␣Delay␣(ms)"
4  set format x "%.0f"
5  set format y "%.0f"
6  set xdata
7  set ydata
8  set grid
9  set mxtics 6
10 set mytics 6
11 set autoscale
12 set nokey
13 set origin 0,0
14
15 set terminal postscript eps 22
16 set size 1,1; set term post landscape color "Times-Roman" 14
17 set output "one_way.eps"
18 plot "one_way_delayvstime" with dots  lt -1 ;
19
20 #set xrange [0:180]
21
22 #set size 0.75, 0.75
23 #set size ratio 0 0.5,0.48
24 #plot "one_way_delayvstime" with dots  lt -1 ;
```

```
#set terminal png x255255255;
#set terminal png tiny
#set output "one_way.png"
#replot
#pause -1;
```

B.2.3.3 jitter.gpl

```
set xlabel "Time (sec)"
set ylabel "Inter-arrival Jitter (ms)"
set format x "%.0f"
set format y "%.0f"

set xdata
set ydata
set grid
set mxtics 6
set mytics 6
set autoscale
set nokey
set origin 0,0

set terminal postscript eps 22
set size 1,1; set term post landscape color "Times-Roman" 14
set output "jitter.eps"
plot "jittervstime" with line lt -1 lw -1 ;
```

B.2.3.4 jitter_smooth.gpl

```
set xlabel "Time (sec)"
set ylabel "Inter-arrival Jitter (ms)"
set format x "%.0f"
set format y "%.0f"

set xdata
set ydata
set grid
set mxtics 6
set mytics 6
set autoscale
set nokey
#set xrange [0:180]
#set yrange [0:5]

set origin 0,0

set terminal postscript eps 22
set size 1,1; set term post landscape color "Times-Roman" 14
set output "smooth_jitter.eps"
plot "timed_final_jitter" with line lt -1 lw -1 ;

#plot "timed_final_pack2packdelay" with dots  lt -1 ;
#plot "timed_final_jitter" with line lt -1 lw -1 ;

#set terminal png x255255255 ;
#set output "smooth_jitter.png"
#show xtics
#show ytics
#replot
```

```
31 #pause -1;
```