ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

"Μελέτη, σχεδιασμός, υλοποίηση και αξιολόγηση ενός συστήματος παρακολούθησης και ελέγχου της δικτυακής απόδοσης μιας πλατφόρμας διαδραστικής επίγειας ψηφιακής ευρυεκπομπής. "

Φοιτητής: Φυτράκης Εμμανουήλ (ΑΜ: 1893)
Εισηγητής: Πάλλης Ευάγγελος

9/ 10 / 2013

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ευάγγελο Πάλλη, για την καθοδήγηση και τη δυνατότητα που μου προσέφερε για τη διεκπεραίωση της πτυχιακής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τον κ. Ανάργυρο Σιδέρη για -την κομβικής σημασίας- βοήθεια που μου προσέφερε καθώς και για την εμπιστοσύνη και κατανόηση που επέδειξε.

Ευχαριστώ θερμά τον Γιώργο Κοντογιωργάκη και το Γιώργο Αλεξίου καθώς και τα υπόλοιπα μέλη του εργατηρίου “ΠΑΣΙΦΑΗ” για την άψογη συνεργασία και τη βοήθειά τους.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την ανεκτίμητη συμπαράσταση και στήριξη όλα αυτά τα χρόνια των σπουδών μου.

Περιεχόμενα

Λίστα σχημάτων

Λίστα πινάκων

Abstract

This thesis uses a semi-Distributed architectural approach to provide real time monitoring services to Interactive DVB-T (IDVB-T) networks. This kind of architecture was chosen because, as opposed to the Centralized approach, it divides the workload between a central and several peripheral nodes while at the same time avoids the complexity and communication overhead of a fully Distributed architecture. The implemented monitoring system uses a passive approach to measure the network performance--because in this way the network traffic is not increased--and supports both online and offline analysis. As far as the monitoring modules concerned, (the Analysis and Presentation modules) reside at a central node of the DVB-T platform while the traffic filtering and capturing modules reside at the intermediate distribution nodes (CMNs) of the IDVB-T network. The monitoring modules were developed using the programming language JAVA while mark up language XML was used for structuring the captured traffic's info. A series of experiments conducted on a real IDVB-T network, proved the efficiency of the implemented system, even in cases of high and unpredicted network load. For the future we aim towards integrating Quality of experience (QoE) assessment capabilities to our monitoring system.

Περίληψη

Η συγκεκριμένη διατριβή παρουσιάζει μια ημικατανεμημένη αρχιτεκτονική προσέγγιση για τον σχεδιασμό και την υλοποίηση ενός συστήματος παροχής, σε πραγματικό χρόνο, υπηρεσιών παρακολούθησης και ελέγχου της δικτυακής απόδοσης διαδραστικών DVB-T (IDVB-T) δικτύων. Αυτό το είδος της αρχιτεκτονικής επιλέχθηκε επειδή, σε αντίθεση με την κεντροποιημένη προσέγγιση, διαμοιράζει το φόρτο εργασίας μεταξύ των κεντρικών και των περιφερειακών κόμβων, ενώ ταυτόχρονα αποφεύγει την πολυπλοκότητα και την επικοινωνιακή επιβάρυνση μιας πλήρως κατανεμημένης αρχιτεκτονικής. Το προτεινόμενο σύστημα παρακολούθησης χρησιμοποιεί την παθητική προσέγγιση για τη μέτρηση της απόδοσης του δικτύου - γιατί με τον τρόπο αυτό η κίνηση στο δίκτυο δεν αυξάνεται - και υποστηρίζει τόσο online όσο και offline ανάλυση. Όσον αφορά τις λειτουργικές μονάδες του συστήματος παρακολούθησης, η μονάδα ανάλυσης και παρουσίασης των αποτελεσμάτων εδρεύει σε ένα κεντρικό κόμβο της DVB-T πλατφόρμας, ενώ η μονάδα σύλληψης και φιλτραρίσματος της δικτυακής κίνησης εδρεύει σε όλους τους ενδιάμεσους κόμβους διανομής (CMNs) του IDVB-T δικτύου. Η σύστημα μας αναπτύχθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού JAVA, ενώ για τη δόμηση των πληροφοριών που συλλέγονται από το IDVB-T δίκτυο χρησιμοποιείται η γλώσσα XML. Μια σειρά πειραμάτων που πραγματοποιήθηκαν σε ένα πραγματικό IDVB-T δίκτυο, απέδειξαν την αποτελεσματικότητα του υλοποιημένου συστήματος, ακόμη και σε περιπτώσεις υψηλού και απρόβλεπτου δικτυακού φόρτου. Για το μέλλον έχουμε ως στόχο να ενσωματώσουμε στο σύστημά μας δυνατότητες εκτίμησης της ποιότητας της εμπειρίας (QoE) των τελικών χρηστών του IDVB-T δικτύου.

1 Εισαγωγή

1.1 Γενικά

Κατά την τελευταία δεκαετία, η αξιοποίηση του προτύπου επίγειας ψηφιακής εκπομπής DVB-T (Digital Video Broadcasting-Terrestrial) [1] οδήγησε στην δημιουργία δικτυακών υποδομών διαδραστικής επίγειας ψηφιακής τηλεόρασης (Interactive DVB-T), ικανών να διασυνδέσουν τους Παροχείς Υπηρεσιών (Web, Mail, Video, VoIP κ.τ.λ) με τους Τελικούς Χρήστες. Οι εν λόγω IDVB-T δικτυακές υποδομές βασίζονταν στην αρχή πάνω σε κεντροποιημένες αρχιτεκτονικές [2,3,4] ενώ μετέπειτα στράφηκαν σε αποκεντρωμένες [4,5,6] αρχιτεκτονικές, καθώς αυτές υπερείχαν των κεντροποιημένων σε θέματα κλιμακοθετησιμότητας (scalability)--μειωμένος όγκος εργασίας στη DVB-T πλατφόρμα αφού η διαχείριση χρηστών και υπηρεσιών μπορεί να γίνει τοπικά από ενδιάμεσους κόμβους διανομής (Cell Main Nodes-CMNs).

Σε τέτοια αποκεντρωμένα IDVB-T δίκτυα--και παρά τα πλεονεκτήματα της κλιμακοθετησιμότητας--ο μη ελεγχόμενος ανταγωνισμός των υπηρεσιών για τους διαθέσιμους δικτυακούς πόρους μπορεί να οδηγήσει σε καταστάσεις συμφόρησης υποβαθμίζοντας έτσι την δικτυακή απόδοση. Για την αποφυγή τέτοιων καταστάσεων είναι απαραίτητη η σε πραγματικό χρόνο παρακολούθηση της δικτυακής απόδοσης του IDVB-T, ώστε να ενεργοποιούνται έγκαιρα μηχανισμοί διασφάλισης της παρεχόμενης δικτυακής ποιότητας υπηρεσίας (Network Quality of Service-NQoS). Προς την κατεύθυνση του σχεδιασμού και υλοποίησης τέτοιων συστημάτων παρακολούθησης υπάρχουν οι εξής τρεις προσεγγίσεις: α) η κεντρικοποιημένη, όπου όλες οι λειτουργικές μονάδες (π.χ. φίλτρων κίνησης, ανάλυσης δικτυακών χαρακτηριστικών) βρίσκονται σε ένα κεντρικό κόμβο; β) ή ημικατανεμημένη όπου οι λειτουργικές μονάδες ενυπάρχουν σε κεντρικούς αλλά και περιφερειακούς κόμβους του δικτύου; και γ) η κατανεμημένη όπου όλες οι λειτουργικές μονάδες εδρεύουν σε περιφερειακούς κόμβους.

1.2 Στόχος Πτυχιακής Εργασίας

Η παρούσα πτυχιακή επικεντρώνεται στη μελέτη, σχεδίαση και υλοποίηση ενός συστήματος ημικατανεμημένης παρακολούθησης και ελέγχου της δικτυακής απόδοσης, σε διαδραστικά συστήματα επίγειας ψηφιακής ευρυεκπομπής, με απώτερο σκοπό την βελτιστοποιημένη απόδοση τους ως προς την εκμετάλλευση των διαθέσιμων δικτυακών πόρων. Επιλέξαμε την ημικατανεμημένη προσέγγιση διότι εν αντιθέσει με την κεντροποιημένη διαμοιράζει τον φόρτο εργασίας σε αρκετούς κόμβους, ενώ ταυτόχρονα αποφεύγει την πολυπλοκότητα--στο συντονισμό και επικοινωνία των περιφερειακών μονάδων--της κατανεμημένης.

1.3 Διάρθρωση Πτυχιακής Εργασίας

Ακολουθώντας την εισαγωγή, το δεύτερο κεφάλαιο παρουσιάζει όλες τις απαραίτητες θεωρητικές πληροφορίες που απαιτούνται ώστε ο αναγνώστης να εξοικειωθεί και να κατανοήσει τη λειτουργία των τεχνολογιών που χρησιμοποιούνται για την επίτευξη αυτής της πτυχιακής εργασίας. Στο τρίτο κεφάλαιο παρουσιάζεται το προτεινόμενο σύστημα ημικατανεμημένης παρακολούθησης και ελέγχου της δικτυακής απόδοσης σε συστήματα αποκεντρωμένης διαδραστικής ψηφιακής τηλεόρασης και περιγράφεται η υλοποίηση του. Στην συνέχεια, στο τέταρτο κεφάλαιο ακολουθεί η αξιολόγηση του προτεινόμενου συστήματος και η παρουσίαση των αποτελεσμάτων που προκύπτουν από τις πειραματικές μετρήσεις που πραγματοποιήθηκαν. Τέλος στο πέμπτο κεφάλαιο διατυπώνονται τα συμπεράσματα της εργασίας αυτής και παρατίθενται κάποιες προτάσεις για μελλοντική έρευνα.

2 Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο γίνεται μία περιγραφή των τεχνολογιών, των πρωτοκόλλων και του λογισμικού που χρησιμοποιήθηκαν για την υλοποίηση της πτυχιακής αυτής.

2.1 ΨΗΦΙΑΚΗ ΤΗΛΕΟΡΑΣΗ

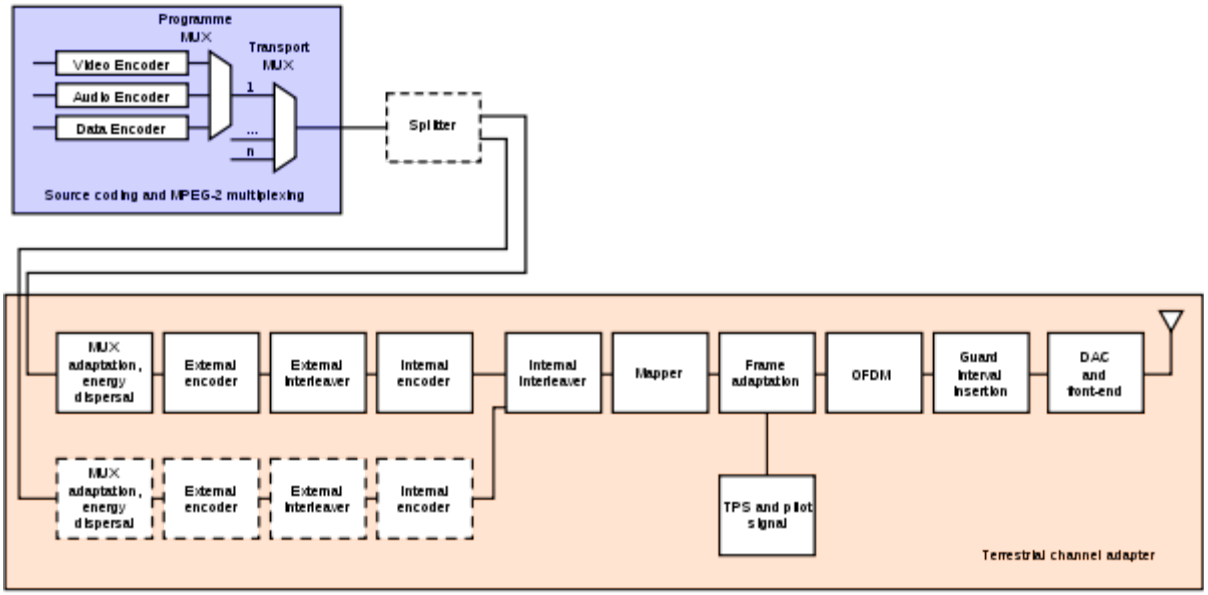
Η ψηφιακή τηλεόραση (Digital Television - DTV) είναι ένας τρόπος μετάδοσης των κινούμενων εικόνων και του ήχου που συνθέτουν ένα τηλεοπτικό πρόγραμμα, μαζί με άλλες υπηρεσίες όπως κείμενο και διαδραστικότητα. Πρόκειται για μια πρωτοποριακή υπηρεσία που αντιπροσωπεύει μια σημαντική εξέλιξη στον τομέα της τεχνολογίας της τηλεόρασης φέρνοντας μαζί της μία πλειάδα πλεονεκτημάτων έναντι της αναλογικής μετάδοσης. Χρησιμοποιεί ψηφιακά διαμορφωμένα δεδομένα, τα οποία είναι συμπιεσμένα και απαιτεί αποκωδικοποίηση από ένα ειδικά σχεδιασμένο σετ τηλεόρασης. Η τεχνολογία της ψηφιακής τηλεόρασης υλοποιείται παγκοσμίως από τρία βασικά πρότυπα: DVB (Digital Video Broadcasting) που χρησιμοποιείται στην Ευρώπη, ATSC(Advanced Television Systems Committee) που χρησιμοποιείται στην Αμερική και ISDB (Integrated Services Digital Broadcasting) που χρησιμοποιείται στην Ιαπωνία. Το σύνολο των προτύπων DVB αποτελείται από τα εξής πρότυπα: DVB-T (Digital Video Broadcasting-Terrestrial)[7], DVB-S (Digital Video Broadcasting-Satellite), DVB-RCS (Digital Video Broadcasting - Return Channel via Satellite)[7] και DVB-H (Digital Video Broadcasting - Handheld)[7]. Η τεχνολογικά προηγμένη υπηρεσία της ψηφιακής μετάδοσης έχει τα εξής χαρακτηριστικά πλεονεκτήματα:

- Σταθερή ποιότητα εικόνας, με μεγαλύτερη ανοχή στις ατέλειες του ασύρματου διαύλου. Παρέχεται, δηλαδή, καλύτερη ικανότητα λήψης, συμπεριλαμβανομένης της εξάλειψης του φαινομένου ghosting και άλλων λαθών μετάδοσης
- Απαιτείται μειωμένος λόγος SNR σε σύγκριση με την αναλογική μετάδοση.
- Αποδοτικότερη αξιοποίηση ηλεκτρομαγνητικού φάσματος
- Πολυπλεξία: Συνύπαρξη πολλών προγραμμάτων και υπηρεσιών, επιλεγόμενης ποιότητας και ευκρίνειας
- Μεταβλητό ρυθμό εκπομπής (bit rate), ανάλογα με τις απαιτήσεις ποιότητας του εκάστοτε προγράμματος
- Δυνατότητα υποστήριξης διαδραστικών υπηρεσιών (προϋποθέτει ύπαρξη reverse channel)
- Δυνατότητα παροχής προγράμματος υψηλής ευκρίνειας και προγραμμάτων ραδιοφώνου

Το πρότυπο που θα μας απασχολήσει, καθώς θα το χρησιμοποιήσουμε στα πειράματά μας, είναι αυτό του DVB-T.

2.1.1 ΠΡΟΤΥΠΟ DVB-T

Το ψηφιακό επίγειο σύστημα τηλεόρασης DVB-T, το οποίο θα χρησιμοποιηθεί και στο δίκτυο που θα γίνουν τα εργαστηριακά πειράματα της συγκεκριμένης πτυχιακής εργασίας, έχει σχεδιαστεί κυρίως για την λήψη από σταθερό σημείο και έχει τα χαρακτηριστικά του φάσματος συχνοτήτων βάση της συμφωνίας της Γενεύης του 2006. Το DVB-T μεταδίδει συμπιεσμένο ψηφιακό ήχο, βίντεο και άλλα δεδομένα σε ένα stream μεταφοράς MPEG [9], χρησιμοποιώντας διαμόρφωση COFDM (Coded Orthogonal Frequency Division Multiplexing). Ο μηχανισμός μετάδοσης, η οποία βασίζεται σε εκπομπή ψηφιακών δεδομένων, χρησιμοποιεί πολλαπλές φέρουσες συχνότητες. Το OFDM, λειτουργεί μέσω διάσπασης της ψηφιακής ροής δεδομένων σε ένα μεγάλο αριθμό μικρότερων ψηφιακών ροών (πακέτα), καθεμία από τις οποίες διαμορφώνει ψηφιακά ένα σύνολο στενά διατεταγμένων φερουσών συχνοτήτων. Το DVB-T προσφέρει τρία διαφορετικά συστήματα διαμόρφωσης (QPSK, 16QAM και 64QAM), η οποία επιλογή γίνεται ανάλογα με την αξιοπιστία μετάδοσης έναντι της ταχύτητας δεδομένων. Το Gurad Interval είναι η διάρκεια (καθυστέρηση) που μεσολαβεί ανάμεσα στα πακέτα προκειμένου να μπορέσει ο δέκτης να χειρίζεται προκαθορισμένη καθυστέρηση στη διαδρομή μετάδοσης. Στα DVB-T συστήματα η μετάδοση επιτυγχάνεται εκπέμποντας σε ένα από τα κανάλια 21-69 της μπάντας των UHF, έχοντας διαθέσιμο εύρος ζώνης 8 MHZ .

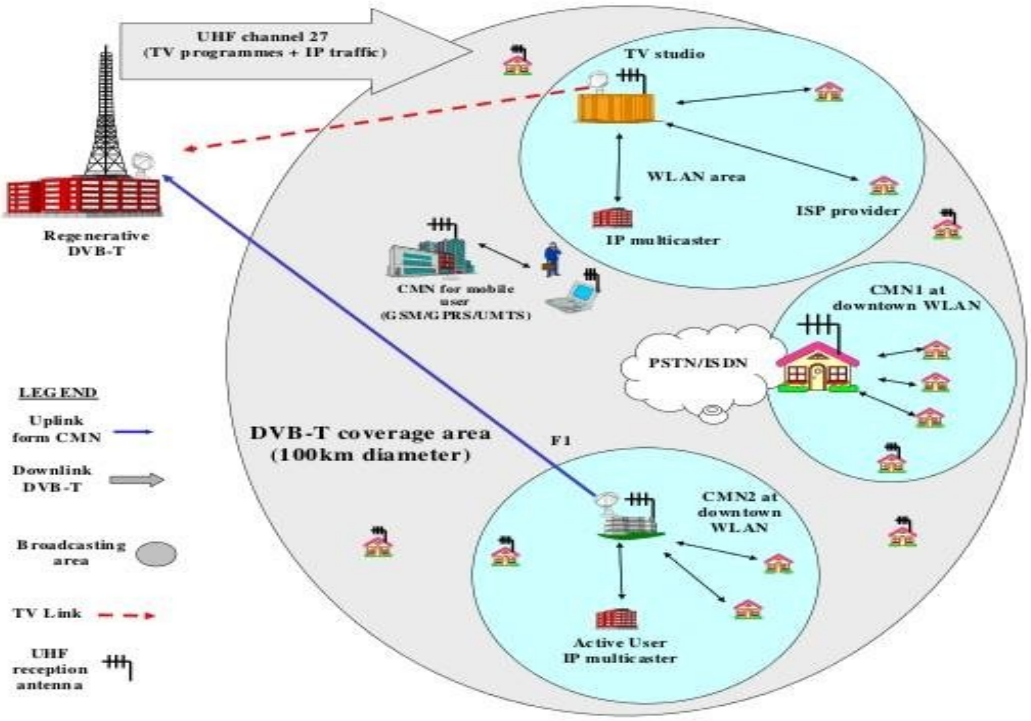


Σχήμα 1: Διάγραμμα DVB-T συστήματος μετάδοσης [8]

Για να υπάρχει η δυνατότητα της αμφίδρομης επικοινωνίας η οποία συνεπάγεται και παροχή IP υπηρεσιών, προϋποθέτει την ύπαρξη ενός καναλιού επιστροφής ώστε να στέλνονται οι αιτήσεις για την κάθε υπηρεσία που ζητάει ο χρήστης. Το συγκεκριμένο κανάλι μπορεί να είναι δίκτυο τεχνολογίας GPRS, DVBS, xDSL καθώς και όποια άλλη τεχνολογία στην οποία μπορεί να εγκατασταθεί η εκάστοτε εφαρμογή και να είναι λειτουργική.

2.1.2 Διαδραστική Ψηφιακή Τηλεόραση (Interactive DVB-T)

Το γενικό μοντέλο της αρχιτεκτονικής του δικτύου διαδραστικής ψηφιακής τηλεόρασης αποτελείται από 1) την πλατφόρμα DVB-T όπου είναι και το κεντρικό σημείο ευρυεκπομπής, 2) από ένα σύνολο ενδιάμεσων κόμβων διανομής(Cell Main Nodes-CMN's) και 3) από τους τελικούς χρήστες. Σε κάθε περιοχή που εκπέμπεται το ψηφιακό σήμα της DVB-T, η περιοχή διαιρείται σε κυψέλες και σε κάθε κυψέλη βρίσκεται και ένας CMN. Ο κάθε CMN είναι υπεύθυνος για τη δρομολόγηση της κίνησης από και προς τους τελικούς χρήστες. Λαμβάνει την κίνηση από την κάρτα τηλεόρασης(DVB-T) και την προωθεί στους τελικούς χρήστες. Παράλληλα όμως δρομολογεί και την κίνηση από τους τελικούς χρήστες προς την πλατφόρμα μέσω του καναλιού επιστροφής που έχει ο CMN. Η αμφίδρομη επικοινωνία μεταξύ κάθε CMN και του κεντρικού σημείου ευρυεκπομπής επιτυγχάνεται με τη χρήση δύο ξεχωριστών μονόδρομων καναλιών επικοινωνίας. Η IP κίνηση ενθυλακώνεται στο ρεύμα μεταφοράς DVB-T MPEG-2 χρησιμοποιώντας την τεχνική Ενθυλάκωσης Πολλαπλών Πρωτοκόλλων (Multi Protocol Encapsulation – MPE) παρέχοντας έτσι ένα ιδεατό δίκτυο κορμού ETHERNET για τις IP υπηρεσίες. Ακολουθεί η πολυπλεξία της IP κίνησης και των τηλεοπτικών προγραμμάτων σε μια ροή DVB-T, η οποία με την σειρά της διαμορφώνεται χρησιμοποιώντας Κωδικοποιημένη Ορθογώνική Πολύπλεξη Συχνότητας (COFDM) και εκπέμπεται μέσω ενός καναλιού UHF .



Σχήμα 2: Γενική Αρχιτεκτονική ενός DVB-T δικτύου

2.2 Παρακολούθηση & Ανάλυση Δικτύων

Η έννοια του Network Monitoring [10] ή αλλιώς η παρακολούθηση δικτύου δημιουργήθηκε από την πρώτη στιγμή που υπήρξαν δίκτυα υπολογιστών και αναφέρεται στη δυνατότητα της συλλογής και της ανάλυσης της κυκλοφορίας ενός δικτύου. Τα περισσότερα routers, switches καθώς και όλοι οι ευφυείς κόμβοι συλλέγουν κάποιο ποσοστό στατιστικών κίνησης του δικτύου. Αυτή η πληροφορία είναι σημαντική για τους διαχειριστές του δικτύου οι οποίοι είναι υπεύθυνοι για την σωστή λειτουργία του. Χωρίς τέτοιου είδους συστήματα παρακολούθησης, θα ήταν εξαιρετικά δύσκολο να εντοπιστούν και να επιλυθούν πολλά δικτυακά προβλήματα. Οι πιο έξυπνες συσκευές δικτύωσης προσφέρουν ανάλυση του επιπέδου (layer) 1 της κυκλοφορίας. Σε αυτό το επίπεδο, η ανάλυση επικεντρώνεται συνήθως στα φυσικά προβλήματα του δικτύου, όπως η κατάσταση της σύνδεσης, τα CRC errors, τις διπολικές παραβιάσεις, καθώς και τα framing errors. Τα συστήματα network monitoring που αναλύουν τα επίπεδα 2 και 3 συνήθως αναφέρονται ως ‘’αναλυτές πρωτοκόλλων’’, επειδή αυτά τα επίπεδα βασίζονται σε ειδικά πρωτόκολλα για τον έλεγχο της μετάδοσης των δεδομένων. Η τελευταία γενιά των προϊόντων δικτυακής παρακολούθησης έχει σχεδιαστεί για να υποστηρίξει πολύ ειδικές λειτουργίες. Για παράδειγμα, ορισμένα προϊόντα παρακολούθησης έχουν σχεδιαστεί να βοηθούν τους διαχειριστές δικτύων να αναγνωρίζουν προβλήματα ασφαλείας , κάποια άλλα για να αναλύουν την απόδοση συγκεκριμένων εφαρμογών και μερικά άλλα για τη συλλογή πρωτογενών δεδομένων και τη μετέπειτα εντατική ανάλυσή τους. Κάθε μία από όλες αυτές τις ειδικότητες μπορεί να προσφέρει μία εστιασμένη λύση που έχει σχεδιαστεί για να αντιμετωπίσει τις ειδικές απαιτήσεις της απαιτητικής αγοράς.

2.2.1 Μεθοδολογίες Παρακολούθησης

Στο Network Monitoring υπάρχουν δύο βασικές μεθοδολογίες: Η παθητική ή αλλιώς η ‘μη παρεμβατική’ παρακολούθηση δικτύων (passive monitoring)[11,12,13] και η ενεργητική ή αλλιώς η ‘παρεμβατική’ παρακολούθηση δικτύων (active monitoring)[10,11,12].

Στο passive monitoring χρησιμοποιούνται μηχανισμοί οι οποίοι παρακολουθούν την ροή του δικτύου καθώς τους διαπερνά. Αυτοί οι μηχανισμοί μπορεί να είναι Sniffer ή OCxMon ή μπορεί να έχουν σχεδιαστεί μέσα σε άλλες συσκευές όπως είναι τα routers, τα switches ή σε κάποιο end-node-host. Παραδείγματα τέτοιου είδους ενσωματώσεων είναι το Remote Monitoring (RMON), το Simple Network Monitoring Protocol (SNMP) και οι συσκευές που υποστηρίζουν netflow ανάλυση. Οι παθητικοί μηχανισμοί παρακολούθησης επικοινωνούν περιοδικά μεταξύ τους και οι πληροφορίες που συλλέγονται χρησιμοποιούνται για την αξιολόγηση της κατάστασης του δικτύου. Η παθητική προσέγγιση δικτυακής παρακολούθησης είναι εξαιρετικά πολύτιμη για τον εντοπισμό δικτυακών προβλημάτων, ωστόσο, περιορίζονται στην ικανότητά τους και δεν μπορούν να δημιουργήσουν σενάρια λύσεων για τυχόν προβλήματα ή να βρουν την ακριβή τοποθεσία τους. Τέλος, δεδομένου ότι το passive monitoring μπορεί να απαιτήσει την προβολή όλων των πακέτων στο δίκτυο, μπορεί να προκύψουν θέματα ασφαλείας σχετικά με το πώς θα αποκτηθεί πρόσβαση ή πως θα υπάρξει η προστασία δεδομένων στις πληροφορίες που συλλέχθηκαν.

Το active monitoring στηρίζεται στην ικανότητα να στέλνει πακέτα δοκιμών στο δίκτυο ή σε servers και διάφορες εφαρμογές, στη συνέχεια τα ‘’ακολουθεί’’ και μετράει τις υπηρεσίες που μπορεί να προσφέρει το δίκτυο. Ως εκ τούτου δημιουργεί επιπλέον κίνηση η οποία είναι τεχνητή. Ο όγκος των πληροφοριών και των πακέτων που εισήχθησαν στη κυκλοφορία του δικτύου είναι πλήρως ρυθμιζόμενα και πρέπει να επισημανθεί πως ακόμα και αν αυτός ο όγκος είναι πολύ μικρός, είναι αρκετός για να αποκτήσουν νόημα οι μετέπειτα μετρήσεις για την εικόνα του δικτύου. Με αυτήν την παρεμβατική μέθοδο παρακολούθησης λοιπόν, έχουμε σαφή έλεγχο σχετικά με την παραγωγή των πακέτων μας και τα σενάρια που θα φτιάξουμε μόλις έχουμε τις απαραίτητες μετρήσεις. Με άλλα λόγια έχουμε τον πλήρη έλεγχο της παραγωγής των πακέτων, τις τεχνικές δειγματοληψίας, του χρόνου, της συχνότητας, των μεγεθών των πακέτων και γενικότερα της διαδρομής και της δομής που θα έχει επιλεχθεί για να γίνει η ενεργητική παρακολούθηση. Το active monitoring προϋποθέτει ότι ο διαχειριστής ενός δικτύου θα μπορεί να ελέγχει ότι θέλει και όποτε το θέλει. Εξομοιώνει έτσι εύκολα τα διάφορα σενάρια που μπορεί να έχει δημιουργήσει και ελέγχει εάν πληρούνται οι προϋποθέσεις ώστε να παρέχονται οι διάφορες ποιότητες υπηρεσιών και να τηρούνται οι δικτυακοί κανόνες.

2.2.2 Μεθοδολογίες Ανάλυσης

Για την ανάλυση των στοιχείων που “παρακολουθούνται” υπάρχουν δύο βασικές μεθοδολογίες: Η online ή αλλιώς η “σε συνεχή σύνδεση” ανάλυση και η offline ή αλλιώς η “εκτός σύνδεσης” ανάλυση.

Οι εφαρμογές monitoring που χρησιμοποιούν online ανάλυση, ουσιαστικά υποστηρίζουν την ανάλυση των δεδομένων σε πραγματικό χρόνο, δηλαδή, ταυτόχρονα με τη συλλογή των δεδομένων από τη ροή του δικτύου. Αντίθετα, οι εφαρμογές που χρησιμοποιούν offline ανάλυση, όπως για παράδειγμα το tcpdump, προορίζονται για την συλλογή και την αποθήκευση των δεδομένων, με σκοπό την ανάλυσή του μετά το τερματισμό της κίνησης του δικτύου και όχι ταυτόχρονα.

2.2.3 Παράμετροι Αξιολόγησης

Σε αυτή την υποενότητα θα παρουσιαστούν οι βασικές μετρήσεις που θα λάβουμε υπόψη χρησιμοποιώντας active monitoring.

Bit rate [13,17]: Στον τομέα των τηλεπικοινωνιών και της πληροφορικής, ο ρυθμός bit rate(μερικές φορές γράφεται bitrate ή ως μεταβλητή R [1]) είναι ο αριθμός των bits που μεταφέρονται ή επεξεργάζονται ανά μονάδα χρόνου.

One way delay [14,17]: Το one way delay ή αλλιώς μονόδρομη καθυστέρηση είναι ο χρόνος που χρειάζεται ένα οποιοδήποτε πακέτο για να μεταφερθεί από μία πηγή Α σε ένα προορισμό Β.

Jitter [15,17]: Ένα, στενά συνδεδεμένο με τη μονόδρομη καθυστέρηση, εργαλείο μέτρησης είναι το jitter ή αλλιώς διακύμανση της καθυστέρησης. Ένα δίκτυο με σταθερή καθυστέρηση δεν παρουσιάζει, όπως είναι λογικό, καμία διακύμανση. Το jitter ουσιαστικά εκφράζεται ως ο μέσος όρος της απόκλισης από τη μέση καθυστέρηση που παρουσιάζει το δίκτυο.

Packet loss [16,17]: Το packet loss συμβαίνει όταν ένα ή περισσότερα πακέτα τα οποία ρέουν μέσα στην κυκλοφορία ενός δικτύου αποτυγχάνουν να φτάσουν στον προορισμό τους.

2.3 XML

Η XML (Extensible Markup Language) [18] είναι ένα είδος “markup language” (οι markup languages έχουν σχεδιαστεί για την επεξεργασία, τον ορισμό και την παρουσίαση ενός κειμένου), η οποία ορίζει ένα σύνολο κανόνων για κωδικοποιημένα αρχεία με τέτοιο τρόπο ώστε να μπορούν να διαβαστούν από τους υπολογιστές και τους ανθρώπους ταυτόχρονα.
Οι στόχοι της xml δίνουν βάση κυρίως στην απλότητα, τη γενικότητα και την ευχρηστία. Ουσιαστικά πρόκειται για μία μορφή δεδομένων κειμένου τα οποία μέσω της unicode υποστήριξης μπορούν να χρησιμοποιηθούν από όλες σχεδόν τις γλώσσες προγραμματισμού. Ο σχεδιασμός των XML επικεντρώνεται κυρίως σε έγγραφα τα οποία χρησιμοποιούνται ευρέως για την αναπαράσταση των διάφορων δομών δεδομένων σε διάφορους τομείς, όπως για παράδειγμα στον τομέα των web services. Τα XML έγγραφα ουσιαστικά χρησιμοποιούν μία περιγραφή του εαυτού τους και μία απλή σύνταξη.

2.4 Λογισμικό

2.4.1 FFMPEG

Το FFmpeg είναι ένα δωρεάν πρόγραμμα λογισμικού το οποίο παράγει βιβλιοθήκες και προγράμματα για το χειρισμό των πολυμεσικών δεδομένων. Περιλαμβάνει τις βιβλιοθήκες libavcodec[19] και libavformat[20] καθώς και ένα πρόγραμμα γραμμής εντολών για το transcoding των πολυμεσικών αρχείων.

Στη συγκεκριμένη πτυχιακή εργασία, χρησιμοποιήθηκε το FFMPEG για να επιτευχθεί η δημιουργία τριών video με διαφορετικό transcoding. Το πρώτο video έγινε transcode στα 1Mb, το δεύτερο στα 2Mb, και το τρίτο στα 3Mb.

2.4.2 VLC

Το VLC media[21] player υποστηρίζει πολλές συμπιέσεις εικόνας και ήχου καθώς και μορφοποιήσεις αρχείων, συμπεριλαμβανομένων των DVD-Video, CD βίντεο και streaming πρωτοκόλλων. Έχει την ικανότητα να κάνει stream μέσω ενός δικτύου υπολογιστών και καθώς και

transcode σε πολυμεσικά αρχεία.

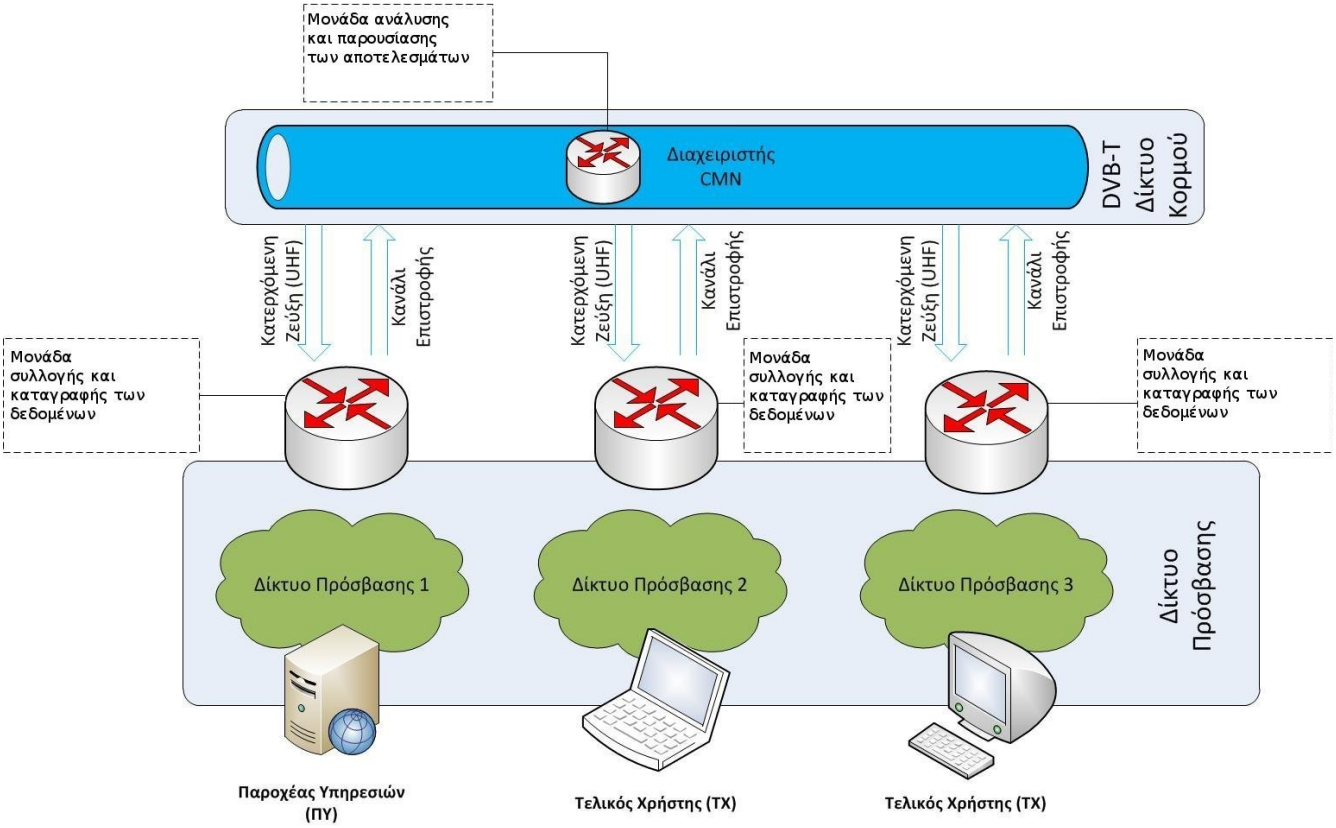
2.4.3 GNU OCTAVE

Το GNU Octave [22]είναι μία υψηλού επιπέδου γλώσσα, που προορίζεται κυρίως για αριθμητικούς υπολογισμούς. Παρέχει δυνατότητες για την αριθμητική επίλυση γραμμικών και μη γραμμικών προβλημάτων, καθώς και για την εκτέλεση άλλων αριθμητικών πειραμάτων. Παρέχει επίσης εκτεταμένες δυνατότητες για τη δημιουργία γραφικών και την απεικόνιση των δεδομένων τους. Το Octave χρησιμοποιείται συνήθως μέσω του διαδραστικού περιβάλλοντος των γραμμών εντολών του, αλλά μπορεί επίσης να χρησιμοποιηθεί και για να γραφτούν μη διαδραστικά προγράμματα. Η Octave γλώσσα είναι αρκετά παρόμοια με τη Matlab, έτσι ώστε τα περισσότερα προγράμματα που φτιάχνονται στη μία είναι εύκολα αναγνωρίσιμα και στην άλλη .

3 Σύστημα ημικατανεμημένης παρακολούθησης και ελέγχου της δικτυακής απόδοσης

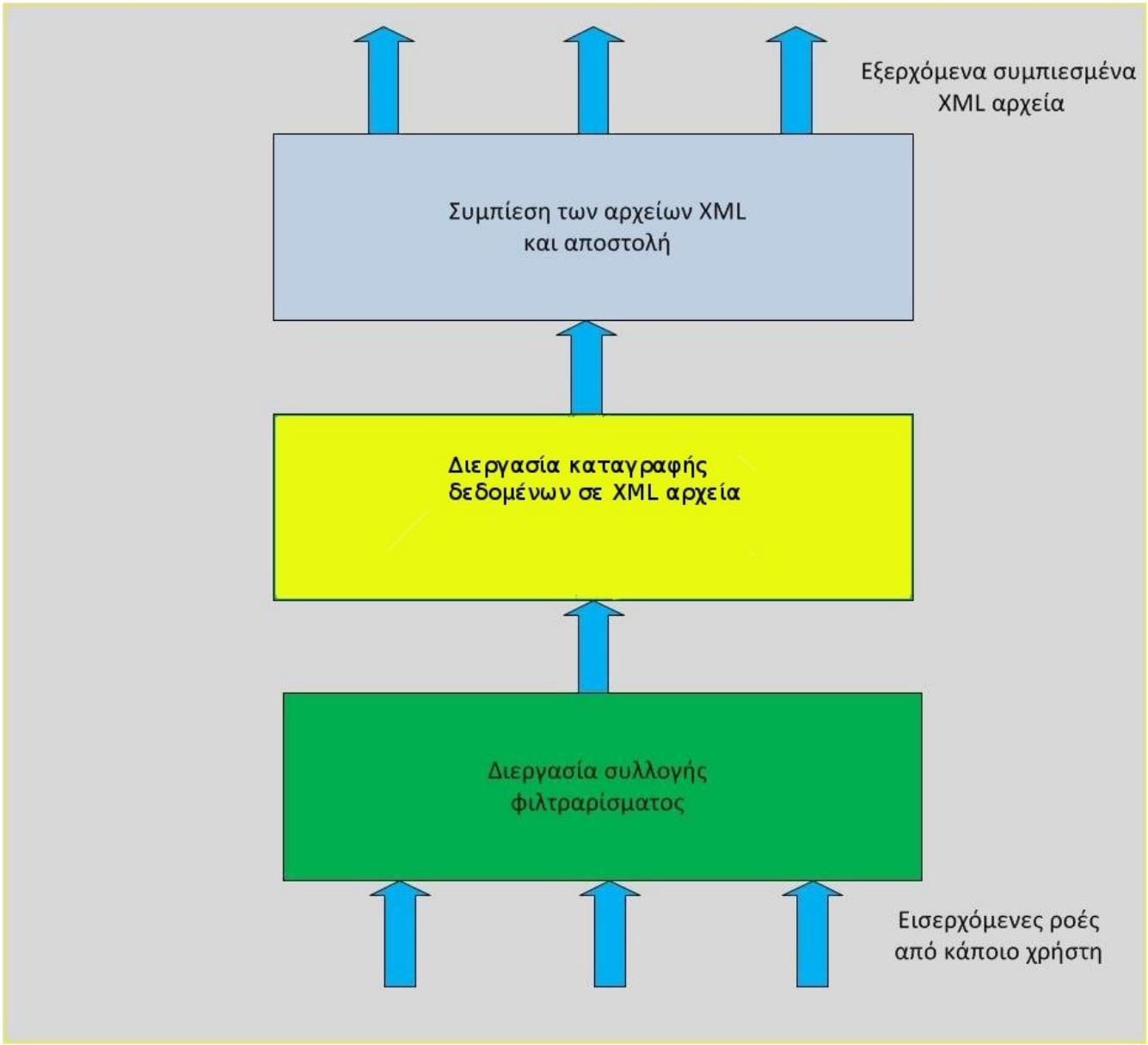
3.1 Σχεδιασμός Συστήματος

Σε αντιδιαστολή με τις μέχρι τώρα κεντρικοποιημένες και κατανεμημένες προσεγγίσεις στη παρακολούθηση και έλεγχο της δικτυακής απόδοσης εμείς προτείνουμε ένα ημικατανεμημένο σύστημα στο οποίο, όπως απεικονίζεται στο Σχήμα 3, οι μονάδες συλλογής και καταγραφής δικτυακής κίνησης εδρεύουν στους ενδιάμεσους κόμβους διανομής, ενώ η μονάδα ανάλυσης και παρουσίασης των αποτελεσμάτων εδρεύει στη DVB-T πλατφόρμα. Ο ημικατανεμημένος τρόπος λειτουργίας του προτεινόμενου συστήματος αντιμετωπίζει προβλήματα κλιμακοθετησιμότητας, αποφεύγοντας τη συγκέντρωση του φόρτου διαχείρισης σε μια κεντρική μονάδα, ενώ διατηρεί σε χαμηλό επίπεδο την επικοινωνιακή επιβάρυνση που απαιτείται για τον συντονισμό των λειτουργικών μονάδων που απαρτίζουν το σύστημα παρακολούθησης και ελέγχου της δικτυακής απόδοσης.



Σχήμα 3: Σύστημα ημικατανεμημένης παρακολούθησης σε IDVB-T δίκτυο

3.1.1 Μονάδα συλλογής και καταγραφής δεδομένων

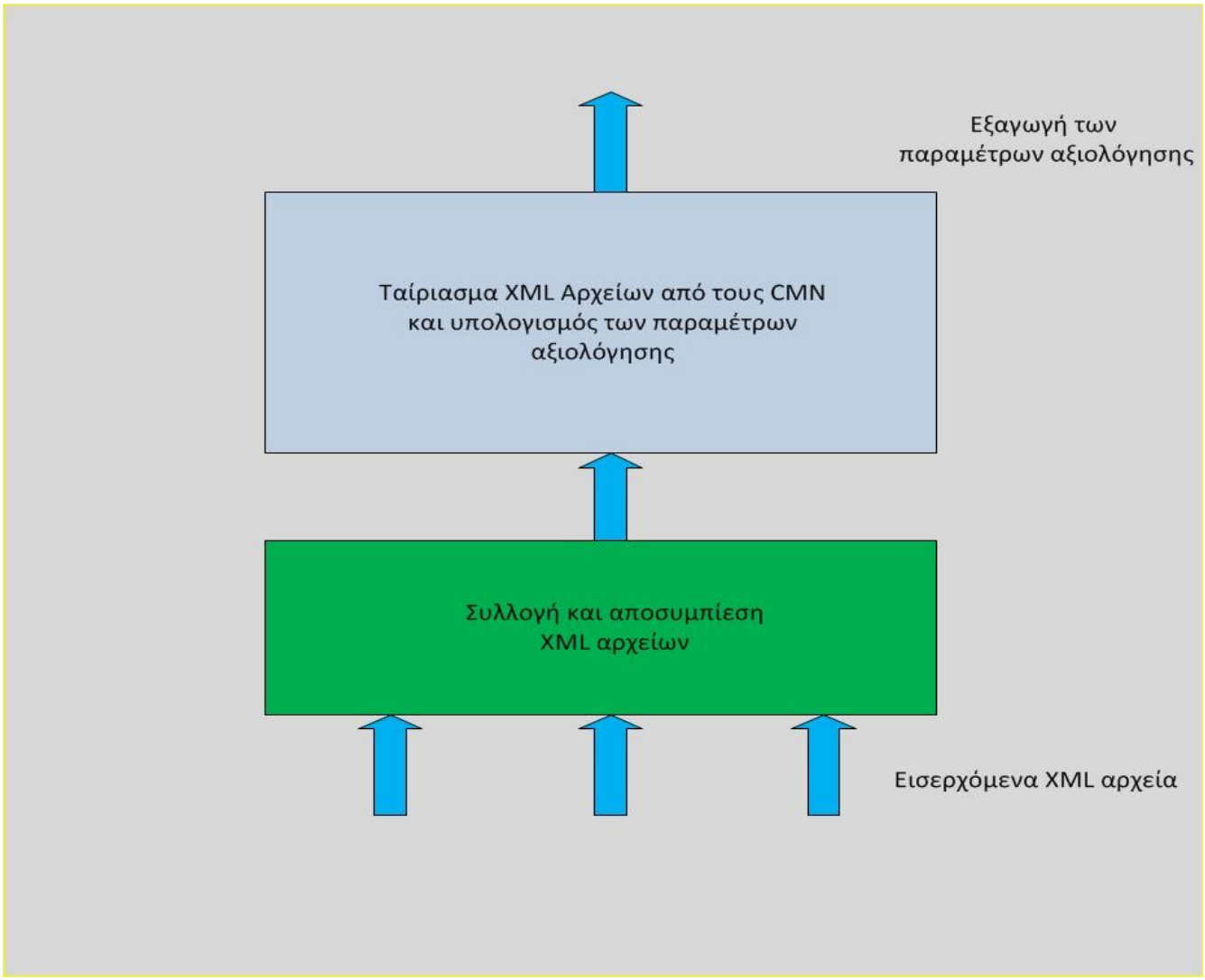


Σχήμα 4: Διεργασίες μονάδας συλλογής και καταγραφής δεδομένων

Αυτό το κομμάτι της εφαρμογής είναι υπεύθυνο για τη καταγραφή και τη διαβίβαση των πληροφοριών σχετικά με την κίνηση που εισέρχεται στο IDVB-T δίκτυο. Ο χρήστης, πριν αρχίσει την καταγραφή της ροής στον CMN 2, θα πρέπει να δώσει κάποιες απαραίτητες παραμέτρους όπως τον αριθμό των ροών που θέλει να παρακολουθεί, την κάρτα δικτύου που συλλέγει τα πακέτα της ροής, η

IP διεύθυνση του προορισμού που θέλει να στείλει τα αποτελέσματα της ροής, το πρωτόκολλο το οποίο χρησιμοποιεί η ροή που θα παρακολουθείται. Όταν ο χρήστης παρέχει τις απαραίτητες παραμέτρους, η εφαρμογή ξεκινά τη συλλογή των πακέτων της ροής που δημιούργησε ο server. Μετά τη συλλογή ενός αριθμού πακέτων, και αφού τα φίλτραρε σύμφωνα με τις απαιτήσεις του χρήστη συγκεντρώνει όλα τα χρήσιμα δεδομένα σε ένα αρχείο XML και στη συνέχεια συμπιέζει το αρχείο και το στέλνει στο κομμάτι της εφαρμογής CORE που βρίσκεται στον CORE Router (βλέπε σχήμα 2). Πιο αναλυτικά, μέσα στο XML αρχείο που δημιουργεί, η εφαρμογή, βάζει ένα στοιχείο (element) σε αυτό για κάθε παρακολουθούμενη ροή. Κάτω από κάθε στοιχείο δημιουργείται ένα ακόμη στοιχείο για κάθε πακέτο που έχει λάβει. Κάτω από κάθε πακέτο υπάρχει ένα χαρακτηριστικό (attribute) που περιέχει τις πληροφορίες σύμφωνα με τις οποίες λάβει μέρος η ανάλυση στη μονάδα του CORE Router. Συγκεκριμένα, τα χαρακτηριστικά αυτά είναι: το id πακέτων ή τον αριθμό σειράς (sequence number), το timestamp, το ωφέλιμο φορτίο (payload), και το checksum.
Η διαδικασία αυτή επαναλαμβάνεται ανά κάποια χρονικά διαστήματα (τα οποία έχει ορίσει ο ίδιος ο χρήστης) μέχρι ο διακομιστής να σταματήσει την αποστολή πακέτων προς τον Πελάτη. Τα δεδομένα που συλλέγονται από το κάθε πακέτο εξαρτώνται από το πρωτόκολλο το οποίο έχει επιλέξει ο διακομιστής. Η εφαρμογή μπορεί να παρακολουθεί τις ροές που χρησιμοποιούν RTP ή TCP πρωτόκολλο.

3.1.2 Μονάδα ανάλυσης και παρουσίασης των αποτελεσμάτων



Σχήμα 5: Διεργασίες της μονάδας ανάλυσης και παρουσίασης

Αυτό το κομμάτι της εφαρμογής είναι υπεύθυνο για τη λήψη των αρχείων XML, που στέλνονται διαρκώς από τους CMN, και την ανάλυση των πληροφοριών που υπάρχουν μέσα σε αυτά. Ο χρήστης, πριν αρχίσει την καταγραφή της ροής, θα πρέπει να δώσει κάποιες απαραίτητες παραμέτρους, όπως η διεύθυνση IP των XML αποστολέων καθώς και την πόρτα (port) στην οποία στέλνονται αυτά τα αρχεία. Όταν οι πληροφορίες στέλνονται, το κομμάτι του CORE ξεκινά τη συλλογή, την αποσυμπίεση, και το ταίριασμα των XML αρχείων και στη συνέχεια υπολογίζει τα χαρακτηριστικά του δικτύου. Αυτά τα χαρακτηριστικά (αναφέρθηκαν εκτενέστερα στο δεύτερο κεφάλειο) είναι το bit rate, η μονόδρομη καθυστέρηση, το jitter, και ο Ρυθμός απωλειών των πακέτων.

Μετά τον υπολογισμό, ο CORE έχει τη δυνατότητα να απεικονίσει τις πληροφορίες σε γραφικές παραστάσεις και διαγράμματα. Η διαδικασία συνεχίζεται μέχρις ότου ο χρήστης αποφασίσει να τη διακόψει.

3.2 Υλοποίηση Συστήματος

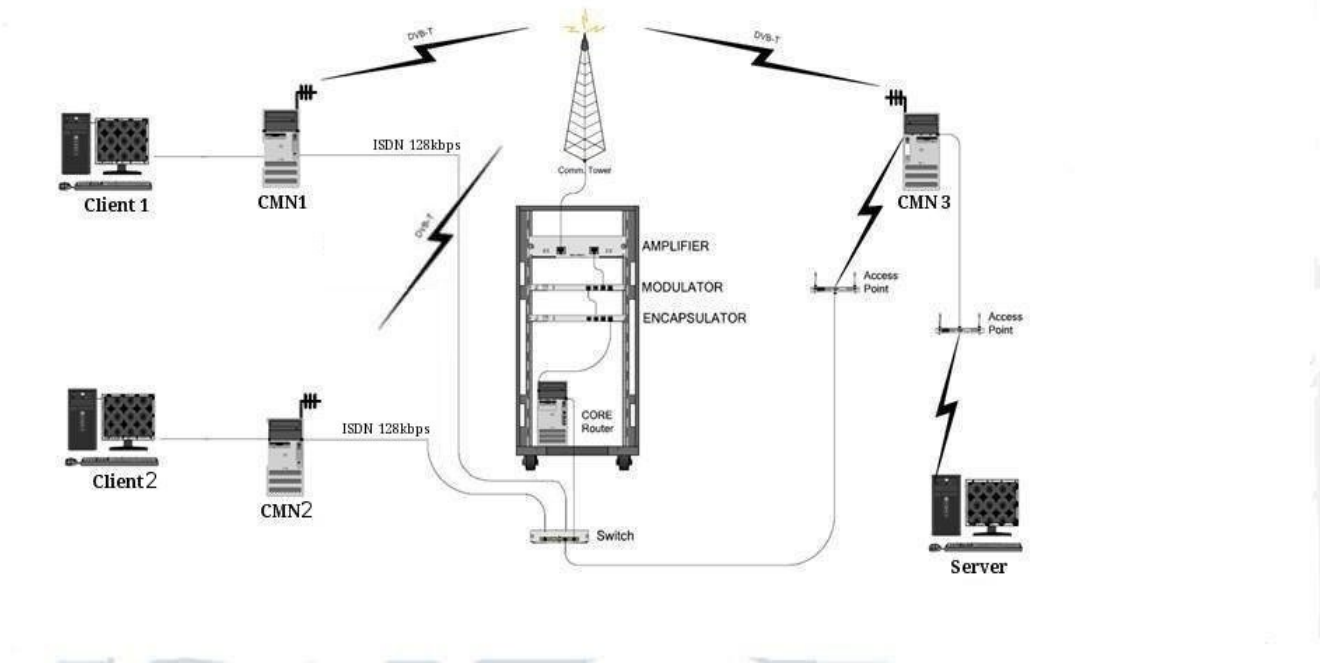
Οι μονάδες της εφαρμογής δημιουργήθηκαν σε γλώσσα προγραμματισμού JAVA. Η μονάδα ανάλυσης και παρουσίασης αποτελείται από 4 κλάσεις, η κάθε μία από τις οποίες είναι υπεύθυνη για κάποια διεργασία. Οι διεργάσιες είναι η ανάλυση, η αποσυμπίεση, το άνοιγμα και το ταίριασμα των xml αρχείων. Η τέταρτη κλάση αποσκοπεί στη σωστή συνεννόηση και λειτουργία των υπολοίπων τριών. Η μονάδα συλλογής και καταγραφής των δεδομένων αποτελείται και αυτή από 4 κλάσεις. Οι κλάσεις έχουν σκοπό να ολοκληρώσουν τις διεργασίες της συλλογής και φιλτραρίσματος των πακέτων, την καταγραφή των δεδομένων τους σε XML αρχεία, και την συμπίεση και αποστολή των αρχείων αυτών. Η τέταρτη κλάση έχει ακριβώς την ίδια λειτουργία και σημαντικότητα με αυτή της μονάδας ανάλυσης και παρουσίασης, η οποία δεν είναι άλλη από τη σωστή λειτουργία και συνεργασία των υπολοίπων τριών κλάσεων.

4 Αξιολόγηση

Στο κεφάλαιο αυτό θα γίνει η παρουσίαση του πειραματικού δικτύου διαδραστικής ψηφιακής τηλεόρασης, των σεναρίων αξιολόγησης του συστήματος ημικατανεμημένης παρακολούθησης και ελέγχου δικτυακής απόδοσης καθώς και των πειραματικών αποτελεσμάτων.

4.1 Πειραματικό Δίκτυο Διαδραστικής Ψηφιακής τηλεόρασης

Το πειραματικό IDVB-T δίκτυο απεικονίζεται στο παρακάτω Σχήμα και αποτελείται από ένα Παροχέα Υπηρεσιών (Server), δύο Πελάτες (Client1, Client2), ένα δρομολογητή (Core Router), τη πλατφόρμα DVB-T και τρεις ενδιάμεσους κόμβους διανομής (CMN1, CMN2, CMN3). Οι Client1, Client2 επικοινωνούν με τους διασυνδεδεμένους με αυτούς CMNs (CMN1, CMN2) μέσω της τεχνολογίας ADSL(1024/8128)Kbps, ενώ ο Server επικοινωνεί με το CMN3 μέσω ασύρματου δικτύου WLAN(802.11g). Όλοι οι CMNs αξιοποιούν ως μονόδρομο κανάλι επιστροφής δίκτυο ISDN(128kbps). Τέλος για την επικοινωνία από την DVB-T πλατφόρμα πρός τους ενδιάμεσους κόμβους διανομής αξιοποιείται το DVB-T κανάλι (10Mbps). Σε αυτό το πειραματικό δίκτυο, οι Πελάτες αποστέλλουν τις αιτήσεις του στον διασυνδεδεμένο με αυτούς ενδιάμεσο κόμβο(CMN). Ο CMN προωθεί μέσω του καναλιού επιστροφής του τις αιτήσεις στο Core Router, ο οποίος τις διαβιβάζει με την σειρά του στη DVB-T πλατφόρμα. Εκεί οι αιτήσεις πολυπλέκονται σε ένα κοινό ρεύμα μεταφοράς (MPEG-TS), το οποίο και διαμορφώνεται κατά το πρότυπο DVB-T, πριν ενισχυθεί και μεταδοθεί σε όλη την περιοχή εκπομπής. Το εκπεμπόμενο σήμα λαμβάνεται από τους CMNs, αποπολυπλέκεται και προωθείται στο Server. Η ανάλογη διαδικασία ακολουθείται και για την αποστολή των υπηρεσιών από το Server προς τους Πελάτες. Στα πλαίσια της αξιολόγησης του συστήματος παρακολούθησης & ελέγχου δικτυακής απόδοσης, η μονάδα συλλογής και καταγραφής δικτυακής κίνησης εγκαταστάθηκε στους CMN1,CMN2 και CMN3, ενώ η μονάδα ανάλυσης και παρουσίασης αποτελεσμάτων εγκαταστάθηκε στο Core Router.



Σχήμα 6: Πειραματική υποδομή του DVB-T δικτύου

4.2 Υπηρεσίες

Στα πλαίσια της αξιολόγησης της αποτελεσματικότητας του υλοποιημένου συστήματος παρακολούθησης και ελέγχου δικτυακής απόδοσης δημιουργήθηκαν οι παρακάτω ροές Video Streaming υπηρεσιών για να μεταφερθούν μέσω του IDVB-T δικτύου.

	Video codec	Encoding Bitrate(Kbps)
Ροή 1	MPEG4	1482
Ροή 2	MPEG4	2007
Ροή 3	MPEG4	3006

Πίνακας 1: Ροές

4.3 Σενάρια

Τα παρακάτω σενάρια περιγράφουν τα πειράματα που εκτελέστηκαν για να αξιολογηθεί το--ενσωματωμένο στο πειραματικό IDVB-T δίκτυο--προτεινόμενο σύστημα ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης. Πιο συγκεκριμένα κάθε σενάριο αναφέρει το στόχο του, τους κόμβους που θα χρησιμοποιηθούν και το είδος της δικτυακής κίνησης που θα διακινηθεί μεταξύ των κόμβων. Όλες οι υπηρεσίες μεταδόθηκαν αξιοποιώντας την εφαρμογή VLC, τα πρωτόκολλα μεταφοράς RTP/UDP και είχαν χρονική διάρκεια 5 λεπτά. Τέλος υπενθυμίζεται, ότι εκτός και εάν αναφερθεί ρητώς κάτι άλλο το διαθέσιμο εύρος ζώνης του DVB-T καναλιού είναι 8 Mbps.

4.3.1 Σεναριο 1

Το πρώτο σενάριο παρακολουθεί τη ροή του δικτύου στην πιο απλή εκδοχή του (βλέπε Εικόνα 4.1). Ο server κάνει stream ένα πεντάλεπτο video κωδικοποίησης 1Mbps στον client. Οι κόμβοι που χρησιμοποιούνται είναι ο client1, ο server,ο CMN1, ο CMN2,και ο CORE.Σε αυτό το σενάριο χρησιμοποιούνται οι κόμβοι: Server, Client, CMN1, CMN2 και Core Router. Ο Server αποστέλλει τη Ροή 1 στο Client, ενώ οι μονάδες συλλογής και καταγραφής των CMN1 και CMN2 συλλέγουν, καταγράφουν και αποστέλλουν πληροφορίες σχετικά με τη μεταδιδόμενη ροή, στη μονάδα ανάλυσης και επεξεργασίας του Core Router για τη περαιτέρω επεξεργασία και εξαγωγή των αποτελεσμάτων δικτυακής απόδοσης του IDVB-T. Ο σκοπός αυτού του σεναρίου είναι η αξιολόγηση της αποτελεσματικότητας του ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης σε συνθήκες χαμηλού δικτυακού φόρτου.

4.3.2 Σενάριο 2

Σε αυτό το σενάριο χρησιμοποιούνται οι κόμβοι: Server, Client, CMN1, CMN2 και Core Router. Ο Server αποστέλλει ταυτόχρονα τις Ροές 1,2,3 στο Client, ενώ οι μονάδες συλλογής και καταγραφής των CMN1 και CMN2 συλλέγουν, καταγράφουν και αποστέλλουν πληροφορίες σχετικά με τις μεταδιδόμενες ροές, στη μονάδα ανάλυσης και επεξεργασίας του Core Router για τη περαιτέρω επεξεργασία και εξαγωγή των αποτελεσμάτων δικτυακής απόδοσης του IDVB-T. Ο σκοπός αυτού του σεναρίου είναι η αξιολόγηση της αποτελεσματικότητας του ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης σε συνθήκες μέτριου δικτυακού φόρτου.

4.3.3 Σενάριο 3

Σε αυτό το σενάριο χρησιμοποιούνται οι κόμβοι: Server, Client, CMN1, CMN2 και Core Router. Ο Server αποστέλλει ανά 10 δευτερόλεπτα και με την εξής σειρά τη: α) Ροή 1, β) Ροή 2 και γ) Ροή 3 στο Client, ενώ οι μονάδες συλλογής και καταγραφής των CMN1 και CMN2 συλλέγουν, καταγράφουν και αποστέλλουν πληροφορίες σχετικά με τις μεταδιδόμενες ροές, στη μονάδα ανάλυσης

και επεξεργασίας του Core Router για τη περαιτέρω επεξεργασία και εξαγωγή των αποτελεσμάτων δικτυακής απόδοσης του IDVB-T. Ο σκοπός αυτού του σεναρίου είναι η αξιολόγηση της αποτελεσματικότητας του ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης σε συνθήκες: α) σταθερού ρυθμού εισόδου και εξόδου υπηρεσιών και β) μέτριου δικτυακού φόρτου.

4.3.4 Σενάριο 4

Σε αυτό το σενάριο χρησιμοποιούνται οι κόμβοι: Server, Client, CMN1, CMN2 και Core Router. Ο Server αποστέλλει τυχαία μέσα από χρονικό διάστημα 1-20 δευτερολέπτων και με την εξής σειρά τη: α) Ροή 1, β) Ροή 2 και γ) Ροή 3 στο Client, ενώ οι μονάδες συλλογής και καταγραφής των CMN1 και CMN2 συλλέγουν, καταγράφουν και αποστέλλουν πληροφορίες σχετικά με τις μεταδιδόμενες ροές, στη μονάδα ανάλυσης και επεξεργασίας του Core Router για τη περαιτέρω επεξεργασία και εξαγωγή των αποτελεσμάτων δικτυακής απόδοσης του IDVB-T. Ο σκοπός αυτού του σεναρίου είναι η αξιολόγηση της αποτελεσματικότητας του ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης σε συνθήκες: α) δυναμικού ρυθμού εισόδου και εξόδου υπηρεσιών και β) μέτριου δικτυακού φόρτου.

4.3.5 Σενάριο 5

Σε αυτό το σενάριο χρησιμοποιούνται οι κόμβοι: Server, Client1, Client2, CMN1, CMN2, CMN3 και Core Router. Ο Server αποστέλλει ταυτόχρονα σε κάθε Client όλες τις υπηρεσίες με την εξής σειρά: α) Ροή 1, β) Ροή 2, γ) Ροή 3, αφήνοντας διάστημα 10 δευτερολέπτων μεταξύ της κάθε Ροής. Παράλληλα, οι μονάδες συλλογής και καταγραφής των CMN1, CMN2 και CMN3 συλλέγουν, καταγράφουν και αποστέλλουν πληροφορίες σχετικά με τις μεταδιδόμενες ροές, στη μονάδα ανάλυσης και επεξεργασίας του Core Router για τη περαιτέρω επεξεργασία και εξαγωγή των αποτελεσμάτων δικτυακής απόδοσης του IDVB-T. Ο σκοπός αυτού του σεναρίου είναι η αξιολόγηση της αποτελεσματικότητας του ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης σε συνθήκες: α) σταθερού ρυθμού εισόδου και εξόδου υπηρεσιών και β) υψηλού δικτυακού φόρτου.

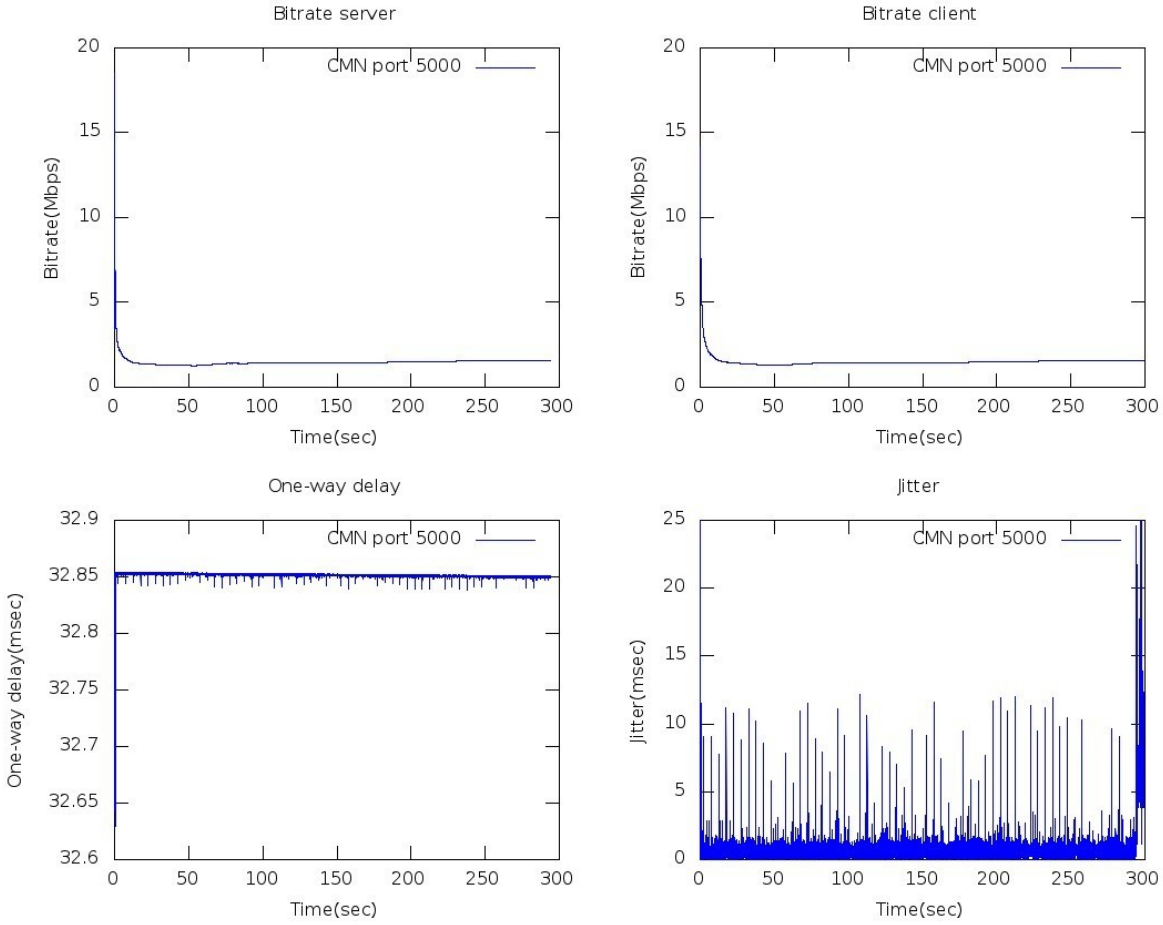
4.3.6 Σενάριο 6

Σε αυτό το σενάριο χρησιμοποιούνται οι κόμβοι: Server, Client1, Client2, CMN1, CMN2, CMN3 και Core Router. Ο Server αποστέλλει ταυτόχρονα σε κάθε Client όλες τις υπηρεσίες με την εξής σειρά: α) Ροή 1, β) Ροή 2, γ) Ροή 3, αφήνοντας μεταξύ της κάθε Ροής τυχαίο χρονικό διάστημα 1-20 δευτερολέπτων. Παράλληλα, οι μονάδες συλλογής και καταγραφής των CMN1, CMN2 και CMN3 συλλέγουν, καταγράφουν και αποστέλλουν πληροφορίες σχετικά με τις μεταδιδόμενες ροές, στη μονάδα ανάλυσης και επεξεργασίας του Core Router για τη περαιτέρω επεξεργασία και εξαγωγή των αποτελεσμάτων δικτυακής απόδοσης του IDVB-T. Ο σκοπός αυτού του σεναρίου είναι η αξιολόγηση της αποτελεσματικότητας του ημικατανεμημένης παρακολούθησης & ελέγχου δικτυακή απόδοσης σε συνθήκες: α) δυναμικού ρυθμού εισόδου και εξόδου υπηρεσιών και β) υψηλού δικτυακού φόρτου.

4.4 Αποτελέσματα – Γραφικές Παραστάσεις

Για την αποφυγή της παρουσίασης πλεοναζόντων (όμοιων) αποτελεσμάτων και για τα Σενάρια 2-6, η μονόδρομη καθυστέρηση (Delay) και η διακύμανση της μονόδρομης καθυστέρησης (Jitter) παρουσιάζεται μόνο για μία από τις μεταδιδόμενες ροές με τις υπόλοιπες να παρατίθονται στο Παράρτημα. Επίσης πρέπει να αναφερθεί εδώ ότι η ορθότητα των αποτελεσμάτων πιστοποιήθηκε καθώς συγκρίθηκαν με αυτά που εξήγαγε μια εφαρμογής offline ανάλυσης της δικτυακής κίνησης.

4.4.1 Σενάριο 1



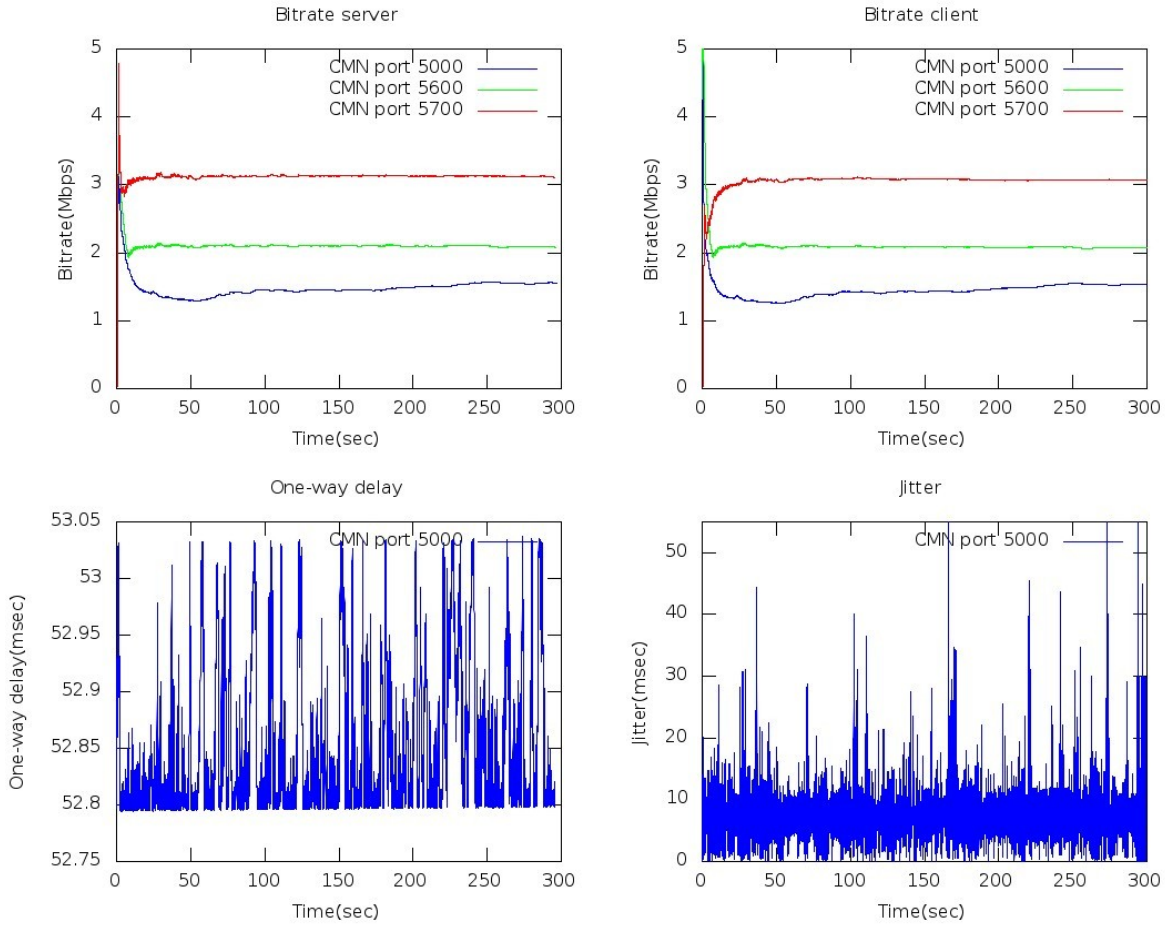
Σχήμα 7: Γραφικές πρώτου σεναρίου: ρυθμός μετάδοσης στον client και στο server, μονόδρομη καθυστέρηση, διακύμανση καθυστέρησης

	ports	**Server Bitrate (Mbps)**	**Client Bitrate (Mbps)**	**Losses (%)**	**Jitter (ms)**	**One-way delay (ms)**
Ροή 1	**5000**	1.8817	1.964	0	6.6554	32.85

Πίνακας 2: Σενάριο 1 : μέσο bitrate client ,μέσο bitrate server, απώλειες πακέτων, μονόδρομη καθυστέρηση, jitter

Τα αποτελέσματα που παρουσιάζονται εδώ υποδεικνύουν ότι ή δικτυακή απόδοση του συστήματος είναι αρκετά καλή. Το αποτέλεσμα αυτό είναι αναμενόμενο καθώς η μεταδιδόμενη υπηρεσία δεν χρειάζεται να ανταγωνιστεί για τους διαθέσιμους δικτυακούς πόρους. Οι μικρές αποκλίσεις στο bitrate του Server και του Client οφείλονται στο ότι ο χρόνος που χρησιμοποιείται για τον υπολογισμό του εν λόγω μέτρου απόδοσης είναι διαφορετικός για τους δύο κόμβους. Η διαφορά αυτή οφείλεται στο jitter που δημιουργείται κατά την είσοδο και έξοδο των πακέτων από τους κόμβους του IDVB-T δικτύου.

4.4.2 Σενάριο 2



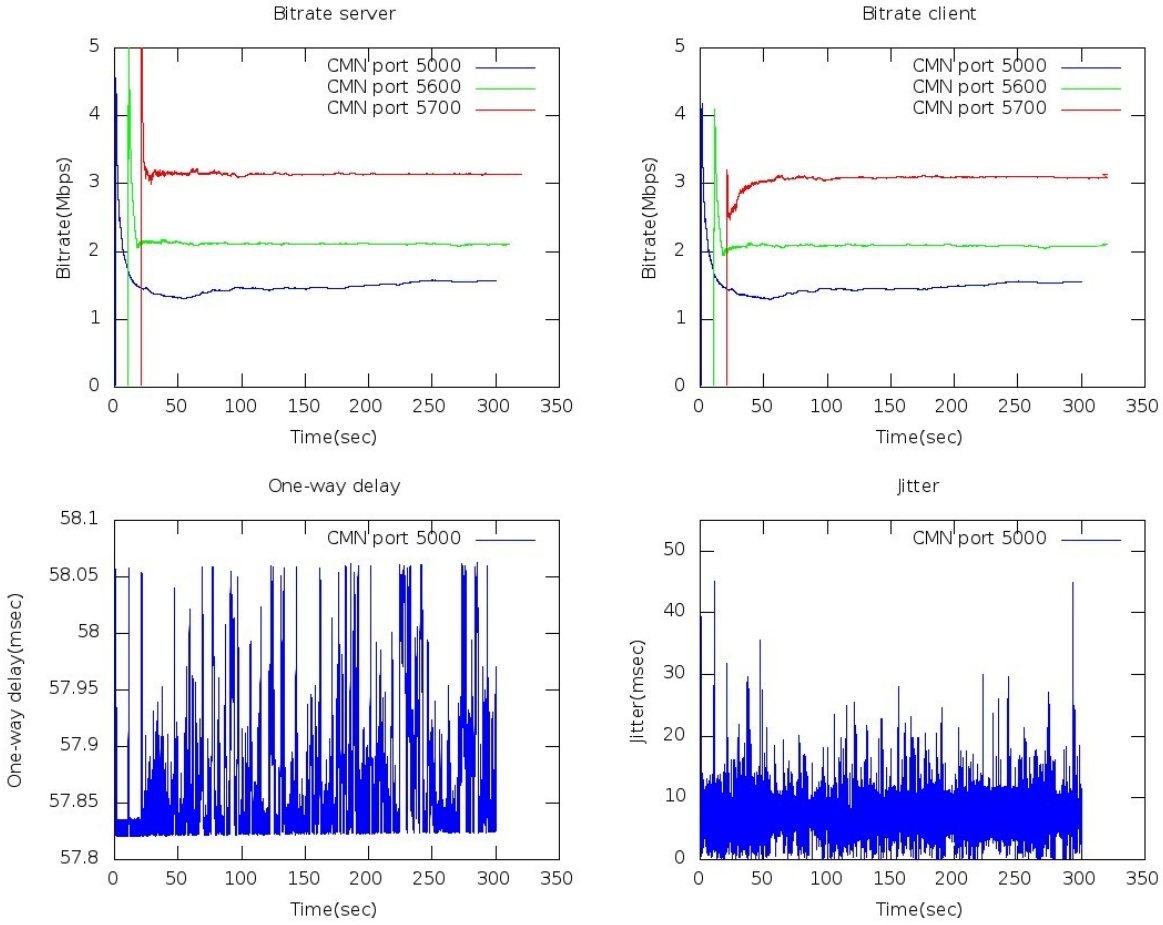
Σχήμα 8: Γραφικές δευτέρου σεναρίου: ρυθμός μετάδοσης στον client και στο server, μονόδρομη καθυστέρηση, διακύμανση καθυστέρησης

	ports	Server Bitrate (Mbps)	Client Bitrate (Mbps)	Losses (%)	Jitter (ms)	One-way delay (ms)
Ροή 1	**5000**	1.8636	1.8502	0	6.3692	52.9
Ροή 2	**5600**	2.6437	2.6611	0	5.5343	52.81
Ροή 3	**5700**	3.9027	3.8147	0	4.6017	52.87

Πίνακας 3: Σενάριο 2 : μέσο bitrate client ,μέσο bitrate server, απώλειες πακέτων, μονόδρομη καθυστέρηση, jitter

Η αυξημένη μονόδρομη καθυστέρηση που παρατηρείται στο παρόν σενάριο οφείλεται στον αυξημένο δικτυακό φόρτο που προκαλεί η μεταφορά τριών υπηρεσιών. Ακόμα και έτσι όμως η δικτυακή απόδοση του IDVB-T δικτύου παραμένει σε ικανοποιητικό επίπεδο, το οποίο είναι και το λογικό καθώς το συνολικά διαθέσιμο εύρος ζώνης του IDVB-T (10 Mbps) δικτύου επαρκεί για την μετάδοση και των τριών υπηρεσιών--και οι τρεις μαζί έχουν bitrate κωδικοποίησης ~6.5 Mbps.

4.4.3 Σενάριο 3



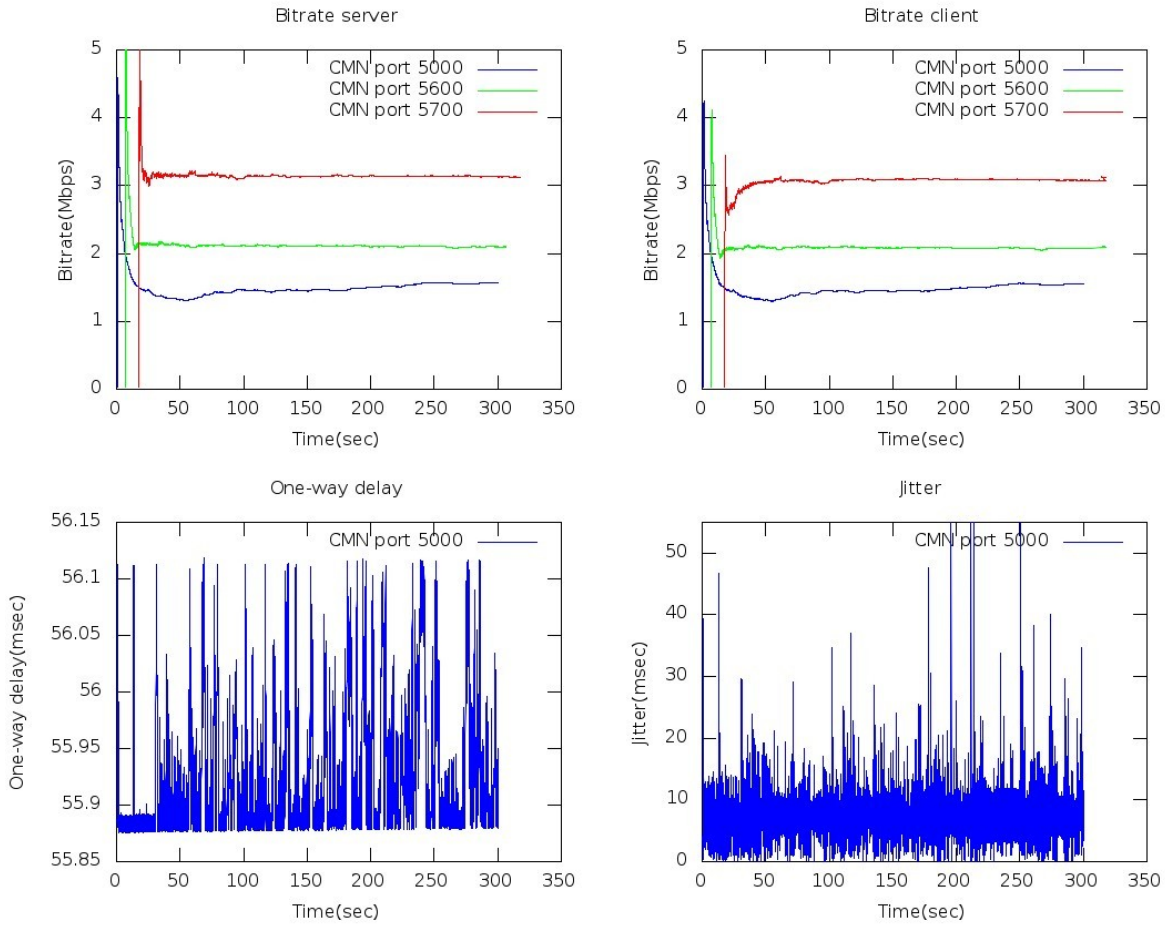
Σχήμα 9: Γραφικές τρίτου σεναρίου: ρυθμός μετάδοσης στον client και στο server, μονόδρομη καθυστέρηση, διακύμανση καθυστέρησης

	ports	Server Bitrate (Bps)	Client Bitrate (Bps)	Losses (%)	Jitter (ms)	One-way delay (ms)
Ροή 1	**5000**	1.8998	1.8797	0	6.3194	57.9
Ροή 2	**5600**	2.6766	2.6236	0	5.5183	57.84
Ροή 3	**5700**	3.9300	3.8360	0	4.6338	57.86

Πίνακας 4: Σενάριο 3 : μέσο bitrate client ,μέσο bitrate server, απώλειες πακέτων, μονόδρομη καθυστέρηση, jitter

Ο σταθερός ρυθμός εισόδου και εξόδου των μεταδιδόμενων ροών δεν επηρεάζει: α) την δικτυακή απόδοση του IDVB-T δικτύου καθώς διαθέσιμο εύρος ζώνης είναι υπεραρκετό για την μεταφορά των υπηρεσιών, και β) την αποτελεσματικότητα του υλοποιημένου συστήματος παρακολούθησης και ελέγχου της δικτυακής απόδοσης καθώς, όπως είναι και το αναμενόμενο, τα αποτελέσματα που εξάγει είναι παρόμοια με αυτά του προηγούμενου σεναρίου (ταυτόχρονη μετάδοση των τριών υπηρεσιών.

4.4.4 Σενάριο 4



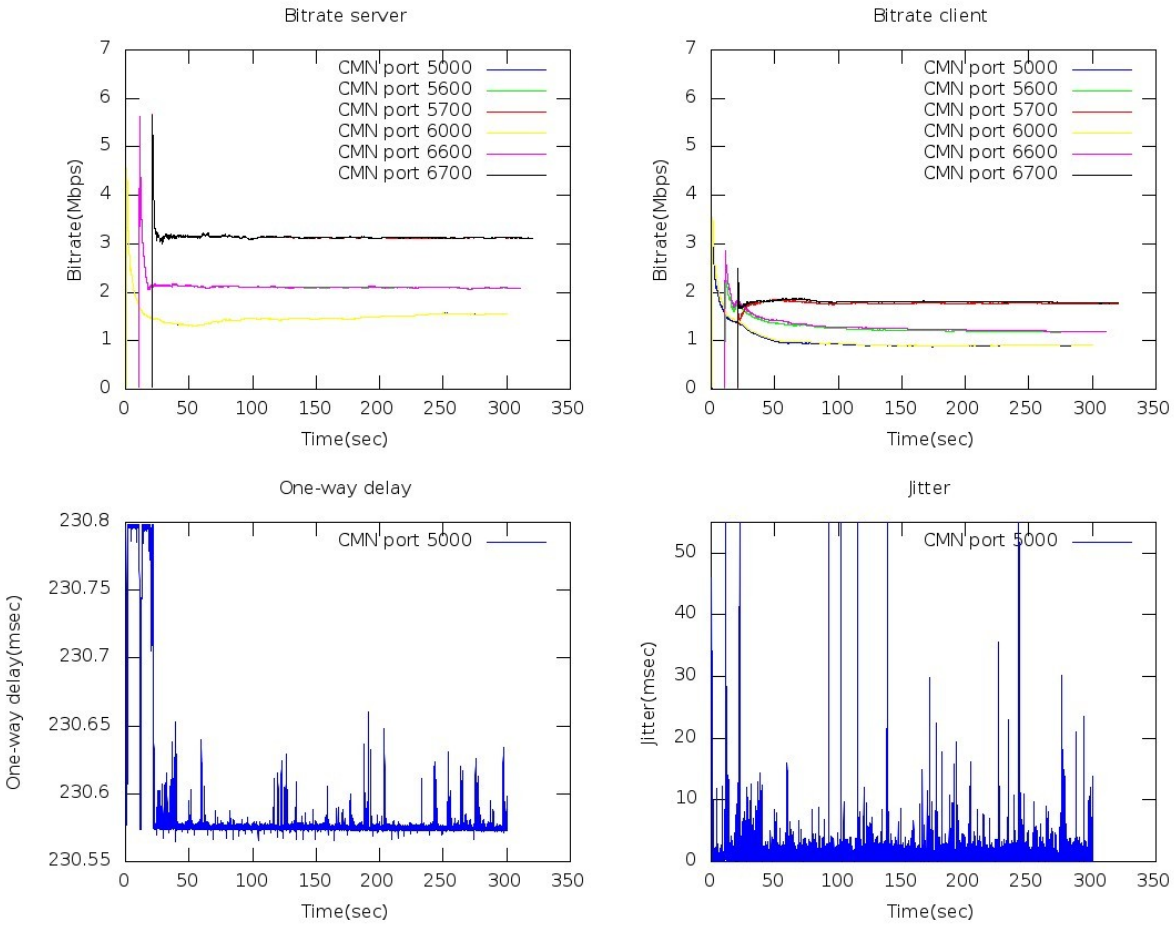
Σχήμα 10: Γραφικές τετάρτου σεναρίου: ρυθμός μετάδοσης στον client και στο server, μονόδρομη καθυστέρηση, διακύμανση καθυστέρησης

	ports	Server Bitrate (Mbps)	Client Bitrate (Mbps)	Losses (%)	Jitter (ms)	One-way delay (ms)
Ροή 1	**5000**	1.9009	1.8816	0	6.3101	55.8
Ροή 2	**5600**	2.6749	2.6206	0	5.5206	55.9
Ροή 3	**5700**	3.9278	3.8352	0	4.6126	56.2

Πίνακας 5: Σενάριο 4 : μέσο bitrate client ,μέσο bitrate server, απώλειες πακέτων, μονόδρομη καθυστέρηση, jitter

Ο τυχαίος ρυθμός εισόδου και εξόδου των μεταδιδόμενων ροών δεν επηρεάζει: α) την δικτυακή απόδοση του IDVB-T δικτύου καθώς διαθέσιμο εύρος ζώνης είναι υπεραρκετό για την μεταφορά των υπηρεσιών, και β) την αποτελεσματικότητα του υλοποιημένου συστήματος παρακολούθησης και ελέγχου της δικτυακής απόδοσης καθώς, όπως είναι και το αναμενόμενο, τα αποτελέσματα που εξάγει είναι παρόμοια με αυτά των προηγούμενων δύο σεναρίων.

4.4.5 Σενάριο 5



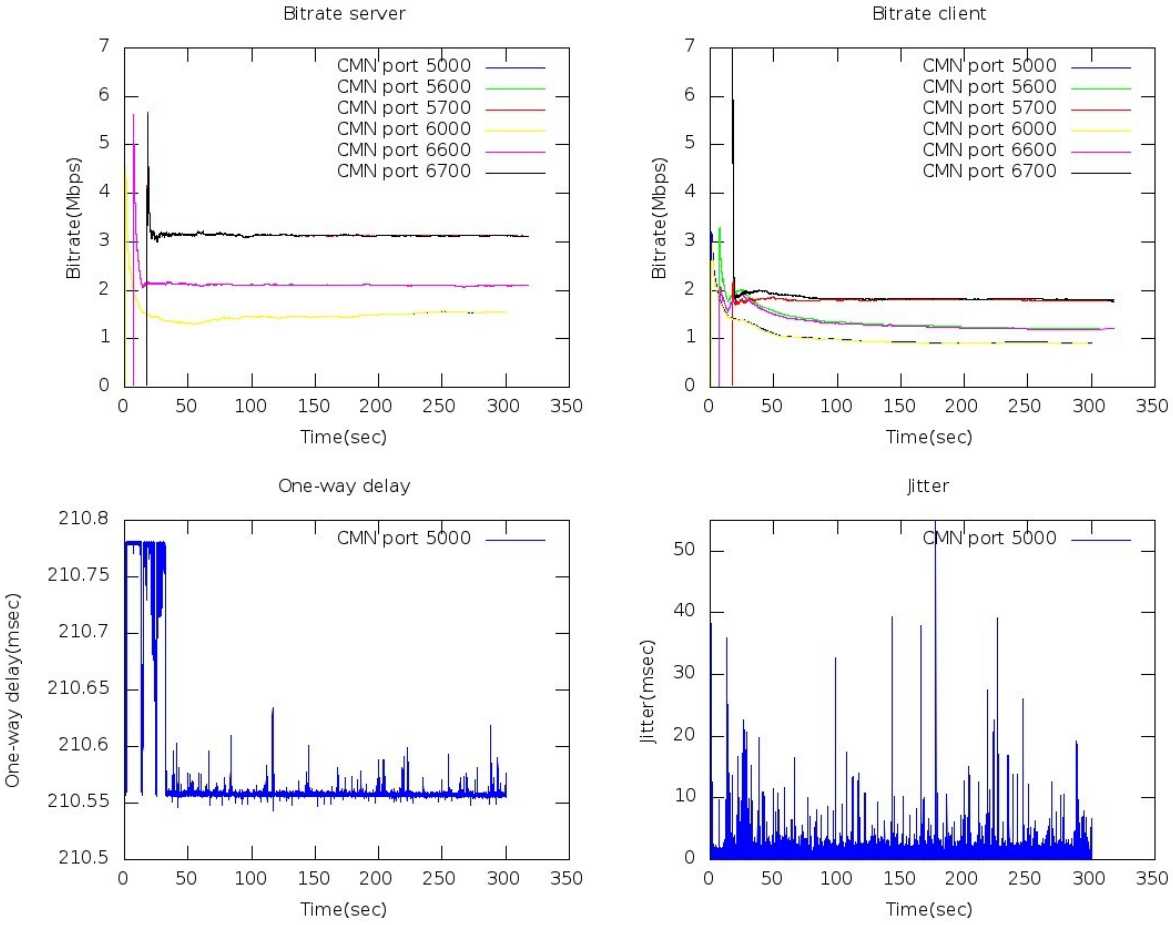
Σχήμα 11: Γραφικές πέμπτου σεναρίου: ρυθμός μετάδοσης στον client και στο server, μονόδρομη καθυστέρηση, διακύμανση καθυστέρησης

	ports	Server Bitrate (MBps)	Client Bitrate (MBps)	Losses (%)	Jitter (ms)	One-way delay (ms)
Ροή 1	**5000**	1.8950	1.2744	2	1.1759	230.60
Ροή 2	**5600**	2.6550	1.5971	3.5	1.0563	230.57
Ροή 3	**5700**	3.9071	2.2010	2.7	9.5201	230.58
Ροή 4	**6000**	1.8951	1.3169	3	5.9312	280.2
Ροή 5	**6600**	2.6687	1.6498	5	5.6778	280.18
Ροή 6	**6700**	3.9147	2.2413	2.2	5.2645	280.21

Πίνακας 6: Σενάριο 5 : μέσο bitrate client ,μέσο bitrate server, απώλειες πακέτων, μονόδρομη καθυστέρηση, jitter

Όπως υποδεικνύουν τα αποτελέσματα--μη μηδενικές απώλειες, μεγάλη μονόδρομη καθυστέρηση, Client bitrate αρκετά μικρότερο του Server bitrate-- η είσοδος ενός νέου κόμβου διανομής και η συνακόλουθη αύξηση των μεταφερόμενων υπηρεσιών οδηγεί σε υποβάθμιση της δικτυακής απόδοσης του IDVB-T δικτύου, ο οποίος προκαλείται από τον μη ελεγχόμενο ανταγωνισμό των υπηρεσιών για τους μη επαρκείς διαθέσιμους δικτυακούς πόρους.

4.4.6 Σενάριο 6



Σχήμα 12: Γραφικές έκτου σεναρίου: ρυθμός μετάδοσης στον client και στο server, μονόδρομη καθυστέρηση, διακύμανση καθυστέρησης

	ports	Server Bitrate (Mbps)	Client Bitrate (Mbps)	Losses (%)	Jitter (ms)	One-way delay (ms)
Ροή 1	**5000**	1.8947	1.3460	3	1.1317	210.59
Ροή 2	**5600**	2.6720	1.7557	2.4	1.0251	210.60
Ροή 3	**5700**	3.9205	2.2444	2.8	1.0819	210.56
Ροή 4	**6000**	1.8953	1.3206	4.7	5.9341	264.27
Ροή 5	**6600**	2.6708	1.6829	2.1	5.6897	265.7
Ροή 6	**6700**	3.9169	2.3129	3.3	5.3458	262.8

Πίνακας 7: Σενάριο 6 : μέσο bitrate client ,μέσο bitrate server, απώλειες πακέτων, μονόδρομη καθυστέρηση, jitter

Ο τυχαίος ρυθμός εισόδου και εξόδου δεν επηρεάζει την αποτελεσματικότητα του υλοποιημένου συστήματος παρακολούθησης και ελέγχου της δικτυακής απόδοσης καθώς, όπως είναι και το αναμενόμενο, τα αποτελέσματα που εξάγει είναι παρόμοια με αυτά του προηγούμενου σεναρίου και υποδεικνύουν ότι υπάρχει υποβάθμιση της δικτυακής απόδοσης.

5 Συμπεράσματα

Σκοπός αυτής της πτυχιακή ήταν ο σχεδιασμός, ανάλυση, υλοποίηση, ενσωμάτωση στο IDVB-T και αξιολόγηση ενός ημικατανεμημένου συστήματος παρακολούθησης και ελέγχου δικτυακής απόδοσης. Η παρεχόμενη από το σύστημα αξιολόγηση της δικτυακής απόδοσης θα επέτρεπε την έγκαιρη--σε περίπτωση υποβαθμισμένης δικτυακής απόδοσης--ενεργοποίηση των μηχανισμών διασφάλισης της παρεχόμενης δικτυακής ποιότητας υπηρεσίας.
Στα πλαίσια αυτά και αξιοποιώντας παθητικούς τρόπους παρακολούθησης (passive monitoring), υλοποιήθηκαν σχεδιάστηκαν και υλοποιήθηκαν μια μονάδα συλλογής και καταγραφής δικτυακής κίνησης και μια μονάδα ανάλυσης και παρουσίασης των αποτελεσμάτων, οι οποίες και ενσωματώθηκαν στους ενδιάμεσους κόμβους διανομής και σε ένα κόμβο της DVB-T πλατφόρμας αντίστοιχα. Η παθητική παρακολούθηση δεν επιβαρύνει με επιπλέον κίνηση το IDVB-T δικτύο, ενώ η ημιπεριφερειακή λειτουργία του συστήματος αποφορτίζει τη DVB-T πλατφόρμα διαμοιράζοντας μέρος του φόρτου στους ενδιάμεσους κόμβους διανομής, αποφεύγοντας ταυτόχρονα την επικοινωνιακή επιβάρυνση που θα προέκυπτε από μια πλήρως κατανεμημένη λειτουργία. Στα πλαίσια της αξιολόγησης του υλοποιημένου συστήματος σχεδιάστηκαν, υλοποιήθηκαν και εκτελέστηκαν μια σειρά από πειράματα σε ένα πραγματικό IDVB-T δίκτυο, τα οποία απέδειξαν την αποτελεσματικότητα--ακόμα και σε καταστάσεις απρόβλεπτης και υψηλής διακύμανσης του δικτυακού φόρτου--του προτεινόμενου συστήματος.

Σαν μελλοντική έρευνα σκοπεύουμε την ενσωμάτωση στο σύστημα μας κριτηρίων αξιολόγησης της ποιότητας εμπειρίας (Quality of Experience-QoE) των τελικών χρηστών του IDVB-T δικτύου. Επίσης θα κινηθούμε προς την κατεύθυνση της δημιουργίας διαλειτουργικών διεπαφών με συστήματα διαχείρισης των δικτυακών πόρων, ώστε αυτά βασισμένα στις πληροφορίες που εξάγει το σύστημα μας να προλαμβάνουν/αντιδρούν έγκαιρα σε περιπτώσεις δικτυακής υποβάθμισης.

6 Βιβλιογραφία

[1] ETS 300 744: Digital Video Broadcasting (DVB): Framing structure, channel coding and modulation for Digital Terrestrial Television (DVB-T), ETSI, 1997

[2] "ETSI, “ETSI EN 301 958 v.1.1.1 Digital Video Broadcasting (DVB): Interaction Channel for Digital Terrestrial Television (RCT) Incorporating Multiple Access OFDM,” 2002."

[3] ETS 300 801 — Digital Video Broadcasting (DVB), “Interaction channel through Public Switched Telecommunications Network (PSTN) / Integrated Services Digital Networks,” (ISDN)."

[4] "G. Xilouris et al., “Reverse Path Technologies in Interactive DVB-T Broadcasting,” Proc. IST Mobile and Wireless Telecommun. Summit, Thessaloniki, Greece, June 2002, pp. 292–95."

[4] "Digital Switchover: Developing Infrastructures for Broadband Access, 6th Information Society Technologies, 6th Framework Programme (ATHENA FP6-507312), (http:www.ist-athena.org)."

[5] "E. Pallis, “Digital Switchover in UHF: the ATHENA Concept for Broadband Access” European Transactions on Telecommunications, vol. 17, no. 2, March 2006, pp. 175–182"

[6] "G. Mastorakis, G. Kormentzas, E. Pallis, A Fusion IP/DVB Networking Environment for Providing Always-On Connectivity and Triple-Play Services to Urban and Rural Areas, IEEE Network Magazine, vol. 21, no. 2, March-April 2007, pp. 21-27"

[7]DVB, http://www.dvb.org/ ,(τελευταία προσπέλαση 5/8/2013)

[8]DVB-T, http://en.wikipedia.org/wiki/DVB-T ,(τελευταία προσπέλαση 7/8/2013)

[9]MPEG, http://en.wikipedia.org/wiki/Category:MPEG ,(τελευταία προσπέλαση 15/8/2013)

[10]Network Monitoring White Paper , ImageStream Internet Solutions, Inc.

[11]Klaus Mochalski and Klaus Irmscher, ”On the Use of Passive Network Measurementsfor Modeling the Internet”

[12]Active vs Passive moritoring http://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html(τελευταία προσπέλαση 11/8/2013)

[13]R. S. Prasad , M. Murray , C. Dovrolis , K. Claffy , “Bandwidth estimation: metrics, measurement techniques, and tools ”

[14]Makoto Aoki, Eiji Oki, and Roberto Rojas-Cessa ,”Measurement Scheme for One-Way Delay Variation with Detection and Removal of Clock Skew ”

[15]SevOne whitepaper, “A Guide to Ensuring Perfect VoIP Calls ”

[16]Broadcom , ”Improving the Quality of Communications with Packet Loss Concealment ”

[17]Costas Courcoubetis and Vasilios A. Siris University of Crete and Institute of Computer Science, “Measurement and analysis of real network traffic”

[18]XML http://www.w3schools.com/xml/xml_whatis.asp , (τελευταία προσπέλαση 13/8/2013)

[19]libavcodec http://ffmpeg.org/doxygen/trunk/libavcodec_2avcodec_8h_source.html ,(τελευταία προσπέλαση 11/8/2013)

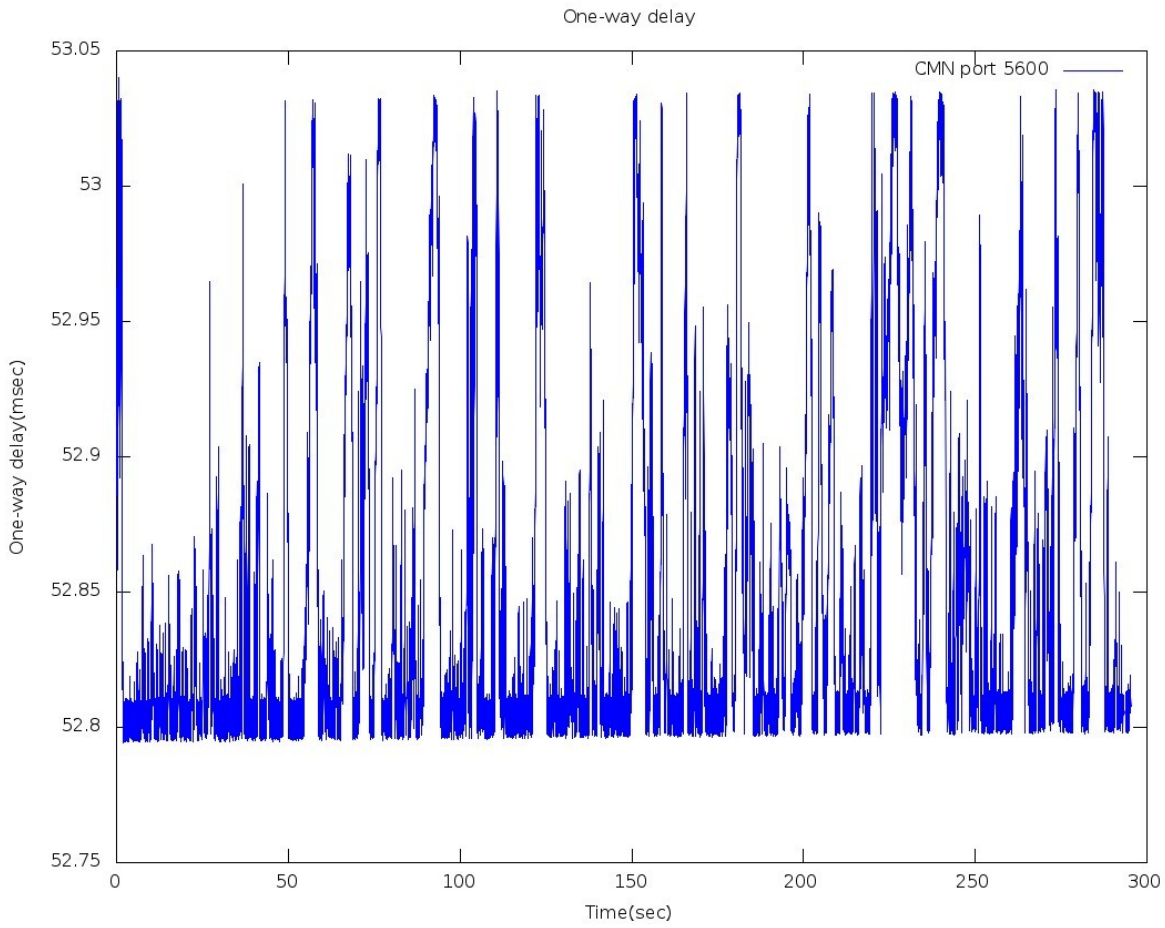
[20]libavformat http://ffmpeg.org/doxygen/trunk/libavformat_2version_8h_source.html ,(τελευταία προσπέλαση 13/8/2013)

[21]VLC media http://en.wikipedia.org/wiki/VLC_media_player ,(τελευταία προσπέλαση 7/8/2013)

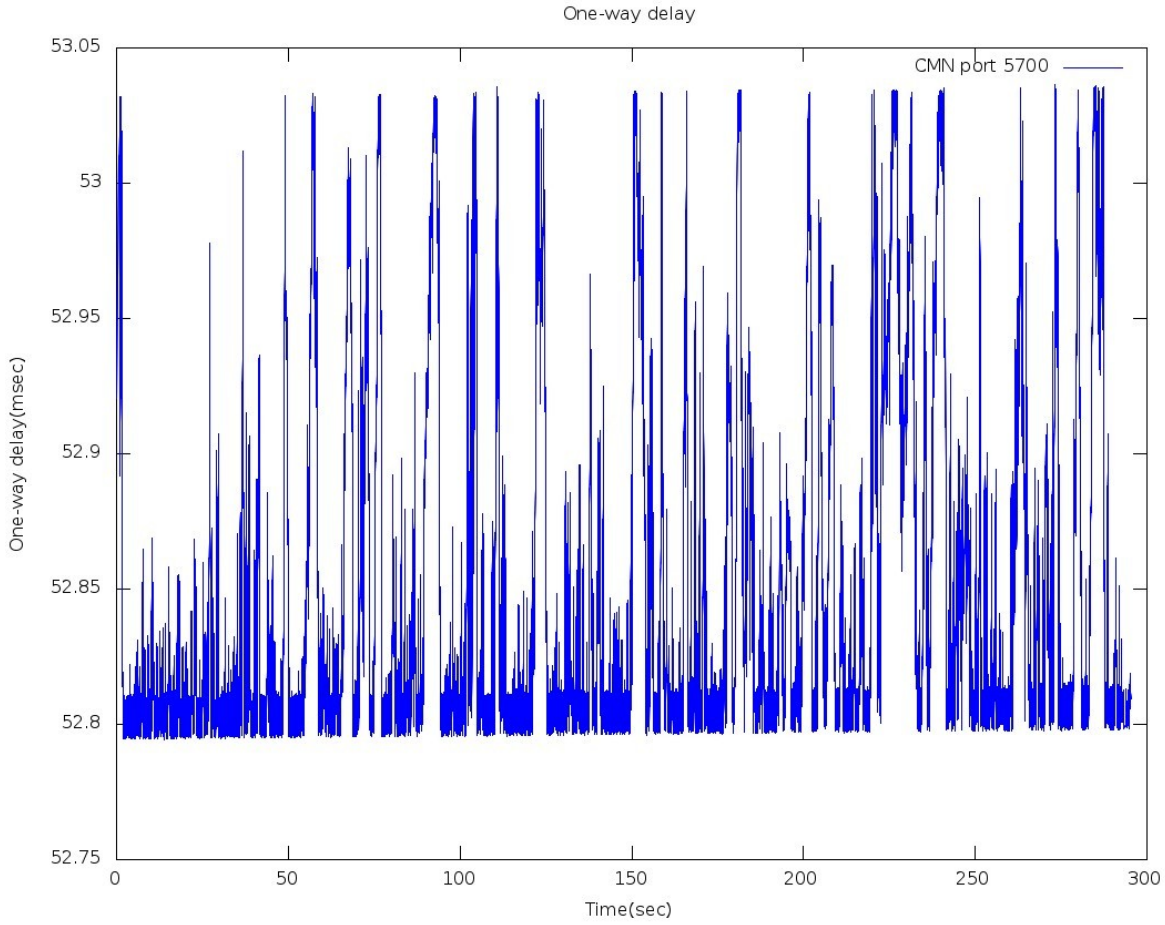
[22]GNU octave http://www.gnu.org/software/octave/ , (τελευταία προσπέλαση 22/8/2013

7 Παράρτημα

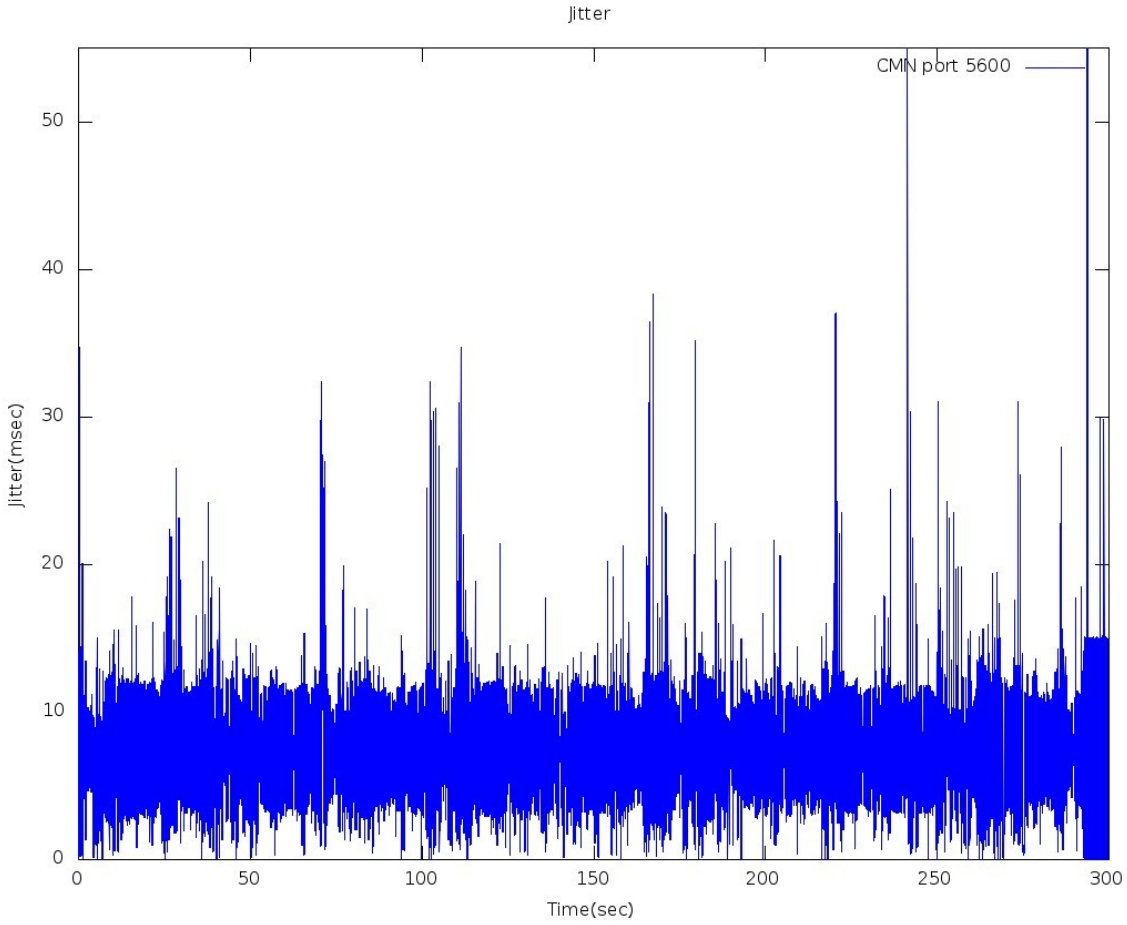
7.1.1 Γραφικές δευτέρου σεναρίου



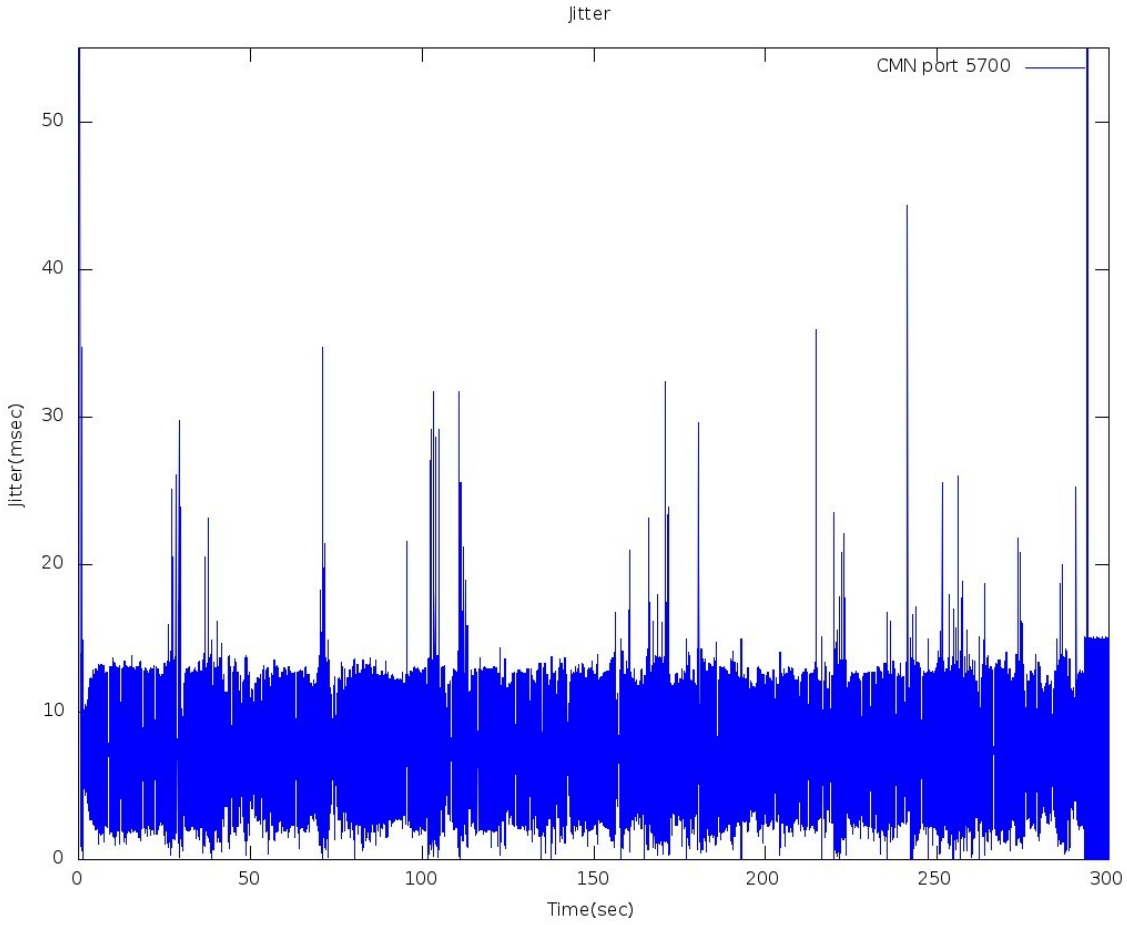
Σχήμα 13: Γραφική δευτέρου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5600



Σχήμα 14: Γραφική δευτέρου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5700

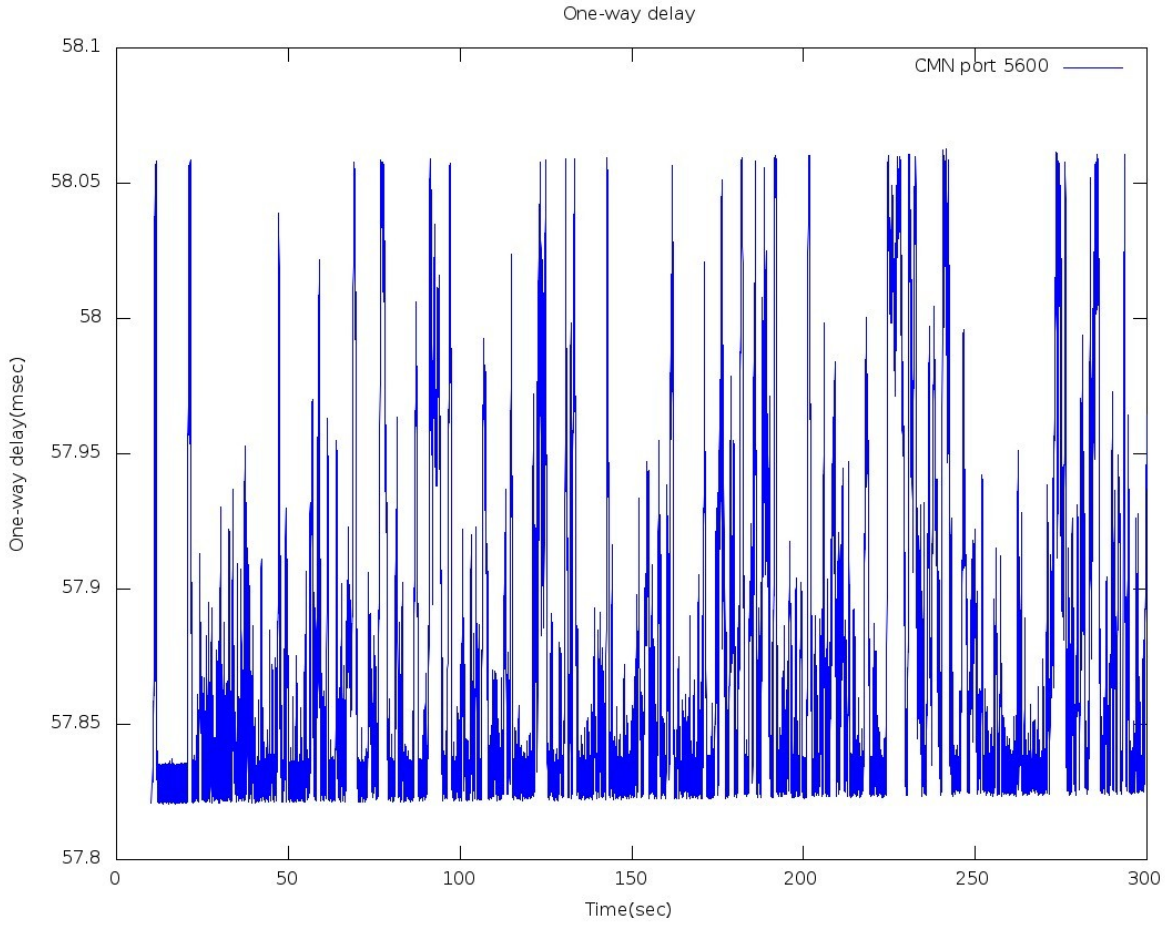


Σχήμα 15: Γραφική δευτέρου σεναρίου: Διακύμανση καθυστέρησης της ροής 5600

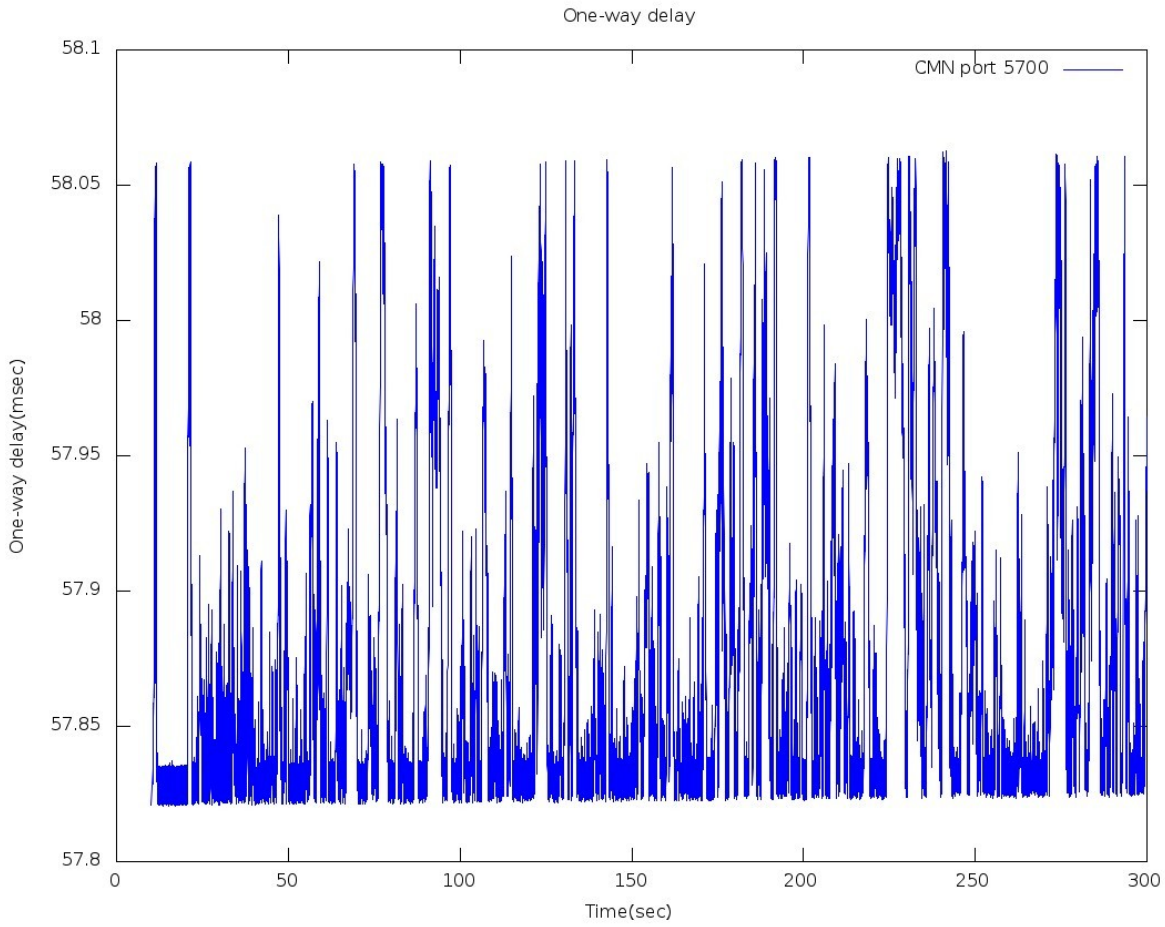


Σχήμα 16: Γραφική δευτέρου σεναρίου: Διακύμανση καθυστέρησης της ροής 5700

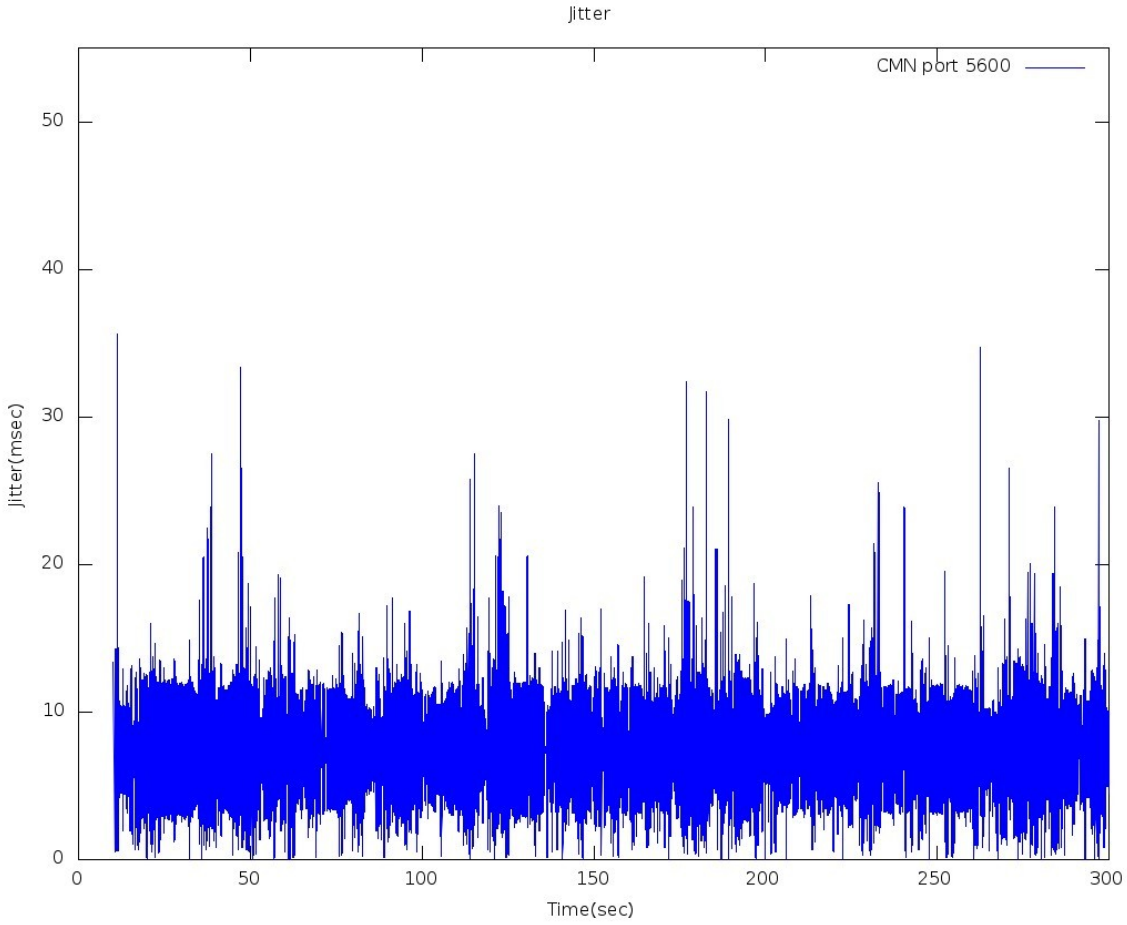
7.1.2 Γραφικές τρίτου σεναρίου



Σχήμα 17: Γραφική τρίτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5600

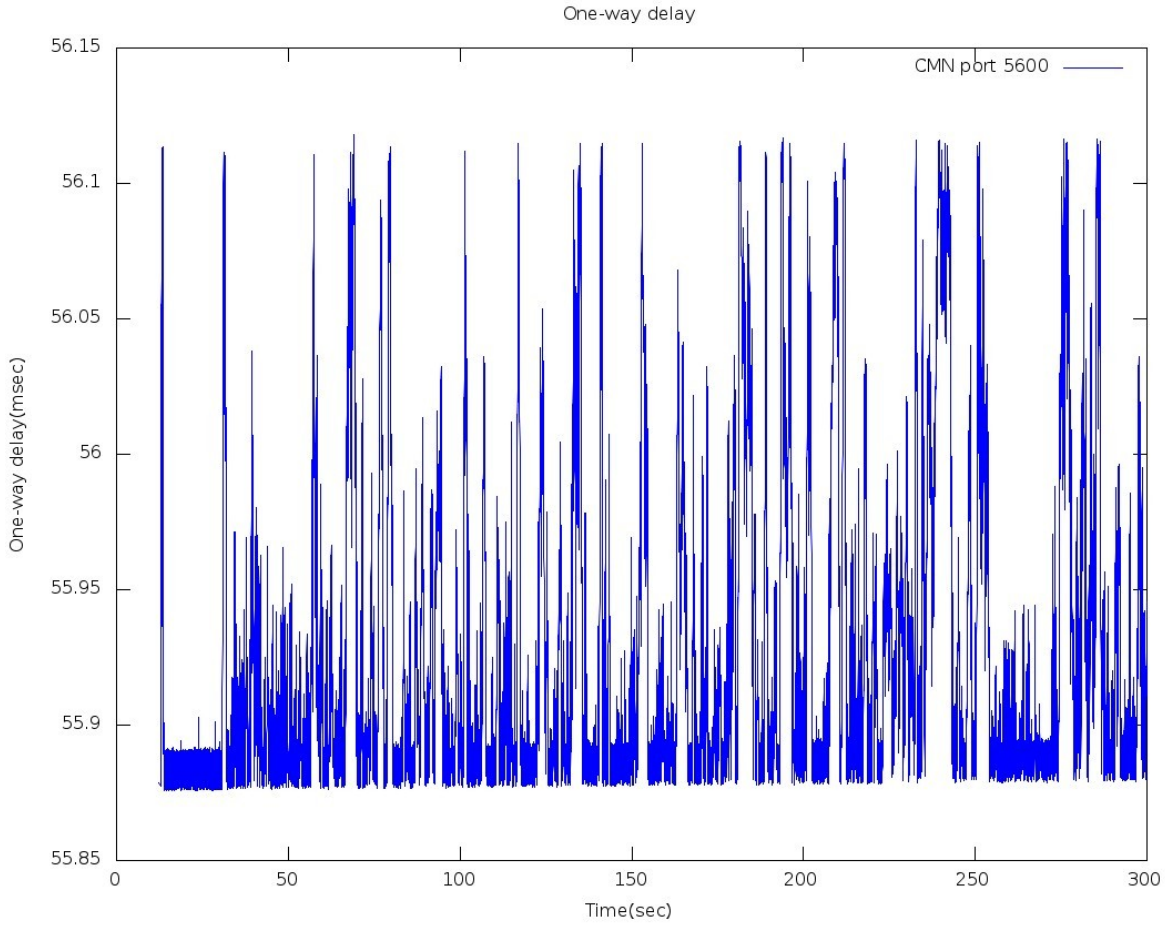


Σχήμα 18: Γραφική τρίτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5700

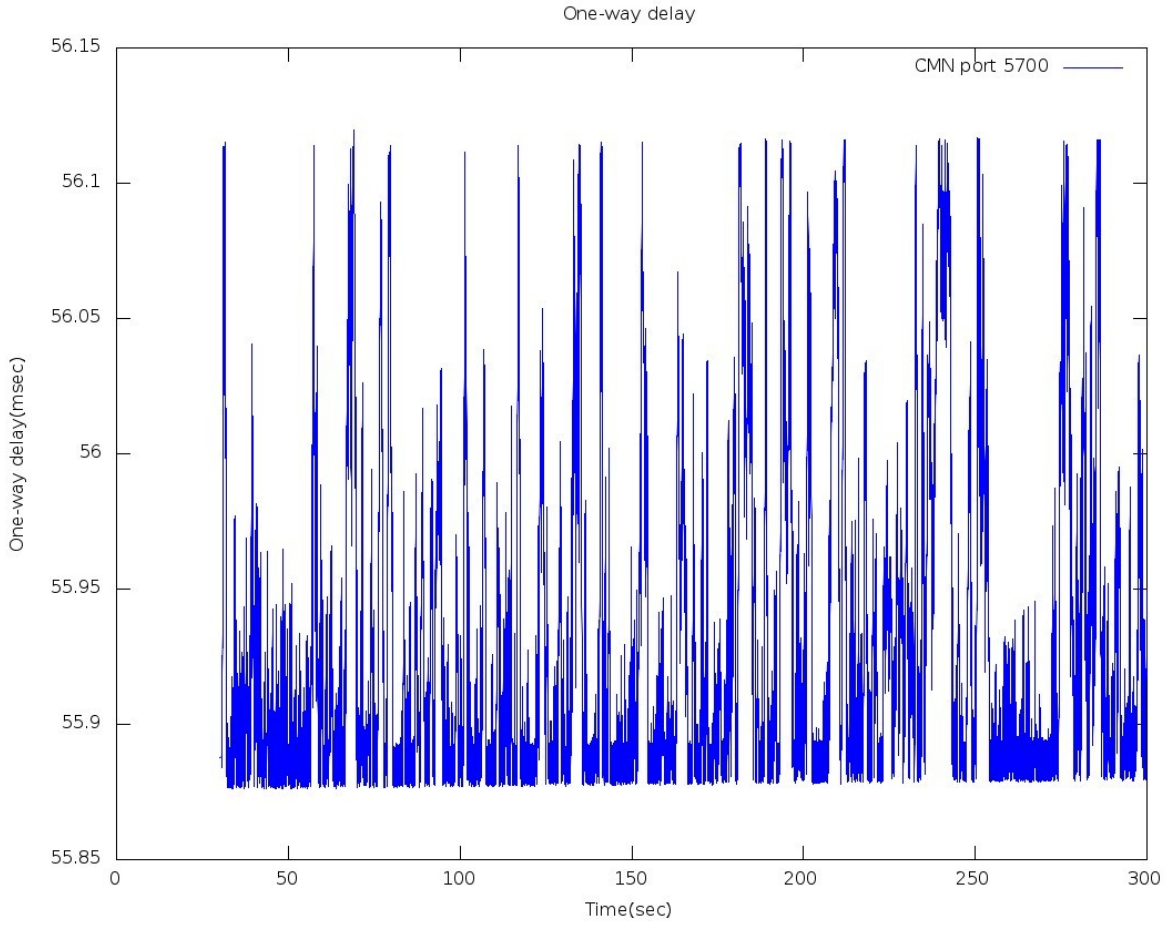


Σχήμα 19: Γραφική τρίτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5600

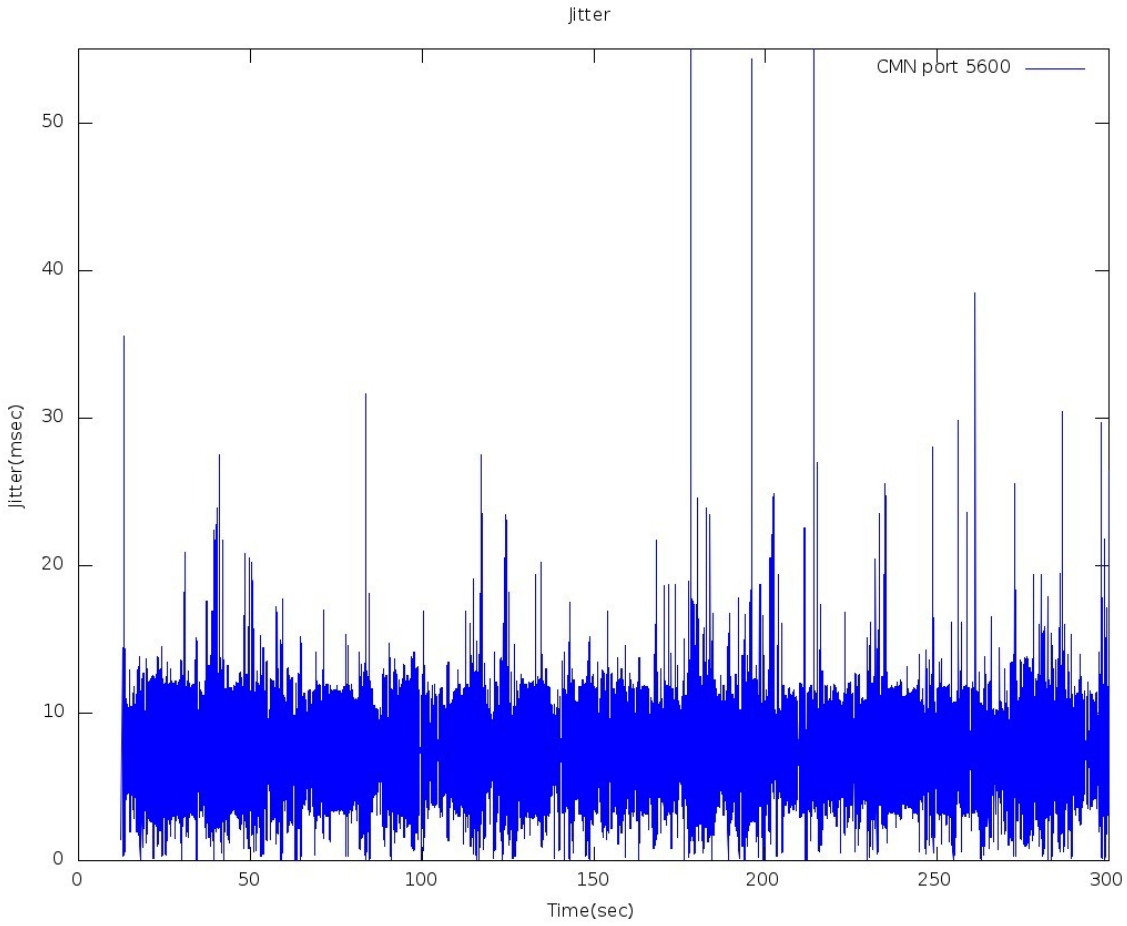
7.1.3 Γραφικές τετάρτου σεναρίου



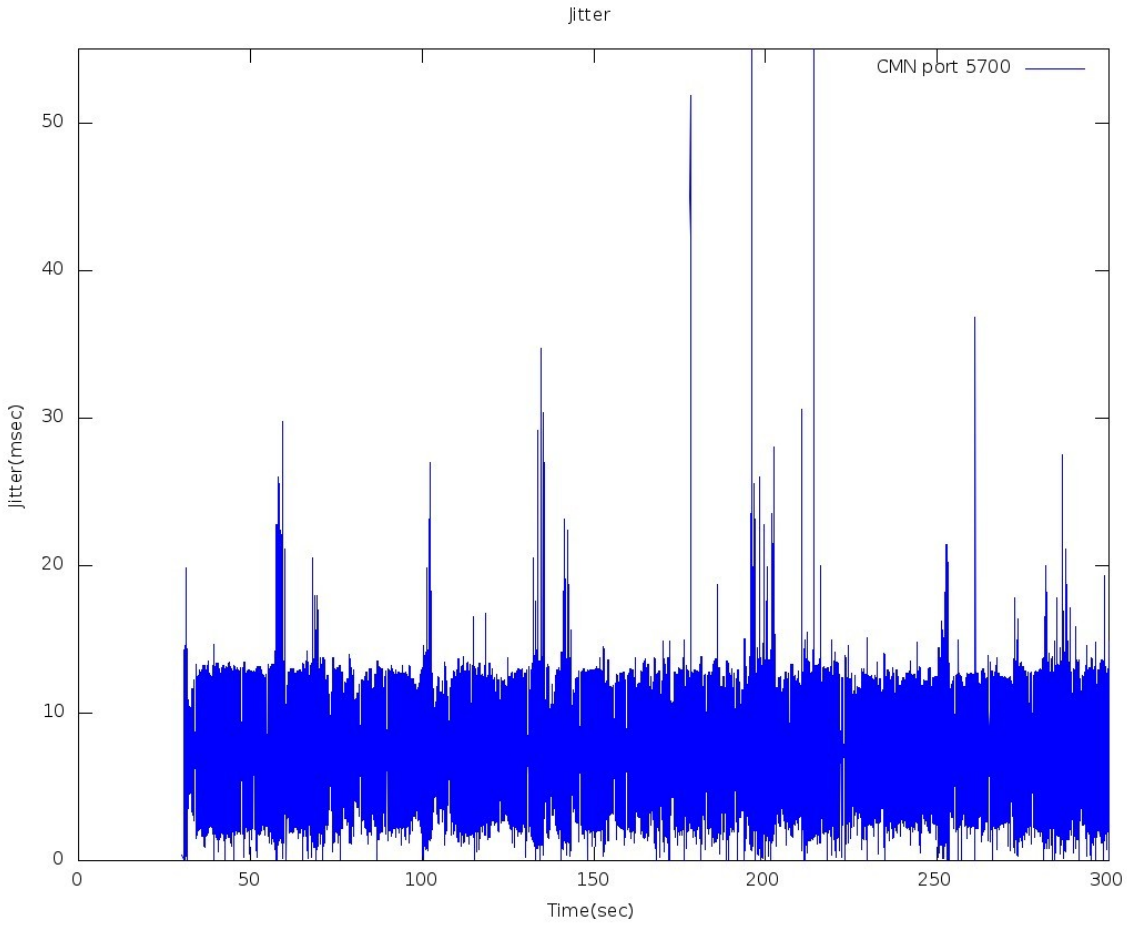
Σχήμα 20: Γραφική τετάρτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5600



Σχήμα 21: Γραφική τετάρτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5700

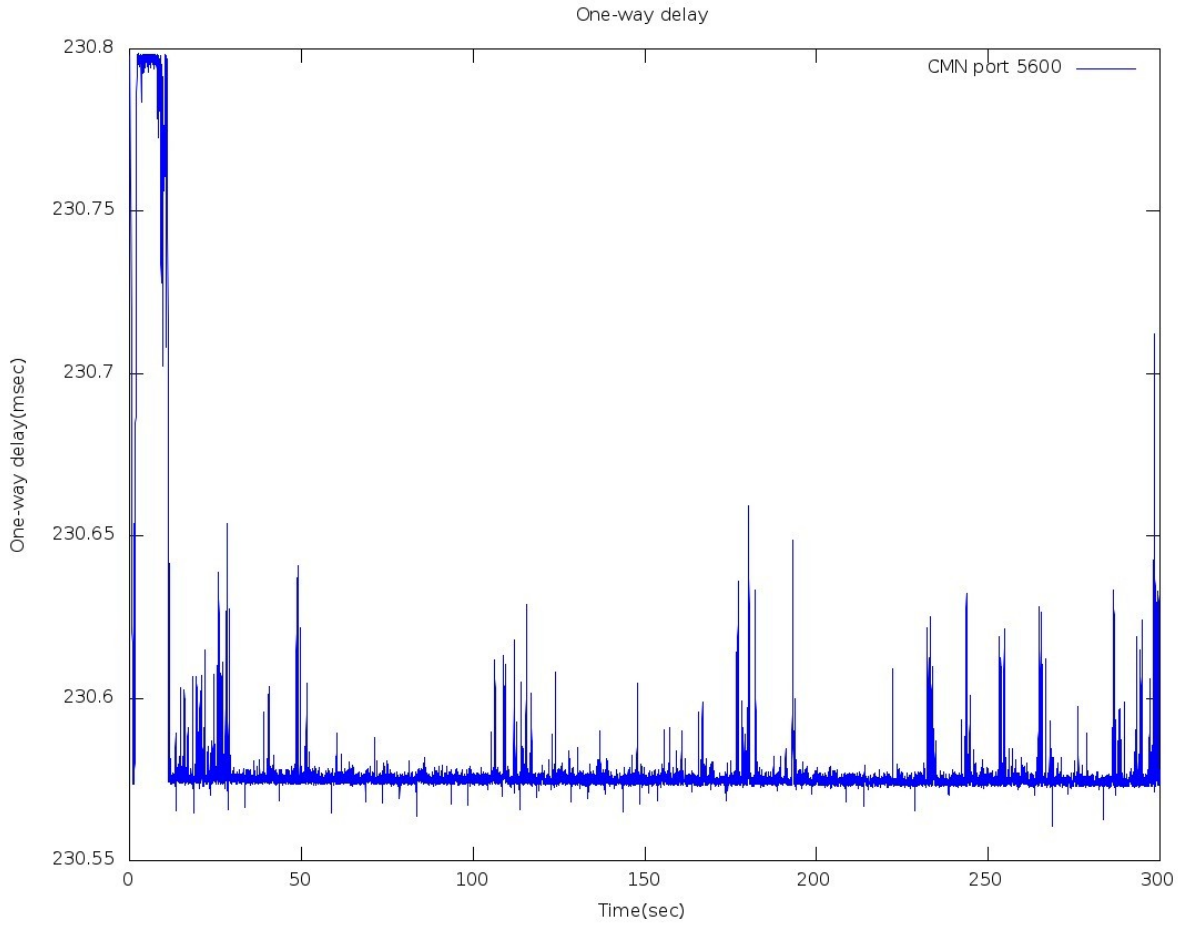


Σχήμα 22: Γραφική τετάρτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5600

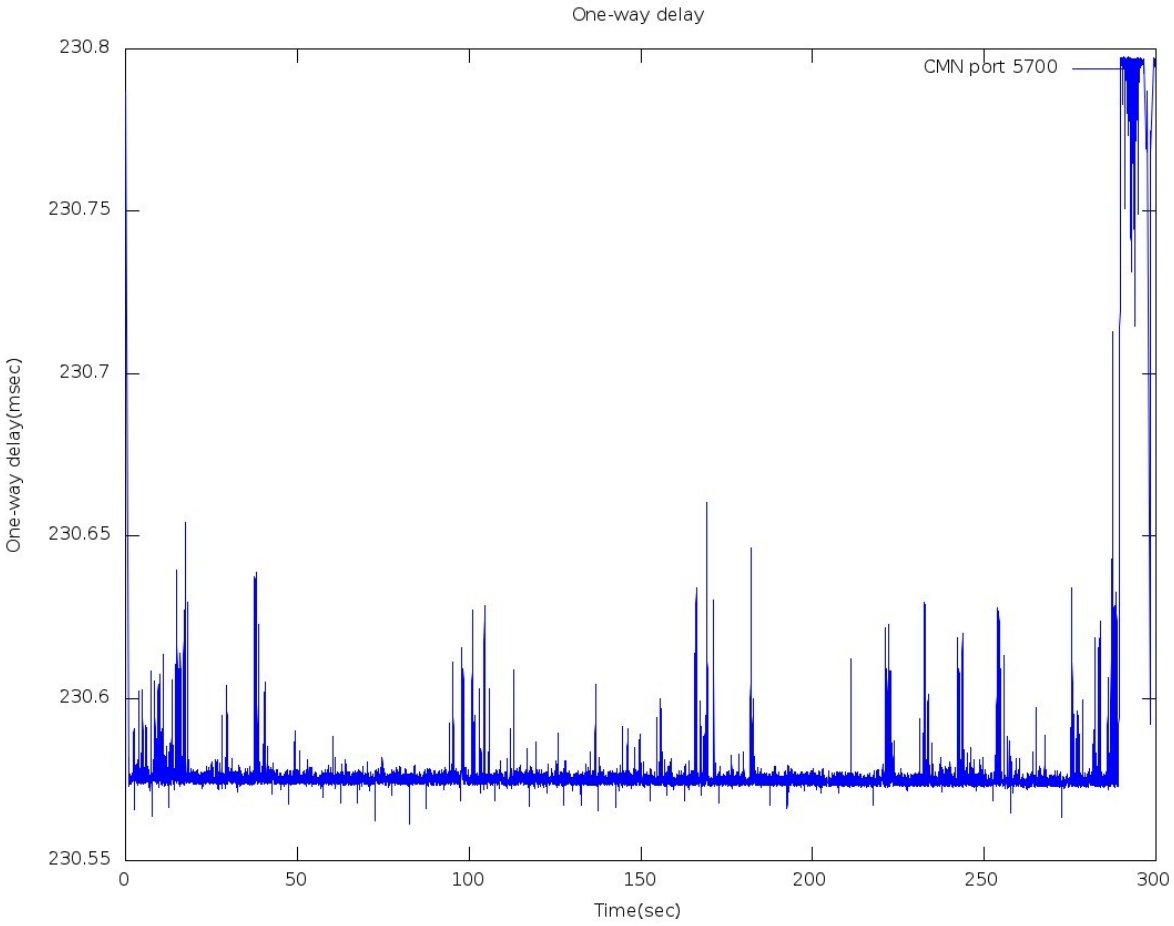


Σχήμα 23: Γραφική τετάρτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5700

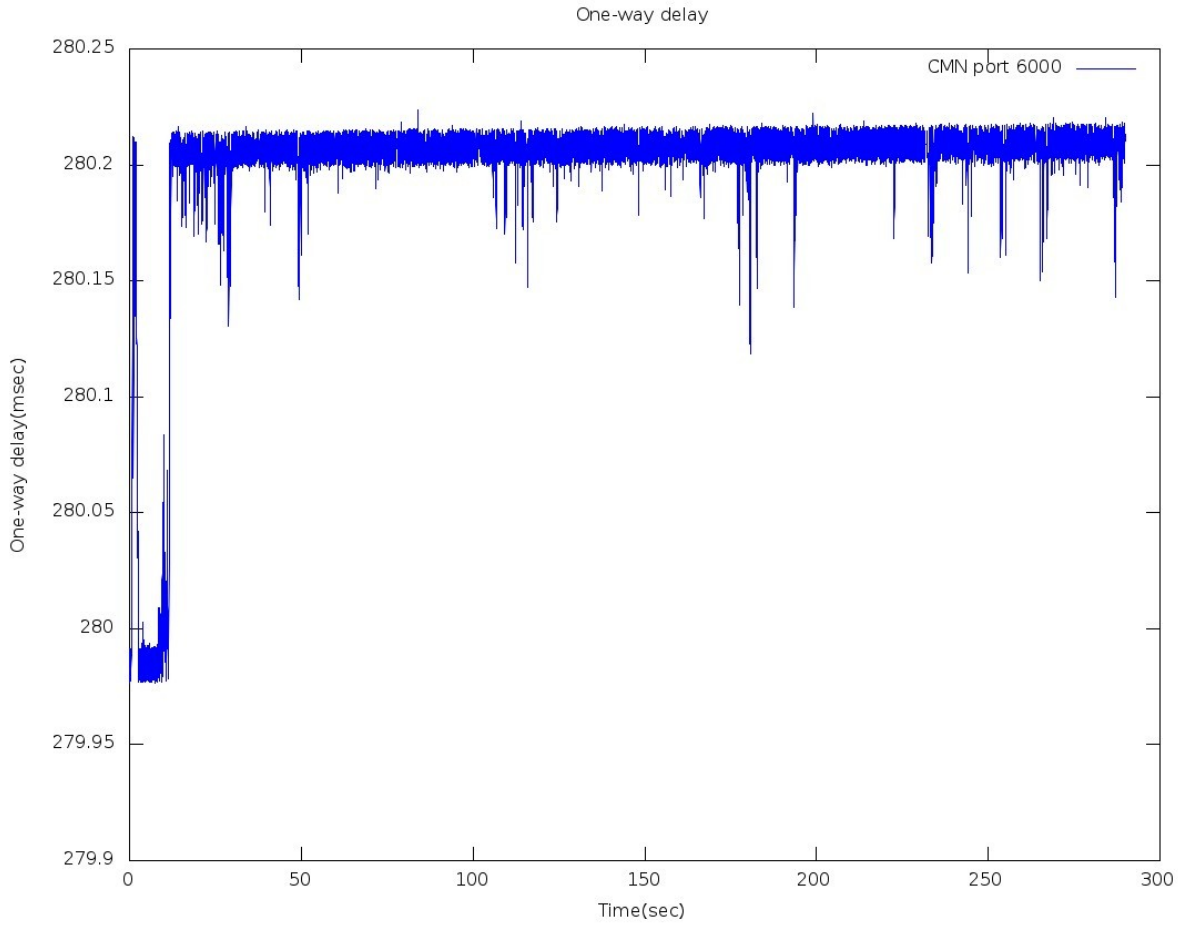
7.1.4 Γραφικές πέμπτου σεναρίου



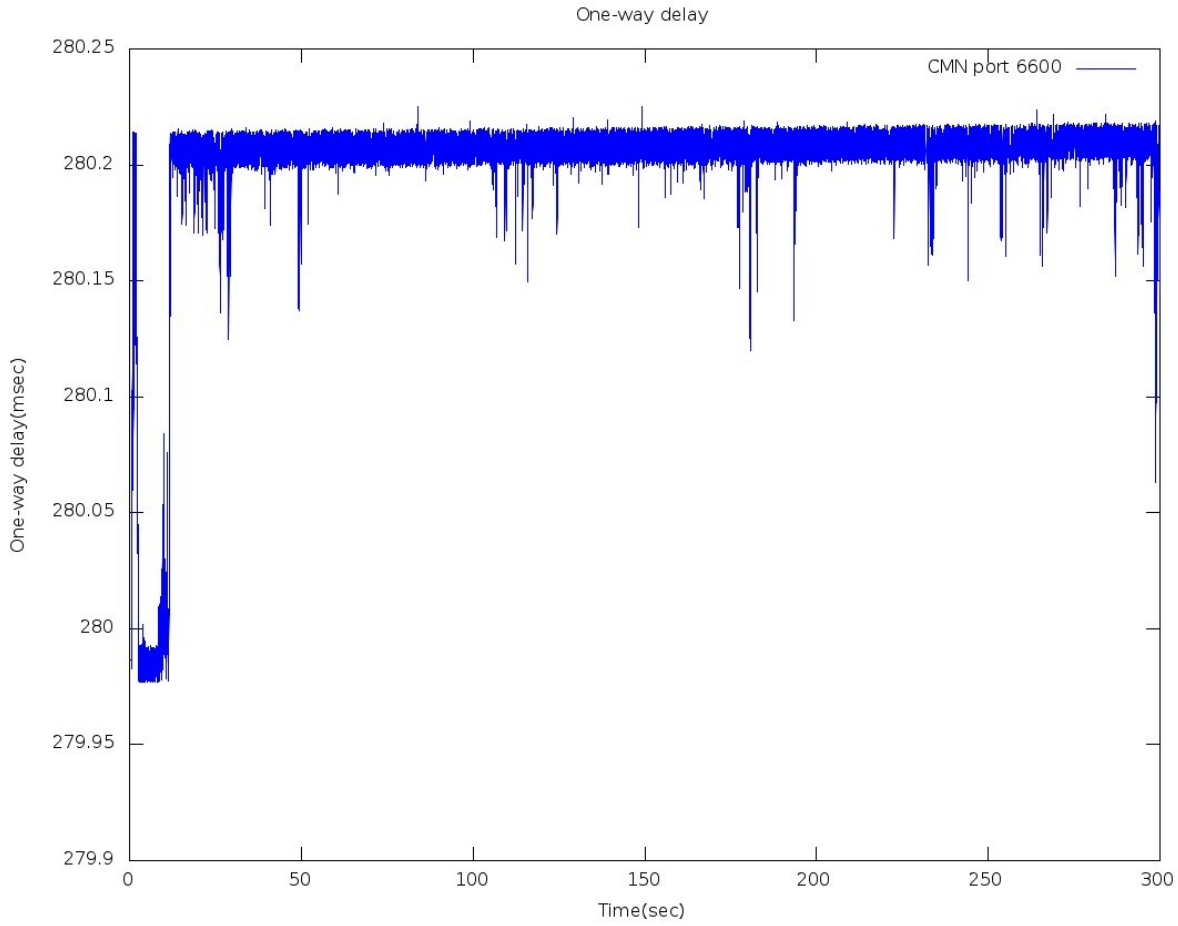
Σχήμα 24: Γραφική πέμπτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5600



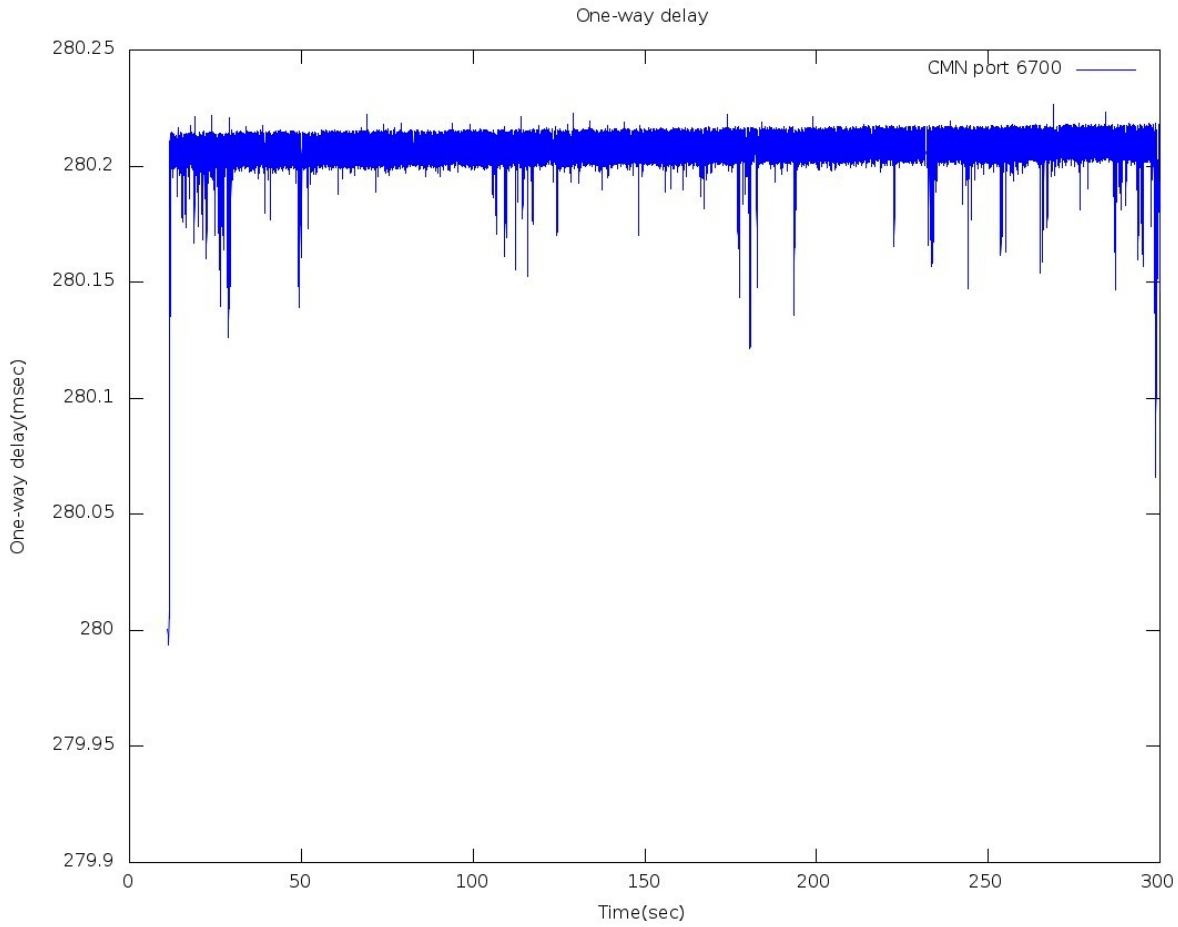
Σχήμα 25: Γραφική πέμπτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5700



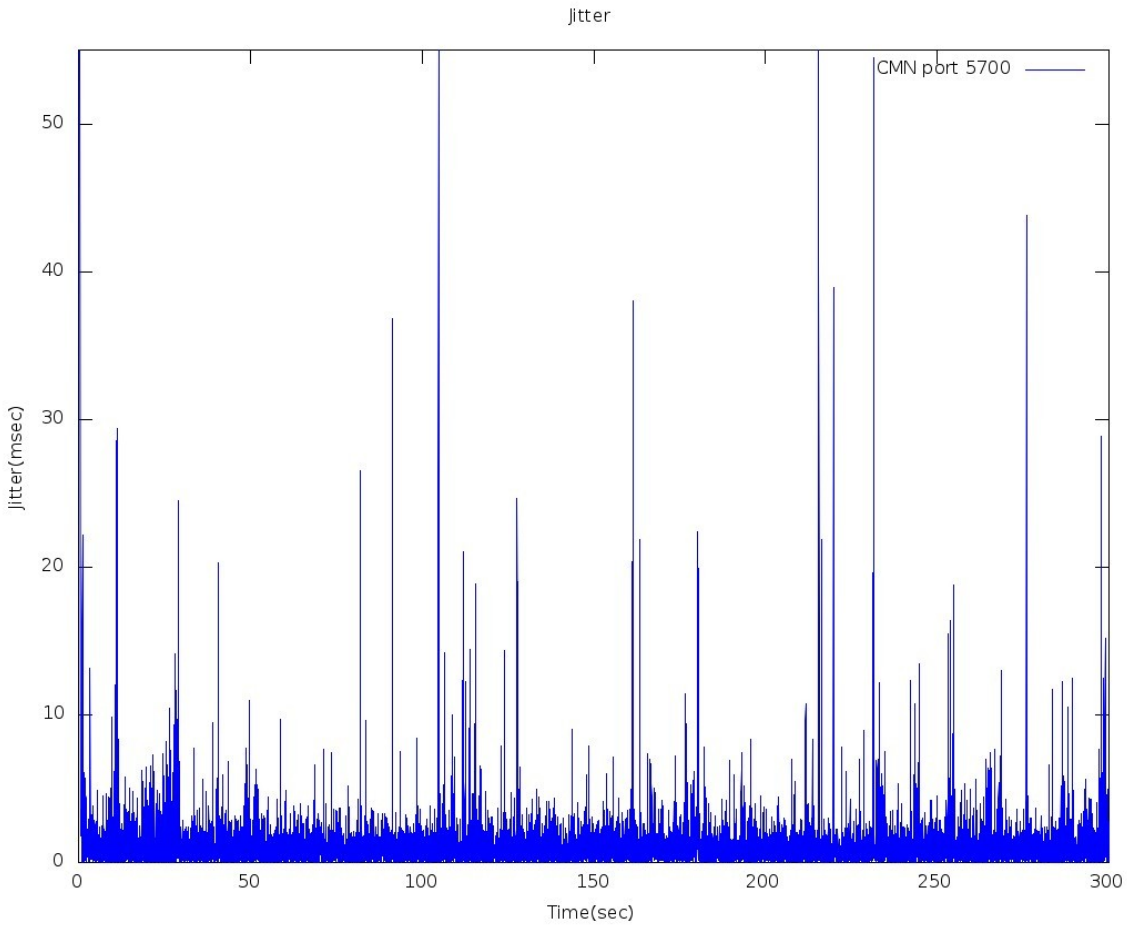
Σχήμα 26: Γραφική πέμπτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 6000



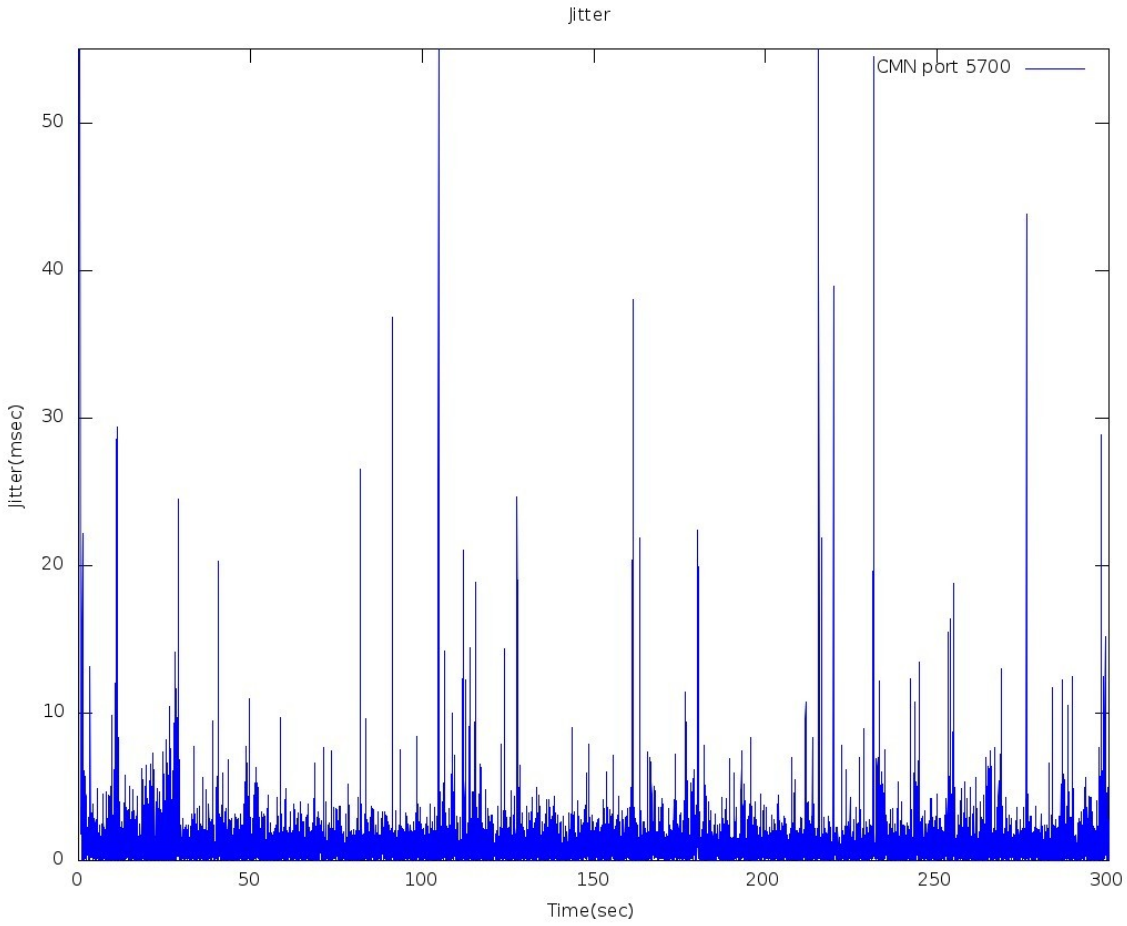
Σχήμα 27: Γραφική πέμπτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 6600



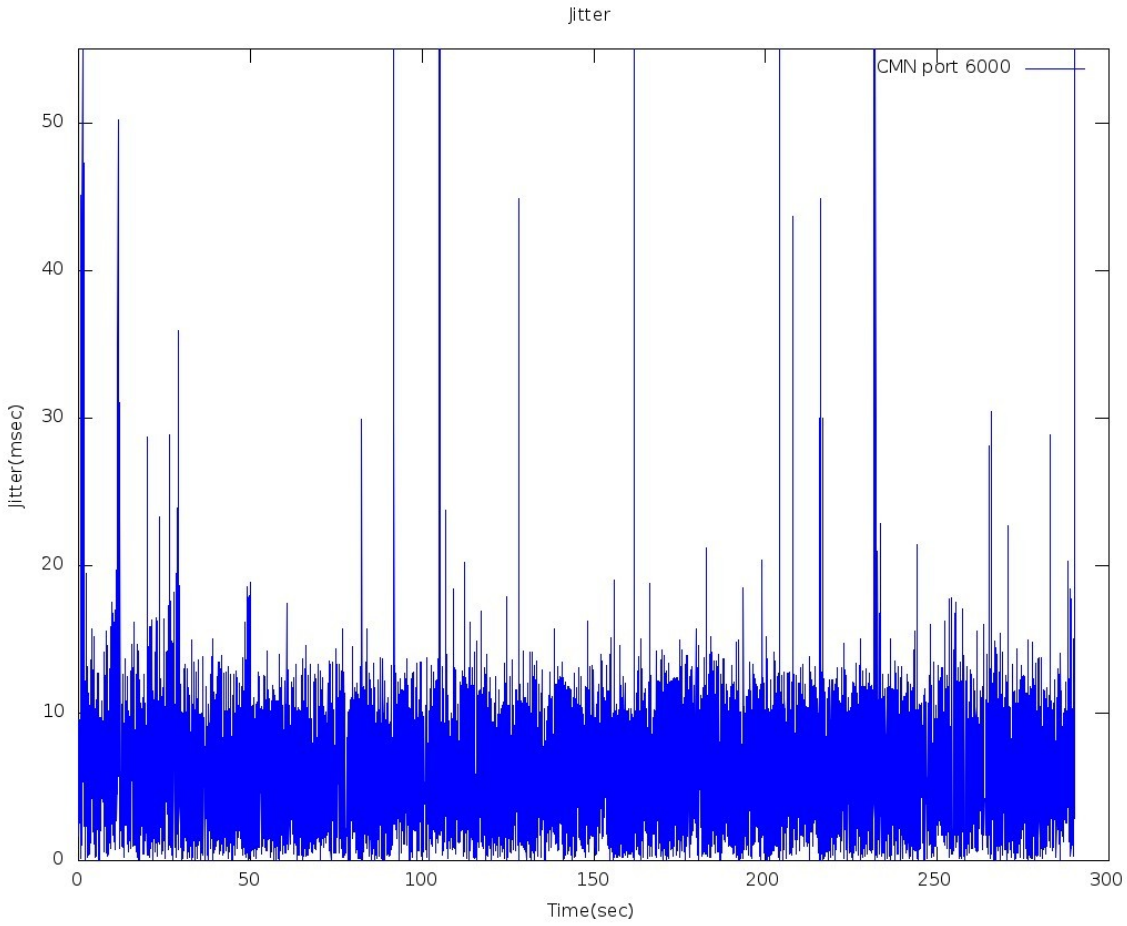
Σχήμα 28: Γραφική πέμπτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 6700



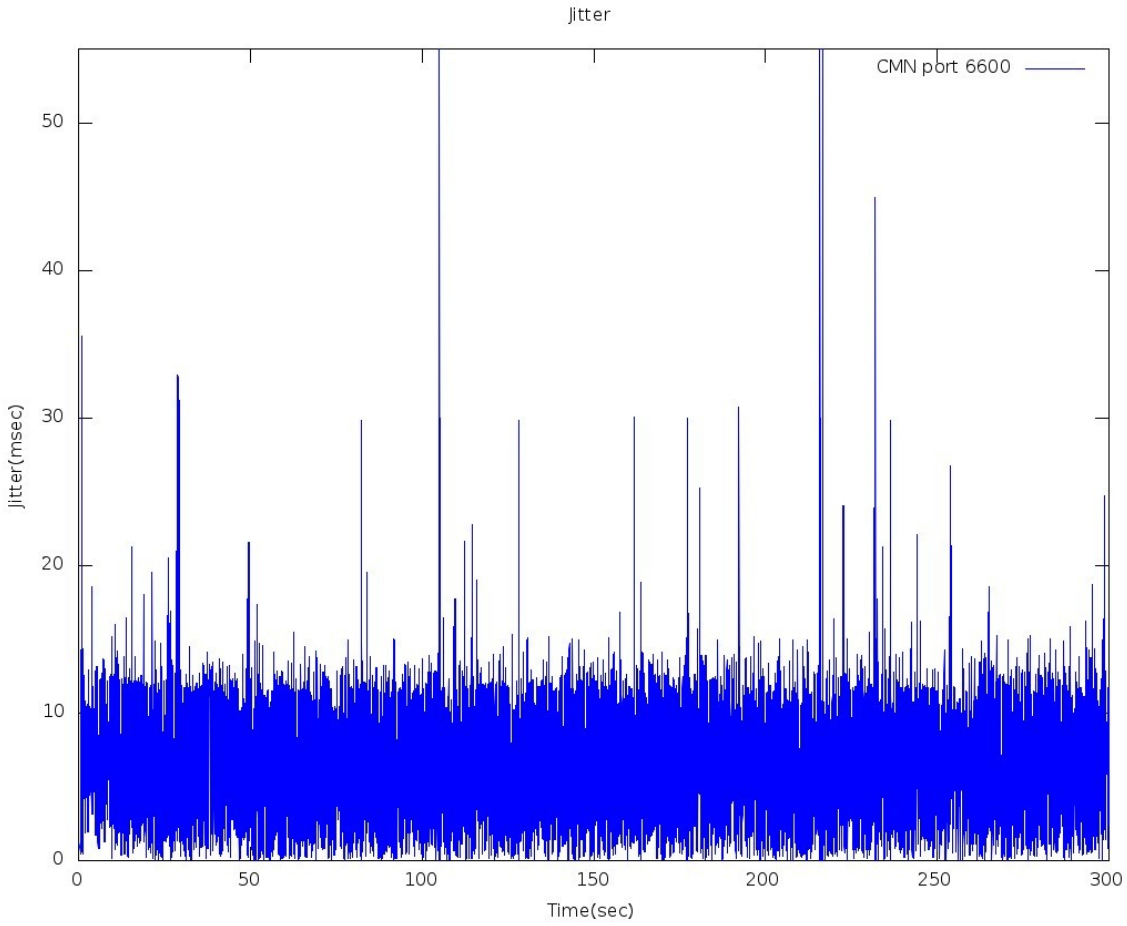
Σχήμα 29: Γραφική πέμπτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5600



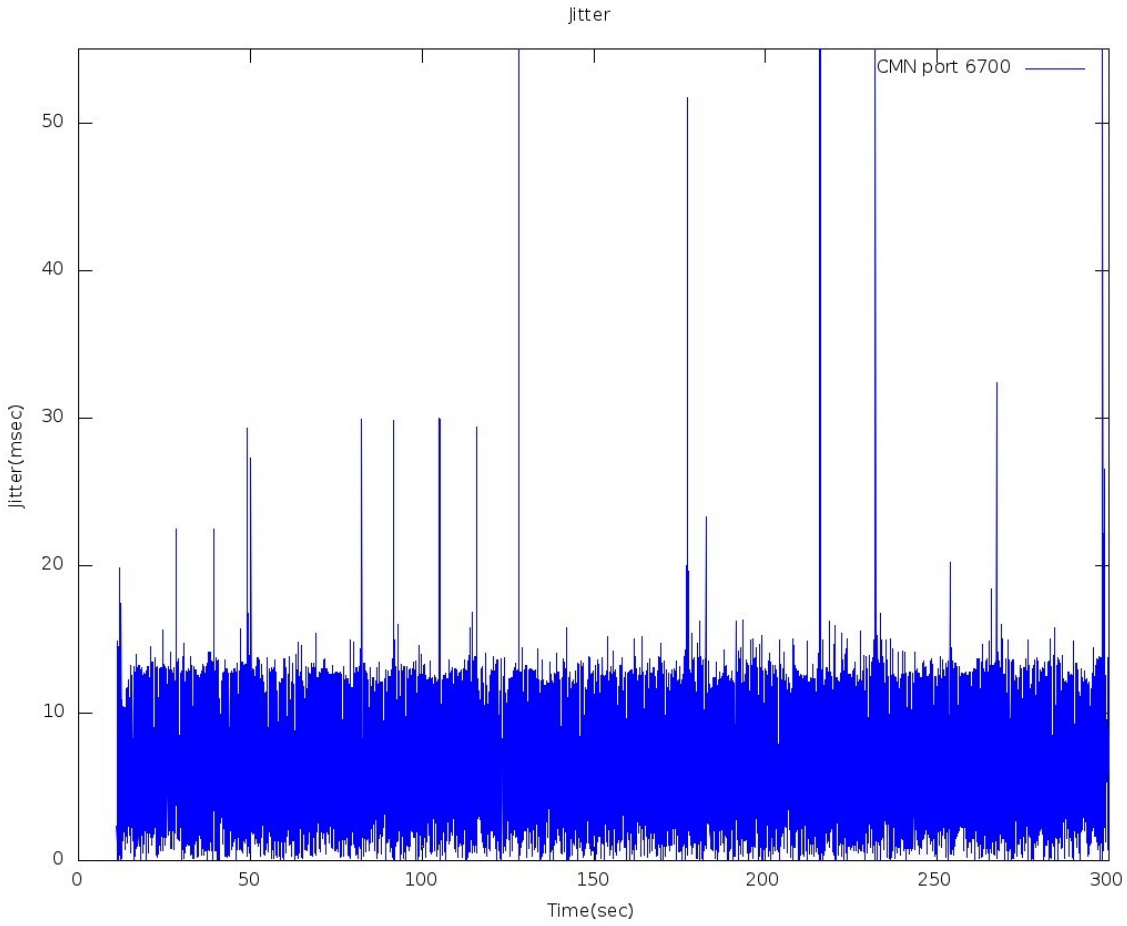
Σχήμα 30: Γραφική πέμπτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5700



Σχήμα 31: Γραφική πέμπτου σεναρίου: Διακύμανση καθυστέρησης της ροής 6000

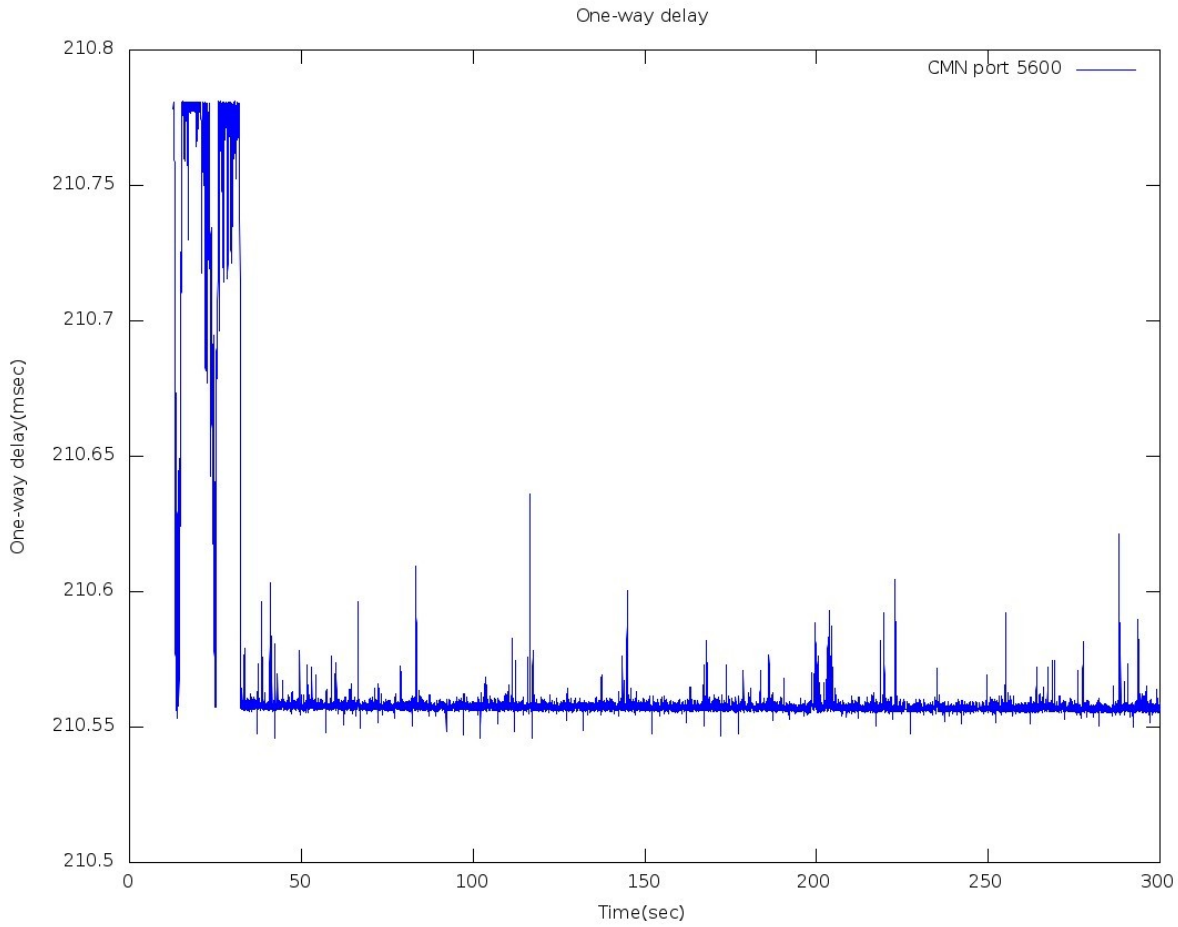


Σχήμα 32: Γραφική πέμπτου σεναρίου: Διακύμανση καθυστέρησης της ροής 6600

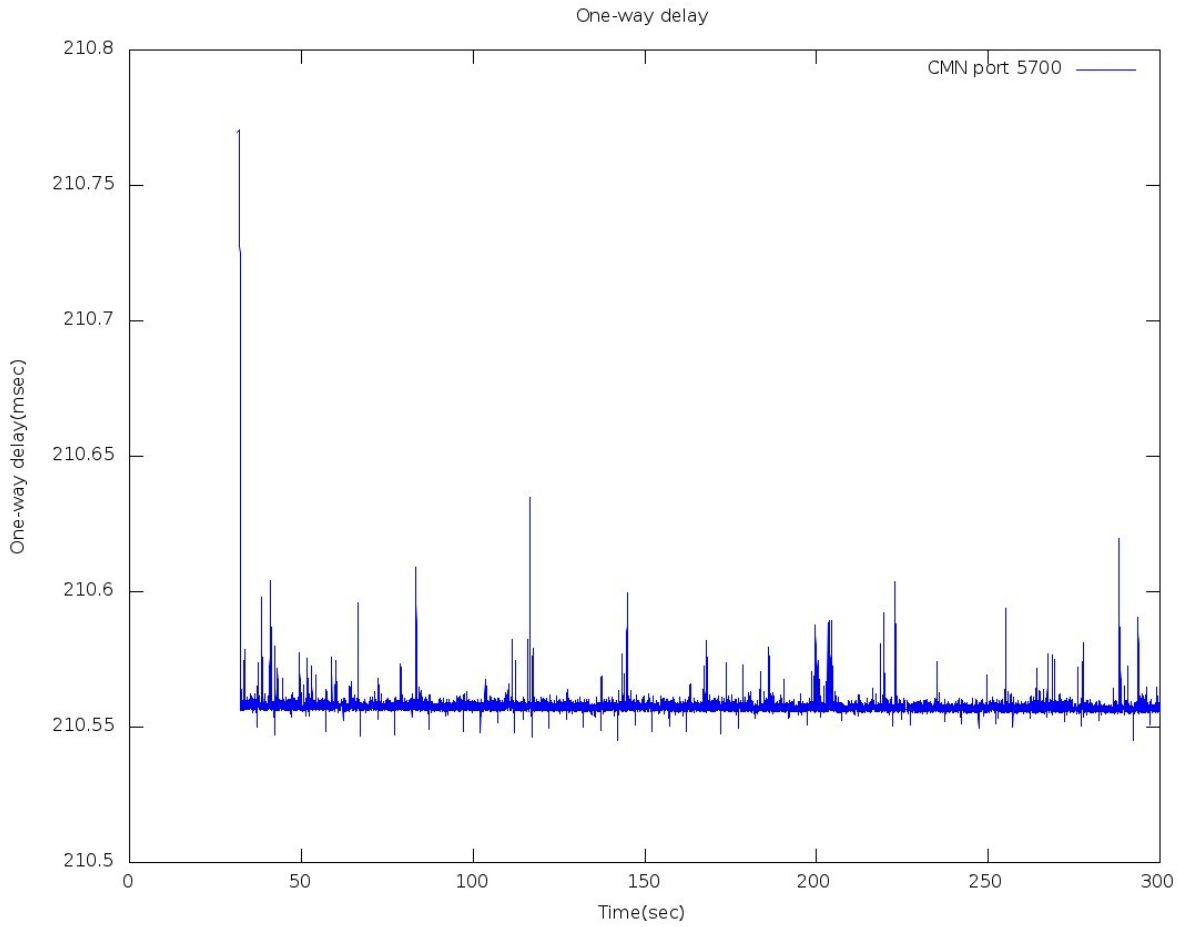


Σχήμα 33: Γραφική πέμπτου σεναρίου: Διακύμανση καθυστέρησης της ροής 6700

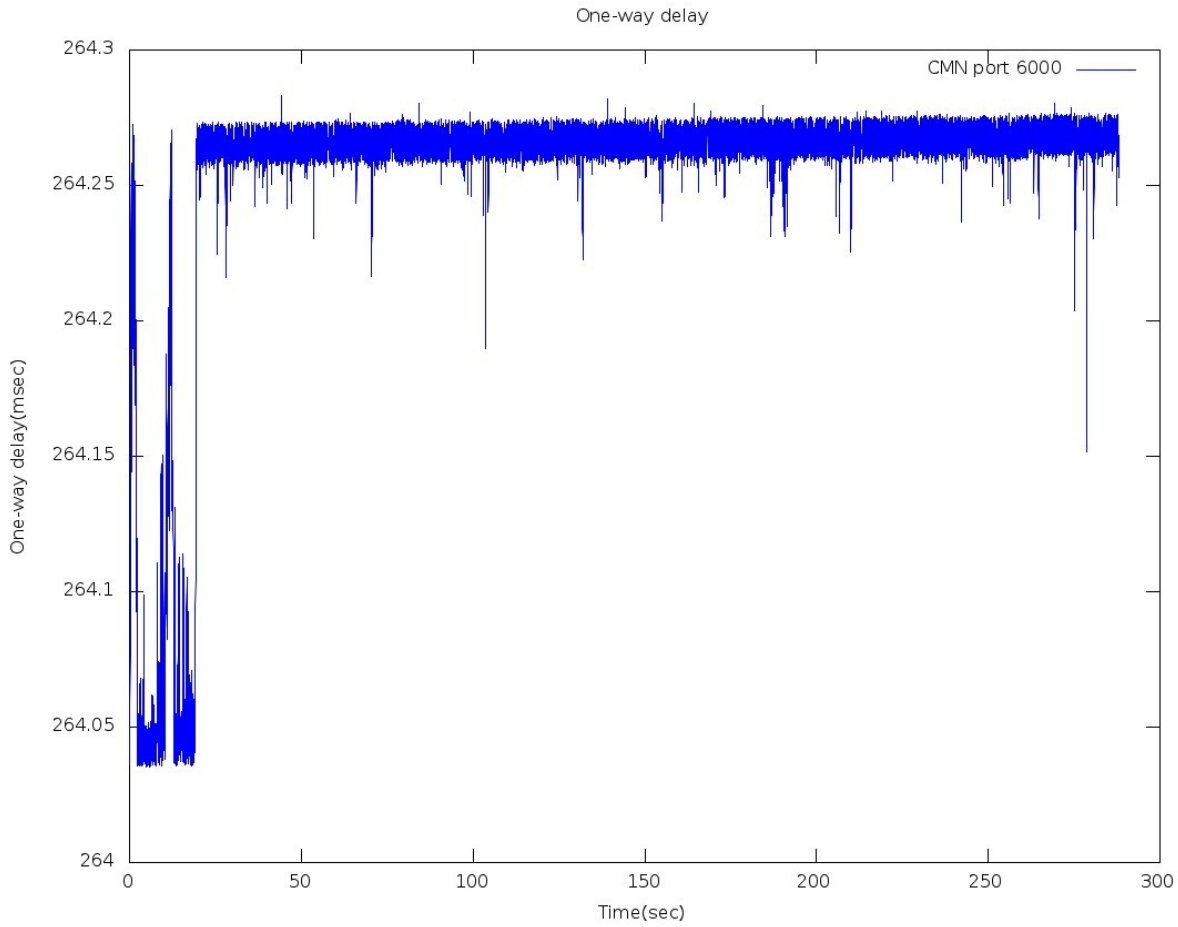
7.1.5 Γραφικές έκτου σεναρίου



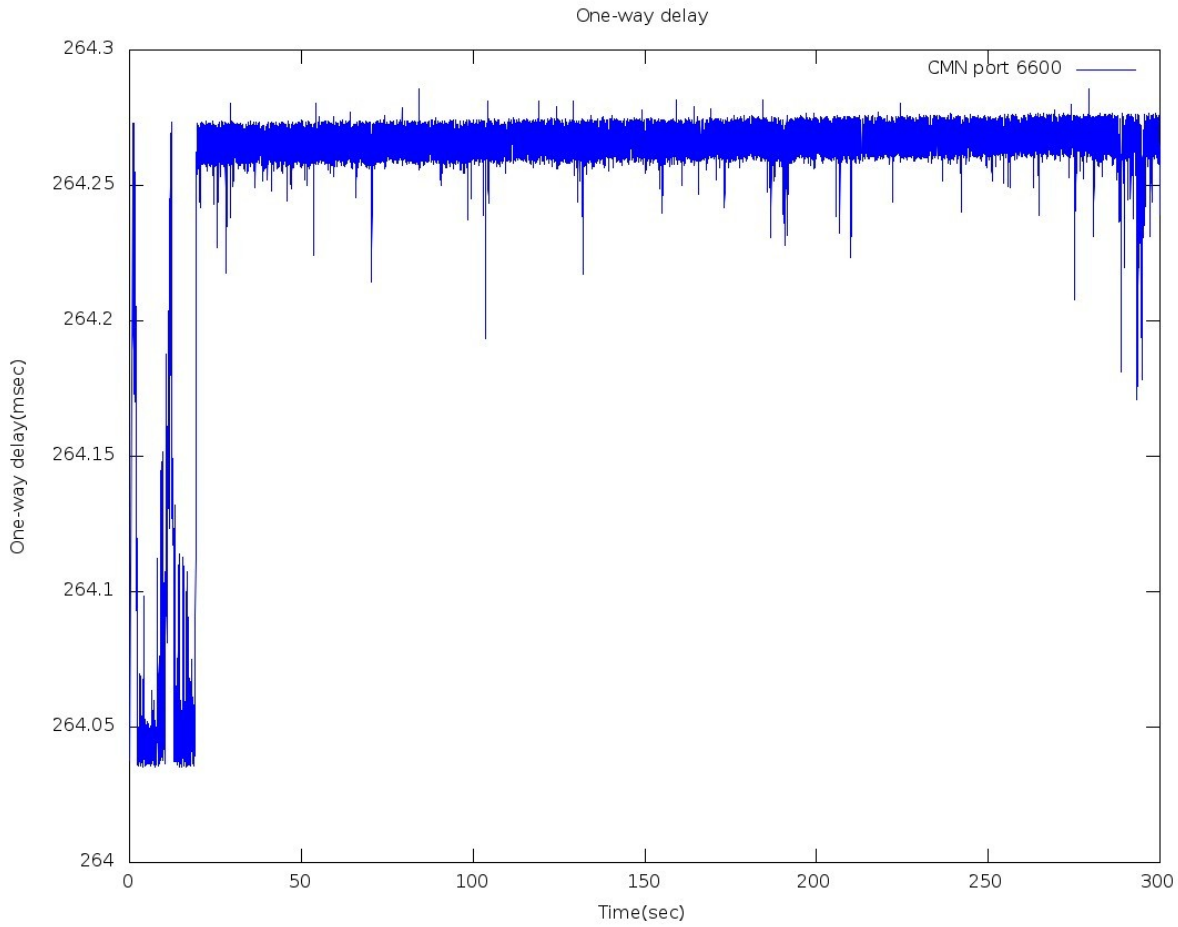
Σχήμα 34: Γραφική έκτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5600



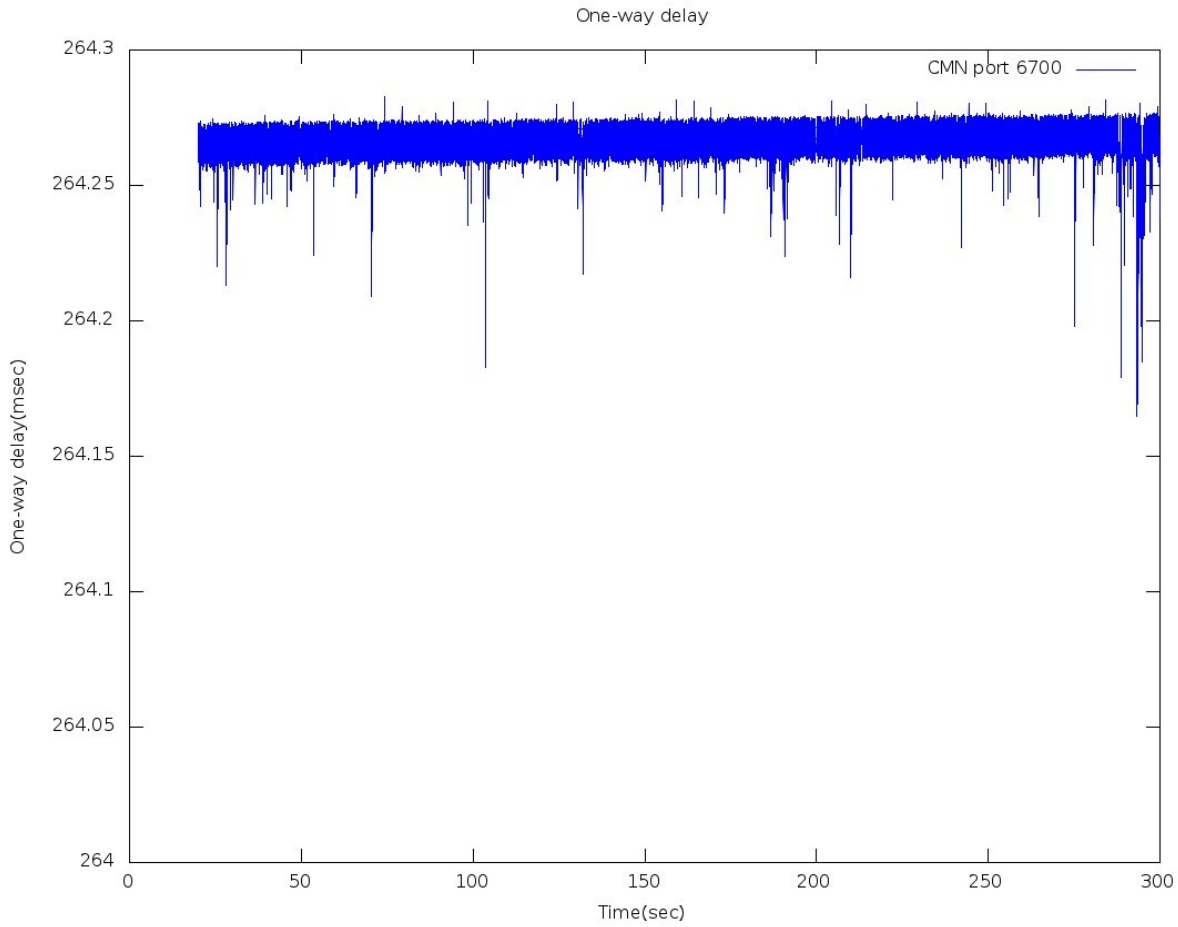
Σχήμα 35: Γραφική έκτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 5700



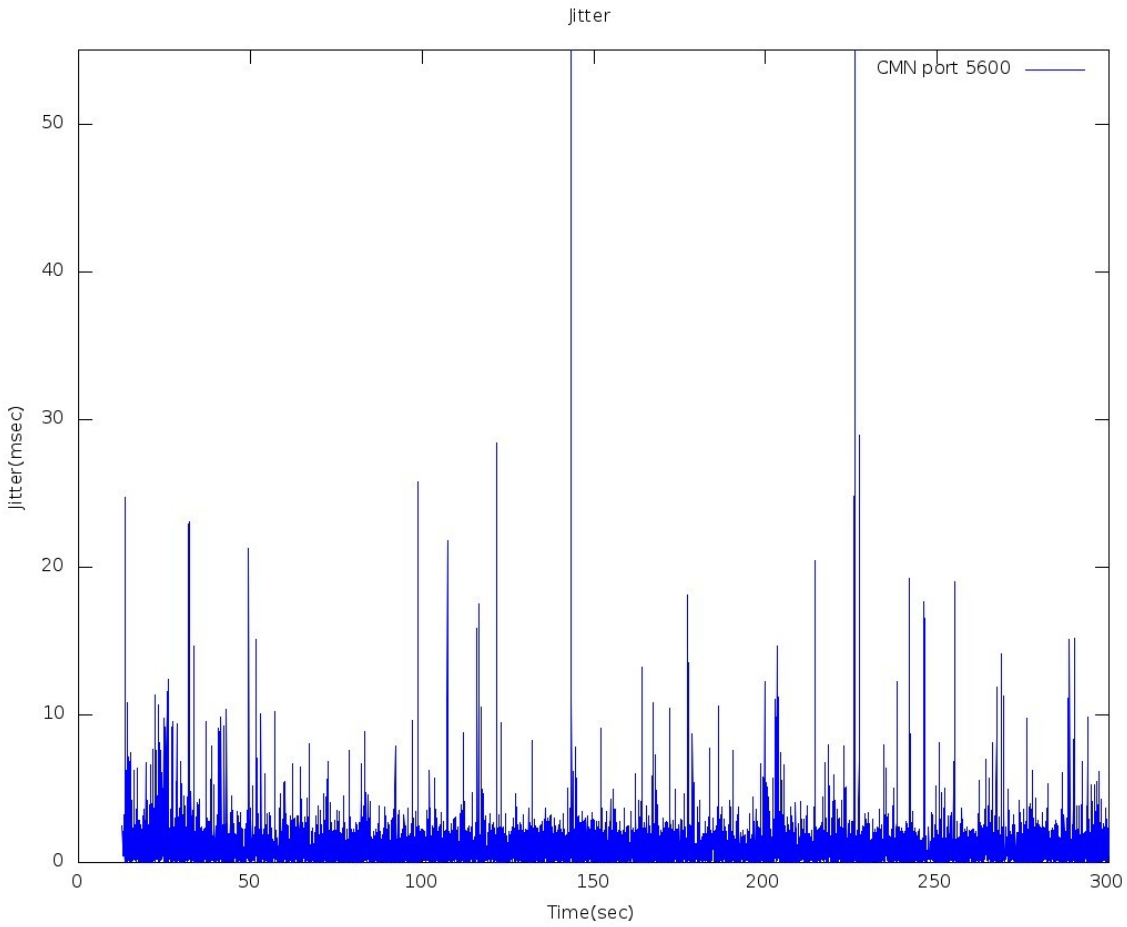
Σχήμα 36: Γραφική έκτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 6000



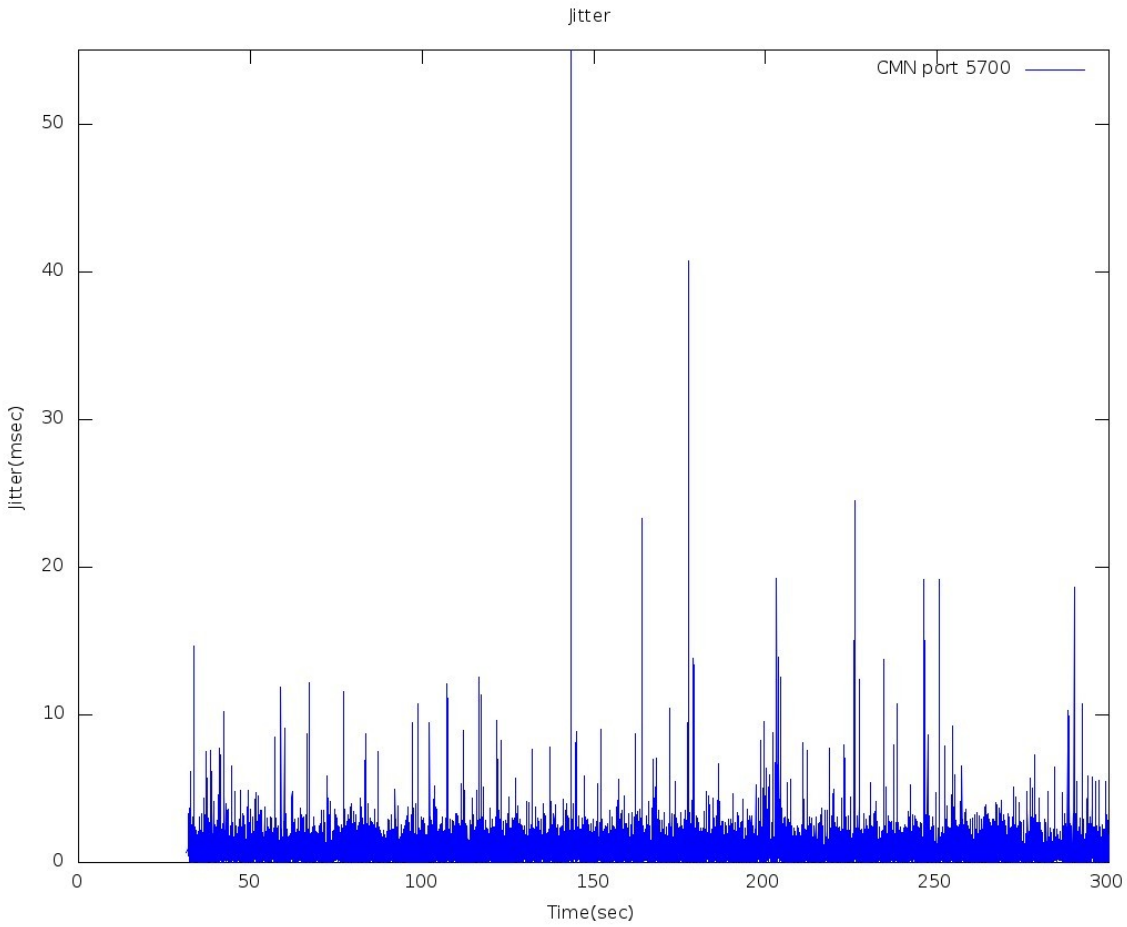
Σχήμα 37: Γραφική έκτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 6600



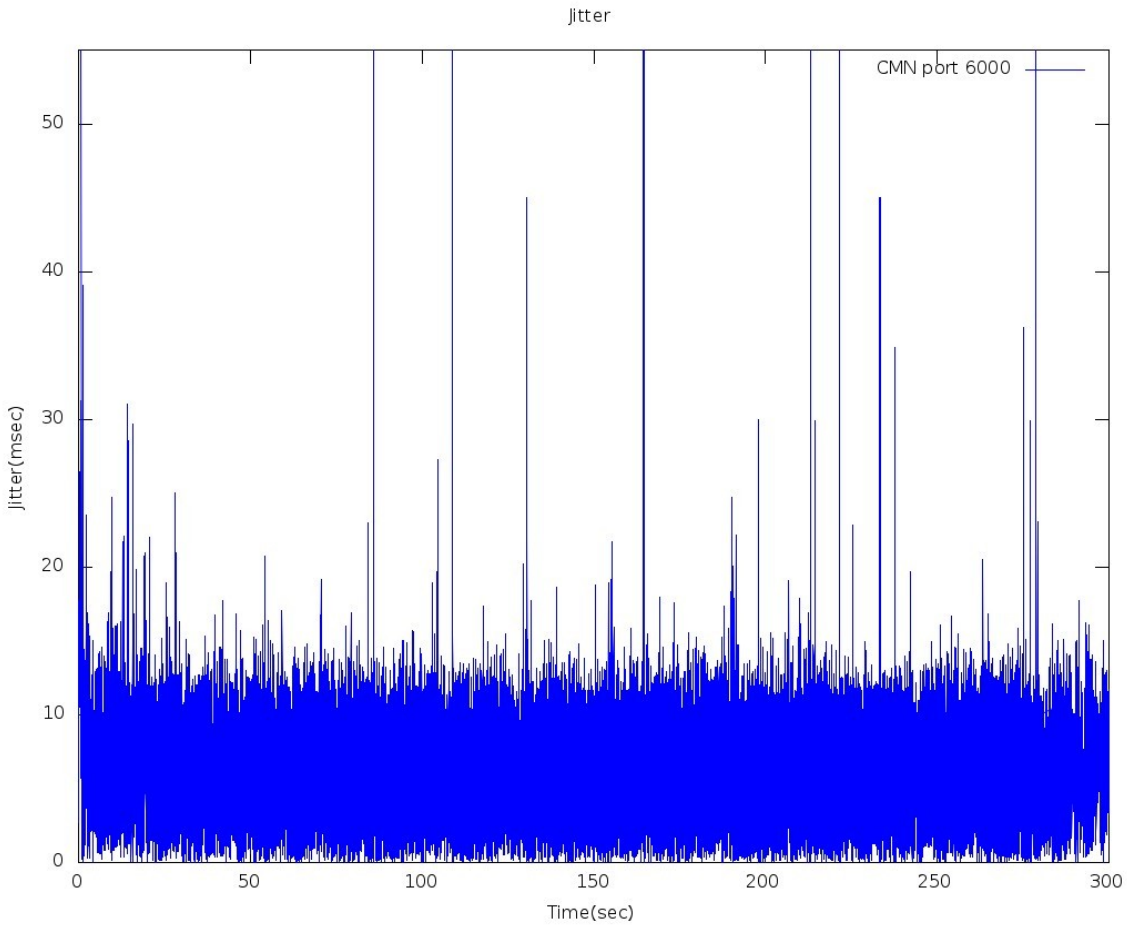
Σχήμα 38: Γραφική έκτου σεναρίου: Μονόδρομη καθυστέρηση της ροής 6700



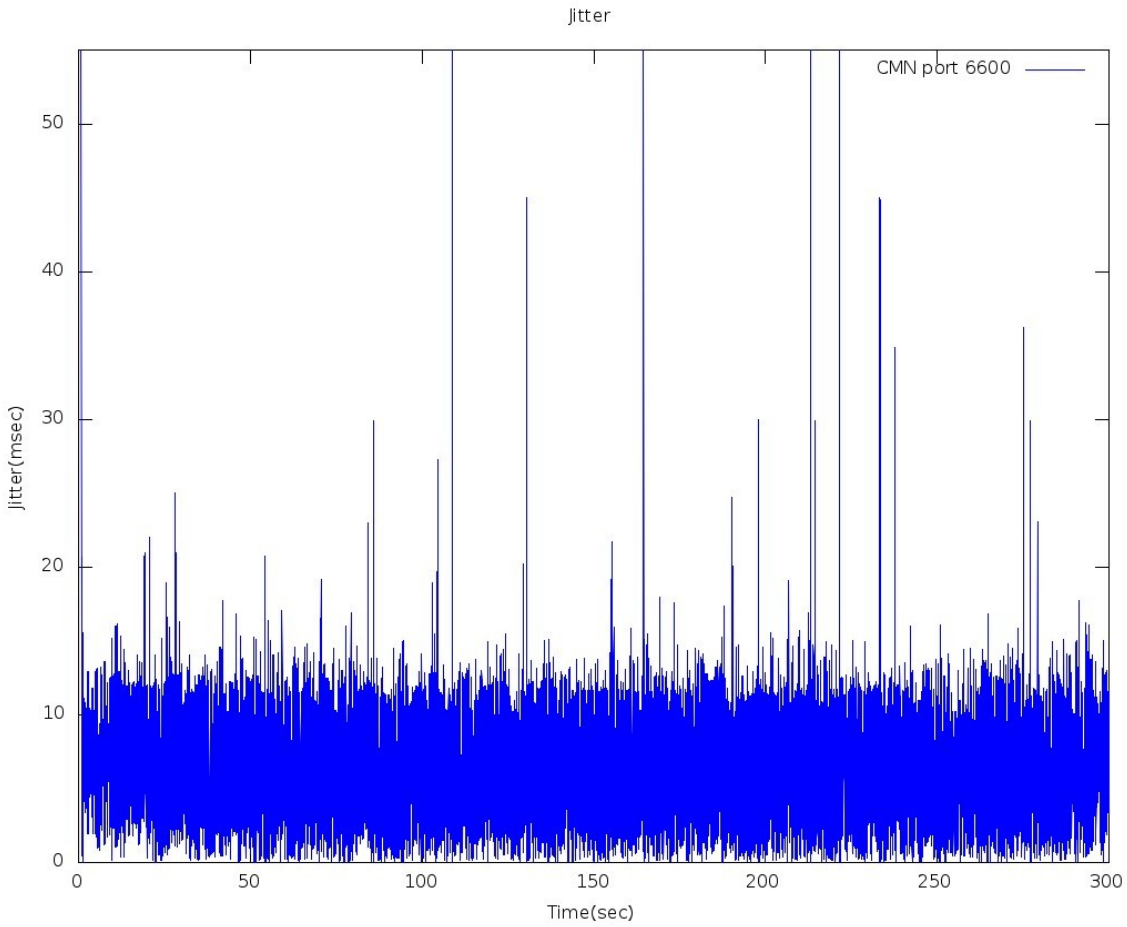
Σχήμα 39: Γραφική έκτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5600



Σχήμα 40: Γραφική έκτου σεναρίου: Διακύμανση καθυστέρησης της ροής 5700



Σχήμα 41: Γραφική έκτου σεναρίου: Διακύμανση καθυστέρησης της ροής 6000



Σχήμα 42: Γραφική έκτου σεναρίου: Διακύμανση καθυστέρησης της ροής 6600

7.1.6 Κώδικας υπολογισμών των παραμέτρων αξιολόγησης

Άνοιγμα των xml πακέτων και υπολογισμός των πραμέτρων αξιολόγησης

Απώλειες πακέτων

```
try {
   System.out.println("Analyzing port " + port + "...");



   System.out.println();
   System.out.println("CALCULATING LOSSES");
   System.out.println();

   int client_packets = client_report.size();
   int server_packets = server_report.size();
   Object sN1;
   Object Pp1;
   double sequenceNumber1 = 0;
   double Port1 = 0;
   Object sN2;
   double sequenceNumber2 = 0;
   int found = 0 ;
   int loss = 0;
   float losses_percent5000 = 0;
   float losses_percent5600 = 0;
   float losses_percent5700 = 0;
   float losses_percent5800 = 0;
```

```
float losses_percent5900 = 0;


int packets_flow_5000 = 0;
int packets_flow_5600 = 0;
int packets_flow_5700 = 0;
int packets_flow_5800 = 0;
int packets_flow_5900 = 0;

int losses5000 = 0;
int losses5600 = 0;
int losses5700 = 0;
int losses5800 = 0;
int losses5900 = 0;
for (int i = 0; i < server_report.size(); i++) {

   sN1 = server_report.get(i).get("sequenceNumber");
   String str1 = sN1.toString();
   sequenceNumber1 = Double.valueOf(str1).doubleValue();

   Pp1 = server_report.get(i).get("port");
   String str3 = Pp1.toString();
   Port1 = Double.valueOf(str3).doubleValue();
   // Sunolika paketa ana roh
   if (Port1 == 5000) {
      packets_flow_5000++;
   }else if(Port1 == 5600)  {
      packets_flow_5600++;
   }else if(Port1 == 5700)  {
      packets_flow_5700++;
   }else if(Port1 == 5800)  {
```

```
   packets_flow_5800++;
}else if(Port1 == 5900)  {
   packets_flow_5900++;
}

   for (int j = 0; j < client_report.size(); j++) {

      sN2 = client_report.get(j).get("sequenceNumber");
      String str2 = sN2.toString();
      sequenceNumber2 = Double.valueOf(str2).doubleValue();

      if ((sequenceNumber1 == sequenceNumber2)) {
         found = 1;
         break;
      }
}
   if(found == 0){
      if (Port1 == 5000) {
   losses5000++;
}else if(Port1 == 5600)  {
   losses5600++;
}else if(Port1 == 5700)  {
   losses5700++;
}else if(Port1 == 5800)  {
   losses5800++;
}else if(Port1 == 5900)  {
   losses5900++;
}

   }
```

```
}
try
      {
        if (Port1 == 5000) {

      losses_percent5000 = (losses5000*100)/packets_flow_5000;

      FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_losses_5000.txt",
true); //true tells to append data.

      String b = String.valueOf(losses5000);

      BufferedWriter out = new BufferedWriter(fstream);

      out.write(b+"\r\n");
      out.close();
        }
        else if(Port1 == 5600){

          losses_percent5600 = (losses5600*100)/packets_flow_5600;

        FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_losses_5600.txt",
true); //true tells to append data.

      String b = String.valueOf(losses5600);

      BufferedWriter out = new BufferedWriter(fstream);

      out.write(b+"\r\n");
      out.close();
        }
```

```
    else if(Port1 == 5700){

        losses_percent5700 = (losses5700*100)/packets_flow_5700;

      FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_losses_5700.txt",
true); //true tells to append data.

     String b = String.valueOf(losses5700);

     BufferedWriter out = new BufferedWriter(fstream);

     out.write(b+"\r\n");
     out.close();

    }
    else if(Port1 == 5800){

        losses_percent5800 = (losses5800*100)/packets_flow_5800;

      FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_losses_5800.txt",
true); //true tells to append data.

     String b = String.valueOf(losses5800);

     BufferedWriter out = new BufferedWriter(fstream);

     out.write(b+"\r\n");
     out.close();

    }
    else if(Port1 == 5900){
```

```
losses_percent5900 = (losses5900*100)/packets_flow_5900;

FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_losses_5900.txt", true); //true tells to append data.

String b = String.valueOf(losses5900);

BufferedWriter out = new BufferedWriter(fstream);

out.write(b+"\r\n");
out.close();
}
}
catch (Exception e)
{
System.err.println("Error: " + e.getMessage());
}
```

Υπολογισμός Bitrate

```
double Temp_sum_payload_client=0.0;
double Temp_sum_payload_server=0.0;
Object tP1 =0;
Object payloadd1 = 0;
Object tP2 =0;
Object payloadd2 = 0;
double Sum_payload1= 0.0;
double Sum_payload2= 0.0;
double bitrate1 = 0.0;
double bitrate2 = 0.0;
double temp_bitrate1 = 0.0;
double temp_bitrate2 = 0.0;
double timestampFirst1 = 0.0;
Object tF1 = 0;
double timestampLast1 = 0.0;
Object tL1 = 0;
double timestampFirst2 = 0.0;
Object tF2 = 0;
double timestampLast2 = 0.0;
Object tL2 = 0;
int q=0;
double timestampTempFirst1 =0.0;
Object tTF1=0;
double timestampTempNext1 =0.0;
Object tTN1=0;
double timestampTempFirst2 =0.0;
Object tTF2=0;
double timestampTempNext2 =0.0;
```

```
Object tTN2=0;
Object Porta1 =0.0;
double ppp = 0.0;
Object Porta2 =0.0;
double ppp2 = 0.0;

for (int i = 0; i < server_report.size(); i++) {
   if (i == 0 ){
      tF1 = server_report.get(i).get("time");
      String str = tF1.toString();
      timestampFirst1 = Double.valueOf(str).doubleValue();
   }
   if (i == server_report.size()-1){
      tL1 = server_report.get(i).get("time");
      String str = tL1.toString();
      timestampLast1 = Double.valueOf(str).doubleValue();
   }


payloadd1 = server_report.get(i).get("payload");
String str = payloadd1.toString();
double payload1 = Double.valueOf(str).doubleValue();
Sum_payload1 = Sum_payload1 + payload1;

Porta1 = server_report.get(i).get("port");
String str15 = Porta1.toString();
double Portaa1 = Double.valueOf(str15).doubleValue();
ppp = Portaa1;
}

bitrate1 = Sum_payload1 / (timestampLast1 - timestampFirst1);
```

```
System.out.println("PORT :  "+ ppp +" Average Bitrate Server : "+ bitrate1);

///////  BITRATE ANA 20 PAKETA SERVER   ////////////
for (int i = 0; i < server_report.size(); i++) {


tP1 = server_report.get(i).get("payload");
String str2 = tP1.toString();
double temp_payload1 = Double.valueOf(str2).doubleValue();
Temp_sum_payload_server = Temp_sum_payload_server + temp_payload1;



    if ((i%20 == 0 && i!=0)) {
      tTF1 = server_report.get(i).get("time");
      String str = tTF1.toString();
      timestampTempNext1 = Double.valueOf(str).doubleValue();

      tTN1 = server_report.get(i-20).get("time");
      String str1 = tTN1.toString();
      timestampTempFirst1 = Double.valueOf(str1).doubleValue();


        temp_bitrate1 = Temp_sum_payload_server / (timestampTempNext1 -
timestampTempFirst1);
         //System.out.println("Bitrate Server per 20 packets : "+ temp_bitrate1);
         Temp_sum_payload_server = 0 ;
```

```
    try
{
   Object PS = server_report.get(i).get("port");
   String strS = PS.toString();
   double PortServer = Double.valueOf(strS).doubleValue();

   if (PortServer == 5000) {



FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_server_5000.txt", true); //true tells to append data.
String a = String.valueOf(temp_bitrate1);

String c = String.valueOf(timestampTempNext1 - timestampTempFirst1);
BufferedWriter out = new BufferedWriter(fstream);

out.write(a +" "+ c +"\r\n");
out.close();
   }
   else if(PortServer == 5600){
      FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_server_5600.txt", true); //true tells to append data.
String a = String.valueOf(temp_bitrate1);

String c = String.valueOf(timestampTempNext1 - timestampTempFirst1);
BufferedWriter out = new BufferedWriter(fstream);

out.write(a +" "+ c +"\r\n");
out.close();
   }
   else if(PortServer== 5700){
```

```
            FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_server_5700.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate1);

            String c = String.valueOf(timestampTempNext1 - timestampTempFirst1);
            BufferedWriter out = new BufferedWriter(fstream);

            out.write(a +" "+ c +"\r\n");
            out.close();

           }
              else if(PortServer== 5800){
              FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_server_5800.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate1);

            String c = String.valueOf(timestampTempNext1 - timestampTempFirst1);
            BufferedWriter out = new BufferedWriter(fstream);

            out.write(a +" "+ c +"\r\n");
            out.close();

           }
              else if(PortServer== 5900){
              FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_server_5900.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate1);

            String c = String.valueOf(timestampTempNext1 - timestampTempFirst1);
            BufferedWriter out = new BufferedWriter(fstream);
```

```
        out.write(a +" "+ c +"\r\n");
        out.close();

       }
       }
    catch (Exception e)
    {
    System.err.println("Error: " + e.getMessage());
    }
    }
}

          for (int i = 0; i < client_report.size(); i++) {


tP2 = client_report.get(i).get("payload");
String str2 = tP2.toString();
double temp_payload2 = Double.valueOf(str2).doubleValue();
Temp_sum_payload_client = Temp_sum_payload_client + temp_payload2;



     if ((i%20 == 0 && i!=0)) {
       tTF2 = client_report.get(i).get("time");
       String str = tTF2.toString();
       timestampTempNext2 = Double.valueOf(str).doubleValue();

       tTN2 = client_report.get(i-20).get("time");
       String str1 = tTN2.toString();
       timestampTempFirst2 = Double.valueOf(str1).doubleValue();
```

```
            temp_bitrate2 = Temp_sum_payload_client / (timestampTempNext2 -
timestampTempFirst2);
             //System.out.println("Bitrate Client per 20 packets : "+ temp_bitrate2);
             Temp_sum_payload_client = 0 ;


                    try
            {
               Object PC = client_report.get(i).get("port");
               String strC = PC.toString();
               double PortClient = Double.valueOf(strC).doubleValue();


               if (PortClient == 5000) {



            FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_client_5000.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate2);


            String c = String.valueOf(timestampTempNext2 - timestampTempFirst2);
            BufferedWriter out = new BufferedWriter(fstream);


            out.write(a +" "+ c +"\r\n");
            out.close();
               }
               else if(PortClient == 5600){
                  FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_client_5600.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate2);


            String c = String.valueOf(timestampTempNext2 - timestampTempFirst2);
```

```
            BufferedWriter out = new BufferedWriter(fstream);

            out.write(a +" "+ c +"\r\n");
            out.close();
              }
              else if(PortClient == 5700){
              FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_client_5700.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate2);

            String c = String.valueOf(timestampTempNext2 - timestampTempFirst2);
            BufferedWriter out = new BufferedWriter(fstream);

            out.write(a +" "+ c +"\r\n");
            out.close();

           }
              else if(PortClient == 5800){
              FileWriter fstream = new
FileWriter("/home/ast3rix/Desktop/manos_bitrate_client_5800.txt", true); //true tells to append data.
            String a = String.valueOf(temp_bitrate2);

            String c = String.valueOf(timestampTempNext2 - timestampTempFirst2);
            BufferedWriter out = new BufferedWriter(fstream);

            out.write(a +" "+ c +"\r\n");
            out.close();

           }
              else if(PortClient == 5900){
              FileWriter fstream = new
```

```
FileWriter("/home/ast3rix/Desktop/manos_bitrate_client_5900.txt", true); //true tells to append data.
                String a = String.valueOf(temp_bitrate2);

                String c = String.valueOf(timestampTempNext2 - timestampTempFirst2);
                BufferedWriter out = new BufferedWriter(fstream);

                out.write(a +" "+ c +"\r\n");
                out.close();

               }
               }
           catch (Exception e)
           {
           System.err.println("Error: " + e.getMessage());
           }

           }
        }

        for (int j = 0; j < client_report.size(); j++) {
           if (j == 0 ){
              tF2 = client_report.get(j).get("time");
              String str = tF2.toString();
              timestampFirst2 = Double.valueOf(str).doubleValue();

           }
           if (j == client_report.size()-1){
              tL2 = client_report.get(j).get("time");
              String str = tL2.toString();
              timestampLast2 = Double.valueOf(str).doubleValue();
           }
```

```
payloadd2 = client_report.get(j).get("payload");
String str = payloadd2.toString();
double payload2 = Double.valueOf(str).doubleValue();
Sum_payload2 = Sum_payload2 + payload2;

Porta2 = client_report.get(j).get("port");
String str16 = Porta2.toString();
double Portaa2 = Double.valueOf(str16).doubleValue();
ppp2 = Portaa2;
}
bitrate2 = Sum_payload2 / (timestampLast2 - timestampFirst2);
System.out.println("PORT :  "+ ppp2 +" Average Bitrate Client : "+ bitrate2);
```

Υπολογισμός Jitter

```
double FirstTimeServer = 0.0;
double SecondTimeServer = 0.0;
double FirstTimeClient = 0.0;
double SecondTimeClient = 0.0;
double TempJitter_Server = 0.0;
double TempJitter_Client = 0.0;
double[] Jitter_Server = new double[client_report.size()];
double[] Jitter_Client = new double[client_report.size()];

double[][] Pinakas_server_report = new double[4][server_report.size()];
double[][] New_Pinakas_server_report = new double[4][client_report.size()];
double[][] Pinakas_client_report = new double[4][client_report.size()];



double SequenceNumberFirst = 0.0;
double SequenceNumberSecond = 0.0;

double Jitter2 =0.0;
double Jitter1 =0.0;
double Jitter =0.0;
double[] Temp_Jitter = new double[Jitter_Server.length];

double Average_Jitter = 0.0;
double Average_one_way_delay = 0.0;
double Temporary_one_way_delay =0.0;
int    counter=0;
double Temp_Sum_one_way_delay =0.0;
```

```
for (int c = 0; c < client_report.size(); c++) {
    Object SNC = client_report.get(c).get("sequenceNumber");
    String s1 = SNC.toString();
    double SequenceNumberClient = Double.valueOf(s1).doubleValue();

    Object TimeC = client_report.get(c).get("time");
    String s2 = TimeC.toString();
    double TimeClient = Double.valueOf(s2).doubleValue();

    Object TimestC = client_report.get(c).get("timestamp");
    String s3 = TimestC.toString();
    double TimestampClient = Double.valueOf(s3).doubleValue();

    Object PortC = client_report.get(c).get("port");
    String s4 = PortC.toString();
    double PortClient = Double.valueOf(s4).doubleValue();

    Pinakas_client_report[0][c] =  SequenceNumberClient;
    Pinakas_client_report[1][c] =  TimeClient;
    Pinakas_client_report[2][c] =  TimestampClient;
    Pinakas_client_report[3][c] =  PortClient;
}

for (int s = 0; s < server_report.size(); s++) {
    Object SNS = server_report.get(s).get("sequenceNumber");
    String s1 = SNS.toString();
    double SequenceNumberServer = Double.valueOf(s1).doubleValue();

    Object TimeS = server_report.get(s).get("time");
    String s2 = TimeS.toString();
    double TimeServer = Double.valueOf(s2).doubleValue();
```

```
    Object TimestS = server_report.get(s).get("timestamp");
    String s3 = TimestS.toString();
    double TimestampServer = Double.valueOf(s3).doubleValue();

    Object PortS = server_report.get(s).get("port");
    String s4 = PortS.toString();
    double PortServer = Double.valueOf(s4).doubleValue();

    Pinakas_server_report[0][s] =  SequenceNumberServer;
    Pinakas_server_report[1][s] =  TimeServer;
    Pinakas_server_report[2][s] =  TimestampServer;
    Pinakas_server_report[3][s] =  PortServer;
}


for (int f = 0; f < client_report.size()-1; f++) {


    SequenceNumberFirst = Pinakas_client_report[0][f];
    SequenceNumberSecond = Pinakas_client_report[0][f+1];

    if (SequenceNumberFirst > SequenceNumberSecond){

        double TempSN = Pinakas_client_report[0][f];
        Pinakas_client_report[0][f] = Pinakas_client_report[0][f+1];
        Pinakas_client_report[0][f+1] = TempSN;

        double TempT = Pinakas_client_report[1][f];
        Pinakas_client_report[1][f] = Pinakas_client_report[1][f+1];
        Pinakas_client_report[1][f+1] = TempT;
```

```
            double TempTst = Pinakas_client_report[2][f];
            Pinakas_client_report[2][f] = Pinakas_client_report[2][f+1];
            Pinakas_client_report[2][f+1] = TempTst;

            double TempPort = Pinakas_client_report[3][f];
            Pinakas_client_report[3][f] = Pinakas_client_report[3][f+1];
            Pinakas_client_report[3][f+1] = TempPort;
        }
    }




    for (int d = 0; d < Pinakas_client_report[0].length; d++) {

        for (int i = 0; i < Pinakas_server_report[0].length; i++) {



            if ((Pinakas_client_report[0][d] == Pinakas_server_report[0][i]) &&
(Pinakas_client_report[2][d] == Pinakas_server_report[2][i])) {



            New_Pinakas_server_report[0][d] = Pinakas_server_report[0][i];
            New_Pinakas_server_report[1][d] = Pinakas_server_report[1][i];
            New_Pinakas_server_report[2][d] = Pinakas_server_report[2][i];
            New_Pinakas_server_report[3][d] = Pinakas_server_report[3][i];

            }
        }
    }
```

```
 for (int j = 1; j <Pinakas_client_report[0].length ; j++) {
    FirstTimeClient = Pinakas_client_report[1][j-1];
    SecondTimeClient = Pinakas_client_report[1][j];
    TempJitter_Client = SecondTimeClient - FirstTimeClient;

      if (TempJitter_Client<0) {
         TempJitter_Client= TempJitter_Client*-1;
      }

    if (TempJitter_Client!= 0.0 ) {

    Jitter_Client[j-1] = TempJitter_Client;

      }
}
 for (int i = 1; i < Pinakas_client_report[0].length; i++) {

    FirstTimeServer = New_Pinakas_server_report[1][i-1];
    SecondTimeServer = New_Pinakas_server_report[1][i];

    TempJitter_Server = SecondTimeServer - FirstTimeServer;

     if (TempJitter_Server<0) {
         TempJitter_Server= TempJitter_Server*-1;
      }

     if (TempJitter_Server != 0.0 ) {

    Jitter_Server[i-1] = TempJitter_Server;
```

```
    }
}

for (int k = 0; k < Jitter_Server.length; k++) {

    Jitter2 = Jitter_Server[k];
    Jitter1 = Jitter_Client[k];
    double Jitter_trial= Jitter2 - Jitter1;

    if (Jitter_trial <0) {
    Jitter= Jitter * -1;
  }
    if (Jitter_trial<1000) {
    Jitter = Jitter_trial;
  }
    try
    {


      if ( Pinakas_client_report[3][k]== 5000) {


      FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_jitter_5000.txt",
true); //true tells to append data.
      String a = String.valueOf(Jitter);
      String b = String.valueOf(Pinakas_client_report[0][k]);
      String c = String.valueOf(Pinakas_client_report[1][k]);
      BufferedWriter out = new BufferedWriter(fstream);

      out.write(a +" "+b+" "+ c +"\r\n");
      out.close();
```

```
                }
                else if( Pinakas_client_report[3][k]== 5600){
                  FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_jitter_5600.txt", true); //true tells to append data.
                String a = String.valueOf(Jitter);
                String b = String.valueOf(Pinakas_client_report[0][k]);
                String c = String.valueOf(Pinakas_client_report[1][k]);
                BufferedWriter out = new BufferedWriter(fstream);

                out.write(a +" "+b+" "+ c +"\r\n");
                out.close();
                }
                else if( Pinakas_client_report[3][k]== 5700){
                FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_jitter_5700.txt", true); //true tells to append data.
                String a = String.valueOf(Jitter);
                String b = String.valueOf(Pinakas_client_report[0][k]);
                String c = String.valueOf(Pinakas_client_report[1][k]);
                BufferedWriter out = new BufferedWriter(fstream);
                out.write(a +" "+b+" "+ c +"\r\n");
                out.close();
               }
                else if(Pinakas_client_report[3][k] == 5800){
                FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_jitter_5800.txt", true); //true tells to append data.
                String a = String.valueOf(Jitter);
                String b = String.valueOf(Pinakas_client_report[0][k]);
                String c = String.valueOf(Pinakas_client_report[1][k]);
                BufferedWriter out = new BufferedWriter(fstream);
                out.write(a +" "+b+" "+ c +"\r\n");
                out.close();
```

```
            }
               else if(Pinakas_client_report[3][k] == 5900){
               FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_jitter_5900.txt",
true); //true tells to append data.
               String a = String.valueOf(Jitter);
               String b = String.valueOf(Pinakas_client_report[0][k]);
               String c = String.valueOf(Pinakas_client_report[1][k]);
               BufferedWriter out = new BufferedWriter(fstream);
               out.write(a +" "+b+" "+ c +"\r\n");
               out.close();
               }
               }
            catch (Exception e)
            {
            System.err.println("Error: " + e.getMessage());
            }

               Temp_Jitter[k] = Jitter;
         }

         for (int p = 0; p < Temp_Jitter.length; p++) {
            Average_Jitter = Average_Jitter + Temp_Jitter[p];
            }
         Average_Jitter = Average_Jitter / Temp_Jitter.length;

         if (Average_Jitter<0) {
            Average_Jitter= (Average_Jitter* -1);
         }
         Average_Jitter = Average_Jitter*1000;
         System.out.println("PORT : "+New_Pinakas_server_report[3][100] +" Average Jitter : "+
Average_Jitter);
```

One-Way delay

```
for (int i = 0; i < New_Pinakas_server_report[0].length; i++) {

for (int j = 0; j < Pinakas_client_report[0].length; j++) {

if ((Pinakas_client_report[0][j] == New_Pinakas_server_report[0][i]) &&
(Pinakas_client_report[2][j] == New_Pinakas_server_report[2][i])){

Temporary_one_way_delay = Pinakas_client_report[1][i]-New_Pinakas_server_report[1][j];

try
{

if (New_Pinakas_server_report[3][j]==5000) {
FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_delay5000.txt",
true); //true tells to append data.


String a = String.valueOf(Temporary_one_way_delay);
String b = String.valueOf(Pinakas_client_report[0][j]);
String c = String.valueOf(Pinakas_client_report[1][j]);
BufferedWriter out = new BufferedWriter(fstream);

out.write(a +" "+ b +" "+ c +"\r\n");
out.close();
}else if (New_Pinakas_server_report[3][j]==5600) {
FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_delay5600.txt",
true); //true tells to append data.
String a = String.valueOf(Temporary_one_way_delay);
```

```
String b = String.valueOf(Pinakas_client_report[0][j]);
String c = String.valueOf(Pinakas_client_report[1][j]);
BufferedWriter out = new BufferedWriter(fstream);
out.write(a +" "+ b +" "+ c +"\r\n");
out.close();
  String a = String.valueOf(Temporary_one_way_delay);
String b = String.valueOf(Pinakas_client_report[0][j]);
String c = String.valueOf(Pinakas_client_report[1][j]);
BufferedWriter out = new BufferedWriter(fstream);
out.write(a +" "+ b +" "+ c +"\r\n");
out.close();
  }else if (New_Pinakas_server_report[3][j]==5600) {
     FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_delay5600.txt",
true); //true tells to append data.
String a = String.valueOf(Temporary_one_way_delay);
String b = String.valueOf(Pinakas_client_report[0][j]);
String c = String.valueOf(Pinakas_client_report[1][j]);
BufferedWriter out = new BufferedWriter(fstream);
out.write(a +" "+ b +" "+ c +"\r\n");
out.close();
  }   else if (New_Pinakas_server_report[3][j]==5700) {
     FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_delay5700.txt",
true); //true tells to append data.
String a = String.valueOf(Temporary_one_way_delay);
String b = String.valueOf(Pinakas_client_report[0][j]);
String c = String.valueOf(Pinakas_client_report[1][j]);
BufferedWriter out = new BufferedWriter(fstream);

out.write(a +" "+ b +" "+ c +"\r\n");
out.close();
  }  else if (New_Pinakas_server_report[3][j]==5800) {
```

```
                    FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_delay5800.txt",
true); //true tells to append data.

                String a = String.valueOf(Temporary_one_way_delay);
                String b = String.valueOf(Pinakas_client_report[0][j]);
                String c = String.valueOf(Pinakas_client_report[1][j]);
                BufferedWriter out = new BufferedWriter(fstream);


                out.write(a +" "+ b +" "+ c +"\r\n");
                out.close();
                  }  else if (New_Pinakas_server_report[3][j]==5900) {
                    FileWriter fstream = new FileWriter("/home/ast3rix/Desktop/manos_delay5900.txt",
true); //true tells to append data.
                String a = String.valueOf(Temporary_one_way_delay);
                String b = String.valueOf(Pinakas_client_report[0][j]);
                String c = String.valueOf(Pinakas_client_report[1][j]);
BufferedWriter out = new BufferedWriter(fstream);

                out.write(a +" "+ b +" "+ c +"\r\n");
                out.close();
                  }
                }
            catch (Exception e)
            {
            System.err.println("Error: " + e.getMessage());
            }
                 Temp_Sum_one_way_delay = Temp_Sum_one_way_delay +
Temporary_one_way_delay;
 counter++;
                }
            }
        }
```

```
Average_one_way_delay= Temp_Sum_one_way_delay  / counter;
System.out.println("PORT : "+New_Pinakas_server_report[3][100]+" Average One-Way
Delay : " +Average_one_way_delay);
```