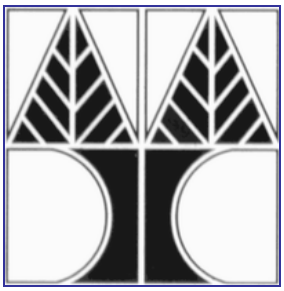
Ατομική Διπλωματική Εργασία

ΧΡΗΣΗ MULTICAST ΓΙΑ ΜΕΤΑΔΟΣΗ ΠΟΛΥΜΕΣΙΚΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΣΕ ΚΙΝΗΤΑ IPv6 ΔΙΚΤΥΑ

Λουίζα Μιχαήλ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ



ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ιούνιος 2007

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΧΡΗΣΗ MULTICAST ΓΙΑ ΜΕΤΑΔΟΣΗ ΠΟΛΥΜΕΣΙΚΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΣΕ ΚΙΝΗΤΑ IPv6 ΔΙΚΤΥΑ

Λουίζα Μιχαήλ

Επιβλέπων Καθηγητής
Δρ. Βάσος Βασιλείου

Ιούνιος 2007

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα, κατ' αρχήν, να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου Δρ. Βάσο Βασιλείου, για την ομαλή συνεργασία που είχαμε καθ' όλο το χρονικό διάστημα εκπόνησης της διπλωματικής αυτής εργασίας. Οι συμβουλές, οι οδηγίες και οι κατευθύνσεις του συνέβαλαν καταλυτικά στην επιτυχή ολοκλήρωση της παρούσας εργασίας.

Θα ήταν παράλειψή μου να μην εκφράσω τις θερμές μου ευχαριστίες στον κύριο Παύλο Αντωνίου, υποψήφιο διδάκτορα στο Πανεπιστήμιο Κύπρου, ο οποίος αφιέρωσε μεγάλο μέρος του πολύτιμου χρόνου του ώστε να με βοηθήσει στην εκπόνηση της διπλωματικής μου εργασίας.

Τέλος, θερμές ευχαριστίες θα ήθελα να απευθύνω και στους γονείς μου για την ψυχολογική στήριξη και ενθάρρυνση που μου παρείχαν καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

ΠΕΡΙΛΗΨΗ

Σκοπός της Διπλωματικής μου εργασίας είναι η multicast μετάδοση πολυμεσικού περιεχομένου σε κινητά IPv6 δίκτυα και συγκεκριμένα η πρόταση ενός πρωτοκόλλου το οποίο να υποστηρίζει τα παραπάνω. Κατέληξα στο συμπέρασμα ότι θα πρέπει να δημιουργηθεί ή να προταθεί ένα πρωτόκολο το οποίο να είναι ικανό να εφαρμόζει multicast σε κινητούς χρήστες οι οποίοι θα έχουν ως πληροφορία ένα πολυμεσικό περιέχομενο video, audio, videotelephony. Αν το παραπάνω θέμα το δούμε ως δύο ανεξάρτητες λειτουργίες, δηλαδή η μία λειτουργία να είναι το mobile multicast και η άλλη λειτουργία να είναι η μετάδοση πολυμεσικού περιεχομένου, τότε μια καλή ιδέα είναι να προταθούν δύο πρωτόκολα από τα οποία το ένα θα αναλαμβάνει να κάνει multicast σε κινητούς χρήστες και το άλλο θα αναλαμβάνει να κάνει τη μετάδοση πολυμεσικού περιεχομένου. Από την άλλη, αν το παραπάνω θέμα το δούμε ως ενιαίο, ως μια λειτουργία δηλαδή, τότε μια καλή σαφώς ιδέα είναι να προταθεί ένα πρωτόκολλο το οποίο να ενσωματώνει και τις δύο λειτουργίες (mobile multicast και μετάδοση πολυμεσικού περιεχομένου). Θεωρώ πως για να καταλήξω στην καλύτερη πρόταση, σωστό είναι να εξεταστούν και οι δύο παραπάνω περιπτώσεις. Κατά συνέπεια, μετά από μελέτη επιστημονικών άρθρων, παρακάτω αναλύονται πρωτόκολα σχετικά με mobile multicast που έχουν προταθεί από επιστήμονες – ερευνητές. Έπειτα, ακολουθούν τα πρωτόκολλα που έχουν προταθεί σχετικά με μετάδοση πολυμεσικού περιεχομένου και στο τέλος αναλύεται ο συνδυασμός δύο πρωτοκόλων που είναι τα καταλληλότερα για το πρόβλημα που εξετάζω. Στο τέλος προτείνεται η δική μου λύση, το δικό μου δηλαδή πρωτόκολλο που να ικανοποιεί τις απαιτήσεις που μου έχουν ζητηθεί. Το πρωτόκολλο αυτό εξηγείται σε θεωρητικό επίπεδο. Σε πρακτικό επίπεδο αξιολογείται απλό και ιεαραρχικό Video (ADIVIS) σε multiple unicast χρήστες καθώς και απλό Video σε multicast χρήστες το οποίο υποστηρίζεται από το σενάριο SSM (Source Specific Multicast).

ΠΕΡΙΕΧΟΜΕΝΑ

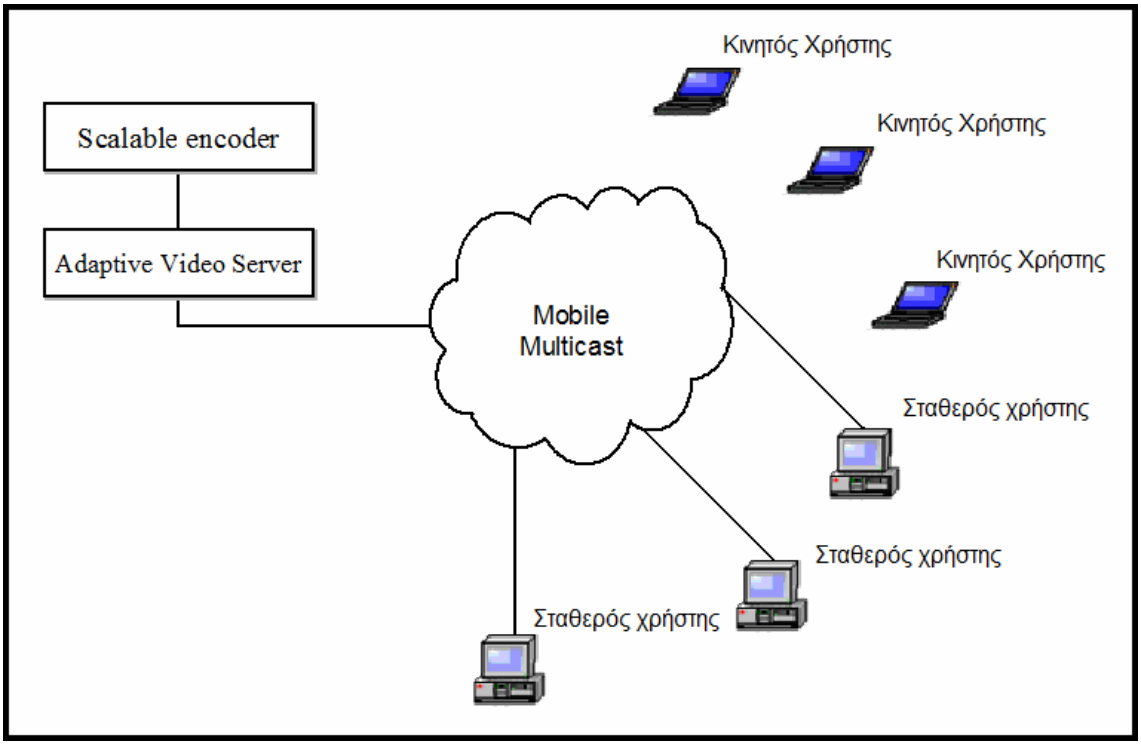
ΚΕΦΑΛΑΙΟ 1

Στόχος εργασίας και μεθοδολογία

1. Εισαγωγή

Στόχος της Διπλωματικής μου εργασίας είναι η πρόταση και αξιολόγηση ενός πρωτοκόλλου που να υποστηρίζει multicast μετάδοση πολυμεσικού περιεχομένου και συγκεκριμένα απλού και ιεραρχικού Video τόσο σε σταθερούς όσο και σε κινητούς χρήστες. Η κύρια μεθοδολογία που ακολουθήθηκε στα πλαίσια της εργασίας αυτής είναι η μελέτη των υπαρχόντων πρωτοκόλλων που να υποστηρίζουν multicast μετάδοση πολυμεσικού περιεχομένου σε κινητούς και σταθερούς χρήστες. Μελετώντας σε βάθος και σε θεωρητικό επίπεδο τα πρωτόκολλα αυτά, επέλεξα το καταλληλότερο πρωτόκολλο που να αρμόζει στις απαιτήσεις της εργασίας αυτής και το αξιολόγησα κάνοντας προσομοιώσεις στο περιβάλλον NS-2.

2. Σχηματική αναπαράσταση πρωτοκόλλου που να ικανοποιεί το στόχο μας



Σχήμα 1.1 – Σχηματική αναπαράσταση του ιδανικού πρωτοκόλλου

ΚΕΦΑΛΑΙΟ 2

Το διαδίκτυο σήμερα

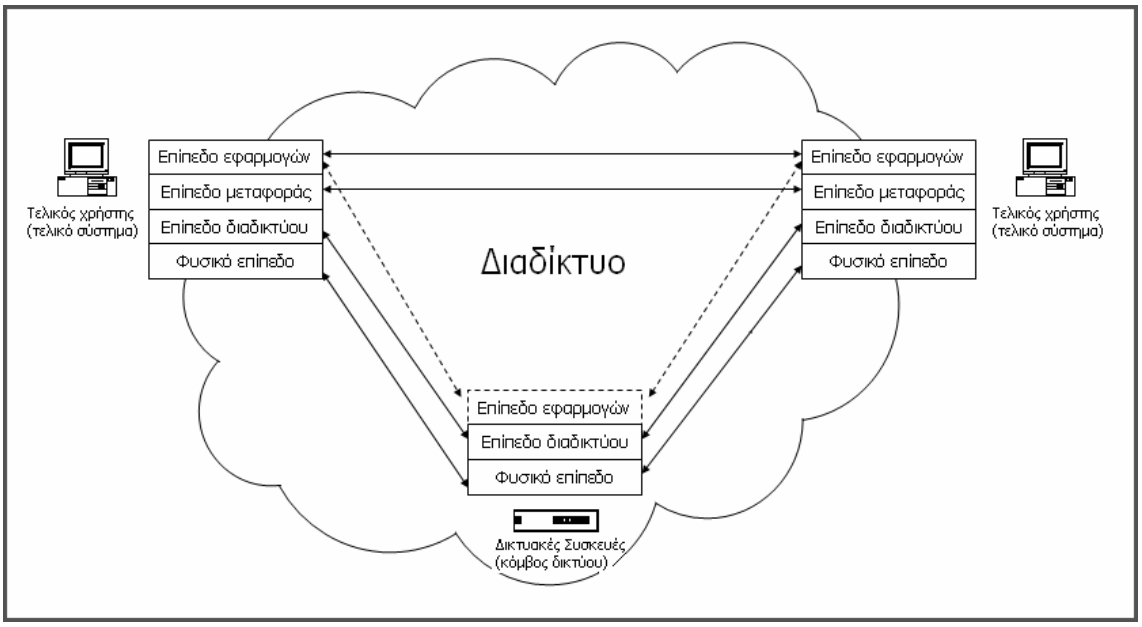
2.1 Εισαγωγή

Τα τελευταία χρόνια παρατηρείται μια αλματώδης ανάπτυξη των υπολογιστικών και δικτυακών τεχνολογιών η οποία έχει επιφέρει δραματικές αλλαγές σε όλους σχεδόν τους τομείς της ανθρώπινης δραστηριότητας και έχει μεταμορφώσει ριζικά τον τρόπο ζωής του σύγχρονου ανθρώπου. Η πρόοδος στον τομέα των δικτυακών τεχνολογιών επέφερε την εξάπλωση των τοπικών δικτύων (Local Area Networks - LAN) και τη διασύνδεσή τους σε δίκτυα ευρείας περιοχής (Wide Area Networks - WAN) με αποτέλεσμα τη δημιουργία του Διαδικτύου (Internet) το οποίο απλώνεται διαρκώς και με ραγδαίους ρυθμούς σε όλες τις δραστηριότητες και αποτελεί πλέον φυσική επέκταση του σημερινού περιβάλλοντος εργασίας του ανθρώπου και όχι μόνο. Σε αυτό το κεφάλαιο παρουσιάζεται η αρχιτεκτονική και τα χαρακτηριστικά του Διαδικτύου τα οποία σχετίζονται με το θέμα της διπλωματικής μου εργασίας.

Πιο συγκεκριμένα σε αυτό το κεφάλαιο παρουσιάζονται τα επίπεδα του Διαδικτύου, τα χαρακτηριστικά ενός δικτυακού μονοπατιού στο Διαδίκτυο, η multicast μετάδοση δεδομένων στο Διαδίκτυο, η περιγραφή του mobile IPv6 και τέλος οι κωδικοποιήσεις πολυμεσικού περιεχομένου για μετάδοση πάνω από δίκτυα.

2.2 Τα επίπεδα του Διαδικτύου

Σε αντίθεση με τα κλασικά τηλεπικοινωνιακά δίκτυα των προηγούμενων δεκαετιών τα οποία χρησιμοποιούσαν τεχνικές μεταγωγής κυκλώματος (circuit switching), το Διαδίκτυο χρησιμοποιεί την τεχνική της μεταγωγής πακέτου (packet switching). Με τη χρήση της μεταγωγής κυκλώματος, κάθε ροή δεδομένων λαμβάνει συγκεκριμένο εύρος ζώνης το οποίο εξομοιώνει ένα δεσμευμένο φυσικό κύκλωμα ανάμεσα στον αποστολέα και τον παραλήπτη για αυτή την ροή δεδομένων. Αντίθετα, με τη χρήση της μεταγωγής πακέτων τα πακέτα πολλών ροών δεδομένων πολυπλέκονται και ανταγωνίζονται για τους διαθέσιμους δικτυακούς πόρους. Επιπλέον με την χρήση της μεταγωγής πακέτου, πακέτα της ίδιας ροής δεδομένων μπορεί να ακολουθήσουν διαφορετικά μονοπάτια στο δίκτυο. Στο παρακάτω σχήμα βλέπουμε την αρχιτεκτονική του Διαδικτύου.



Σχήμα 2.1 – Αρχιτεκτονική Διαδικτύου

Στην αρχιτεκτονική του Διαδικτύου γίνεται διαχωρισμός ανάμεσα στα τελικά συστήματα (όπου εκτελούνται οι εφαρμογές του τελικού χρήστη) και στους κόμβους του δικτύου. Σε ένα σενάριο χρήσης του Διαδικτύου, μια ροή δεδομένων μεταδίδεται από ένα τελικό σύστημα σε ένα άλλο διασχίζοντας ένα ή περισσότερους κόμβους του δικτύου. Σε γενικές γραμμές στα τελικά συστήματα εκτελούνται οι εφαρμογές του τελικού χρήστη οι οποίες παράγουν δεδομένα. Τα δεδομένα των εφαρμογών του τελικού χρήστη χωρίζονται σε πακέτα δεδομένων τα οποία μεταδίδονται στη διεύθυνση προορισμού τους χρησιμοποιώντας τη λειτουργικότητα που παρέχει το επίπεδο μεταφοράς του Διαδικτύου. Το επίπεδο του Διαδικτύου είναι υπεύθυνο για τη δρομολόγηση των πακέτων δεδομένων μέσα στο δίκτυο ώστε αυτά να φτάσουν στον τελικό προορισμό τους. Στους κόμβους του δικτύου τα εισερχόμενα πακέτα μπορεί να αποθηκεύονται τοπικά για κάποιο χρονικό διάστημα σε μνήμες προσωρινής αποθήκευσης (buffers), μπορεί να τυγχάνουν επεξεργασίας, μπορεί να προωθούνται σε κάποιον άλλο κόμβο του δικτύου ή να προωθούνται στο τελικό σύστημα που είναι ο προορισμός τους. Θα πρέπει να τονίσουμε πως δεν είναι απαραίτητο όλοι οι κόμβοι του δικτύου να υλοποιούν τις λειτουργικότητες όλων των επιπέδων του Διαδικτύου, για παράδειγμα μπορεί να υπάρχουν κόμβοι οι οποίοι απλά αντιγράφουν τα πακέτα τα οποία λαμβάνουν, στη θύρα εισόδου τους στη θύρα εξόδου τους στο φυσικό επίπεδο χωρίς να χρησιμοποιούν κάποια λειτουργία του επιπέδου Διαδικτύου.

2.2.1 Το επίπεδο του Διαδικτύου

Το επίπεδο του Διαδικτύου είναι υπεύθυνο για τη δρομολόγηση των πακέτων δεδομένων στον τελικό τους προορισμό. Κατά τη διάρκεια μετάδοσης των πακέτων δεδομένων δε χρησιμοποιείται κάποια σύνδεση και είναι πιθανόν πακέτα δεδομένων της ίδιας ροής δεδομένων να ακολουθήσουν διαφορετικές διαδρομές για να φτάσουν στον προορισμό τους. Το πρωτόκολλο του επιπέδου Διαδικτύου το οποίο χρησιμοποιείται στον Διαδίκτυο είναι το IP (Internet Protocol).

Θα πρέπει να τονίσουμε ότι το επίπεδο Διαδικτύου δεν παρέχει καμία εγγύηση για την παράδοση των πακέτων δεδομένων κατά συνέπεια τα πακέτα δεδομένων μπορεί να φτάσουν καθυστερημένα στον παραλήπτη τους, να φτάσουν σε διαφορετική σειρά από τη σειρά την οποία μεταδόθηκαν ή να μην φτάσουν καθόλου (δηλαδή να χαθούν στο δίκτυο). Η απώλεια των πακέτων μπορεί να οφείλεται είτε σε λάθη κατά τη μετάδοσή τους στο φυσικό μέσο είτε λόγω υπερχείλισης της μνήμης προσωρινής αποθήκευσης (buffer) των δικτυακών συσκευών. Στα σημερινά δίκτυα η απώλεια πακέτων οφείλεται κυρίως στην υπερχείλιση της μνήμης προσωρινής αποθήκευσης των δικτυακών συσκευών και λιγότερο στα λάθη που παρουσιάζονται στο φυσικό μέσο. Η σημαντικότερη υπηρεσία την οποία παρέχουν οι δικτυακές συσκευές οι οποίες λειτουργούν στο επιπέδου του Διαδικτύου (δηλαδή οι δρομολογητές του Διαδικτύου) είναι η δρομολόγηση των πακέτων δεδομένων από τον αποστολέα στον παραλήπτη. Λόγω του γεγονότος ότι τα πακέτα δεδομένων σε μια εισερχόμενη γραμμή ενός δρομολογητή μπορεί να λαμβάνονται με διαφορετικό ρυθμό από το ρυθμό τον οποίο η δικτυακή συσκευή τα μεταδίδει σε μια εξερχόμενη γραμμή, πολλές φορές τα εισερχόμενα πακέτα δεδομένων χρειάζεται να αποθηκευτούν στη μνήμη προσωρινής αποθήκευσης πριν προωθηθούν σε μία εξερχόμενη γραμμή. Σημαντικός παράγοντας στη λειτουργία της δικτυακής συσκευής είναι ο τρόπος με τον οποίο η δικτυακή συσκευή διαχειρίζεται τα πακέτα δεδομένων τα οποία βρίσκονται αποθηκευμένα στη μνήμη προσωρινής αποθήκευσης. Πολλές τεχνικές έχουν προταθεί για αυτό τον σκοπό (γνωστές ως τεχνικές - αλγόριθμοι δρομολόγησης) και στη συνέχεια παρουσιάζονται οι πιο διαδεδομένες τεχνικές ή τεχνικές οι οποίες σχετίζονται με την παρούσα διπλωματική εργασία.

FIFO (First In First Out)

Σε αυτήν την τεχνική (η οποία είναι μια από τις πιο απλές τεχνικές αλλά και από τις πλέον χρησιμοποιούμενες στους δρομολογητές του Διαδικτύου) η μνήμη προσωρινής

αποθήκευσης αποτελείται από μία FIFO ουρά στην οποία τα πακέτα δεδομένων προωθούνται στις εξερχόμενες γραμμές με τη σειρά με την οποία έφτασαν στις εισερχόμενες γραμμές του δρομολογητή. Σε περίπτωση που η FIFO ουρά φτάσει στο μέγιστο μέγεθός της τα πακέτα τα οποία συνεχίζουν να έρχονται στις εισερχόμενες γραμμές απορρίπτονται μέχρι να υπάρχει διαθέσιμος χώρος στη FIFO ουρά. Αυτή η τεχνική μπορεί να υλοποιηθεί σχετικά εύκολα αλλά πάσχει από προβλήματα συγχρονισμού και άδικου διαχωρισμού του εύρος ζώνης ανάμεσα σε ροές δεδομένων οι οποίες ανταγωνίζονται για το διαθέσιμο εύρος ζώνης. Το πρόβλημα συγχρονισμού των ροών δεδομένων είναι ιδιαίτερα έντονο όταν TCP ροές δεδομένων διασχίζουν δρομολογητές οι οποίοι είναι υπερφορτωμένοι και απορρίπτουν συχνά πακέτα. Λόγω της συμφόρησης η οποία παρουσιάζεται, όλες οι TCP συνδέσεις χρησιμοποιούν τον μηχανισμό αποτροπής της συμφόρησης τον οποίο υλοποιούν και μειώνουν το ρυθμό μετάδοσης δεδομένων περίπου την ίδια χρονική στιγμή με αποτέλεσμα οι δρομολογητές να μην είναι πλέον υπερφορτωμένοι. Στη συνέχεια οι TCP συνδέσεις αυξάνουν το ρυθμό μετάδοσης δεδομένων μέχρι οι δρομολογητές να υπερφορτωθούν ξανά όποτε οι TCP συνδέσεις ξανά μειώνουν το ρυθμό μετάδοσης δεδομένων. Το φαινόμενο αυτό επαναλαμβάνεται συνεχώς και οδηγεί σε μη πλήρη αξιοποίηση των υπαρχόντων δικτυακών πόρων. Επιπλέον, η FIFO ουρά μπορεί να οδηγήσει και σε προβλήματα δικαιοσύνης λόγω του ότι η πιθανότητα να απορριφθεί ένα πακέτο δεδομένων είναι η ίδια για όλες τις ροές δεδομένων οι οποίες συναγωνίζονται για το διαθέσιμο εύρος ζώνης. Το γεγονός αυτό έχει ως αποτέλεσμα μια άπληστη πηγή δεδομένων (η οποία δεν εφαρμόζει κάποιο μηχανισμό αποφυγής της συμφόρησης και μεταδίδει δεδομένα με υψηλό ρυθμό μετάδοσης με αποτέλεσμα να χρησιμοποιεί μεγάλο μέρος του διαθέσιμου εύρους ζώνης) να έχει το ίδιο ποσοστό απώλειας των πακέτων δεδομένων της με μια πηγή δεδομένων η οποία καταναλώνει πολύ μικρότερο μέρος του διαθέσιμου εύρους ζώνης. Αυτό δεν είναι δίκαιο λόγω του γεγονότος ότι παρόλο που το πρόβλημα της συμφόρησης οφείλεται κυρίως στην πηγή με τον υψηλό ρυθμό μετάδοσης δεδομένων αυτή η πηγή συνεχίζει να λαμβάνει μεγάλο μέρος του διαθέσιμου εύρους ζώνης.

RED (Random Early Detection)

Η τεχνική RED αποτελείται από μια ουρά FIFO στην οποία έχουν ορισθεί δύο όρια. Όταν το μέγεθος της ουράς ξεπεράσει το πρώτο όριο η ουρά απορρίπτει τα εισερχόμενα πακέτα δεδομένων με κάποια ορισμένη πιθανότητα. Όταν το μέγεθος της

ουράς ξεπεράσει και το δεύτερο όριο η ουρά απορρίπτει όλα τα εισερχόμενα πακέτα δεδομένων. Με αυτή την τεχνική το μήκος της ουράς διατηρείται μικρό και ταυτόχρονα αποφεύγεται το φαινόμενο του συγχρονισμού. Η τεχνική RED παρουσιάζει τα προβλήματα δικαιοσύνης τα οποία παρουσιάζει και η ουρά FIFO. Η τεχνική RED υποστηρίζεται από τους περισσότερους σύγχρονους δρομολογητές αν και δεν χρησιμοποιείται σε πολλούς εγκατεστημένους δρομολογητές του Διαδικτύου.

Δίκαιη διαχείριση ουρών (Fair queuing mechanism)

Οι περισσότερες από αυτές τις τεχνικές πετυχαίνουν ομοιόμορφη κατανομή του εύρους ζώνης και απομονώνουν τις άπληστες πηγές δεδομένων. Κάποιες άλλες τεχνικές δεσμεύουν συγκεκριμένο εύρος ζώνης για κάθε ροή δεδομένων και εγγυώνται συγκεκριμένα όρια τόσο στον εύρος ζώνης όσο και στην καθυστέρηση. Οι παραπάνω τεχνικές είναι σε γενικές γραμμές πολύπλοκες στην χρήση τους (ειδικά στην περίπτωση που οι δρομολογητές δε διαχειρίζονται από τον ίδιο οργανισμό όπως στην περίπτωση του Διαδικτύου) και στις μέρες μας βρίσκουν εφαρμογή κυρίως στα εσωτερικά δίκτυα οργανισμών και λιγότερο στους κεντρικούς δρομολογητές του Διαδικτύου.

2.2.2 Τρόποι μετάδοσης των πακέτων δεδομένων στο Διαδίκτυο

Στο επίπεδο του Διαδικτύου τα πακέτα δεδομένων μπορούν να μεταδοθούν με τους παρακάτω τρόπους:

Unicast

Σε αυτήν την περίπτωση τα πακέτα δεδομένων μεταδίδονται από την πηγή - αποστολέα των δεδομένων σε ένα συγκεκριμένο παραλήπτη.

Broadcast

Σε αυτήν την περίπτωση τα πακέτα δεδομένων μεταδίδονται από την πηγή - αποστολέα δεδομένων σε όλους τους σταθμούς του δικτύου. Η broadcast μετάδοση συνήθως χρησιμοποιείται στα τοπικά δίκτυα για την επικοινωνία με όλους του σταθμούς του δικτύου.

Multicast

Σε αυτήν την περίπτωση τα πακέτα δεδομένων μεταδίδονται από την πηγή - αποστολέα σε μια ομάδα από παραλήπτες και είναι η περίπτωση που θα μας απασχολήσει για την εκπόνηση της συγκεκριμένης διπλωματικής εργασίας.

Στην τελευταία έκδοση του πρωτοκόλλου IP, την έκδοση 6 (IP version 6 - IPv6) ορίζεται και ένας επιπλέον τρόπος μετάδοσης:

Anycast

Σε αυτή την περίπτωση τα πακέτα δεδομένων μεταδίδονται σε ένα παραλήπτη (όποιος λάβει το πακέτο δεδομένων πρώτος) από μια συγκεκριμένη ομάδα παραληπτών.

Η multicast μετάδοση δεδομένων είναι ιδιαίτερα σημαντική λόγω του ότι οδηγεί σε σημαντική εξοικονόμηση δικτυακών πόρων. Στο Διαδίκτυο, για την υλοποίηση της multicast μετάδοσης χρησιμοποιούνται οι multicast διευθύνσεις οι οποίες αντιπροσωπεύουν μια ομάδα από αποστολείς και παραλήπτες την οποία ομάδα ονομάζουμε multicast σύνοδο (multicast session). Οι αποστολείς μεταδίδουν τα πακέτα δεδομένων τα οποία έχουν ως αποδέκτη την multicast διεύθυνση της multicast συνόδου. Οι παραλήπτες πρέπει να ενημερώσουν το δίκτυο (πιο συγκεκριμένα τον δρομολογητή στον οποίο συνδέονται) για την πρόθεσή τους να συμμετέχουν στην multicast σύνοδο ώστε να λαμβάνουν τα multicast πακέτα δεδομένων ή για την πρόθεσή τους να αποχωρήσουν ώστε να σταματήσουν να λαμβάνουν τα multicast πακέτα δεδομένων.

2.2.3 Το επίπεδο μεταφοράς

Το επίπεδο μεταφοράς του Διαδικτύου είναι υπεύθυνο για την από άκρο σε άκρο μετάδοση των πακέτων δεδομένων στο τελικό σύστημα και πιο συγκεκριμένα στην διεργασία του τελικού συστήματος στην οποία απευθύνονται τα πακέτα δεδομένων. Στο επίπεδο μεταφοράς του Διαδικτύου υπάρχουν δύο πρωτόκολλα μεταφοράς: Το TCP (Transmission Control Protocol) και το UDP (User Datagram Protocol). Το TCP είναι ένα πρωτόκολλο το οποίο χρησιμοποιεί την έννοια της σύνδεσης (connection oriented) και παρέχει αξιόπιστη μεταφορά των πακέτων δεδομένων στην σειρά με την οποία μεταδόθηκαν. Αντίθετα, το UDP είναι ένα πρωτόκολλο το οποίο δε χρησιμοποιεί την έννοια της σύνδεσης και δεν παρέχει οποιεσδήποτε εγγυήσεις για την αξιόπιστη μετάδοση των πακέτων δεδομένων.

2.2.3.1 Το πρωτόκολλο Transmission Control Protocol (TCP)

Το πρωτόκολλο TCP [1] παρέχει μια αξιόπιστη υπηρεσία βασισμένη σε συνδέσεις η οποία εγγυάται την αξιόπιστη μετάδοση των πακέτων δεδομένων στην σειρά με την οποία μεταδόθηκαν ανάμεσα σε δύο τελικά συστήματα. Η λειτουργία του TCP είναι αρκετά περίπλοκη αν σκεφτεί κανείς ότι λειτουργεί πάνω από το επίπεδο του Διαδικτύου το οποίο δεν παρέχει ούτε υπηρεσίες σύνδεσης ούτε εγγυήσεις για τη

μετάδοση των πακέτων δεδομένων. Οι υπηρεσίες τις οποίες παρέχει το TCP μπορούν να διαχωριστούν στις παρακάτω:

Εδραίωση και τερματισμός σύνδεσης

Πριν την έναρξη της μετάδοσης των πακέτων δεδομένων τα τελικά συστήματα θα πρέπει να ανταλλάξουν κατάλληλα μηνύματα συγχρονισμού (SYN messages) ώστε να εδραιωθεί η σύνδεση μεταξύ τους. Επίσης για να τερματιστεί η σύνδεση μεταξύ των δύο τελικών συστημάτων θα πρέπει να ανταλλαχθούν τα κατάλληλα μηνύματα (FIN messages).

Αξιόπιστη μετάδοση

Προκειμένου να επιτύχει την αξιόπιστη μετάδοση των πακέτων δεδομένων, ο αποστολέας της TCP κίνησης απαιτεί από τον παραλήπτη να επιβεβαιώνει την ορθή λήψη των πακέτων που του μεταδίδει. Για το σκοπό αυτό ο αποστολέας της TCP κίνησης διατηρεί ένα αντίγραφο κάθε πακέτου το οποίο μεταδίδει μέχρι αυτό να επιβεβαιωθεί από τον παραλήπτη. Εάν ο αποστολέας της TCP κίνησης δεν λάβει επιβεβαίωση για ένα πακέτο δεδομένων μετά από χρόνο t_{out} από τότε που το μετάδωσε, το ξαναμεταδίδει. Η τιμή του t_{out} πρέπει να επιλεγεί με προσοχή ώστε ο αποστολέας να αντιλαμβάνεται έγκαιρα τις απώλειες των πακέτων αλλά και να μην ξαναμεταδίδει πακέτα τα οποία βρίσκονται καθοδόν προς τον παραλήπτη τους. Ο παρακάτω αλγόριθμος χρησιμοποιείται για τον υπολογισμό του t_{out}, ο οποίος βασίζεται σε υπολογισμούς του χρόνου καθυστέρησης μετάδοσης μετά επιστροφής ενός πακέτου, χρόνος RTT (Round Trip Time):

$$t_{diff} = t_{measured} - t_{RTT}$$

$$t_{RTT} = t_{RTT} + \delta * t_{diff}$$

$$\sigma_n = \sigma_{n-1} + \rho(|t_{diff}| - \sigma_{n-1})$$

Όπου $t_{measured}$ είναι ο χρόνος RTT ο οποίος μετριέται στο δίκτυο, όπου t_{RTT} είναι ο φιλτραρισμένος χρόνος RTT, σ_n η φιλτραρισμένη μέση διακύμανση και t_{out} είναι ο χρόνος λήξης για την επαναμετάδοση ενός πακέτου δεδομένων. Ο χρόνος $t_{measured}$ μετριέται βάσει του χρόνου που απαιτείται από την αποστολή ενός πακέτου από τον

αποστολέα μέχρι αυτός να λάβει την επιβεβαίωση από τον παραλήπτη. Η παράμετρος δ έχει τιμές από 0 έως 1 και καθορίζει πόσο ευαίσθητος θα είναι ο υπολογισμός του t_{RTT} στις μετρήσεις οι οποίες λαμβάνονται από το δίκτυο και συνήθως έχει την τιμή 0.125. Η παράμετρος ρ καθορίζει πόσο γρήγορα τα νέα δείγματα επηρεάζουν την μέση διακύμανση σ_n και συνήθως έχει την τιμή 0.25. Η παράμετρος η καθορίζει το πόσο η μέση διακύμανση σ_n επηρεάζει την τιμή του χρόνου t_{out}. Αρχικά είχε προταθεί η τιμή 2 για την παράμετρο η όμως στην έκδοση Reno του TCP προτείνεται η τιμή 4.

Έλεγχος ροής και συμφόρησης

Το TCP παρέχει έναν ιδιαίτερα αξιόπιστο μηχανισμό για τον έλεγχο ροής και συμφόρησης στον οποίο οφείλεται και η σημαντική χρήση και εξάπλωση του πρωτοκόλλου TCP. Ο μηχανισμός αυτός βασίζεται στην τεχνική του κυλιόμενου παραθύρου για τη μετάδοση των πακέτων δεδομένων με την χρήση του στην οποία αποφεύγονται φαινόμενα συμφόρησης που οδηγούν σε απώλειες πακέτων δεδομένων. Ο μηχανισμός αυτό λειτουργεί ως εξής: Ο αποστολέας της TCP κίνησης διατηρεί μια μνήμη προσωρινής αποθήκευσης (buffer) η οποία ονομάζεται «παράθυρο μετάδοσης» (transmission window) στην οποία εισάγει ένα αντίγραφο από κάθε πακέτο δεδομένων το οποίο μεταδίδει. Όσο υπάρχει διαθέσιμος χώρος στη μνήμη προσωρινής αποθήκευσης, ο αποστολέας μεταδίδει πακέτα στο δίκτυο και σε περίπτωση που η μνήμη προσωρινής αποθήκευσης γεμίσει, ο αποστολέας «παγώνει» τη μετάδοση πακέτων. Ο παραλήπτης των TCP πακέτων επιβεβαιώνει τη λήψη των πακέτων δεδομένων και στο πακέτο επιβεβαίωσης το οποίο στέλνει, περιλαμβάνει την ελεύθερη χωρητικότητα της δικής του μνήμης προσωρινής αποθήκευσης και τον αριθμό του τελευταίου πακέτου το οποίο έλαβε από τον αποστολέα. Ο αποστολέας όταν λαμβάνει ένα πακέτο επιβεβαίωσης από τον παραλήπτη, διαγράφει από τη μνήμη προσωρινής αποθήκευσής του τα πακέτα δεδομένων που επιβεβαιώνει ο παραλήπτης και έτσι δημιουργείται ελεύθερος χώρος στην μνήμη προσωρινής αποθήκευσης για τη μετάδοση επιπλέον πακέτων δεδομένων. Η πρώτη έκδοση του TCP λάμβανε υπόψη της μόνο τα θέματα της αξιοπιστίας και του έλεγχου ροής και προσπαθούσε να μη μεταδίδει περισσότερα πακέτα δεδομένων από αυτά τα οποία μπορεί να διαχειριστεί ο παραλήπτης. Η τακτική αυτή έχει ικανοποιητικά αποτελέσματα όταν ο περιοριστικός παράγοντας στη μετάδοση των δεδομένων είναι ο παραλήπτης και αντίθετα, δε λειτουργεί ικανοποιητικά όταν περιοριστικός παράγοντας είναι το δίκτυο. Σε αυτή την

περίπτωση, η παραπάνω τακτική μπορεί να οδηγήσει στο φαινόμενο του «congestion collapse». Σύμφωνα με το φαινόμενο αυτό, ενώ υπάρχει υψηλή κίνηση στο δίκτυο, το ποσοστό της «ωφέλιμης» κίνησης είναι μικρό και το μεγαλύτερο μέρος της κίνησης οφείλεται στην επαναμετάδοση πακέτων που είτε έχουν παραδοθεί στον παραλήπτη (αλλά ο αποστολέας δεν έχει λάβει ακόμη την επιβεβαίωση από τον παραλήπτη) είτε βρίσκονται στο δίκτυο ακόμη. Για το λόγο αυτό προτείνονται οι παρακάτω καταστάσεις, με τη χρήση των οποίον υπολογίζεται το μέγεθος του παραθύρου μετάδοσης:

«Αργό ξεκίνημα» (slow start)

Στην κατάσταση «αργό ξεκίνημα», ο αποστολέας της TCP κίνησης χρησιμοποιεί ένα ακόμα «παράθυρο μετάδοσης» το οποίο ονομάζεται «παράθυρο συμφόρησης» (congestion window) η χωρητικότητα του οποίου μετριέται σε αριθμό πακέτων. Όταν εδραιώνεται μια TCP σύνδεση, το παράθυρο συμφόρησης έχει μέγεθος ένα και κάθε φορά που λαμβάνεται μια επιβεβαίωση από τον παραλήπτη, το παράθυρο συμφόρησης αυξάνεται κατά το αριθμό των πακέτων που επιβεβαιώνονται. Στη συνέχεια, ο αποστολέας μπορεί να μεταδίδει τόσα πακέτα δεδομένων όσο η μικρότερη τιμή από το παράθυρο συμφόρησης και το μέγεθος παραθύρου που περιέχεται στο πακέτο επιβεβαίωση.

«Αποφυγή συμφόρησης» (congestion avoidance)

Με τη χρήση του παραπάνω μηχανισμού «αργό ξεκίνημα», η TCP σύνδεση αυξάνει το ρυθμό μετάδοσης των δεδομένων και όταν η χωρητικότητα του δικτύου φτάσει στο άνω όριο της, τότε οι ενδιάμεσοι δρομολογητές αρχίζουν να απορρίπτουν πακέτα δεδομένων. Προκείμενου να αποφευχθεί το φαινόμενο αυτό (η κατάσταση συμφόρησης) το μέγεθος του παραθύρου συμφόρησης αυξάνεται το πολύ κατά ένα πακέτο κάθε χρόνο RTT σε αντίθεση με την κατάσταση «αργό ξεκίνημα».

Από τη δημιουργία του Διαδικτύου, διάφορες εκδόσεις του TCP έχουν προταθεί, και στη συνέχεια αναφέρουμε τις πιο σημαντικές εκδόσεις του TCP:

Tahoe - TCP

Το Tahoe - TCP υλοποιήθηκε το 1988 στο 4.3 BSD λειτουργικό. Το Tahoe – TCP χρησιμοποιεί τους αλγορίθμούς «αργό ξεκίνημα» και «αποφυγή συμφόρησης» όπως επίσης και αλγόριθμο «γρήγορης επαναμετάδοσης» (fast retransmission).

Reno - TCP

Το Reno - TCP υλοποιήθηκε στο 1990 και αποτελεί την πιο ευρέως χρησιμοποιούμενη έκδοση του TCP. Το Reno - TCP χρησιμοποιεί τους αλγορίθμους «αργό ξεκίνημα», «αποφυγή συμφόρησης», «γρήγορης επαναμετάδοσης» και επιπλέον χρησιμοποιεί ένα μηχανισμό «γρήγορης επαναφοράς» (fast recovery) σύμφωνα με τον οποίο ο αποστολέας της TCP κίνησης ξανά μεταδίδει ένα χαμένο πακέτο δεδομένων αφού λάβει τρεις διπλές επιβεβαιώσεις πακέτων.

New - Reno TCP

Η έκδοση Tahoe - TCP μειώνει το παράθυρο μετάδοσης στο ένα πακέτο μετά από μια απώλεια πακέτου, συμπεριφορά η οποία είναι αρκετά συντηρητική και μπορεί να οδηγήσει σε μικρή αξιοποίηση των πόρων του δικτύου. Αντίθετα το Reno - TCP μπορεί να οδηγήσει σε μικρή αξιοποίηση των πόρων του δικτύου όταν υπάρχουν πολλαπλές απώλειες πακέτων στον χρόνο RTT.

SACK - TCP

Στο SACK - TCP (Selective ACKnowledgment TCP) γίνεται επιλεκτική επιβεβαίωση πακέτων από τον παραλήπτη. Ο παραλήπτης συμπεριλαμβάνει σε κάθε πακέτο επιβεβαίωσης έναν αριθμό από επιβεβαιώσεις οι οποίες επιβεβαιώνουν μη συνεχόμενες ομάδες πακέτων δεδομένων που έχουν ληφθεί σωστά. Αυτό επιτρέπει στον παραλήπτη να αντιλαμβάνεται έγκαιρα τις απώλειες δεδομένων και να αποφεύγει περιττές καθυστερήσεις με αποτέλεσμα την καλύτερη απόδοση.

Vegas - TCP

Το Vegas - TCP χρησιμοποιεί μια βελτιωμένη μέθοδο για να υπολογίζει το μέγεθος των παραθύρων η οποία βασίζεται στην απόκλιση των μετρήσεων του χρόνου RTT.

ECN - TCP

Έχουν προταθεί διάφορες επεκτάσεις στο TCP που εκμεταλλεύονται τις πληροφορίες για πρόωρη ανίχνευση της συμφόρησης (ECN - Early Congestion Notification) που δύναται να παρέχει το δίκτυο. Σε αυτήν την περίπτωση, οι δρομολογητές του δικτύου μαρκάρουν ένα συγκεκριμένο bit στις επικεφαλίδες των πακέτων δεδομένων όταν αρχίζουν να υπερφορτώνονται. Το TCP λαμβάνοντας αυτήν την πληροφορία μπορεί να μειώσει το ρυθμό μετάδοσης πριν αρχίσουν να απορρίπτονται πακέτα δεδομένων και με αυτόν τον τρόπο πετυχαίνει καλύτερη απόδοση. Το βασικότερο μειονέκτημα αυτών

την επεκτάσεων είναι ότι απαιτούνται αλλαγές στην υποδομή του δικτύου και όχι μόνο στα τελικά συστήματα όπως συμβαίνει στις άλλες επεκτάσεις του TCP.

2.2.3.2 Το πρωτόκολλο User Datagram Protocol (UDP)

Αντίθετα με το TCP το οποίο είναι ένα πρωτόκολλο προσανατολισμένο στη σύνδεση και απαιτεί την εδραίωση σύνδεσης πριν την μετάδοση των δεδομένων, το UDP [2] είναι ένα πρωτόκολλο το οποίο δε χρησιμοποιεί συνδέσεις και παρέχει μια πολύ πιο απλή υπηρεσία. Το UDP μεταδίδει πακέτα δεδομένων (τα οποία ονομάζονται datagrams) από ένα αποστολέα σε ένα παραλήπτη χωρίς να εγγυάται τη μετάδοση των πακέτων στον προορισμό τους. Το γεγονός ότι το TCP απαιτεί από τον παραλήπτη να επιβεβαιώνει στον αποστολέα τη λήψη κάθε πακέτου δεδομένων, μπορεί να δημιουργήσει το φαινόμενο της «πλημμύρας πληροφοριών ανάδρασης» (feedback implosion) στην περίπτωση μιας σύνδεσης ενός σημείου προς πολλά σημεία (point to multipoint). Κατά συνέπεια η multicast μετάδοση δεδομένων υλοποιείται αποτελεσματικότερα με τη χρήση του UDP στο Διαδίκτυο. Επιπλέον το TCP μπορεί να διακόψει τη μετάδοση των δεδομένων όσο περιμένει ένα πακέτο επιβεβαίωσης ή μπορεί να μειώσει το ρυθμό μετάδοσης δεδομένων δραστικά όταν αντιληφθεί απώλεια πακέτων. Αυτή η συμπεριφορά είναι αποδεκτή για εφαρμογές που η εγγυημένη παράδοση δεδομένων είναι απαραίτητη όμως δεν είναι κατάλληλη για εφαρμογές όπως οι εφαρμογές πολυμέσων οι οποίες λαμβάνουν πληροφορία την οποία παρουσιάζουν στο χρήστη. Πολλές φορές σε αυτές τις εφαρμογές η έγκαιρη μετάδοση είναι πιο σημαντική από 100% ορθή μετάδοση των δεδομένων. Για τους παραπάνω λόγους οι εφαρμογές πολυμέσων στο Διαδίκτυο (πχ τηλεφωνία με χρήση του πρωτοκόλλου IP) στηρίζονται στη μετάδοση δεδομένων με την χρήση του πρωτοκόλλου UDP. Το γεγονός ότι το πρωτόκολλο UDP δεν υποστηρίζει έλεγχο ροής και αξιόπιστη μετάδοση δεδομένων, έχει ως αποτέλεσμα τα παραπάνω χαρακτηριστικά να υλοποιούνται στο επίπεδο των εφαρμογών.

2.2.4 Το επίπεδο εφαρμογών

Οι εφαρμογές είναι υπεύθυνες να παράγουν δεδομένα όπως για παράδειγμα βίντεο, ήχο, εικόνες και κείμενο. Η επικοινωνία ανάμεσα σε εφαρμογές είναι από άκρο σε άκρο πράγμα που σημαίνει ότι μια εφαρμογή μεταδίδει τα δεδομένα της σε μια άλλη εφαρμογή χωρίς να απαιτείται η άμεση αλληλεπίδραση των εφαρμογών με τους κόμβους του δικτύου. Στις περισσότερες περιπτώσεις οι κόμβοι του δικτύου προωθούν

και δρομολογούν τα δεδομένα στο Διαδίκτυο. Επιπλέον οι εφαρμογές μπορεί να χρησιμοποιούν πρωτόκολλα στο επίπεδο εφαρμογών τα οποία υλοποιούν συγκεκριμένες λειτουργίες (όπως για παράδειγμα το μαρκάρισμα κάποιον πλαισίων ενός βίντεο) και παρέχουν πληροφορίες για τα δεδομένα ή επιπλέον δυνατότητες για τη διαχείριση των δεδομένων. Τέτοια πρωτόκολλα στο επίπεδο των εφαρμογών είναι και τα πρωτόκολλα RTP / RTCP.

2.3 Mobile IPv6

Τα κύρια χαρακτηριστικά του IPv6 [3] που είναι σημαντικά για τα κινητά και ασύρματα δίκτυα είναι ο ικανοποιητικός αριθμός IP διευθύνσεων, η υλοποίηση header ασφαλείας, οι επιλογές προορισμού για ικανοποιητικό rerouting, ο αυτόματος καθορισμός διευθύνσεων, η ανάκαμψη από λάθη χωρίς στην κατάσταση soft – state bottleneck και άλλα.

2.3.1 Βασικές Ορολογίες

Care-of-address

Μια IP διεύθυνση που συσχετίζεται με έναν κινητό κόμβο όταν αυτός βρίσκεται σε ένα ξένο υποδίκτυο και κατ' επέκταση σε μια ξένη σύνδεση. Το πρόθεμα της διεύθυνσης αυτής είναι το πρόθεμα του ξένου δικτύου. Ανάμεσα στις διάφορες care-of addresses που μπορεί να έχει ένας κινητός κόμβος, αυτή που συσχετίζεται με τον Home Agent(HA) του κόμβου λέγεται primary care-of address.

Δέσιμο (Binding)

Η συσχέτιση του Home Agent (HA) ενός κινητού κόμβου με μια care-of address (COA), μαζί με την εναπομείναντα διάρκεια ζωής της σύνδεσης αυτής.

Correspondent Node (CN)

Ένας κόμβος με τον οποίο ο κινητός κόμβος είναι σε επικοινωνία. Ο CN μπορεί να είναι είτε κινητός είτε σταθερός.

Foreign Link (FL)

Οποιαδήποτε άλλη σύνδεση εκτός από το Home Link του κόμβου.

Foreign Subnet Prefix (FSP)

Οποιοδήποτε άλλο IP πρόθεμα υποδικτύου εκτός από το home πρόθεμα του δικτύου.

Home Agent (HA)

Ένας δρομολογητής στο home link του κινητού κόμβου στον οποίο έχει εγγράψει ο κινητός κόμβος την COA του. Καθώς ο κινητός κόμβος είναι εκτός του home network του, ο HA δέχεται πακέτα που προορίζονται για την home address του κόμβου αυτού, τα ενθυλακώνει και τα στέλνει στην εγγεγραμμένη COA του κόμβου.

Home Link (HL)

Η σύνδεση στην οποία καθορίζεται το Home Subnet Prefix ενός κινητού κόμβου. Οι τυπικοί μηχανισμοί δρομολόγησης, θα παραδώσουν πακέτα που προορίζονται για κάποιο κόμβο στο HL του.

Home Registration

Εγγραφή ενός κινητού κόμβου σε μια COA.

Home Subnet Prefix

Το IP subnet prefix που αντιστοιχεί στο home address ενός κινητού κόμβου.

2.3.2 Επιλογές προορισμού

Binding Update

Χρησιμοποιείται από ένα MN για να ενημερώσει το HA ή κάθε άλλο CN για την τρέχουσα care-of-address.

Binding Acknowledgement

Χρησιμοποιείται για την επιβεβαίωση λήψης ενός μηνύματος Binding Update.

Binding Request

Χρησιμοποιείται από κάποιο κόμβο για να ζητήσει από τον MN να στείλει πληροφορίες με την μορφή Binding Update με την τρέχουσα διεύθυνση care-of-address.

Home Address

Χρησιμοποιείται σε πακέτα τα οποία στέλνονται από τον κινητό κόμβο για να ενημερώσουν τον παραλήπτη για τη διεύθυνση home address του MN.

2.3.3 Δομές δεδομένων

Binding Cache

Κάθε Mobile IPv6 κόμβος έχει μια binding Cache μνήμη η οποία χρησιμοποιείται για να φυλάει τα bindings για άλλους κόμβους. Για παράδειγμα όταν ένας κόμβος λάβει ένα Binding Update, θα το φυλάξει στην Binding Cache του. Κάθε φορά που ένα πακέτο θα αποσταλεί, γίνεται μια αναζήτηση στη Binding Cache για μια εγγραφή που

αφορά τον παραλήπτη. Αν βρεθεί μια εγγραφή, το πακέτο αποστέλνεται στη COA του παραλήπτη και όχι στον HA του.

Binding Λίστα Ενημέρωσης

Κάθε κινητός κόμβος έχει μια binding λίστα ενημέρωσης η οποία χρησιμοποιείται για να αποθηκεύονται πληροφορίες σχετικά με κάθε Binding Update που έχει σταλεί από αυτόν τον κινητό κόμβο και του οποίου δεν έχει λήξη ακόμη η διάρκεια ζωής του. Περιέχει όλα τα Binding Updates που έχουν σταλεί σε οποιοδήποτε CN, είτε σταθερό είτε κινητό και στον HA του.

Home Agents List

Για κάθε HL όπου ένας κόμβος λειτουργεί σαν HA, δημιουργεί μια λίστα η οποία περιέχει πληροφορίες για όλους τους άλλους HA σε αυτή τη σύνδεση. Οι πληροφορίες λαμβάνονται από τις διαφημίσεις των δρομολογητών που στέλνουν οι HAs. Στις διαφημίσεις αυτές, είναι ενεργοποιημένο το home agent bit, αν ο αποστολέας είναι HA σε αυτή τη σύνδεση. Οι πληροφορίες για τους άλλους HAs χρησιμοποιούνται από τον μηχανισμό Dynamic Home Agent Discovery. Ο μηχανισμός αυτός επιτρέπει σε ένα κινητό κόμβο να ανακαλύπτει δυναμικά την IP διεύθυνση ενός HA στο HL του, στον οποίο μπορεί να εγγράψει την primary care-of address του κατά την διάρκεια που βρίσκεται μακριά από το home δίκτυό του.

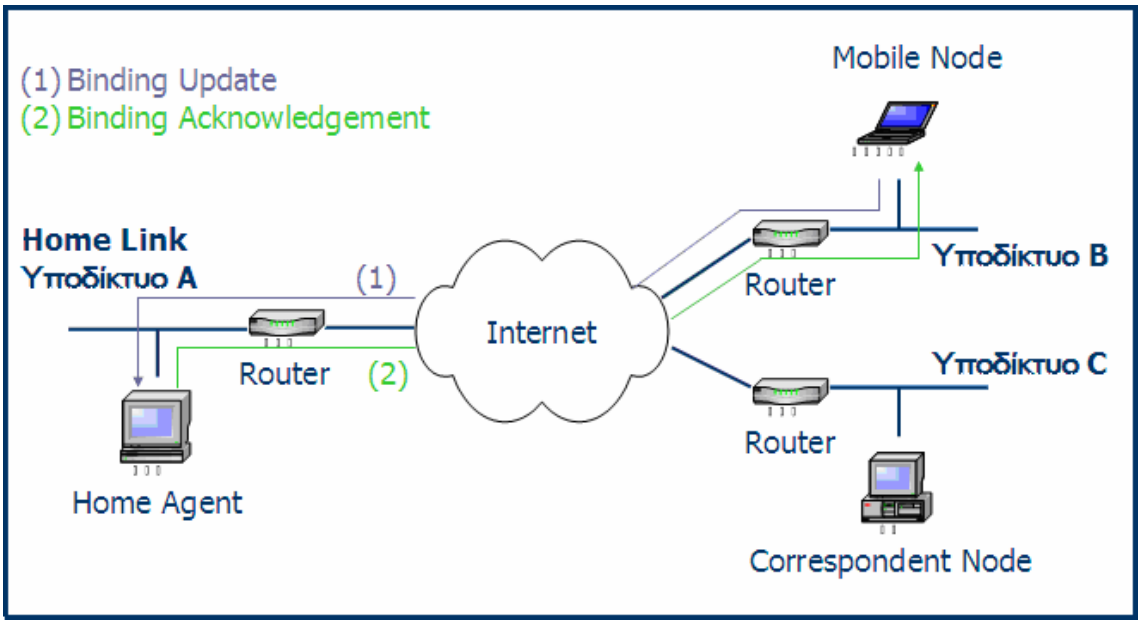
2.3.4 Λειτουργία του Mobile IPv6

Για να περιγράψουμε τη βασική λειτουργία του Mobile IPv6, χρησιμοποιούμε ένα δίκτυο το οποίο περιέχει τρία υποδίκτυα (Υποδίκτυο Α, Υποδίκτυο Β, Υποδίκτυο C). Στο Υποδίκτυο Α υπάρχει ένας δρομολογητής ο οποίος λειτουργεί σαν HA. Αυτή η σύνδεση είναι επίσης το HL κάποιο κινητού κόμβου ο οποίος έχει μετακινηθεί από τη σύνδεση Α σε κάποια άλλη σύνδεση (έστω σύνδεση Β). Ακόμη, υπάρχει ένας CN (κινητός ή σταθερός) στην τρίτη σύνδεση (έστω σύνδεση Γ).

Διαδικασία Εγγραφής Home Agent (Home Agent Registration)

Μόλις ένας κινητός κόμβος αναγνωρίσει ότι έχει μετακινηθεί από μια σύνδεση σε άλλη και ανακαλύπτει ένα καινούριο default δρομολογητή, κάνει τη λειτουργία του address autoconfiguration (stateful ή stateless). Χρησιμοποιεί αυτή τη νέα διεύθυνση σαν την COA του. Το πρόθεμα της COA είναι το πρόθεμα της σύνδεσης στην οποία είναι τώρα ο κινητός κόμβος. Έτσι όλα τα πακέτα που προορίζονται για τον κόμβο αυτό,

αποστέλνονται στην σύνδεση στην οποία βρίσκεται. Ο κινητός κόμβος εγγράφει την COA του στον HA του, μέσω της HL.



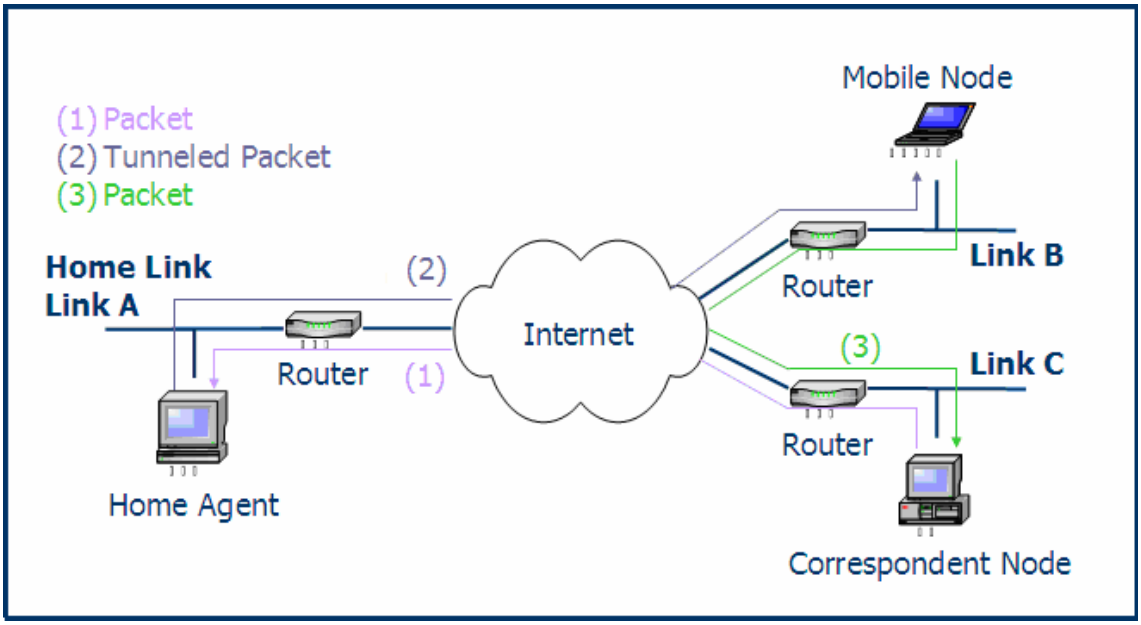
Σχήμα 2.2 – Διαδικασία εγγραφής του Home Agent [4]

Ο κινητός κόμβος στο υποδίκτυο B στέλνει ένα πακέτο στον HA του που περιέχει μια Binding Update επιλογή προορισμού. Ο HA με τη σειρά του εγγράφει αυτό το Binding και επιστρέφει ένα πακέτο με μια Binding Acknowledgement επιλογή προορισμού

Βελτιστοποίηση Διαδρομής (Route Optimization) .

Αφού ο HA έχει εγγράψει την COA του κινητού κόμβου, αναλαμβάνει πακέτα που προορίζονται στο home address του κινητού δικτύου. Έτσι χρησιμοποιεί τη μέθοδο Proxy Neighbor Discovery. Με την μέθοδο αυτή, ο HA στέλνει μια διαφήμιση γείτονα (Neighbor Advertisement) στο HL εκ μέρους του κινητού κόμβου. Ο HA απαντά και σε γειτονικούς ερεθισμούς (Neighbor Solicitations) εκ μέρους του κινητού κόμβου. Κάθε πακέτο που αποστέλνεται στο home address του κινητού κόμβου, παραλαμβάνεται από τον HA και αποστέλνεται στην COA του κόμβου, με IPv6 ενθυλάκωση.

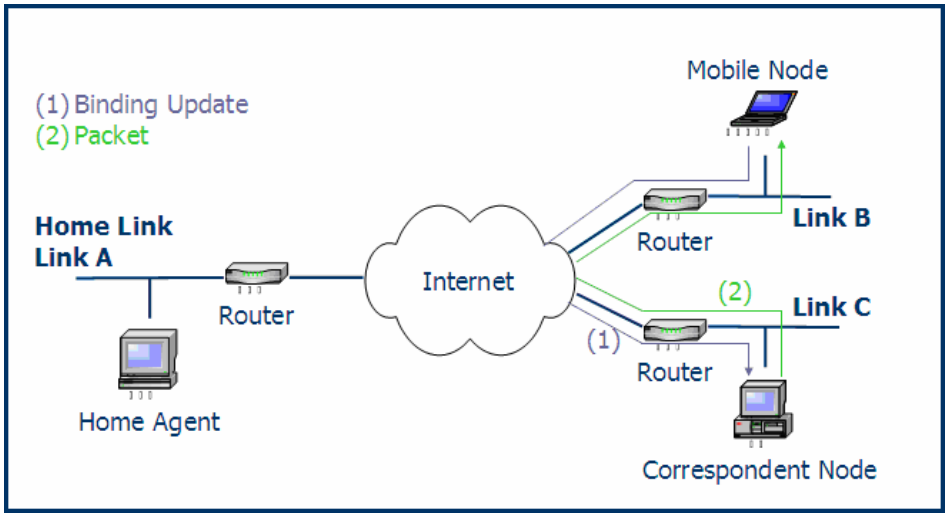
Αν ο κινητός κόμβος στείλει πακέτα σε ένα οποιοδήποτε άλλο κόμβο, τα στέλνει απευθείας στον προορισμό τους. Ο κινητός κόμβος θέτει τη διεύθυνση αποστολέα στα πακέτα που στέλνει την COA του και συμπεριλαμβάνει μια Home Address επιλογή προορισμού. Επειδή η home address είναι στατική, σε αντίθεση με την COA, επιτρέπει κάθε σε κάθε CN τη «διαφανή» χρήση της COA για στρώματα πάνω από την υποστήριξη του IPv6. Τα πιο υψηλά επίπεδα, συμπεριλαμβανομένου και των προγραμμάτων εφαρμογών δεν προσέχουν την COA, μόνο την home address.



Σχήμα 2.3 – Δρομολόγηση πακέτου [4]

Αν ένας κινητός κόμβος επικοινωνεί με ένα CN καθώς είναι μακριά από το home subnet του, τα πακέτα δρομολογούνται από τον CN στον HA, από τον HA στον κινητό κόμβο και από τον κινητό κόμβο στον CN. Αυτή η «ανωμαλία» δρομολόγησης λέγεται Τριγωνική Δρομολόγηση.

Για την αποφυγή της Τριγωνικής Δρομολόγησης, ένας κινητός κόμβος μπορεί να στείλει Binding Updates σε οποιοδήποτε CN, είτε κινητό είτε σταθερό. Αυτό επιτρέπει σε IPv6 CN να φυλάξουν την τρέχουσα COA του κόμβου αυτού και να στέλνουν πακέτα απευθείας.



Σχήμα 2.4 – Διαδικασία Binding Update [4]

Οποιοσδήποτε IPv6 κόμβος που στέλνει ένα πακέτο, ελέγχει πρώτα την Binding Cache του για την συγκεκριμένη διεύθυνση προορισμού. Αν υπάρχει μια εγγραφή, θα στείλει το πακέτο στον κινητό κόμβο χρησιμοποιώντας ένα routing header. Η διαδρομή που καθορίζεται από αυτό το routing header έχει δυο hops. Το πρώτο hop είναι το COA και το δεύτερο το home address του κινητού κόμβου. Έτσι το πακέτο πηγαίνει απευθείας στην COA του κόμβου. Στην συνέχεια, αφού ο κινητός κόμβος λάβει το πακέτο, το προωθεί στο επόμενο hop που καθορίζεται στο routing header. Αφού το τελευταίο hop είναι η home address του κινητού κόμβου, το πακέτο θα σταλεί στο home address.
Αν η Binding Cache δεν έχει καμία εγγραφή, το πακέτο θα σταλεί κανονικά. Μετά θα δρομολογηθεί στο συγκεκριμένο δίκτυο και θα παραληφθεί από το κόμβο προορισμού. Στην περίπτωση που ο προορισμός είναι ένας κινητός κόμβος μακριά από το home subnet του, το πακέτο θα παραληφθεί από τον HA του στο HL και θα σταλεί στον κινητό κόμβο. Με την παραλαβή του πακέτου αυτού, ο κινητός κόμβος θα στείλει στον CN ένα Binding Update με την COA του.

Διαχείριση binding μηνυμάτων

Ένας κινητός κόμβος ο οποίος έχει κάνει configure ένα νέο COA σαν την πρωτεύων COA του, πρέπει να εγγράψει αυτή τη διεύθυνση στον HA του και στους CN που έχουν ήδη ενημερωθεί για το binding του κινητού κόμβου. Για αυτό τον σκοπό ο κινητός κόμβος στέλνει ένα Binding Update που περιέχει τη νέα του διεύθυνση. Για επιβεβαίωση ότι ο παραλήπτης πράγματι παρέλαβε το Binding Update, ο κινητός κόμβος μπορεί να ζητήσει αναγνώριση ενεργοποιώντας το Acknowledgment bit στο Binding Update (μέχρι την παραλαβή της επιβεβαίωσης, ο κινητός κόμβος θα στέλνει περιοδικά το Binding Update). Ένας κινητός κόμβος πρέπει να ενεργοποιεί το Acknowledgment bit σε Binding Updates που προορίζονται προς τον HA του. Μπορεί ακόμη να ενεργοποιήσει το Acknowledgment bit όταν στέλνει σε CNs αλλά δεν είναι αναγκαίο γιατί αν το Binding Update δεν παραληφθεί για οποιοδήποτε λόγο από τον CN, ο κινητός κόμβος το καταλαβαίνει όταν εξακολουθεί να λαμβάνει πακέτα από CNs, μέσω του HA του.
Πριν τη λήξη μιας καταχώρησης στην Binding Cache για ένα κινητό κόμβο, ο CN μπορεί να ανανεώσει την καταχώρηση στέλνοντας ένα Binding Request στον κινητό κόμβο. Κατά συνέπεια, ο κινητός κόμβος θα απαντήσει με ένα Binding Update.

Ανίχνευση Κίνησης

Κατά τη διάρκεια που ένας κινητός κόμβος είναι μακριά από το home υποδίκτυό του, επιλέγει ένα δρομολογητή σαν τον default δρομολογητή και ένα πρόθεμα που διαφημίζεται από αυτόν τον δρομολογητή για να το χρησιμοποιήσει στην πρωτεύουσα COA του.

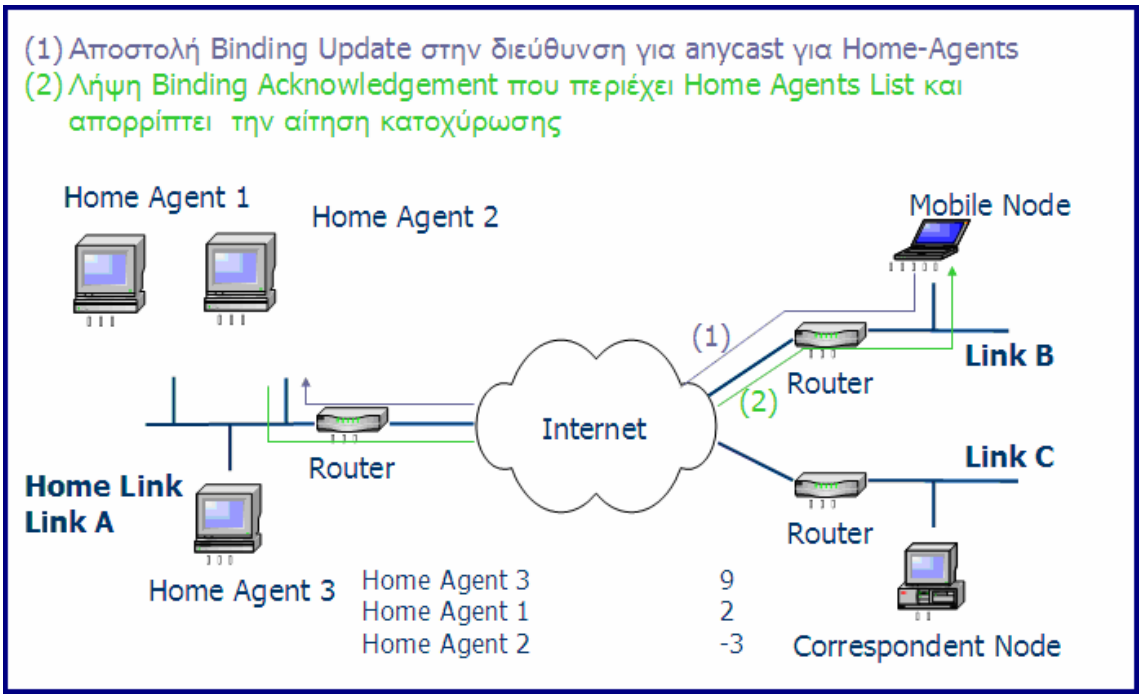
Στην συνέχεια, ο κινητός κόμβος μπορεί να χρησιμοποιήσει οποιοδήποτε συνδυασμό από διαθέσιμους μηχανισμούς για να ανιχνεύσει πότε έχει κινηθεί από μια σύνδεση σε μια άλλη. Μια δυνατότητα είναι ο κόμβος να περιμένει για τα Router Advertisements που στέλνονται περιοδικά. Αν δεν παραλάβει διαφήμιση για ένα συγκεκριμένο χρονικό διάστημα θα υποθέσει ότι ο default δρομολογητής δεν είναι πλέον διαθέσιμος και συνδέεται με άλλον δρομολογητή από τον οποίο είχε λάβει διαφήμιση αυτό το διάστημα.

Όταν ο κινητός κόμβος καταλάβει ότι έχει κινηθεί σε άλλη σύνδεση, στέλνει ένα Binding Update στον HA του και στους CN τους οποίους έχει καταχωρημένους στην Binding Λίστα Ενημέρωσής του. Έτσι ο κινητός κόμβος τους ενημερώνει για την νέα COA του και κατά συνέπεια για την μετακίνησή του.

Μηχανισμός Εξεύρεσης Home Agent

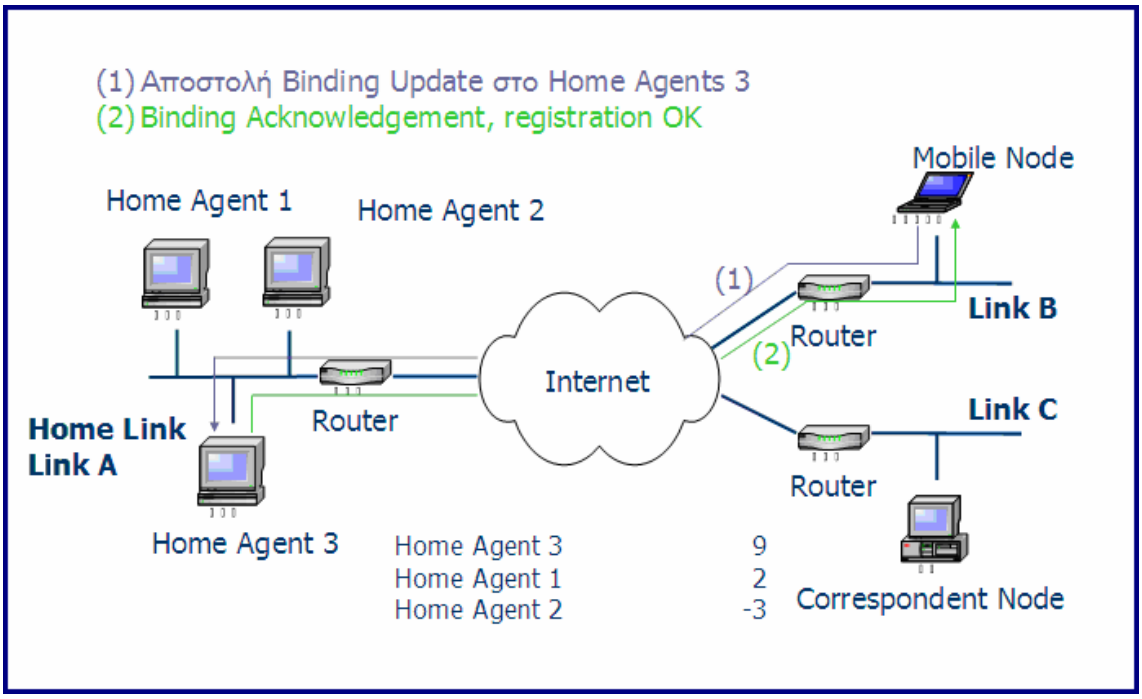
Υποθέτοντας ότι ο κινητός κόμβος δε γνωρίζει το IP του HA του, το πρωτόκολλο Mobile IPv6 προσφέρει ένα μηχανισμό που επιτρέπει στον κινητό κόμβο να ανακαλύπτει δυναμικά την IP διεύθυνση κάποιου HA στο HL του, στον οποίο θα μπορεί να εγγράψει την COA του όταν θα είναι μακριά το home subnet του.

Ο κινητός κόμβος στέλνει ένα Binding Update στο “Home Agents’ anycast address” για το δικό του home subnet πρόθεμα και κατά συνέπεια καταφέρνει να επικοινωνήσει με ένα από τους δρομολογητές στο HL του, που εκείνη τη στιγμή λειτουργεί σαν HA. Αν ο HA απορρίψει το Binding Update, θα επιστρέψει μια λίστα με όλους τους HA στο HL. Αυτή η λίστα διατηρείται από κάθε HA και δημιουργείται μέσω των περιοδικών αποστολών Routing Advertisements. Ο κινητός κόμβος στέλνει ένα Binding Update σε μια από τις διευθύνσεις στην λίστα και περιμένει για το ανάλογο Binding Acknowledgement. Αν δεν το λάβει, ή αν απορριφθεί , δοκιμάζει να εγγραφεί σε ένα άλλο HA της λίστας. Η επιλογή των HAs στην λίστα, γίνεται με την σειρά που καταγράφονται στην λίστα, γιατί η πρώτη διεύθυνση είναι του πιο διαθέσιμου HA και η τελευταία του λιγότερου διαθέσιμου.



Σχήμα 2.5 – Επιλογή του Home Agent στη λίστα [4]

Ο κινητός κόμβος τώρα θα στείλει το Binding Update του στον Home Agent 3 γιατί έχει απορριφθεί το Binding Update που είχε στείλει στον Home Agent 2. Ο Home Agent 2 του στέλλει μια λίστα με διευθύνσεις, όπου ο Home Agent 3 καλείται να επιλέξει μια από αυτές. Η επιλογή γίνεται με την σειρά που καταγράφονται στην λίστα, γιατί η πρώτη διεύθυνση είναι του πιο διαθέσιμου HA και η τελευταία του λιγότερου διαθέσιμου.



Σχήμα 2.6 – Επιτυχής εγγραφή του Home Agent 3 [4]

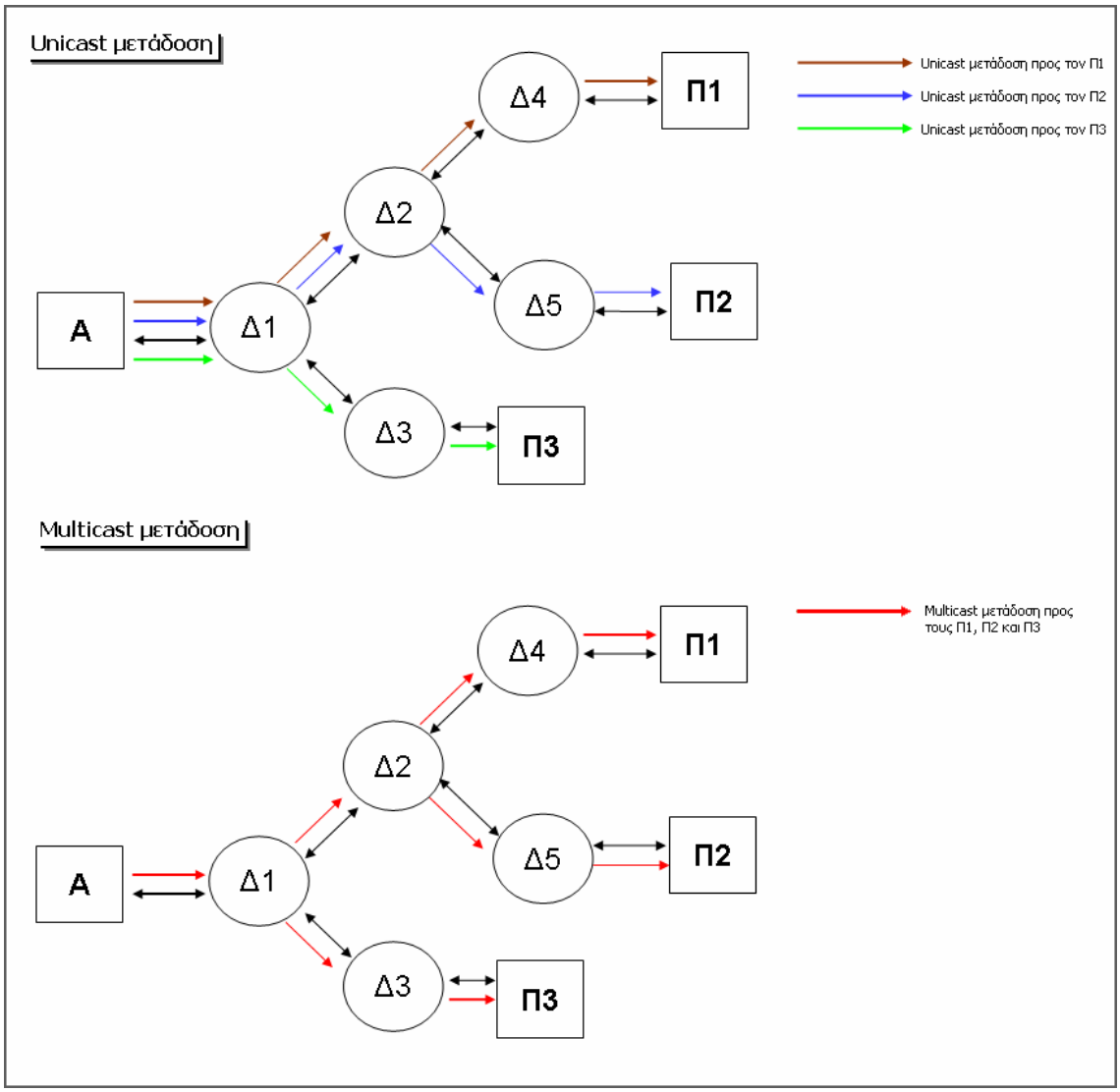
2.4 Multicast μετάδοση δεδομένων

2.4.1 Περιγραφή της multicast μετάδοσης δεδομένων

Το multicast είναι μια τεχνολογία η οποία αναπτύχθηκε προκειμένου να επιτευχθεί η μετάδοση δεδομένων, με χρήση της στοίβας πρωτοκόλλων TCP / IP του Διαδικτύου, από ένα σταθμό προς πολλούς με κύριο γνώμονα την αποτελεσματική χρήση των διαθέσιμων δικτυακών πόρων.

Πιο συγκεκριμένα η βασική αρχή λειτουργίας του multicast βασίζεται στην εξής ιδέα: Σε περιπτώσεις που έχουμε μετάδοση των ίδιων δεδομένων από έναν αποστολέα προς *n* άλλους παραλήπτες τότε με τον συμβατικό unicast τρόπο μετάδοσης, θα δημιουργηθούν *n* διαφορετικές ροές δεδομένων (data flows) με κατεύθυνση από τον αποστολέα που μεταδίδει προς τους *n* παραλήπτες που λαμβάνουν τα δεδομένα. Όταν κομμάτια των μονοπατιών που οδηγούν στους παραλήπτες είναι κοινά, τότε τα ίδια δεδομένα επαναλαμβάνονται πάνω στο ίδιο δικτυακό σύνδεσμο (network link) με σαφές αποτέλεσμα να γίνεται κακή χρήση του εύρους ζώνης που αυτό έχει, αλλά και επίσης να αυξάνεται σημαντικά ο φόρτος του σταθμού που στέλνει τα δεδομένα γιατί έχει να εξυπηρετήσει *n* διαφορετικές ροές.

Στο σχήμα 2.7, συγκρίνεται η multicast με την unicast μετάδοση δεδομένων για την περίπτωση που ένας αποστολέας (Α) θέλει να μεταδώσει την ίδια πληροφορία σε τρεις παραλήπτες (Π1, Π2 και Π3) μέσα από ένα δίκτυο πέντε δρομολογητών (Δ1, Δ2, Δ3, Δ4 και Δ5). Κατά τη διάρκεια της unicast μετάδοσης ο αποστολέας μεταδίδει μια ξεχωριστή ροή δεδομένων για κάθε ένα από του παραλήπτες. Στην περίπτωση της multicast μετάδοσης, ο αποστολέας μεταδίδει μια ροή δεδομένων σε όλους τους παραλήπτες και η ροή δεδομένων διακλαδώνεται μόνο στα σημεία του δικτύου που απαιτείται με αποτέλεσμα να έχουμε τη βέλτιστη χρήση των πόρων του δικτύου λόγω του ότι η ροή των δεδομένων μεταδίδεται μόνο μια φορά σε κάθε γραμμή του δικτύου. Για να επιτευχθεί αυτό πρέπει να γίνουν τα παρακάτω: Αρχικά ο αποστολέας μεταδίδει τη multicast ροή δεδομένων στον δρομολογητή Δ1. Όσο δεν υπάρχει εκδηλωμένο κάποιο ενδιαφέρον από κάποιον παραλήπτη για τη multicast ροή δεδομένων ο δρομολογητής Δ1 δεν την προωθεί περαιτέρω. Όταν για παράδειγμα ο παραλήπτης Π1 εκδηλώσει ενδιαφέρον για τη multicast ροή δεδομένων στέλνοντας ένα κατάλληλο μήνυμα στον δρομολογητή Δ4 στον οποίο συνδέεται, ο δρομολογητής Δ4 προωθεί το αίτημα αυτό στον δρομολογητή Δ2 και αυτός με την σειρά του το προωθεί στον δρομολογητή Δ1. Στη συνέχεια ο δρομολογητής Δ1 προωθεί τη multicast ροή δεδομένων στο δρομολογητή Δ2 και αυτός με την σειρά του την προωθεί στον δρομολογητή Δ4 ο οποίος τελικά την προωθεί στον παραλήπτη Π1. Η ίδια διαδικασία επαναλαμβάνεται και για τους άλλους παραλήπτες όταν αυτοί εκδηλώσουν ενδιαφέρον για τη multicast ροή δεδομένων. Για παράδειγμα όταν ο παραλήπτη Π2 εκδηλώσει ενδιαφέρον για την multicast ροή δεδομένων ο δρομολογητής Δ5 θα προωθήσει το αίτημα στον δρομολογητή Δ2 και ο δρομολογητής Δ2 θα αρχίσει να προωθεί τη multicast ροή δεδομένων στο δρομολογητή Δ5. Σε περίπτωση που κάποιος παραλήπτης δε θέλει να λαμβάνει πλέον την multicast ροή δεδομένων στέλνει το αντίστοιχο μήνυμα στο δρομολογητή στον οποίο συνδέεται και ο δρομολογητής σε περίπτωση που δεν υπάρχει άλλος παραλήπτης ο οποίος να λαμβάνει την multicast ροή δεδομένων σταματά τη μετάδοση της multicast ροής δεδομένων. Έτσι, στην περίπτωση που ο παραλήπτης Π1 σταματήσει να λαμβάνει τη multicast ροή δεδομένων, ο δρομολογητής Δ4 θα σταματήσει να λαμβάνει τη multicast ροή δεδομένων από τον δρομολογητή Δ2, όμως ο δρομολογητής Δ2 θα συνεχίσει να λαμβάνει τη multicast ροή δεδομένων από το δρομολογητή Δ1 γιατί πρέπει να εξυπηρετήσει το δρομολογητή Δ3.



Σχήμα 2.7 – Μετάδοση multicast και unicast

Θα πρέπει να τονίσουμε πως στο Διαδίκτυο σήμερα η multicast μετάδοση δεδομένων δεν υποστηρίζεται από όλους τους δρομολογητές και για αυτό τον σκοπό έχει δημιουργηθεί ένα ιδεατό δίκτυο το οποίο αποτελείται από δρομολογητές του Διαδικτύου οι οποίοι υποστηρίζουν multicast μετάδοση και ονομάζεται MBONE (Multicast Backbone) και στο οποίο δίκτυο μπορεί να λάβει χώρα multicast μετάδοση δεδομένων. Στο MBONE είναι δυνατή η μετάδοση multicast κίνησης ανάμεσα στους δρομολογητές οι οποίοι δεν υποστηρίζουν multicast μετάδοση δεδομένων με τη χρήση της τεχνικής του tunneling κατά την οποία τα multicast πακέτα δεδομένων εισάγονται σε unicast πακέτα δεδομένων και μεταδίδονται ως unicast κίνηση μέχρι τον επόμενο δρομολογητή ο οποίος υποστηρίζει multicast όπου εξάγονται από τα unicast πακέτα και στη συνέχεια μεταδίδονται ως multicast κίνηση.

Η τεχνολογία multicast έχει γίνει πλέον αναπόσπαστο στοιχείο των σημερινών δικτύων εξαιτίας των ολοένα και αυξανόμενων εφαρμογών πολυμέσων που μεταδίδουν δεδομένα πάνω από το Διαδίκτυο. Αξίζει να αναφερθεί πως το μέγεθος της οικονομίας που επιτυγχάνεται με αυτόν τον τρόπο είναι αρκετά μεγάλο αν αναλογιστούμε το κόστος των απομακρυσμένων συνδέσεων υψηλών ταχυτήτων (Broadband WAN Links).

2.4.2 Το multicast ως υπηρεσία του δικτύου

Το multicast είναι μια τεχνολογία που υλοποιείται στο επίπεδο του δικτύου και παρέχεται ως υπηρεσία στις εφαρμογές ανώτερων επιπέδων. Αυτές με τη σειρά τους για να την εκμεταλλευτούν πρέπει να διαθέτουν τις κατάλληλες βιβλιοθήκες.

Προκειμένου οι multicast ροές δεδομένων να ξεχωρίζουν από τις υπόλοιπες και άρα να χειρίζονται διαφορετικά από το επίπεδο δικτύου, χρησιμοποιούν ένα ειδικό κομμάτι από το διαθέσιμο χώρο διευθύνσεων του IP πρωτοκόλλου. Συγκεκριμένα έχει διατεθεί για χρήση από το multicast η κλάση D (Class D) των διευθύνσεων του Διαδικτύου της οποίας το εύρος των διευθύνσεων είναι από 224.0.0.0 έως 239.255.255.255.

2.4.3 Πρωτόκολλα δρομολόγησης για την υποστήριξη του multicast

Η τεχνολογία multicast ουσιαστικά είναι μια τεχνική δρομολόγησης η οποία αναλαμβάνει να μεταφέρει μια ροή δεδομένων σε όλους τους σταθμούς που ενδιαφέρονται να τη λάβουν. Μιλώντας πιο συγκεκριμένα, κάθε επικοινωνία που χρησιμοποιεί το multicast ονομάζεται σύνοδος (session) και σε αυτήν αντιστοιχίζεται μια μοναδική multicast IP διεύθυνση (ή multicast διεύθυνση).

Πρέπει να γίνει ξεκάθαρο πως για την επίτευξη οποιαδήποτε επικοινωνίας μέσω multicast πρέπει να εξασφαλιστούν δύο βασικά συστατικά.

- Οι σταθμοί που επιθυμούν να ξεκινήσουν μια καινούρια ή να συμμετάσχουν σε μία προϋπάρχουσα σύνοδο πρέπει να έχουν έναν τρόπο να ενημερώσουν για αυτό τον τοπικό δρομολογητή τους.
- Οι δρομολογητές μεταξύ τους πρέπει να έχουν κάποιο πρωτόκολλο επικοινωνίας έτσι ώστε να ανταλλάσσουν πληροφορίες όσον αναφορά την δρομολόγηση της multicast κίνησης. Αυτές οι πληροφορίες στις περισσότερες των περιπτώσεων αφορούν κατά πόσο οι σταθμοί που βρίσκονται πίσω από ένα δρομολογητή επιθυμούν ή όχι να λάβουν μέρος σε κάποια μορφή multicast επικοινωνίας και

συνεπώς ο δρομολογητής να προωθηθεί ή να σταματήσει να προωθείται multicast κίνηση προς τους σταθμούς αυτούς.

Η δρομολόγηση της multicast κίνησης συνήθως βασίζεται σε κάποιο προϋπάρχον unicast πρωτόκολλο δρομολόγησης όπως τα OSPF , RIP , BGP. Στην περίπτωση όμως που οι χρήστες μας στο δίκτυο είναι κινητοί, τότε υπάρχουν άλλα πρωτόκολλα που να υποστηρίζουν και να διαχειρίζονται multicast με κινητικότητα των χρηστών.

Στις επόμενες παραγράφους παρουσιάζονται τα πιο διαδεδομένα πρωτόκολλα multicast δρομολόγησης για κινητούς χρήστες μιας και αυτό μας ενδιαφέρει στην παρούσα έρευνα και περιγράφεται ο τρόπος λειτουργίας τους έτσι ώστε σε μελλοντικό στάδιο να γίνει σύγκριση μεταξύ τους και να επιλεχθεί το καταλληλότερο πρωτόκολλο για μετάδοση πολυμεσικού περιεχομένου.

Remote Subscription [5]

Παράδοση των multicast δεδομένων μέσω των βέλτιστων μονοπατιών από την πηγή στους παραλήπτες. Το συχνό handoff (μεταφορά ενός κινητού host από ένα δίκτυο σε άλλο), επιφέρει περίπου το κόστος του επανακτισίματος του multicast δέντρου, καθώς ο αριθμός των Multicast reconstruction tree εξαρτάται από τη συχνότητα των handoffs στους κινητούς hosts.

Bidirectional Tunneling [5]

Το κόστος του multicast reconstruction tree είναι μειωμένο μιας και η multicast δρομολόγηση δεν επηρεάζεται από την κινητικότητα του χρήστη, όμως το μονοπάτι δρομολόγησης για multicast μεταφορά ίσως να απέχει πολύ από το βέλτιστο και παράλληλα υπάρχει tunnel convergence και συγκεκριμένα packet duplication.

MoM (Mobile Multicast) [6]

Χρησιμοποιεί Designated Multicast Service Provider (DMSP), έτσι ώστε τα duplicated δεδομένα δεν γίνονται tunneled στον common FA (Foreign Agent). Είναι υπεύθυνο το DMSP για την προώθηση των multicast δεδομένων στον FA. Αν ο αριθμός των μελών του multicast group είναι μικρός και τα handoff rates των κινητών hosts αυξάνονται, τότε απαιτείται συχνά DMSP handoff. Επίσης, υπάρχει μεγάλο μονοπάτι δρομολόγησης όταν ένας κινητός χρήστης μετακινείται σε ξένο δίκτυο μακριά από τον HA (Home Agent) του.

RBMoM (Range Based Mobile Multicast) [7]

Υβριδικό πρωτόκολλο των remote subscription και bi-directional tunnelling. Σύμφωνα με το εύρος υπηρεσιών του RBMoM [7] πρωτοκόλλου, η ανταλλαγή μεταξύ του μήκους του tunnelling μονοπατιού και της συχνότητας του multicast tree reconstruction έχει να δείξει τα καλύτερα αποτελέσματα όταν multicast δεδομένα παραδίδονται στο κοντινότερο tunnelling μονοπάτι από MHA σε FA χωρίς να καταβάλλεται το υψηλό κόστος του multicast reconstruction tree. Το RBMoM πρωτόκολλο αποφασίζει την τιμή του service range σύμφωνα με τα handoff rates των κινητών hosts και τον αριθμό των μελών του multicast group. Οι δύο αυτές παράμετροι όμως δεν μπορούν να αποτελούν κριτήριο για καθορισμό μιας βέλτιστης τιμής του πεδίου υπηρεσιών καθώς σε πραγματικές συνθήκες αλλάζουν δυναμικά. Για την επίλυση αυτού του προβλήματος προτείνεται ο ένας αλγόριθμος μείωσης του κόστους του multicast tree reconstruction και παράδοσης multicast data στο κοντινότερο μονοπάτι tunnelling από MHA σε FA. Προτείνεται, για το σκοπό αυτό, μια ειδική παράμετρος συστήματος η οποία ονομάζεται tunnelling service range. Το tunnelling πεδίο υπηρεσίας ικανοποιεί το όριο του end-to-end delay και το delay variation, και παίρνει το μέγιστο tunnelling πεδίο υπηρεσίας από το MHA. Το κόστος του multicast tree reconstruction μειώνεται κατά ένα σημαντικό βαθμό. Οι κινητοί χρήστες αποτελούν ληφθέντα δεδομένα multicast από το παρόν MHA μέχρι να τεθούν εκτός του πεδίου υπηρεσιών. Αν το service range έχει μεγάλη τιμή, τότε το μήκος του μονοπατιού tunnelling από το MHA γίνεται μακρύ. Για να μειωθεί το μήκος αυτό, όταν ένας κινητός host κινείται σε ξένο δίκτυο, τα multicast δεδομένα παραδίδονται από το κοντινότερο MHA ψάχνοντας τον MHA πίνακα του FA. Όμως, υπάρχει απώλεια δεδομένων που συμβαίνει όταν ένας κινητός χρήστης μετακινείται εκτός του παρόντος εύρους υπηρεσιών και κινείται σε ξένο δίκτυο. Σε αυτήν την περίπτωση, αν το ξένο δίκτυο δεν είναι μέλος του παρόντος multicast group , τότε το multicast packet loss συμβαίνει κατά τη διάρκεια του χρόνου που απαιτείται για να ενωθεί με το παρόν multicast group.

TBMoM (Timer-based Mobile Multicast) [8]

Νέο υβριδικού τύπου πρωτόκολλο το οποίο χρησιμοποιεί το JOIN timer ενός κινητού χρήστη. Η νέα multicast agent οντότητα ονομάζεται FMA. Καθώς ένας κινητός χρήστης ταξιδεύει σε ξένα δίκτυα, επιλέγει το FMA δυναμικά και αιτείται όπως το FMA να γίνει μέλος σε ένα multicast δένδρο αν το χρονόμετρο του κινητού χρήστη έχει λήξη. Όταν το FMA ολοκληρώσει τη διαδικασία ένωσης με το χρήστη, ξεκινά να λαμβάνει multicast datagrams από το multicast δένδρο, και τότε αναλαμβάνει την ευθύνη παράδοσης multicast datagrams για τον κινητό χρήστη μέσα σε ένα συγκεκριμένο πλαίσιο χρόνου (time interval). Το service range τώρα δείχνει διαφορετικό κάθε φορά που συμβαίνει ένα time – out. Το FMA μεταφέρει τα multicast datagrams χρησιμοποιώντας unicast tunneling. Παρέχει καλύτερη βελτιστοποίηση μονοπατιού και μικρότερο αριθμό DMSP handoffs συγκριτικά με το MoM, MMA και RBMoM. Παράλληλα, δείχνει τη λιγότερη διάσπαση σε Multicast υπηρεσίες χάρη στα handoffs.

Connection and connectionless modes [9]

Μια νέα μέθοδος για mobile multicast με μια ανάμεικτη μέθοδο connection και connectionless προσαρμογή. Στη μέθοδο αυτή, ένας αποστολέας καθορίζει έναν παραλήπτη ο οποίος επικοινωνεί με connection mode. Οι άλλοι παραλήπτες, ενόσω ο αποστολέας μεταδίδει data frames στον καθορισμένο παραλήπτη, διακόπτουν τα frames με connectionless modes και τα αποθηκεύουν αν δεν υπάρχει λάθος. Μετά το τέλος της μετάδοσης στον καθορισμένο παραλήπτη, τα frames που δεν έχουν παραληφθεί ξαναστέλνονται από τον host με connection mode. Δεν υπάρχει σύγκρουση από την ανάδραση διότι ο αριθμός του παραλήπτη που επιστρέφει ACK ή NACK της παραλαβής των frames, παραμένει ένας καθόλη τη διάρκεια της περιόδου μετάδοσης. Ως αποτέλεσμα, ο συνολικός χρόνος μετάδοσης μειώνεται (ο χρόνος μέχρι όλοι οι παραλήπτες να παραλάβουν ολόκληρα όλα τα data frames).

CMMR (Core-Manager Based Scalable Multicast Routing) [10]

Ο CM (Core Manager) γνωρίζει, «κρατεί», τα ίχνη των core (για κάθε multicast μεταφορά) των δέντρων έτσι ώστε να έχει μια σύντομη εικόνα όλων των δέντρων. Όποτε ένα νέο μέλος θέλει να ενταχθεί σε ένα group, συμβουλεύεται τον CM για να του δώσει έναν core. Τα νέα μέλη σχεδόν πάντα κατευθύνονται σε έναν κοντινό core ενός

δέντρου έτσι ώστε το κόστος του δέντρου να είναι περιορισμένο. Παρόλα αυτά, δεν είναι ανάγκη κάθε νέο μέλος να πρέπει να συμβουλεύεται τον CM για να ενταχθεί σε ένα Group μιας και στο CMMR όλοι οι δρομολογητές που βρίσκονται στο δέντρο, αναλαμβάνουν τοπικά τις αιτήσεις για ένταξη στο group (δέντρο). Υπάρχει επίσης ιεραρχική αρχιτεκτονική στην οποία ολόκληρη η multicast περιοχή χωρίζεται σε υποπεριοχές με διαφορετικά ιεραρχικά επίπεδα, και οι IP multicast διευθύνσεις καθορίζονται ανάλογα με την ιεραρχία. Αφού το CMMR είναι αναδρομικό, όλοι οι δρομολογητές θα τρέχουν τον ίδιο αλγόριθμο άσχετα με το ιεραρχικό επίπεδο στο οποίο βρίσκονται. Με το πρωτόκολλο αυτό, επιλύθηκε το πρόβλημα της καθυστέρησης επανασύνδεσης (reconnection latency).

RRBMoM (Reliable Range Based Mobile Multicast) [11]

Επέκταση του RBMoM, παρέχει ACK-based αξιοπιστία και loss recovery του sender και χρησιμοποιεί μια tree-based ιεραρχική αρχιτεκτονική δομή για να ελαφρύνει το πρόβλημα συγκέντρωσης με τα ACK.

MMROP (Mobile Multicast με Routing Optimization) [12]

Είναι επέκταση του MIP – RS και έχει τα πλεονεκτήματα της υψηλής αποδοτικότητας δρομολόγησης. Χρησιμοποιεί έναν join and leave μηχανισμό και δρομολογεί την missing data sequence λόγω του προβλήματος “out of synch”, στους άλλους agents των γειτονικών δικτύων μέσω tunneling. Με το MMROP ένας foreign ή home agent, επεκτείνεται έτσι ώστε να βοηθά στο multicast για κινητούς χρήστες. Ένας κινητός agent δεν είναι κατανάγκη ένας multicast δρομολογητής. Από την οπτική γωνία του multicast δρομολογητή, ένας κινητός agent είναι ένα μέλος του group. Από την οπτική γωνία των κινητών χρηστών, ένας agent εξυπηρετεί όπως το proxy των multicast υπηρεσιών. Ως αποτέλεσμα, το MMROP κερδίζει μη απώλεια πακέτων από περιπλανόμενους (όπως συμβαίνει στο MIP-BT) και επίσης κερδίζει optimal routing efficiency (όπως συμβαίνει στο MIP-RS). Σε σχέση με τα πρωτόκολλα MIP-RS, MIP-BT,MoM, το MMROP έχει περισσότερη απόδοση βελτιστοποίησης δρομολόγησης, χαμηλό κόστος μεταφοράς και υψηλή αντοχή σε λάθη (ευρωστία).

2.5 Χαρακτηριστικά ενός δικτυακού μονοπατιού στο Διαδίκτυο

Συνήθως οι μόνες συσκευές οι οποίες έχουν σαφή εικόνα για την κατάσταση ενός δικτύου και τους διαθέσιμους πόρους του δικτύου είναι οι δρομολογητές του δικτύου. Δυστυχώς στο Διαδίκτυο σήμερα δεν υπάρχει κάποιος μηχανισμός προκειμένου οι δρομολογητές να μπορούν να ενημερώνουν τις εφαρμογές στα τελικά συστήματα για την τρέχουσα κατάσταση του δικτύου. Το γεγονός αυτό έχει ως συνέπεια οι εφαρμογές στα τελικά συστήματα του Διαδικτύου, να πρέπει να εκτιμήσουν από μόνες τους τις τρέχουσες δικτυακές συνθήκες μετρώντας διάφορες παραμέτρους του δικτύου χωρίς να υποστηρίζονται για αυτό τον σκοπό άμεσα από τις δικτυακές συσκευές. Με βάση το μέτρημα αυτών των παραμέτρων οι εφαρμογές μπορούν να εξάγουν συμπεράσματα για την κατάσταση του δικτύου και να προσαρμόσουν ανάλογα τη μετάδοση των δεδομένων τους. Οι παράμετροι τις οποίες μπορούν να εκτιμήσουν οι εφαρμογές και βοηθούν στην εξαγωγή συμπερασμάτων για την κατάσταση του δικτύου είναι οι παρακάτω:

- Εύρος ζώνης (bandwidth)
- Ρυθμός απώλειας πακέτων
- Διακύμανση καθυστέρησης (Jitter)
- Χρόνος καθυστέρησης μετάδοσης μετά επιστροφής RTT (Round Trip Time)

Γενικά μπορούμε να πούμε πως η ποιότητα αναπαραγωγής πολυμέσων σε μια εφαρμογή πραγματικού χρόνου εξαρτάται από την απώλεια πακέτων και τη διακύμανση καθυστέρησης (Jitter). Επίσης όταν παρουσιάζονται τα παραπάνω φαινόμενα αποτελούν ένδειξη πως υπάρχει κάποιο πρόβλημα με τη μετάδοση των δεδομένων και αντιστοίχως πρέπει να ληφθούν μέτρα από την εφαρμογή ώστε αυτά τα φαινόμενα να εξαφανιστούν και αν αυτό δεν είναι δυνατόν, να περιοριστούν.

Στη συνέχεια αναλύουμε τις παραπάνω παραμέτρους και περιγράφουμε πώς αυτές μπορούν να χρησιμοποιηθούν για την εξαγωγή συμπερασμάτων σχετικά με την κατάσταση του δικτύου.

2.5.1 Εύρος ζώνης

Με τον όρο εύρος ζώνης (bandwidth) αναφερόμαστε στην ποσότητα της πληροφορίας η οποία μπορεί να μεταφερθεί σε ένα δεδομένο χρονικό διάστημα (συνήθως ένα δευτερόλεπτο) πάνω από ένα δικτυακό σύνδεσμο. Ένας πιο τεχνικός ορισμός του εύρους ζώνης είναι το εύρος των συχνοτήτων τις οποίες χρησιμοποιεί ένα σήμα σε ένα μέσο μετάδοσης. Κάθε ψηφιακό ή αναλογικό σήμα έχει ένα εύρος ζώνης. Στα ψηφιακά συστήματα το εύρος ζώνης εκφράζεται σε bit δεδομένων ανά δευτερόλεπτο (bit per second - bps). Για παράδειγμα ένα modem το οποίο λειτουργεί στα 57.600 bps έχει διπλάσιο εύρος ζώνης σε σχέση με ένα modem το οποίο λειτουργεί στα 28.800 bps.

Θα πρέπει να τονίσουμε ότι τα δικτυακά μονοπάτια συνήθως αποτελούνται από μια σειρά από δικτυακούς συνδέσμους που ο καθένας έχει το δικό του εύρος ζώνης. Εάν ένας από αυτούς τους δικτυακούς συνδέσμους είναι σημαντικά αργότερος σε σχέση με τους άλλους τότε θεωρούμε τον δικτυακό αυτό σύνδεσμο ως σημείο συμφόρησης του δικτυακού μονοπατιού γιατί περιορίζει το εύρος ζώνης όλου του δικτυακού μονοπατιού.

2.5.2 Ρυθμός απώλειας πακέτων

Ο ρυθμός απώλειας πακέτων ορίζεται ως το κλάσμα των συνολικών πακέτων που για κάποιο λόγο δε φτάνουν στον παραλήπτη. Σε περίπτωση συμφόρησης στο δίκτυο οι δρομολογητές του δικτύου μπορεί να αρχίσουν να απορρίπτουν πακέτα αντί να τα προωθούν. Οι απώλειες αυτές μπορούν να ανιχνευτούν με αριθμούς ακολουθίας (sequence numbers) που τοποθετούνται στα πακέτα δεδομένων. Ο παραλήπτης μπορεί να υπολογίσει το ρυθμό των απωλειών των πακέτων δεδομένων μετρώντας τον αριθμό των πακέτων δεδομένων που χάθηκαν στο δίκτυο με τον αριθμό των πακέτων δεδομένων που μεταδόθηκαν (βάσει του αριθμού ακολουθίας) για ένα χρονικό διάστημα. Πολλές φορές η επιλογή του χρονικού διαστήματος στο οποίο θα γίνει η μέτρηση είναι σημαντική και μπορεί να επηρεάσει του υπολογισμούς. Για αυτόν το λόγο οι μετρήσεις του ρυθμού απώλειας των πακέτων δεδομένων φιλτράρονται με τη χρήση κατάλληλων φίλτρων ώστε να μας δίνουν πιο αξιόπιστες μετρήσεις. Τα πρωτόκολλα RTP / RTCP παρέχουν τον απαραίτητο μηχανισμό ώστε ο παραλήπτης να ενημερώνει τον αποστολέα για το ρυθμό απώλειας των πακέτων δεδομένων. Στα σημερινά δίκτυα οι απώλειες πακέτων δεδομένων οφείλονται κυρίως σε συμφόρηση του δικτύου και όχι σε λάθη μετάδοσης όπως συνέβαινε στα παλαιότερα δίκτυα.

2.5.3 Διακύμανση καθυστέρησης (Jitter)

Γενικά η διακύμανση καθυστέρησης είναι μια έννοια η οποία είναι δύσκολο να ορισθεί. Υπάρχουν ορισμοί της διακύμανσης καθυστέρησης οι οποίοι σχετίζονται με τη μέση ή τη μέγιστη τιμή της. Ορισμένοι ερευνητές ορίζουν τη διακύμανση καθυστέρησης ως την διαφορά ανάμεσα στην μέγιστη και ελάχιστη καθυστέρηση για μια περίοδο χρόνου. Κάποιοι άλλοι ορίζουν τη διακύμανση καθυστέρησης ως τη μέγιστη καθυστέρηση ανάμεσα σε δύο διαδοχικά πακέτα για μια περίοδο χρόνου. Σε αυτή τη διπλωματική εργασία, χρησιμοποιούμε για να ορίσουμε τη διακύμανση καθυστέρησης το ορισμό σύμφωνα με το οποίο η διακύμανση καθυστέρησης ορίζεται ως η μέση απόκλιση (σε ομαλοποιημένη (smoothed) απόλυτη τιμή) της διαφοράς D σε χρόνους πακέτων δεδομένων στον παραλήπτη συγκρινόμενη με τον αποστολέα για ένα ζεύγος πακέτων. Όπως φαίνεται και στην παρακάτω σχέση, ο παραπάνω ορισμός είναι ισοδύναμος με τη διαφορά σε σχετικούς χρόνους μετάδοσης για δύο πακέτα (ως σχετικό χρόνο μετάδοσης εννοούμε την διαφορά ανάμεσα στην χρονοσήμανση (timestamp) που φέρει το πακέτο και την τιμή του ρολογιού του παραλήπτη την στιγμή που λαμβάνει το πακέτο). Εάν S_i είναι η χρονοσήμανση του πακέτου i και R_i είναι ο χρόνος που λαμβάνεται το πακέτο, τότε για δύο πακέτα i και j, το D ορίζεται ως εξής:

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i)$$

Το φαινόμενο της διακύμανσης καθυστέρησης οφείλεται κυρίως στους παρακάτω λόγους:

- Καθυστερήσεις στις ουρές (queues) του δικτύου: Εάν όλα τα πακέτα κατά την κυκλοφορία τους στο δίκτυο συναντούν τις ίδιες ουρές, και τα ίδια μήκη στις ουρές, στο μονοπάτι το οποίο διανύουν, καθυστερούν περίπου τον ίδιο χρόνο, με αποτέλεσμα παρόλο που η καθυστέρηση από άκρο σε άκρο μπορεί να είναι μεγάλη, να μην υπάρχουν διακυμάνσεις στην καθυστέρηση. Το φαινόμενο της διακύμανσης καθυστέρησης παρουσιάζεται όταν διαδοχικά πακέτα καθυστερούν διαφορετικές χρόνους στις ουρές που συναντούν. Οι διαφορετικές καθυστερήσεις σχετίζονται άμεσα με το μοντέλο εξυπηρέτησης το οποίο χρησιμοποιείται από την ουρά (για παράδειγμα αν το μοντέλο είναι FIFO, RED κλπ.) και τη διασταυρωμένη κίνηση (cross traffic) στην ουρά.

- Καθυστερήσεις λόγω του αποστολέα: Συχνά το φαινόμενο της διακύμανσης καθυστέρησης δεν οφείλεται αποκλειστικά σε δικτυακά φαινόμενα αλλά σχετίζεται και με προβλήματα τα οποία οφείλονται στον αποστολέα. Αυτό συμβαίνει γιατί ο υπολογιστής του αποστολέα δε συμπεριφέρεται πάντα όπως αναμένουμε, ιδιαίτερα αν αυτός ο υπολογιστής εκτελεί περισσότερες από μια εργασίες ταυτόχρονα. Αν το λογισμικό ή το υλικό το οποίο χρησιμοποιείται για την κωδικοποίηση των πολυμέσων έχει μη αναμενόμενη συμπεριφορά μπορεί να παρουσιαστεί το φαινόμενο της διακύμανσης καθυστέρησης ακόμη και αν δεν υπάρχουν προβλήματα στο δίκτυο. Αυτό οφείλεται στο γεγονός ότι η χρονοσήμανση η οποία τοποθετείται σε κάθε πακέτο αντιστοιχεί στον χρόνο που η πληροφορία που περιέχει το πακέτο κωδικοποιήθηκε και κατά συνέπεια προσδιορίζει τον χρόνο στον οποίο πρέπει να παρουσιασθεί στην εφαρμογή του παραλήπτη. Επιπλέον αυτή η χρονοσήμανση, χρησιμοποιείται για τον υπολογισμό της καθυστέρησης του δικτύου και κατά συνέπεια στον υπολογισμό της διακύμανσης καθυστέρησης. Εάν περάσει αξιοσημείωτος χρόνος από την στιγμή που το πακέτο κωδικοποιήθηκε μέχρι το πακέτο να τοποθετηθεί στο δίκτυο (λόγω του ότι ο επεξεργαστής είναι απασχολημένος με άλλες εργασίες) τότε επηρεάζονται και οι υπολογισμοί της διακύμανσης καθυστέρησης. Αν δε ληφθεί υπόψη αυτός ο παράγοντας, μπορούμε να οδηγηθούμε σε λάθος συμπεράσματα για την κατάσταση του δικτύου.

Σαν συμπέρασμα μπορούμε να πούμε πως η διακύμανση καθυστέρησης από μόνη της δεν αρκεί για να εξάγουμε συμπεράσματα για την κατάσταση του δικτύου. Κατά συνέπεια η διακύμανση καθυστέρησης θα πρέπει να συνδυάζεται με άλλες παραμέτρους, όπως για παράδειγμα τις απώλειες πακέτων για να εξάγουμε ασφαλή συμπεράσματα για την κατάσταση του δικτύου.

2.5.4 Χρόνος Round Trip Time (RTT)

Ο χρόνος καθυστέρησης μετάδοσης μετά επιστροφής RTT (Round Trip Time) αντιπροσωπεύει το χρόνο που απαιτείται για ένα πακέτο δεδομένων να πάει από ένα τελικό σύστημα σε ένα άλλο και στην συνέχει να επιστρέψει στο αρχικό σύστημα. Ο χρόνος RTT περιλαμβάνει τον χρόνο μετάδοσης του πακέτου στα φυσικά μέσα, τον χρόνο αναμονής και επεξεργασίας στις ουρές των δικτυακών συσκευών και τον χρόνο επεξεργασίας στα τελικά συστήματα. Ο πιο απλός τρόπος για τον υπολογισμό του χρόνου RTT ανάμεσα σε δύο τελικά συστήματα Α και Β είναι να μεταδώσουμε ένα

πακέτο από το τελικό σύστημα Α, το οποίο περιέχει τον χρόνο $T_{ΑΠ}$ που μεταδόθηκε (timestamp), στο τελικό σύστημα Β και το τελικό σύστημα Β μόλις λάβει το πακέτο δεδομένων να το επιστρέφει στο τελικό σύστημα Α και το τελικό σύστημα Α να το λαμβάνει την χρονική στιγμή $T_{ΛΗ}$. Σε αυτή την περίπτωση ο χρόνος RTT μπορεί να υπολογιστεί ως η διαφορά των χρόνων $T_{ΛΗ}$ και $T_{ΑΠ}$:

$$t_{RTT} = T_{ΛΗ} - T_{ΑΠ}$$

Η μέτρηση της καθυστέρησης προς τη μια κατεύθυνση είναι πιο περίπλοκη. Λόγω της πιθανής ασυμμετρίας στις γραμμές του δικτύου που συνδέσουν τα δύο τελικά συστήματα δεν μπορούμε να υπολογίσουμε την καθυστέρηση προς τη μια κατεύθυνση ως το μισό του χρόνου RTT. Για να εκτιμήσουμε την καθυστέρηση προς τη μια κατεύθυνση θα μπορούσε το τελικό σύστημα Α να στείλει ένα πακέτο δεδομένων με μία χρονοσήμανση (timestamp) $T_{ΑΠ}^{A}$ στο τελικό σύστημα Β και το τελικό σύστημα Β να το παραλάβει την χρονική στιγμή $T_{ΛΗ}^{B}$. Σε αυτή την περίπτωση η καθυστέρηση προς τη μια κατεύθυνση θα μπορούσε να υπολογιστεί ως:

$$t_{oneway} = T_{ΛΗ}^{B} - T_{ΑΠ}^{A}$$

Για να ισχύει ο παραπάνω υπολογισμός θα πρέπει τα τελικά συστήματα να έχουν συγχρονισμένα ρολόγια. Γενικά ο συγχρονισμός των ρολογιών στα δικτυακά περιβάλλοντα δεν είναι απλή διαδικασία και χρησιμοποιούνται είτε πρωτόκολλα (πχ Πρωτόκολλο Χρόνου Δικτύου, Network Time Protocol, συσκευές GPS (Global Positioning System) που έχουν υψηλό κόστος. Επίσης εκτός από το πρόβλημα του συγχρονισμού υπάρχει και το πρόβλημα του ότι τα ρολόγια μπορεί να τρέχουν με διαφορετική ταχύτητα γεγονός που μπορεί να οδηγήσει σε επιπλέον προβλήματα.

2.6 Κωδικοποιήσεις πολυμεσικού περιεχομένου για μετάδοση πάνω από δίκτυα

2.6.1 ITU H.261

Το 1984 το CCITT Study Group XV καθιέρωσε μια εξειδικευμένη ομάδα στην κωδικοποίηση για οπτική τηλεφωνία για να αναπτύξει συστάσεις για μετάδοση video στα Mx384 Kbps (M=1-5). Αργότερα μελετήθηκε τυποποίηση στα Px64 Kbps (P=1-30), καθώς η τεχνολογία συμπίεσης βελτιώθηκε και έγινε εφικτή η μετάδοση video σε χαμηλότερους ρυθμούς. Οι συστάσεις που δημιουργήθηκαν είναι γνωστές σαν "υποδομή για οπτικοακουστικές υπηρεσίες". Η ITU-T/CCITT σύσταση H.261,

"κωδικοποίηση video για οπτικοακουστικές υπηρεσίες στα Px64Kbps", καθορίζει έναν κώδικα για συμπιεσμένο ψηφιακό video ο οποίος ολοκληρώθηκε και εγκρίθηκε το Δεκέμβριο του 1990. Μία από τις εφαρμογές αυτού του τύπου είναι το video-τηλέφωνο και η video-διάσκεψη. Συνεπώς, ο αλγόριθμος κωδικοποίησης video πρέπει να είναι ικανός να λειτουργεί σε πραγματικό χρόνο με τη μικρότερη καθυστέρηση. Για P=1 ή 2, μόνο επιτραπέζια άμεση οπτική επικοινωνία ("videotelephony") μπορεί να επιτευχθεί πρακτικά. Πιο πολύπλοκες εικόνες, για παράδειγμα, στην περίπτωση της video-διάσκεψης απαιτούν P>=6. Το H.261 είναι μια από τις τυποποιήσεις της ITU-T H.320 οικογένειας για video-τηλέφωνο και τηλεδιάσκεψη σε ρυθμούς bits κυμαινόμενους από 64Kbps σε 2Mbps. Για να εξασφαλίσουμε την ενδοεπικοινωνία μεταξύ διαφορετικών συστημάτων τηλεόρασης (π.χ. PAL και NTSC) και να μειώσουμε το ρυθμό bit, το σύνηθες μέγεθος του video στο H.261 είναι CIF.

2.6.2 ITU H.263

Η σύσταση H.263 σχεδιάστηκε για επικοινωνία σε χαμηλούς ρυθμούς δεδομένων. Η σύσταση H.263 χρησιμοποιεί μία block motion-compensated DCT μέθοδο για την κωδικοποίηση του video. Η κωδικοποίηση κατά H.263 είναι πολύ πιο αποτελεσματική από ότι η κωδικοποίηση κατά H.261. Στην σύσταση H.263, η κωδικοποίηση πραγματοποιείται τεμαχίζοντας κάθε εικόνα σε macroblocks. Κάθε macroblock αποτελείται από 16x16 luminance blocks και 8x8 chrominance blocks. Κάθε macroblock μπορεί να κωδικοποιηθεί είτε ως intra (χωρίς συσχετίσεις ανάμεσα σε διαδοχικά frames) είτε ως inter (με συσχετίσεις ανάμεσα σε διαδοχικά frames). Η τεχνική DCT η οποία χρησιμοποιείται, παρέχει spatial redundancy και temporal redundancy. Η σύσταση H.263 στηρίζεται στην H.261 και παρέχει κάποιες επεκτάσεις για την υποστήριξη πιο αποτελεσματικής κωδικοποίησης.

2.6.3 ITU H.264

Η σύσταση H.264 ή αλλιώς H.264/AVC σχεδιάστηκε με σκοπό την αποδοτική κωδικοποίηση σε χαμηλότερα bit rates από τις άλλες συστάσεις που είδαμε πιο πάνω, χωρίς να αυξάνει την πολυπλοκότητα της σχεδίασης σε τέτοιο βαθμό ώστε να είναι αδύνατη η υλοποίησή της. Παράλληλα, ένας άλλος σκοπός της σύστασης αυτής είναι η παροχή αρκετής ευελιξίας έτσι ώστε να μπορεί να εφαρμόζεται σε ποικίλες εφαρμογές όπως για παράδειγμα σε υψηλή και χαμηλή ευκρίνεια πολυμεσικού περιεχομένου καθώς και για να λειτουργεί αποδοτικά σε διάφορα δίκτυα και συστήματα όπως broadcast, DVD αποθήκευση, RTP/IP packet networks και ITU-T πολυμεσικά

τηλεφωνικά συστήματα. Η παρουσίαση της πρώτης έκδοσης του H.264/AVC ολοκληρώθηκε το 2003.

2.6.4 JPEG

Η μέθοδος συμπίεσης JPEG χρησιμοποιείται για κωδικοποίηση full-motion video, ειδικά, σε σήματα NTSC TV. Είναι γνωστή και ως Motion JPEG. Αν και το JPEG δε σχεδιάστηκε για full-motion βίντεο, μπορεί να το εξυπηρετήσει με κάποιους περιορισμούς. Ένας από τους περιοριστικούς παράγοντες της χρήσης του αλγορίθμου είναι ότι αυτός λειτουργεί ανεξάρτητα από πλαίσιο σε πλαίσιο, γι' αυτό, δεν μπορεί να μειώσει τον πλεονασμό που υπάρχει μεταξύ των πλαισίων. Μερικοί εκτίμησαν το γεγονός ότι το JPEG εκτελεί μόνο intra-frame συμπίεση σαν ένα κέρδος με την αίσθηση ότι προσφέρει "γρήγορη" τυχαία πρόσβαση σε οποιοδήποτε πλαίσιο του υλικού του βίντεο. Άλλες τεχνικές full-motion συμπίεσης βίντεο εκτελώντας intraframe συμπίεση βασίζονται στην περιοδική μετάδοση ενός πλαισίου αναφοράς- αν το πλαίσιο αναφοράς στέλνεται κάθε 20 πλαίσια, ένας ίσως πρέπει να περιμένει για 19 πλαίσια πριν το πλαίσιο αναφοράς ληφθεί. Αυτό θα μπορούσε να ισοδυναμεί με μια αναμονή 20/30 ή 0.66 δευτερολέπτων. Με το JPEG, ο χρόνος αναμονής είναι τόσος όσος αυτός που απαιτείται για την αποκωδικοποίηση ενός πλαισίου, ο οποίος είναι 0.04 δευτερολέπτων. Για την απεικόνιση του video σε μια οθόνη PC σε μια μέτρια ανάλυση, δηλαδή, 640 x 480 pixels και 24 bits για αναπαράσταση χρώματος, το JPEG επιτυγχάνει συμπίεση περίπου 1 MB ανά πλαίσιο, ή 30 Mbps. Ο κανόνας σήμερα είναι να υπάρχει βίντεο σε ένα μικρό παράθυρο του PC. Για παράδειγμα, το DVI λειτουργεί με ένα παράθυρο 256 x 240, το οποίο κόβει τον ρυθμό του bit από ένα παράγοντα πέντε, στο 1.2 Mbps.

2.6.5 MPEG

Το MPEG [13] αναπτύχθηκε το 1988 στο πλαίσιο εργασίας των ενωμένων ISO/IEC τεχνικών επιτροπών στην τεχνολογία πληροφορίας. Ο σκοπός της ομάδας ήταν να αναπτύξει πρότυπα για κωδικοποιημένη αναπαράσταση video και ήχου, καθώς και του συνδυασμού τους, όταν χρησιμοποιείται για αποθήκευση και για ανάκτηση στα ψηφιακά αποθηκευτικά μέσα (DSM). Η έννοια των DSM περιλαμβάνει μεταξύ άλλων και τηλεπικοινωνιακά κανάλια όπως το ISDN και τοπικά δίκτυα (LAN).

Το πρότυπο MPEG έχει τρία μέρη MPEG-video, MPEG-audio και MPEG-systems. Το MPEG-video ασχολείται με τη συμπίεση σημάτων video, το MPEG-audio ασχολείται με την συμπίεση σημάτων ήχου και το MPEG-systems ασχολείται με το θέμα του

συγχρονισμού και της πολύπλεξης των πολλαπλών συμπιεσμένων bit streams video και ήχου.

Σημειώνεται ότι ένα από τα πιο σημαντικά χαρακτηριστικά των τύπων του MPEG είναι ότι οι τύποι καθορίζουν μόνο τη σύνταξη των κωδικοποιημένων bit streams έτσι ώστε οι αποκωδικοποιητές (decoders) ακολουθώντας αυτά τα πρότυπα να μπορούν να αποκωδικοποιήσουν το bit stream. Αυτό επιτρέπει ευελιξία στο σχεδιασμό και υλοποίηση κωδικοποιητών (encoders).

2.6.6 MPEG-4

Το MPEG-4 είναι μια πρωτοβουλία μέσα στην όλη διαδικασία του MPEG με στόχο να βελτιώσει την κωδικοποίηση των δεδομένων όταν αυτά πρέπει να μεταδοθούν με χαμηλό ρυθμό μετάδοσης. Με την ολοκλήρωση του, το πρότυπο MPEG-4 θέτει σε λειτουργία ένα ευρύ φάσμα καινούριων εφαρμογών, περιλαμβάνοντας multimedia εφαρμογές σε κινητά δίκτυα (mobile networks), video-τηλεφωνία με απλή (παλιά) υπηρεσία τηλεφώνου ή με ασύρματα δίκτυα.

ΚΕΦΑΛΑΙΟ 3

Περιγραφή υπαρχόντων πρωτοκόλλων

3.1 Multicast μετάδοση πολυμεσικού περιεχομένου σε σταθερούς χρήστες

Ο «One Multicast Stream» μηχανισμός [14], είναι ένας μηχανισμός ο οποίος μπορεί να χρησιμοποιηθεί για multicast multimedia data με τη χρήση ενός multicast stream πάνω από ετερογενή δίκτυα όπως το διαδίκτυο, και έχει την ικανότητα να προσαρμόζει τη μετάδοση των πολυμεσικών δεδομένων ανάλογα με τις αλλαγές του δικτύου. Ο μηχανισμός αυτός βασίζεται σε real time πρωτόκολα. Χρησιμοποιεί μια δίκαια inter-receiver συνάρτηση έτσι ώστε να συμπεριφέρεται δίκαια στις ομάδες των παραληπτών που βρίσκονται σε ένα ετερογενές περιβάλλον.

Κατά τη διάρκεια της μετάδοσης multicast των πολυμεσικών δεδομένων σε ένα μόνο Multicast stream, η εφαρμογή του αποστολέα πρέπει να επιλέξει το ρυθμό της μετάδοσης έτσι ώστε να ικανοποιεί περισσότερο τους παραλήπτες που έχουν τις συνθήκες του τρέχοντος δικτύου. Για να γίνει αυτό, υπάρχουν τρεις προσεγγίσεις:

1. Equation Based
2. Network feedback based
3. Συνδυασμός των δύο πιο πάνω προσεγγίσεων

Ο προτεινόμενος μηχανισμός επικεντρώνεται στον έλεγχο των συνθηκών του δικτύου καθώς και στην πρόβλεψη του ρυθμού της μετάδοσης έτσι ώστε να ικανοποιούνται περισσότερο οι ομάδες των παραληπτών.

Περίληψη του Adaptive Multicast Transmission μηχανισμού

Η εφαρμογή του αποστολέα χρησιμοποιεί RTP/RTCP πρωτόκολα για τη μετάδοση των πολυμεσικών δεδομένων. Οι παραλήπτες παραλαμβάνουν τα πολυμεσικά δεδομένα και ενημερώνουν τον αποστολέα για την ποιότητα της μετάδοσης, μέσω RTCP reports. Ο αποστολέας, από την πλευρά του, παραλαμβάνει τα RTCP reports, τα αναλύει και καθορίζει το ρυθμό της μετάδοσης r, ο οποίος να ικνοποιεί περισσότερο τις ομάδες των παραληπτών ανάλογα με τις συνθήκες του δικτύου. Ο αποστολέας έχει τις πληροφορίες για κάθε παραλήπτη i , και κάθε φορά που παραλαμβάνει ένα RTCP report από τον

παραλήπτη i, εκτιμά το ρυθμό μετάδοσης ri που επιθυμεί να έχει ο παραλήπτης i σαν να ήταν ο μόνος στο δίκτυο. Ο αποστολέας χρησιμοποιεί IRF και RFi συναρτήσεις έτσι ώστε να καθορίσει τον ιδανικό ρυθμό μετάδοσης. Η συνάρτηση RFi είναι η εξής:

$$RFi = \min(ri, r) / \max(ri, r)$$

ri: ρυθμός μετάδοσης που προτιμά ο παραλήπτης

r: ρυθμός μετάδοσης που σκοπεύει να χρησιμοποιήσει ο αποστολέας

Το ιδανικό είναι ri = r.

Η συνάρτηση IRF για μία ομάδα με n παραλήπτες είναι η εξής:

$$IRF(r) = \sum_{i=1}^{n} a_i * RF_i(r)$$

όπου $\text{to } \sum_{i=1}^{n} a_i = 1 \text{ and } 0 \leq a_i \leq 1, 0, i = 1, ..., n.$

όπου r: ρυθμός μετάδοσης που σκοπεύει να χρησιμοποιήσει ο αποστολέας

αi: το βάρος του παραλήπτη ri σε σχέση με την τιμή IRF.

Παρατηρούμε ότι για μεγαλύτερες τιμές του IRF, η ομάδα των παραληπτών είναι πιο ικανοποιημένη, ενώ για μικρότερες τιμές του IRF, είναι λιγότερο ικανοποιημένη.

Αφού ο αποστολέας συμπεριφέρεται ίσα και δίκαια στους παραλήπτες, τότε αi = 1/n. Από το RTCP πρωτόκολο μπορούμε εύκολα να υπολογίσουμε τον αριθμό των παραληπτών.

Αλγόριθμοι για το μηχανισμό Multicast μετάδοσης

Ο προτεινόμενος μηχανισμός χρησιμοποιεί δύο αλγόριθμους. Feedback analysis και update sender rate. Ο αλγόριθμος Feedback Analysis, αναλύει τις feedback πληρφορίες τις οποίες ο παραλήπτης i, αποστέλλει κάθε φορά που ο αποστολέας παραλαμβάνει ένα RTCP report από τον παραλήπτη i, τρέχει τον αλγόριθμο feedback analysis έτσι ώστε να εκτιμήσει τον ιδανικότερο ρυθμό μετάδοσης που να ικανοποιεί τον παραλήπτη i. Οι αλλαγές ανάμεσα στις συνθήκες δικτύου για τον παραλήπτη i, βασίζονται στις τιμές της απώλειας πακέτων και delay jitter για τον συγκεκριμένο παραλήπτη. Συγκεκριμένα, για το ρυθμό απώλειας πακέτων, ορίζονται δύο τιμές, η LRc (congestion packet loss rate) και η LRu (unload packet loss rate) οι οποίες με βάση τους παρακάτω κανόνες ελέγχουν τις αλλαγές των συνθηκών του δικτύου:

$$if(LR^{i}_{new} \geq LR_c) \rightarrow congestion$$
$$if(LR_u < LR^{i}_{new} < LR_c) \rightarrow load$$
$$if(LR^{i}_{new} \leq LR_u) \rightarrow unload$$

Ο αποστολέας εκτιμά το ρυθμό μετάδοσης r μέσω της χρήσης του αλγόριθμου update sender rate. Η εκτίμηση αυτή αποσκοπεί στην αύξηση της ικανοποίησης της ομάδας των παραληπτών που δίδεται από τη συνάρτηση IRF. Ο αλγόριθμος update sender rate, χρησιμοποιεί τον Additive Increase Multiplicative Decrease (AIMD) μηχανισμό, έτσι ώστε να εκτιμήσει το νέο ρυθμό μετάδοσης.

Τώρα, εξετάζοντας το πρόβλημα ως μια λειτουργία, σαφώς το πρωτόκολο που θα αναλυθεί είναι ένα και θα συνδυάζει τις δύο λειτουργίες (mobile multicast και μετάδοση πολυμεσικού περιεχομένου). Ακολουθεί η ανάλυση ενός προταθέντος πρωτοκόλλου, του Multimedia Multicast in Mobile Computing [15].

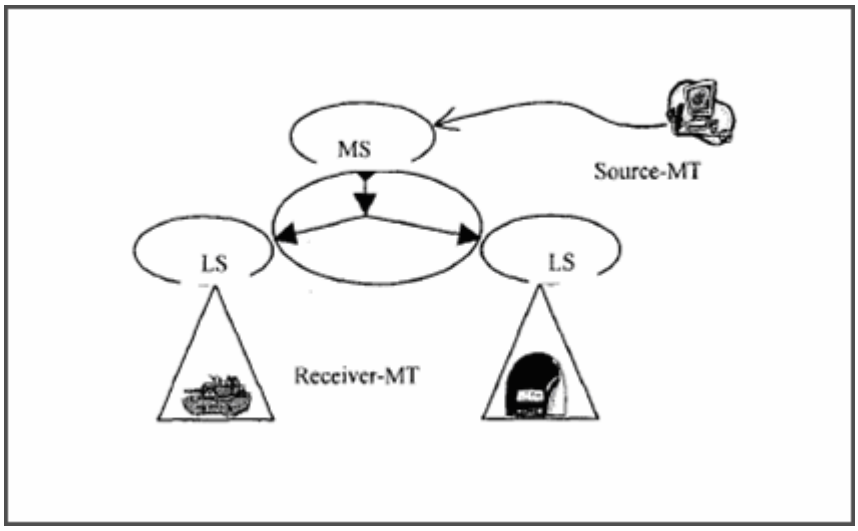
3.2 Multicast μετάδοση πολυμεσικού περιεχομένου σε κινητούς χρήστες

Για να καταστεί επιτυχής η μετάδοση πολυμεσικού περιεχομένου (audio, video, data), είναι απαραίτητο να παρέχεται ποιότητα υπηρεσίας (QoS) ανάμεσα στους τελικούς χρήστες. Η πρόβλεψη της ποιότητας υπηρεσίας, σημαίνει ότι η ροή πολυμεσικού περιεχομένου πρέπει να παίρνει προβλέψιμη υπηρεσία από τους διαθέσιμους πόρους (CPU χρόνος, εύρος ζώνης) στο σύστημα επικοινωνίας. Το πρωτόκολλο επικοινωνίας θα πρέπει να εγγυάται αποδεκτές παραμέτρους όπως δεσμευμένη καθυστέρηση και αποδεκτο jitter. Η διαφορά στις καθυστερήσεις του δικτύου προκαλεί σοβαρό πρόβλημα στο playback του πολυμεσικού περιεχομένου(multimedia stream). Πρέπει τα media units να παρουσιάζονται συνέχεια στο χρόνο. Ένα άλλο πρόβλημα που επηρεάζει σοβαρά το συνεχόμενο playback του multimedia stream, είναι το handoff. Όταν ένας κινητός χρήστης μετακινείται από μια τοποθεσία σε άλλη έχοντας ενεργό stream, το μονοπάτι που θα ακολουθούν τα media units αλλάζει. Αν η νέα τοποθεσία στην οποία μετακινείται ο κινητός χρήστης είναι υπερφορτωμένη, το διαθέσιμο εύρος ζώνης στη νέα αυτή τοποθεσία μπορεί να μην είναι αρκετό για να παρέχει το throughput το οποίο έπαιρνε ο κινητός χρήστης στην προηγούμενη τοποθεσία που βρισκόταν. Κατά συνέπεια, ο κινητός χρήστης πιθανόν να υποφέρει από προσωρινή αποκοπή της

υπηρεσίας κατά τη διάρκεια του handoff ενόσω η σύνδεση αποκόπτεται από το παλιό μονοπάτι και εγκαθιδρύεται στο νέο μονοπάτι.

Άρα, για να γίνει κατορθωτό το συνεχόμενο playback προτείνεται ένα συγχρνισμένο πρωτόκολο και μηχανισμοί για τη διαχείριση του buffer όταν συμβαίνει ένα handoff.

Το πρωτόκολο αυτό παρέχει ένα δεδομένο δεσμευμένο jitter, καθώς και μηχανισμούς που να διαχειρίζονται το buffer και να επιτρέπουν ασταμάτητο playback όταν οι κινητοί χρήστες μετακινούνται. Παράλληλα, εξετάζονται οι περιπτώσεις όπου οι πόροι προκρατούνται ή όχι, και δείχνεται ότι όταν οι πόροι δεσμεύονται από πριν, το μέγεθος του Buffer του κινητού χρήστη μειώνεται, κατά συνέπεια δεν υπάρχει scalability.



Σχήμα 3.1 – Multicast επικοινωνία στο Mobile Computing

Κάθε MT επικοινωνεί μέσω ενός Mobile Support Station (MSS). Το MSS επικοινωνεί με το δίκτυο μέσω ενός Mobility Router (MR) ή Switch. Υποθέτουμε ότι MSS και MR είναι ομαδοποιημένα στην ίδια οντότητα, δηλαδή στον Local Server (LS). Ένα MT όταν επικοινωνεί, ελέγχεται από ένα LS. Το σύνολο των MT που ενώνονται με τα LS λέγεται group (αντί cell).

Στην περίπτωση που έχουμε multicast, μια πηγή-MT που έχει multimedia stream, κάνει multicast το stream της μέσω του LS της που ονομάζεται Master Server (MS). Κάθε MT παραλαμβάνει το Multimedia stream μέσω του LS του. Τα LS ενώνονται με το MS. Τα MT (πηγές ή παραλήπτες), μετακινούνται από ένα LS σε άλλο. Ένα ξένο LS όταν το group του δεν είναι άδειο, μπορεί να είναι ήδη ενωμένο στο MS, αλλιώς πρέπει να ζητήσει να ενωθεί με το MS. Αν η MT πηγή μετακινηθεί σε ένα άλλο LS (ξένο MS),

τότε νέες συνδέσεις πρέπει να γίνουν μεταξύ των ήδη υπάρχοντων LS και του ξένου MS.

Η πηγή-MT σπάζει το Multimedia stream σε media units τα οποία συμβολίζονται με το τ_s. Η τιμή αυτή περιλαμβάνει στον υπολογισμό της την καθυστέρηση συλλογής, πακετοποίηση, την καθυστέρηση του τελικού compression και encapsulation. Η τιμή του playback στον παραλήπτη-MT και η έξοδος των media units στους ενδιάμεσους κόμβους είναι η ίδια με την αρχική.

Για κάθε MT, η καθυστέρηση επικοινωνίας d, ποικίλει όμως είναι δεσμευμένη μεταξύ μιας ελάχιστης $d^{\min}$ και μέγιστης $d^{\max}$ τιμής για κάθε ραδιοκανάλι που ενώνονται στο LS. Σε ένα καλωδιομένο δίκτυο, η καθυστέρηση των άκρων από τον MS σε ένα $\mathbf{LS}_i$ συμβολίζεται με το $\mathbf{D}_i$. Η καθυστέρηση αυτή είναι δεσμευμένη μεαξύ του $D_i^{\max}$ που αντιπροσωπεύει τη μέγιστη δυνατή καθυστέρηση των άκρων, και του $D_i^{\min}$ που αντιπροσωπεύει την ελάχιστη δυνατή καθυστέρηση των άκρων. Οι τιμές αυτές καθορίζονται στην αρχή του πρωτοκόλου. Συγκεκριμένα, η τιμή $D^{\max}=\max_{i=1}^{n}(D_i^{\max})$ ανάμεσα σε όλα τα LS και MS.

Το πρωτόκολο χωρίζεται σε δύο βήματα. Στο πρώτο βήμα, υπάρχει ένα πρωτόκολο ανάμεσα στα LS και τον Master Server MS όπου υπάρχει μια δεσμευμένη καθυστέρηση jitter στο καλωδιομένο δίκτυο. Με τον τρόπο αυτό τα LS ξεκινούν να μεταφέρουν τα media units σε όλα τα μέλη του group τους την ίδια χρονική στιγμή. Παράλληλα, υπάρχει ένας μηχανισμός που διευθύνει τον LS που εντάσσεται ή φεύγει από ένα Multicast group του MS. Στο δεύτερο βήμα, υπάρχει μια στρατηγική που επιτρέπει το συγχρονισμό των MT (πηγές ή παραλήπτες) όταν μετακινούνται από κελί σε κελί. Γίνεται pre-buffering από media units σε κάθε MT έτσι ώστε να επιτυγχάνεται intra-stream και να εμποδίζεται διακοπή όταν κάνει handoff, σε κάθε LS έτσι ώστε να επιτυγχάνεται intra-stream και inter-LS και στον MS έτσι ώστε να επιτυγχάνεται intra-stream και να εμποδίζεται διακοπή όταν η πηγή-MT μετακινείται.

Συγχρονισμός των τοπικών εξυπηρετητών

Intra-Stream συγχρονισμός

Γενικά, ο intra-stream συγχρονισμός καθορίζεται ως ένα συνηθισμένο Playback των media units που παραλαμβάνονται. Πετυγχαίνεται με το να καθυστερεί ο χρόνος εξόδου του πρώτου παραληφθέντος media unit του.

δ= maximum end-to-end delay – minimum end-to-end delay (1)

Η διαφορά αυτή ονομάζεται jitter. Από τον ορισμό αυτό, καθορίζουμε το μέγεθος του buffer για τον παραλήπτη έτσι ώστε να διατηρεί intra-stream συγχρονισμό.

$$B = \delta * \tau_y \qquad (2)$$

Για να γίνει εφικτός ο multimedia συγχρονισμός, πρέπει να γίνει intra-stream συγχρονισμός για τον master server MS(από την πηγή-MT στον MS), τους LS και τους παραλήπτες-MT. Ορίζουμε ως $\delta \mathrm{MS} = d^{\max} - d^{\min}$ τη διαφορά μεταξύ της μέγιστης και ελάχιστης καθυστέρησης από τον MT στον MS. Τότε το αντίστοιχο μέγεθος του buffer είναι: $BMS = \lceil (\delta MS * \tau_s) \rceil$. Λέγοντας intra-stream συγχρονισμό για το $\mathrm{LS_i}$ εννοούμε το stream από τον MS στον $\mathrm{LS_i}$. Ορίζουμε ως $\delta \mathrm{LS_i} = D_i^{\max} - D_i^{\min}$ τη διαφορά ανάμεσα στη μέγιστη και ελάχιστη καθυστέρηση από τον MS στον LS. Τότε, το αρμόζον μέγεθος του buffer για τον $\mathrm{LS_i}$, είναι $BLS_i = (\delta LS_i * \tau_s)$. Ο intra-stream συγχρονισμός για τον παραλήπτη-MT, αφορά το stream από τον LS στον παραλήπτη-MT. Ορίζουμε ως $\delta \mathrm{rMT} = d^{\max} - d^{\min}$ τη διαφορά ανάμεσα στη μέγιστη και ελάχιστη καθυστέρηση από τον LS στον MT. Τότε, το αντίστοιχο μέγεθος του buffer για τον παεραλήπτη-MT είναι $BrMT = (\delta rMT * \tau_s)$.

Inter-LS συγχρονισμός

Σε ένα multicast group, οι παραλήπτες πρέπει να παίζουν τα media units την ίδια στιγμή. Ο κανόνας αυτός ορίζεται με τον inter-LS συγχρονσμό. Στην περίπωσή μας, εφαρμόζεται για να συγχρονίζει τους LS μεταξύ τους για αυτό και λέγεται inter-LS συγχρονισμός. Επίσης, αντί να κάνουν playback τα medias, κάθε LS κάνει multicast τα media units στο δικό του group από παραλήπτες-MT.

Για να διατηρήσουμε inter-LS συγχρονισμό, πρέπει να έχουμε ίδιους χρόνους εξόδου για το πρώτο media unit για όλους τους LS καθώς και όλα τα media units που θα ακολουθούν πρέπει να μεταδίδονται την ίδια χρονική στιγμή. Για να διατηρήσουμε Inter-LS συγχρονισμό, η έξοδος του πρώτου media unit πρέπει να καθυστερήσει μέχρι όλοι οι LS να παραλάβουν το πρώτο media unit χωρίς να υπάρχει απώλεια του intra-stream συγχρονισμού. Για να λάβουμε υπόψη το συγχρονισμό για το πολυμεσικό περιεχόμενο, και για να τηρήσουμε intra-stream και inter-LS συγχρονισμό, είναι απαραίτητο να καθορίσουμε ένα μηχανισμό ο οποίος να υπολογίζει το χρόνο εξόδου για το πρώτο media unit αντίστοιχα για τον κάθε LS. Καθορίζουμε ένα αρχικό πρωτόκολο

στο οποίο ο MS υπολογίζει την end-to-end καθυστέρηση σε κάθε LS. Η καθυστέρηση αυτή επιτρέπει στον LS να υπολογίσει το χρόνο στον οποίο θα ξεκινήσει να κάνει multicast.

Πρωτόκολλο Εκκίνησης

Κατά τη διάρκεια του πρωτοκόλου αυτού, η πηγή-MT δεν μετακινείται. Αντί να χρησιμοποιήσουμε end-to-end καθυστερήσεις, μπορούμε να χρησιμοποιήσουμε roundtrip καθυστερήσεις ούτως ώστε να υπολογίσουμε σε ποια χρονική στιγμή θα ξεκινήσει το multicast.

Σε χρόνο r_0 η πηγή-MT στέλνει ένα request μήνυμα REQUEST_MT στον Master Server MS της. Σε χρόνο t_{start} ο MS παραλαμβάνει το REQUEST_MT και κάνει broadcast REQUEST μήνυμα σε όλους τους LS που είναι γραμμένοι στο Multicast group του MS. Κάθε LS παραλαμβάνει το μήνυμα αυτό στον τοπικό του χρόνο s_i και απαντά αμέσως στέλνοντας στον MS το $\text{RESPONSE}(D_i^{\max})$ μήνυμα. Σε χρόνο a_i, ο MS παραλαμβάνει την απόκριση του LS_i και υπολογίζει $D_i=(a_i-t_{start})/2$. Μπορούμε επίσης να χρησιμοποιήσουμε την roundtrip καθυστέρηση. Παράλληλα, ο MS υπολογίζει:

- $a^{\max}=\max_{i=1}^{n}(a_i)$ που είναι ο χρόνος παραλαβής της τελευταίας απόκρισης
- Για όλα τα $i=1..n,\ \psi_i=a^{\max}-a_i\ \ \psi_i=a^{\max}-a_i$ που είναι η διαφορά μεταξύ του χρόνου άφιξης της τελευταίας απόκρισης και αυτού του LS_i.
- $D^{\max}=\max_{i=1}^{n}(D_i^{\max}),\qquad \delta_i^{\max}=D^{\max}-D_i^{\max}$
- $\phi=a^{\max}-t_{start}$ που είναι προσωρινή μεταβλητή και καθορίζει τη μέγιστη roundtrip καθυστέρηση ανάμεσα στον MS και στους LS.

Ο MS αποκρίνεται στην πηγή-MT στέλνοντας RESPONSE_MT(φ). Σε χρόνο r_f που αντιστοιχεί στην παραλαβή του μηνύματος αυτού, η πηγή-MT υπολογίζει την end-to-end καθυστέρηση στον MS. $d=(r_f-r_0-\phi)/2$.

Στη συνέχεια, επιβεβαιώνει αμέσως στέλλοντας CONFIRM_MT στον MS και αρχίζει τη μετάδοση των media της.

Όταν παραλάβει CONFIRM_MT από την πηγή-MT, ο MS στέλνει στον κάθε LS_i $OUTPUT(D_i, \psi_i, D^{max}, d, \Delta MS)$ μήνυμα και ξεκινά να κάνει broadcast τα media units, μετά από $\Delta MS+\delta MS$ time units.

Παραλαμβάνοντας το μήνυμα LS_i $OUTPUT(D_i, \psi_i, D^{max}, d, \Delta MS)$ ο LSi υπολογίζει το χρόνο εξόδου για το πρώτο media unit $T_{out_i}^1$.

$$T_{out_i}^1 = s_i + D_i + \psi_i + D^{\max} + 2d + \delta MS + \Delta MS \quad (3)$$

Οι LS ξεκινούν να εξάγουν τα media units σε χρόνο $T_{out_i}^1$. Μετά, κάθε $\frac{1}{\tau_s}$ κάνουν Multicast το επόμενο media unit.

Μέγεθος Buffer

Για να γίνει intra-stream συγχρονισμός, κάθε LS χρειάζεται ένα μέγεθος buffer ίσο με:

$\delta_i * \tau_s$ media units

Επίσης, για να γίνει inter-LS συγχονισμός, κάθε LS χρειάζεται ένα μέγεθος buffer ίσο με:

$\delta_i^{\max} * \tau_s$ media units.

Το συνολικό μέγεθος του buffer που απαιτείται για τον LSi για να κάνει καθολικό συγχρονισμό, ισούται με:

$B_i = (\delta_i^{\max} + \delta_i) * \tau_s = (D^{\max} - D_i^{\min}) * \tau_s$ media units.

Όταν ο LSi γεμίσει το buffer του με Bi media units, μπορεί να ξεκινήσει να εξάγει media units.

Δυναμική Διαχείριση

Προσθήκη και διαγραφή τοπικού εξυπηρετητή LS

Ένας εξυπηρετητής LSi που επιθυμεί να γίνει μέλος του multicast group από MT, πρέπει να στείλει ένα JOIN μήνυμα στον Master Server MS. Υποθέτω ότι αυτό γίνεται σε χρόνο Tjoin i. Ο MS αποκρίνεται αμέσως στέλνοντας μήνυμα $RESP\_JOIN(D^{max}, d, \Delta h, \Delta MS)$. Σε χρόνο T_{resp_i} ο LSi παραλαμβάνει την απόκριση αυτή και υπολογίζει την καθυστέρηση Di στον MS ίση με $D_i=(T_{resp_i} - T_{join_i})/2$.

Τώρα, ο LSi είναι ενωμένος στον MS και πρέπει να κάνει Buffer τα media units για να γίνει ο συγχρονισμός όπως καθορίστηκε προηγουμένως. Άρα, ο χρόνος στον οποίο αρχίζουν να εξάγονται τα media units στο group του MT είναι:

$$T_{out_i}^{1} = T_{join_i} + D_i + D^{\max}.$$

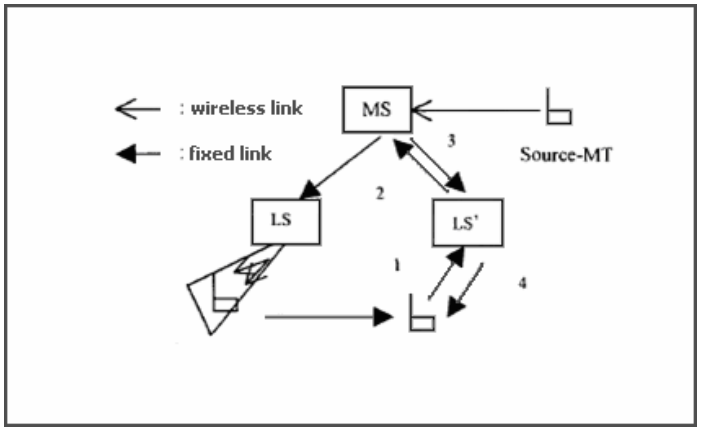
Με τον ίδιο τρόπο, ένας LSi μπορεί να φύγει από το multicast group αν δεν έχει κανένα μέλος στο MT group του. Στην περίπτωση αυτή, στέλνει ένα LEAVE μήνυμα στον MS. Ο MS στη συνέχεια αφαιρεί τον LSi από το multicast group του.

Receiver-MT new call

Ένας παραλήπτης-MT που επιθυμεί να ενταχθεί στο multicast group, πρέπει να στείλει αίτηση στον τοπικό του εξυπηρετητή LS. Δύο περιπτώσεις πρέπει να διαχωριστούν. Αν ο LS ήδη παραλαμβάνει το multimedia stream τότε ο παραλήπτης-MT ξεκινά αμέσως να παραλαμβάνει. Αλλιώς, αν ο LS δεν είναι στο multicast group, πρέπει να γίνει σύνδεση του LS με τον MS. Τότε, αφού γίνει η σύνδεση, ο παραλήπτης-MT ξεκινά να παραλαμβάνει τα media units. Πριν ο παραλήπτης αρχίσει να παίζει τα media units, πρέπει να κάνει pre-buffer τα media units για να τηρήσει intra-stream συγχρονισμό και ασταμάτητο playback όταν συμβεί handoff.

Διαχείριση του handoff

Ο κάθε εξυπηρετητής διαχειρίζεται το multicast group του. Αυτό επιτρέπει σε ένα LS να μην είναι ενωμένος με τον MS αν δεν υπάρχει κάποιο μέλος στο Multicast group του. Διαχωρίζουμε δύο είδη handoff, το πρώτο αφορά τον παραλήπτη-MT και το άλλο αφορά την πηγή-MT.



Σχήμα 3.2 – Handoff του παραλήπτη

Μετακίνηση του παραλήπτη-MT

Στατική περίπτωση(Γνωστές Τοποθεσίες)

Οι τοποθεσίες στις οποίες ο παραλήπτης-MT μπορεί πιθανόν να επισκεφθεί καθώς μετακινείται μέσω της ανοιχτής του σύνδεσης, είναι γνωστές και οι πόροι δεσμεύονται από πριν. Έτσι, όλοι οι δυνατοί εξυπηρετητές LS, ενώνονται στον MS από πριν. Όταν ένας MT αλλάζει τοποθεσία σε ένα ξένο εξυπηρετητή LS', παραλαμβάνει αμέσως, μετά τη σύνδεση στον ξένο εξυπηρετητή LS', το multimedia stream. Το πλεονέκτημα της λύσης αυτής είναι ότι δεν υπάρχει παύση στο playback. Επιπρόσθετα, ο intra-media συγχρονισμός επιτυγχάνεται. Άρα, το μέγεθος του Buffer δε μεταβάλλεται. Ο εξυπηρετητής LS' στέλνει μήνυμα για να ενημερώσει τον LS ότι ο MT έχει αλλάξει την τρέχουσά του τοποθεσία και επομένως μπορεί να αφαιρεθεί από το Multicast group του.

Δυναμική περίπτωση(Άγνωστες Τοποθεσίες)

Οι εξυπηρετητές ενώνονται στον MS όταν διαχειρίζονται τουλάχιστον ένα MT στο πεδίο τους. Αν ένας εξυπηρετητής δεν έχει κανένα MT στο group του, θα αποσυνδεθεί από τον MS. Όταν ένας MT αλλάζει τοποθεσία από τον LS στον LS', δύο περιπτώσεις υπάρχουν:

1. Ο ξένος εξυπηρετητής LS' ήδη παραλαμβάνει το multimedia stream και οι πόροι έχουν ήδη κρατηθεί, άρα έχουμε στατική περίπτωση. Έτσι, ο LS θα αποσυνδεθεί από τον MS αν κανένας MT δεν πραμείνει στο multicast group του. Έπειτα, απελευθερώνει τους πόρους που δέσμευσε για αυτό το multicast stream.
2. Ο ξένος εξυπηρετητής LS' δεν παραλαμβάνει Multimedia stream. Στην περίπτωση αυτή, πρέπει να εγκαθιδρύσει σύνδεση με τον MS. Δύο περιπτώσεις για να έχουμε ασταμάτητο Playback, είναι:
    I. Να γίνεται buffer στον παραλήπτη-MT ένας αριθμός από media units που να αντιστοιχεί στο χρόνο μετακίνησης (handoff).
    II. Να γίνει buffer στον προηγούμενο LS ένας αριθμός από media units που να αντιστοιχεί στο χρόνο μετακίνησης (handoff).

Στην περίπτωση αυτή, ο παραλήπτης-MT συνεχίζει να παραλαμβάνει media units από τον τελευταίο LS μέχρι να συνδεθεί με τον LS' και να ξεκινήσει την παραλαβή. Αυτή η λύση είναι δυνατή αν ο MT εισέρχεται στην τομή των δύο τοποθεσιών.

Προτιμούμε την πρώτη λύση αν έχουμε αρκετή μνήμη στον παραλήπτη-MT. Αλλιώς, η δεύτερη είναι η καλύτερη λύση, παρόλη την απόσβεση του σήματος παραλαβής στην

τομή των δύο τοποθεσιών. Και στις δύο περιπτώσεις πρέπει να εκτιμήσουμε το χρόνο της μετακίνησης και να εμποδίσουμε την κράτηση των media units. Ως Δh ορίζουμε το χρόνο αυτό. Ο χρόνος που χρειάζεται για τη μετακίνηση δεσμεύεται από:

- Τη μέγιστη καθυστέρηση από τον παραλήπτη-MT στον LS': d^{max}
- Τη μέγιστη καθυστέρηση για την αίτηση σύνδεσης από τον LS' στον MS: D^{max}
- Τη μέγιστη καθυστέρηση από τον LS' μέχρι να παραλάβει το πρώτο media unit από τον MS: D^{max}
- Τη μέγιστη καθυστέρηση για τον παραλήπτη-MT μέχρι να παραλάβει το πρώτο media unit από τον MS: d^{max}

Η συνολική καθυστέρηση είναι: $2 * D^{max} + 2 * d^{max}$

Όταν οι πόροι είναι ήδη προκρατημένοι, και για να έχουμε ασταμάτητο stream, τότε ο παραλήπτης-MT πρέπει να κάνει Pre-buffer ένα δεσμευμένο αριθμό media units κατά τη διάρκεια $\Delta h = 2 * D^{max} + 2 * d^{max}$

Και στις δύο περιπτώσεις, για κάθε παραλήπτη-MT χρειάζεται να τηρείται intra-stream συγχρονισμός. Το μέγεθος του buffer που απαιτείται για να υπάρχει συγχρονισμός σε κάθε παραλήπτη-MT είναι:

$$BrMT = (\delta_r MT + \Delta h) * \tau_s$$

Μετακίνηση της πηγής-MT

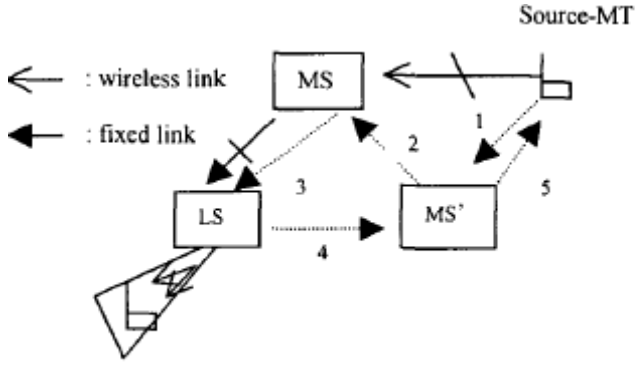


Figure 3. Source-MT handoff.

Σχήμα 3.3 – Handoff του αποστολέα

Η πηγή-MT μετακινείται από το πεδίο του MS σε ένα MS'. Ο MS' πρέπει να ενημερώσει τον MS για το γεγονός αυτό. Επίσης, ο MS πρέπει να ενημερώσει τους LS

που είναι μέλη του για την αλλαγή αυτή. Κάθε LS πρέπει να αλλάξει τη σύνδεσή του στον MS'. Όταν γίνουν όλες οι συνδέσεις με τον MS', οι LS ξεκινούν την παραλαβή του stream που γίνεται broadcast στο group με τα μέλη.

Κάθε εξυπηρετητής δεσμεύεται από μια D^{max} καθυστέρηση. Ο χρόνος που χρειάζεται για την πιο πάνω διαδικασία είναι δεσμευμένος από:

- Τη μέγιστη καθυστέρηση από τον MS' στον MS: D^{max}
- Τη μέγιστη καθυστέρηση από τον MS σε κάθε LS: D^{max}
- Τη μέγιστη καθυστέρηση για σύνδεση του κάθε LS στον MS': D^{max}
- Τη μέγιστη καθυστέρηση για κάθε LS για να παραλάβει το πρώτο media unit από τον MS': D^{max}

Άρα, η συνολική καθυστέρηση είναι $4 * D^{max}$

Για να μην διακοπεί το stream εξόδου σε κάθε LS, ο MS πρέπει να κρατήσει τουλάχιστον $\Delta MS = 4 * D^{max}$ media units για το buffer του.

Διαχείριση του Buffer

Και στις δύο περιπτώσεις που αναλύσαμε σε προηγούμενη ενότητα για τη μετακίνηση του παραλήπτη-MT, υπάρχει μια διακοπή της παραλαβής των media units κατά τη διάρκεια του χρόνου Δh. Στο χρόνο αυτό, το μέγεθος του buffer μειώνεται. Αν σε κάθε μετακίνηση το μέγεθος του buffer μειώνεται , τότε είναι πιθανόν να σταματήσει το playback. Επιλέγουμε να παίξουμε πιο αργά το playback κατά τη διάρκεια του χρόνου μετακίνησης μέχρι ο παραλήπτης-MT να μην έχει φυλάξει το μέγεθος του buffer.

Ως τ_s' ορίζουμε το νέο ρυθμό για το playback στον παραλήπτη-MT. Ως B_0=BrMT ορίζουμε το αρχικό μέγεθος του buffer και ως Δr το χρόνο που χρειάζεται για να ξαναφυλαχθεί το αρχικό μέγεθος του buffer.

Όσο μειώνεται το Δh, τόσο μειώνεται το Δr. Άρα, ο χρόνος για να ξαναφυλαχθεί το αρχικό μέγεθος του buffer είναι $\Delta r = \frac{\Delta h * \tau_s}{\tau_s - \tau_s'}$, where $\tau_s \geq \tau_s'$ αλλιώς το playback σταματά.

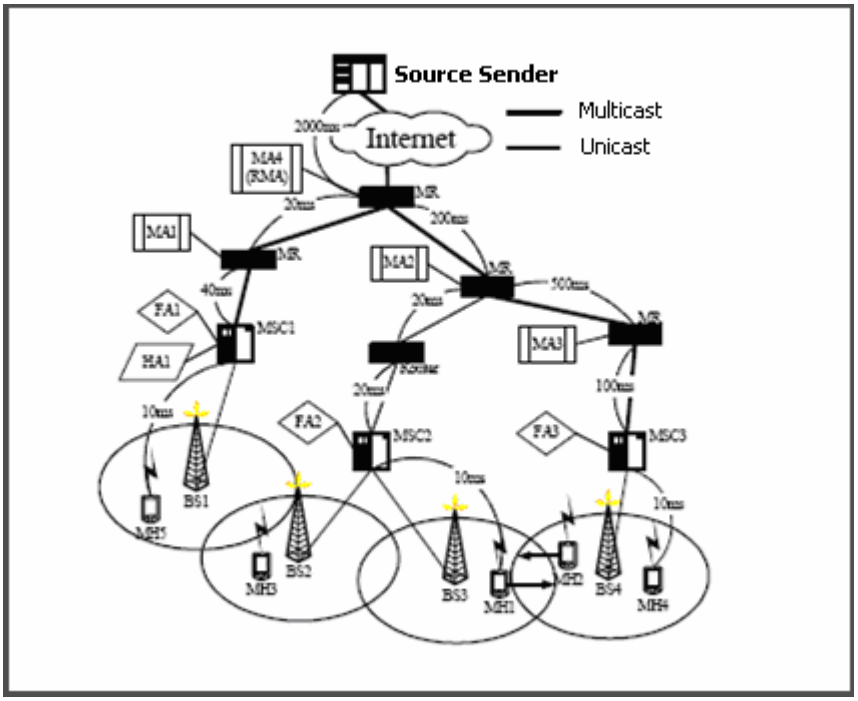
Στο τέλος του Δr, το buffer παίρνει το αρχικό του μέγεθος Bo. Ο παραλήπτης-MT, όταν μετακινείται από μια τοποθεσία σε άλλη, πρέπει να περνά κάποιο χρόνο στη νέα τοποθεσία πριν να μετακινηθεί αλλού. Ο χρόνος αυτός πρέπει να είναι μεγαλύτερος από $\Delta h + \Delta r$.

Διαθεσιμότητα πόρων

Ο χρόνος για τη μετακίνηση από μια τοποθεσία σε μια άλλη, κυμαίνεται μεταξύ $2* d^{min} \leq \Delta h \leq 2* D^{max} + 2* d^{max}$ αν οι πόροι είναι διαθέσιμοι αμέσως για τον παραλήπτη-MT. Σε αντίθετη περίπτωση, προτείνεται πρωτόκολο όταν δεν υπάρχουν διαθέσιμοι πόροι:

Όταν μετακινείται, ο παραλήπτης-MT ζητά να συνδεθεί με τον ξένο LS. Αν υπάρχουν ήδη δεσμευμένοι πόροι ή υπάρχουν διαθέσιμοι, η σύνδεση γίνεται αποδεκτή και ο LS απαντά απευθείας στέλνοντας media units(μετά προσθέτει τον παραλήπτη-MT στο multicast group). Αλλιώς, κάθε $n*\Pi$ χρόνο, ο παραλήπτης-MT πρέπει να ξανακάνει αίτηση. Αν μετά από m φορές δεν υπάρχεο απόκριση τότε η σύνδση χάνεται. Η παράμετρος n είναι ένας ακέραιος που κυμαίνεται από 2 μέχρι m και Π είναι η ελάχιστη interval για την επανάληψη του αιτήματος.

3.2.1 Synchronized Multimedia Multicast – SMM



Σχήμα 3.4 – Αναπαράσταση πρωτοκόλλου

Ένα άλλο πρωτόκολο, είναι το Synchronized Multimedia Multicast SMM [16] το οποίο παρέχει seamless playback συνεχούς media στον κινητό χρήστη MH, με λίγα buffers,

ακόμη και όταν ο κινητός χρήστης μετακινείται σε άλλα κινητά δίκτυα με διαφορετικές end-to-end καθυστερήσεις (EED).
Υπάρχουν δύο πιθανές περιπτώσεις για τον κινητό χρήστη για να παραλάβει πακέτα μέσω multicast. Η πρώτη λέγεται Remote Subscription (RS) και η δεύτερη λέγεται Home Subscription (HS).

Remote Subscription (RS)

Στην περίπτωση του RS, όποτε ο MH επιθυμεί να ενταχθεί σε ένα συγκεκριμένο Multicast group, ο FA(Foreign Agent) του κινητού δικτύου στο οποίο ο MH ανήκει, είναι υπεύθυνος γι'αυτό. Τότε, αυτός ο FA μεταφέρει τα παραληφθέντα Multicast πακέτα στον MH μέσω των κινητών του καναλιών. Το πλεονέκτημα της RS τεχνικής είναι ότι μπορεί να κάνει Multicast δεδομένα στους FA και να πετυγχαίνει τα βέλτιστα δρομολόγια. Όμως, απαιτεί να υπάρχει ένας Multicast δρομολογητής (MR) σε κάθε κινητό δίκτυο στο οποίο ο MH μπορεί να μοιράζεται. Άρα η τεχνική αυτή συνεπάγεται υψηλό κόστος και λιγότερη ευελιξία.

Home Subscription (HS)

Η HS τεχνική βασίζεται στον HA(Home Agent) του MH για να ενταχθεί στο multicast group. Ο source sender(SS) μεταφέρει τα multicast πακέτα πρώτα στον HA του κάθε κινητού χρήστη. Στη συνέχεια, ο HA ενώνει τα multicast πακέτα σε ένα ενιαίο πακέτο και το στέλνει στην τρέχουσα care-of-address του MH όταν μετακινείται σε ένα ξένο δίκτυο. Σαφώς το κόστος εφαρμογής είναι χαμηλότερο από την τεχνική RS. Παρόλα αυτά, η τεχνική HS έχει δύο σημαντικά μειονεκτήματα. Το πρώτο, είναι ότι το πακέτο πρέπει να δρομολογηθεί από τον SS στον HA και στη συνέχεια στον MH παρόλο που θα μπορούσε να παέι απευθείας από τον SS στον MH. Το δεύτερο είναι ότι όλοι οι MH σε ένα συγκεκριμένο κελί δεν μπορούν να μοιράζονται τα ίδια πακέτα ακόμη και όταν βρίσκονται στο ίδιο multicast group.

Η Multicast αρχιτεκτονική του SMM πρωτοκόλου

Η τεχνική αυτή κτίζει μια δομή όπου κάθε MA στον MR ελέγχει διάφορα FA για να βοηθήσει τον MH να ενταχθεί στο multicast group και να μεταφέρει τα παραληφθέντα multicast πακέτα στον FA. Ο αρχικός MA (RMA) λειτουργεί ως το σημείο συγχρονισμού για να οργανώσει όλους τους FA να μεταφέρουν πακέτα.

3.2.1.1 Λειτουργίες του SMM

Το βήμα ένταξης στο Multicast group

Όποτε ένας κινητός χρήστης ζητά να ενταχθεί σε ένα multicast group, πρέπει να εκτελεστούν οι τρεις πιο κάτω λειτουργίες.

1. Ο κινητός χρήστης στέλνει ένα νέο Membership Request μήνυμα στον ξένο ατζέντη του (FA). Ο FA συγκρίνει το μήνυμα που παρέλαβε με τις εισόδους στον πίνακα πληροφοριών του.
2. Αν ο ξένος ατζέντης υποστηρίζει IP Multicast, μεταφέρει το IGMP membership report στον MA για να ενταχθεί στο multicast group.
3. Ο MA στέλνει το IGMP membership report για να ενταχθεί στο multicast group. Γεφυρώνει τα Multicast πακέτα στον FA μέσω unicast σύνδεσης όταν παραλάβει τα πακέτα αυτά από τον SS.

Βήμα Συγχρονισμού

Για να επιτευχθεί συγχρονισμένο playback του συνεχούς media stream για όλους τους κινητούς χρήστες, το πρωτόκολο SMM πρέπει πρώτα να υπολογίσει τις end-to-end EED καθυστερήσεις μεταξύ του αρχικού multicast ατζέντη(RMA) και κάθε ξένου ατζέντη FA για να συντονίσει πότε να μεταφέρει το multicast πακέτο στον κινητό χρήστη ταυτόχρονα όταν ξεκινά το multicast.

1. Ο RMA περιοδικά κάνει Multicast το summarized membership report query (SMRQ) μήνυμα σε όλους τους FA στο group και ξεκινά τα timers για τις αποκρίσεις αυτού του SMRQ μηνύματος από όλους τους MA.
2. Όταν ο MA παραλάβει το SMRQ μήνυμα από τον RMA, το μεταφέρει σε όλους τους FA που ελέγχει και ξεκινά να χρονομετρά για το μήνυμα αυτό.
3. Μόλις ο FA παραλάβει το SMRQ μήνυμα, απαντά με ένα Summarized Membership Report(SMR) μήνυμα στον MA του. Μετά, ο MA ενημερώνει τις πληροφορίες στον group πίνακά του αν χρειάζεται και μετρά τον round-trip χρόνο (RTTMF) ανάμεσα σε αυτόν και τον FA σύμφωνα με την τιμή του timer. Όταν ο MA παραλάβει το SMR από όλους τους FA του, υπολογίζει τη μέγιση DMFi,j και μεταφέρει το SMR πίσω στον RMA.

$$DMF_{i,j}^{t} = \alpha * DMF_{i,j}^{t-1} + (1-\alpha) * (RTTMF_{i,j} / 2), 0 < \alpha < 1 \quad (1)$$

4. Όταν ο RMA παραλάβει το SMR μήνυμα από το MA, μετρά το roundtrip χρόνο(RTTRMMr,i) ανάμεσα σε αυτόν και τον MA και υπολογίζει την τρέχουσα end-to-end καθυστέρηση. Μόλις ο RMA παραλάβει όλα τα SMR μηνύματα από όλους τους MA του, μπορεί να καθορίσει τη μέγιστη EED μεταξύ αυτού και όλων των MA.

$$DRMM_{r,i}^{t} = \beta * DRMM_{r,j}^{t-1} + (1-\beta) * (RTTRMM_{r,j} / 2), 0 < \beta < 1 \quad (2)$$

5. Τα πιο πάνω επαναλαμβάνονται μέχρι ο RMA να παραλάβει το πρώτο multicast πακέτο από τον SS. Μετά, ο RMA κάνει multicast το πακέτο σε κάθε MA μαζί με τη MAX(DRMMr,i) και το DRMMr,i, τιμή αυτού του MA.
6. Κάθε MA υπολογίζει την EED ανάμεσα στο RMA και κάθε ελεγχόμενο FA, όταν παραλαμβάνει το πακέτο από τον RMA. Το πακέτο στέλνεται σε κάθε ελεγχόμενο FA μαζί με τη MAX(DRMMr,i) και την DRMMr,j του FA.

$$DRMF_{r,j}^{t} = DRMM_{r,i}^{t} - Max(DMF_{i,j}^{t}) + DMF_{i,j}^{t} \quad (3)$$

7. Όταν ο FA παραλάβει τα multicast πακέτα από τον MA του, θα κάνει buffer όλα τα εισερχόμενα πακέτα για μια συγκεκριμένη χρονική διάρκεια, η οποία ισούται με τη χρονική διαφορά μεταξύ της μέγιστης καθυστέρησης MAX(DRMMr,i) στο multicast group και στην DRMFr,j τιμή από τον RMA και μετά συγχρονισμένα μεταφέρει τα πακέτα αυτά στον MH.

$$TF_{j} = Max(DRMM_{r,i}) - DRMF_{r,j} \quad (4)$$

Το βήμα μετακίνησης

Όταν ο κινητός χρήστης ολοκληρώσει τις handoff λειτουργίες και ξεκινά να παίρνει multicast πακέτα από το νέο BS, το buffer του μικραίνει στο μηδέν όταν το multicast δέντρο αλλάξει.

3.2.2 Multicast μετάδοση πολυμεσικών δεδομένων με τη χρήση μιας multicast ροής δεδομένων

Σε αυτήν την ενότητα παρουσιάζεται ο σχεδιασμός και η ανάλυση ενός μηχανισμού για την multicast μετάδοση πολυμέσων με την χρήση μιας multicast ροής δεδομένων [17]. Η αξιολόγησή του προτεινόμενου μηχανισμού έγινε με χρήση μοντέλου που αναπτύχθηκε σε περιβάλλον εξομοιωτή. Επιπλέον, ο μηχανισμός ο οποίος αναπτύχθηκε χρησιμοποιεί ένα μηχανισμό για την ποσοτική μέτρηση της «ικανοποίησης» κάθε πελάτη από την υπηρεσία. Με τη χρήση του μηχανισμού που αναπτύχθηκε ο

εξυπηρετητής / αποστολέας μπορεί να εξυπηρετεί «δίκαια» τους πελάτες. Με αυτόν τον τρόπο αντιμετωπίζεται το βασικότερο μειονέκτημα της multicast μετάδοσης πολυμέσων με την χρήση μιας multicast ροής δεδομένων. Βασικό συμπέρασμα της αξιολόγησης του μηχανισμού ο οποίος παρουσιάζεται είναι πως ο μηχανισμός που αναπτύχθηκε έχει φιλική συμπεριφορά προς το πρωτόκολλο TCP με την έννοια ότι επιτρέπει την μετάδοση TCP κίνησης με ικανοποιητική απόδοση και δεν κυριαρχεί επί της TCP κίνησης.

Το πιο σημαντικό χαρακτηριστικό του προτεινόμενου μηχανισμού είναι το γεγονός ότι προσπαθεί ταυτόχρονα να ικανοποιήσει την ετερογενή ομάδα των παραληπτών (με τις τρέχουσες δικτυακές συνθήκες) και ταυτόχρονα προσπαθεί να έχει φιλική συμπεριφορά προς τους άλλους χρήστες του δικτύου. Επιπλέον οι δυνατότητες καταγραφής των δικτυακών συνθηκών του προτεινόμενου μηχανισμού χρησιμοποιούν ένα συνδυασμό παραμέτρων για την εξαγωγή συμπερασμάτων σχετικά με τις δικτυακές συνθήκες. Επιπλέον όλα τα απαραίτητα συστατικά για την εκτέλεση του προτεινόμενου μηχανισμού εγκαθίστανται μόνο στην εφαρμογή του αποστολέα γεγονός που επιτρέπει σε οποιαδήποτε εφαρμογή η οποία μπορεί να συμμετέχει σε RTP συνόδους (για παράδειγμα MBONE εργαλεία) να μπορούν να εκμεταλλευτούν τα πλεονεκτήματα του προτεινόμενου μηχανισμού χωρίς να απαιτούνται αλλαγές στην πλευρά του παραλήπτη.

Εισάγεται η έννοια της συνάρτησης δικαιοσύνης *RF* (Receiver Fairness) και της συνάρτησης δικαιοσύνης των παραληπτών *IRF* (Inter - Receiver Fairness). Ο αποστολέας του μηχανισμού χρησιμοποιεί τις συναρτήσεις *RF* και *IRF* προκειμένου να υπολογίσει το ρυθμό μετάδοσης της multicast ροής δεδομένων που ικανοποιεί περισσότερο την ομάδα των παραληπτών. Η συνάρτηση *RF* για τον παραλήπτη *i* ορίζεται ως εξής:

$$RF_i(r) = \frac{\min(r_i, r)}{\max(r_i, r)}$$

Όπου r_i είναι ο ρυθμός μετάδοσης δεδομένων που ο παραλήπτης *i* προτιμά (το r_i αντιπροσωπεύει τον ρυθμό μετάδοσης δεδομένων τον οποίο ο παραλήπτης *i* θα προτιμούσε αν ήταν ο μοναδικός παραλήπτης της multicast ροής δεδομένων) και *r* είναι ο ρυθμός μετάδοσης δεδομένων τον οποίο ο αποστολέας σκοπεύει να χρησιμοποιήσει. Από την παραπάνω σχέση είναι φανερό ότι η συνάρτηση *RF* έχει τιμές στο διάστημα [0.0, 1.0], και ο παραλήπτης *i* είναι ικανοποιημένος όταν $RF_i \approx 1.0$

και πλήρως ικανοποιημένος όταν $RF_i = 1.0$ (στην περίπτωση που $r_i = r$). Ο παραλήπτης i δεν είναι ικανοποιημένος όταν $RF_i << 1.0$. Ο παραλήπτης i μπορεί να μην είναι ικανοποιημένος είτε λόγω απωλειών πακέτων (όταν $r_i < r$) είτε λόγω της μη εκμετάλλευσης διαθέσιμου εύρους ζώνης (όταν $r_i > r$). Η συνάρτηση IRF ορίζεται για μια ομάδα n παραληπτών ως εξής:

$$IRF(r) = \sum_{i=1}^{n} a_i * RF_i(r)$$

Υπό τον όρο $\sum_{i=1}^{n} a_i = 1$ και $0 \le a_i \le 1, i = 1,...,n$

Όπου r είναι ο ρυθμός μετάδοσης δεδομένων τον οποίο ο αποστολέας σκοπεύει να χρησιμοποιήσει και a_i είναι το βάρος του παραλήπτη i στον υπολογισμό της τιμής της συνάρτησης IRF . Από την παραπάνω σχέση, είναι φανερό ότι για μεγάλες τιμές της συνάρτησης IRF η ομάδα των παραληπτών είναι συνολικά περισσότερο ικανοποιημένη, ενώ για μικρές τιμές της συνάρτησης IRF η ομάδα των παραληπτών είναι συνολικά λιγότερο ικανοποιημένη. Χρησιμοποιούμε τις συναρτήσεις RF και IRF προκειμένου να καθορίσουμε τον ρυθμό μετάδοσης δεδομένων ο οποίος ικανοποιεί περισσότερο την ομάδα των παραληπτών.

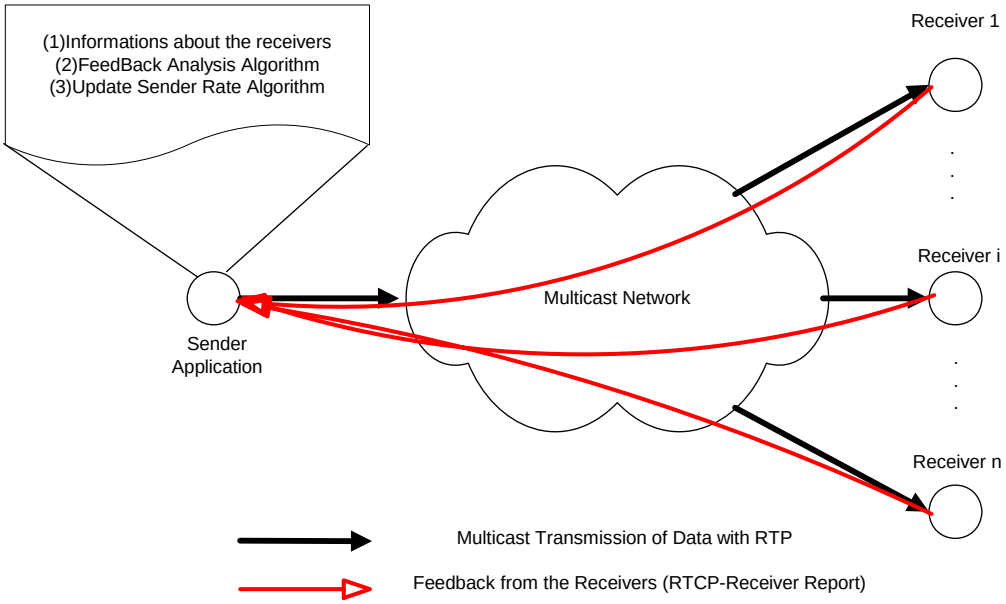
3.2.2.1 Περιγραφή του προτεινόμενου μηχανισμού

Αρχιτεκτονική του προτεινόμενου μηχανισμού

Η παράγραφος αυτή παρουσιάζει την αρχιτεκτονική του μηχανισμού για την multicast μετάδοση πολυμεσικών δεδομένων με την χρήση μιας multicast ροής δεδομένων. Υποθέτουμε ότι έχουμε ένα αποστολέα ο οποίος μεταδίδει πολυμεσικά δεδομένα σε μια ομάδα από n παραλήπτες με την χρήση μιας multicast ροής δεδομένων. Ο αποστολέας χρησιμοποιεί τα πρωτόκολλα RTP / RTCP για τη μετάδοση των πολυμεσικών δεδομένων. Οι παραλήπτες λαμβάνουν τα πολυμεσικά δεδομένα και πληροφορούν τον αποστολέα για την ποιότητα της μετάδοσης με τη χρήση RTCP αναφορών παραλήπτη. Ο αποστολέας συλλέγει τις RTCP αναφορές παραληπτών, τις αναλύει και επιλέγει τον ρυθμό μετάδοσης r ο οποίος ικανοποιεί περισσότερο την ομάδα των παραληπτών με τις δεδομένες δικτυακές συνθήκες.

Ο αποστολέας κρατά πληροφορίες για κάθε παραλήπτη i και κάθε φορά που λαμβάνει μια RTCP αναφορά παραλήπτη από τον παραλήπτη i, εκτιμά το ρυθμό μετάδοσης δεδομένων τον οποίο προτιμά αυτός ο παραλήπτης r_i, ο οποίος αντιπροσωπεύει τον ρυθμό μετάδοσης δεδομένων τον οποίο ο παραλήπτης i θα προτιμούσε αν ήταν ο μοναδικός παραλήπτης της multicast ροής δεδομένων. Η εκτίμηση του επιθυμητού ρυθμού μετάδοσης δεδομένων για τον παραλήπτη i, r_i γίνεται με την χρήση του αλγόριθμου ο οποίος παρουσιάζεται πιο κάτω.

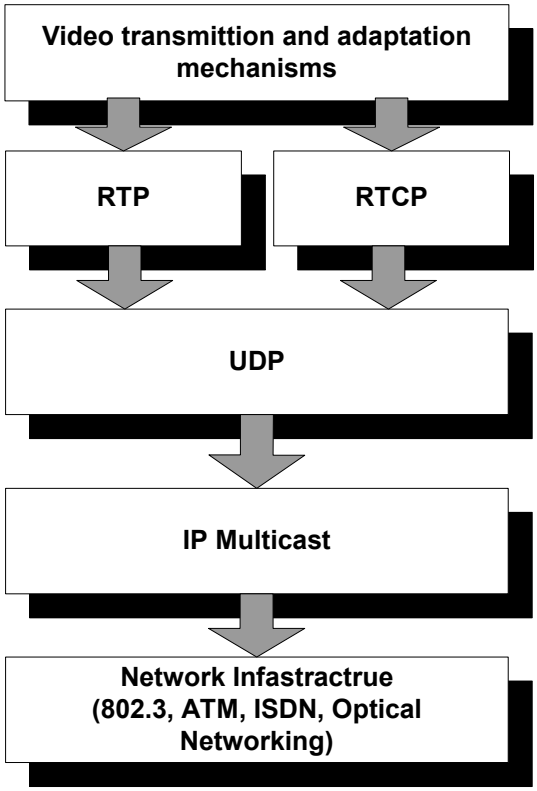
Ο αποστολέας σε επαναλαμβανόμενα χρονικά διαστήματα εκτιμά το ρυθμό μετάδοσης δεδομένων r για τη multicast ροή δεδομένων των πολυμεσικών δεδομένων. Με το ρυθμό μετάδοσης δεδομένων r, ο αποστολέας προσπαθεί να ικανοποιήσει περισσότερο την ομάδα των n παραληπτών με τις δεδομένες δικτυακές συνθήκες. Ο αποστολέας χρησιμοποιεί ως μέσο σύγκρισης της συνολικής ικανοποίησης των παραληπτών τη συνάρτηση IRF και συνήθως αντιμετωπίζει όλους τους παραλήπτες ως ίσους, που σημαίνει ότι τα βάρη a_i για όλους τους παραλήπτες $i, i = 1...n$ στη συνάρτηση IRF είναι $a_i = \frac{1}{n}$. Αν ο αποστολέας επιθυμεί να μεταχειριστεί όχι ισοδύναμα την ομάδα των παραληπτών μπορεί να ορίσει προτεραιότητες με τον χρησιμοποιεί διαφορετικές τιμές στα βάρη a_i. Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική του μηχανισμού μετάδοσης πολυμεσικών δεδομένων.



Σχήμα 3.5 – Αρχιτεκτονική του μηχανισμού μετάδοσης πολυμεσικών δεδομένων

Η ποιότητα παρουσίασης των πολυμεσικών δεδομένων εξαρτάται από το ρυθμό απώλειας των πακέτων δεδομένων και στη διακύμανση καθυστέρησης (delay jitter) κατά τη διάρκεια της μετάδοσης πάνω από το δίκτυο. Απώλειες πακέτων δεδομένων ή απότομη αύξηση της διακύμανσης καθυστέρησης μπορεί να θεωρηθούν ως ενδείξεις προβλημάτων κατά τη μετάδοση των πολυμεσικών δεδομένων πάνω από το δίκτυο. Σε αυτήν την περίπτωση ο αποστολέας θα πρέπει να προσαρμόσει το ρυθμό μετάδοσης των πολυμεσικών δεδομένων προκειμένου να αποφευχθεί το φαινόμενο της συμφόρησης του δικτύου. Οι εφαρμογές πολυμέσων έχουν ανώτερα και κατώτερα όρια στο ρυθμό απώλειας των πακέτων και στη διακύμανση καθυστέρησης. Αν ο ρυθμός απώλειας των πακέτων ή η διακύμανση καθυστέρησης βρεθούν πάνω από αυτά τα όρια, η μετάδοση των πολυμεσικών δεδομένων είναι προβληματική και πρέπει να σταματήσει.

Για την υλοποίηση της προσαρμογής των πολυμέσων στον επιθυμητό ρυθμό μετάδοσης της multicast ροής συνήθης τεχνικές είναι η χρήση της τεχνικής προσαρμογής του ρυθμού μετάδοσης με αλλαγή των παραμέτρων του κωδικοποιητή (rate shaping) ή η χρήση της τεχνικής απόρριψης πλαισίων (frame dropping) ή ένας συνδυασμός των παραπάνω. Η υλοποίηση του μηχανισμού προσαρμογής των πολυμέσων εξαρτάται από την κωδικοποίηση της πολυμεσικής πληροφορίας. Για παράδειγμα αν θέλουμε να χρησιμοποιήσουμε την τεχνική απόρριψης πλαισίων για την προσαρμογή ενός MPEG βίντεο, θα πρέπει να χρησιμοποιήσουμε επιλεκτική απόρριψη πλαισίων λόγω του γεγονότος ότι η MPEG κωδικοποίηση χρησιμοποιεί κωδικοποίηση ανάμεσα στα πλαίσια (inter - frame encoding) και επομένως κάποια πλαίσια περιέχουν πληροφορίες σχετικές με άλλα πλαίσια. Η στοίβα πρωτοκόλλων του προτεινόμενου μηχανισμού είναι η εξής:



Σχήμα 2.6 – Στοίβα πρωτοκόλλων

Ο μηχανισμός οποίος προτείνεται βασίζεται στη multicast μετάδοση πολυμέσων. Στο επίπεδο συνόδου, σύμφωνα με το μοντέλο αναφοράς OSI, γίνεται χρήση των RTP / RTCP πρωτοκόλλων. Τα πρωτόκολλα RTP / RTCP χρησιμοποιούνται επίσης λόγω των δυνατοτήτων που παρέχουν για την παροχή πληροφοριών ανάδρασης με την χρήση των RTCP αναφορών. Τα πρωτόκολλα RTP / RTCP χρησιμοποιούν το πρωτόκολλο UDP στο επίπεδο μεταφοράς, ενώ το πρωτόκολλο IP multicast χρησιμοποιείται στο επίπεδο δικτύου. Ως υποδομή δικτύου μπορεί να χρησιμοποιηθεί οποιαδήποτε δικτυακή τεχνολογία υποστηρίζει το πρωτόκολλο IP multicast. Η στοίβα πρωτοκόλλων που χρησιμοποιούνται από τον προτεινόμενο μηχανισμό παρουσιάζεται στο παραπάνω σχήμα.

Εκτίμηση του επιθυμητού ρυθμού μετάδοσης κάθε παραλήπτη

Ο αλγόριθμος ανάλυσης των πληροφοριών του δικτύου αναλύει τις πληροφορίες που ο παραλήπτης i αποστέλλει στον αποστολέα (με την χρήση RTCP αναφορών παραλήπτη) σχετικά με την ποιότητα μετάδοσης της πολυμεσικής πληροφορίας. Κάθε φορά που ο αποστολέας λαμβάνει μια RTCP αναφορά από τον παραλήπτη i, εκτελεί τον αλγόριθμο ανάλυσης των πληροφοριών του δικτύου προκειμένου να υπολογίσει τον

επιθυμητό ρυθμό μετάδοσης r_i για τον παραλήπτη i . Ο ρυθμός μετάδοσης r_i αντιπροσωπεύει τον ρυθμό μετάδοσης δεδομένων τον οποίο ο παραλήπτης i θα προτιμούσε εάν ήταν ο μοναδικός παραλήπτης της multicast ροής δεδομένων.

Ο αλγόριθμος ανάλυσης των πληροφοριών του δικτύου, για τον υπολογισμό της νέας τιμής του επιθυμητού ρυθμού μετάδοσης r_i για τον παραλήπτη i χρησιμοποιεί την παρακάτω διαδικασία:

$$if\,(network = unload) \rightarrow r_{i-new} = r_{i-old} + R_{increase}$$

$$if\,(network = load) \rightarrow r_{i-new} = r_{i-old}$$

$$if\,(network = congestion) \rightarrow r_{i-new} = r_{i-old} * (1 - LR_{new})$$

$$r_{i-old} = r_{i-new}$$

Όπου:

r_{i-new}: η νέα τιμή του επιθυμητού ρυθμού μετάδοσης r_i για τον παραλήπτη i.

r_{i-old}: η παλιά τιμή του επιθυμητού ρυθμού μετάδοσης r_i για τον παραλήπτη i.

$R_{increase}$: είναι ο παράγοντας με τον οποίο ο αποστολέας αυξάνει τον ρυθμό μετάδοσης δεδομένων στην περίπτωση που υπάρχει διαθέσιμο εύρος ζώνης.

Όταν η δικτυακή κατάσταση του παραλήπτη i είναι κατάσταση μη φόρτου (unload), αυξάνεται ο επιθυμητός ρυθμός μετάδοσης r_i με το να προστίθεται ο παράγοντας $R_{increase}$, προκειμένου να μειωθεί η δυσαρέσκεια του παραλήπτη i λόγω του ανεκμετάλλευτου εύρους ζώνης. Όταν η δικτυακή κατάσταση του παραλήπτη i είναι κατάσταση συμφόρησης (congestion), μειώνεται ο επιθυμητός ρυθμός μετάδοσης r_i πολλαπλασιάζοντας τον με τον παράγοντα $1 - LR_{new}$ (το οποίο σημαίνει ότι ο επιθυμητός ρυθμός μετάδοσης r_i του παραλήπτη i είναι ο μέγιστος ρυθμός μετάδοσης ο οποίος δεν προκαλεί απώλειες πακέτων στον παραλήπτη i) προκειμένου να μειωθεί η δυσαρέσκεια του παραλήπτη i λόγω των απωλειών των πακέτων. Όταν η δικτυακή κατάσταση του παραλήπτη i είναι κατάσταση φόρτου (load), δε μεταβάλλεται ο επιθυμητός ρυθμός μετάδοσης r_i του παραλήπτη i λόγω του ότι ο παραλήπτης i είναι ικανοποιημένος με τον τρέχον ρυθμό μετάδοσης δεδομένων. Επιπλέον ο επιθυμητός

ρυθμός μετάδοσης r_i για τον παραλήπτη i δεν μπορεί να είναι μεγαλύτερος από μια τιμή $r_{\max}$ και δεν μπορεί να είναι μικρότερος από μια τιμή $r_{\min}$. Οι τιμές των $r_{\max}$ και $r_{\min}$ εξαρτάται από τον τύπο του δικτύου στον οποίο εκτελείται η εφαρμογή και από τη φύση της εφαρμογής.

Η συμπεριφορά και λειτουργία του αλγόριθμου ανάλυσης των πληροφοριών του δικτύου επηρεάζεται από τις παραμέτρους οι οποίες χρησιμοποιούνται ($\alpha, \beta, \gamma, LR_c, LR_u, R_{increase}$). Η επιλογή των παραπάνω παραμέτρων εξαρτάται από το δίκτυο στον οποίο εκτελείται η εφαρμογή και το κυρίαρχο μοντέλο κίνησης σε αυτό το δίκτυο. Οι κατάλληλες τιμές των παραπάνω παραμέτρων για κάθε τύπο δικτύου μπορεί καθοριστούν μέσα από μια σειρά πειραμάτων και εξομοιώσεων.

Εκτίμηση του ρυθμού μετάδοσης δεδομένων

Ο αποστολέας σε επαναλαμβανόμενα χρονικά διαστήματα υπολογίζει το ρυθμό μετάδοσης δεδομένων r για την multicast μετάδοση των πολυμεσικών δεδομένων με τη χρήση του αλγόριθμου ανανέωσης του ρυθμού μετάδοσης δεδομένων. Η εκτίμηση του ρυθμού μετάδοσης δεδομένων r του αποστολέα στοχεύει στο να αυξήσει συνολικά την ικανοποίηση της ομάδας των παραληπτών βασιζόμενος στο μέτρο ικανοποίησης το οποίο παρέχει η συνάρτηση IRF. Όταν ο αποστολέας εκτιμά το νέο ρυθμό μετάδοσης δεδομένων r προσπαθεί να παρέχει στην ομάδα των παραληπτών την καλύτερη δυνατή ικανοποίηση με τις δεδομένες δικτυακές συνθήκες.

Ο αλγόριθμος ανανέωσης του ρυθμού μετάδοσης δεδομένων χρησιμοποιεί ένα αλγόριθμο προσθετικής αύξησης και πολλαπλασιαστικής μείωσης (Additive Increase Multiplicative Decrease - AIMD) προκείμενου να υπολογίσει τον νέο ρυθμό μετάδοσης δεδομένων r. Ο αλγόριθμος ανανέωσης του ρυθμού μετάδοσης δεδομένων είναι παρόμοιος με τον αλγόριθμο που καθορίζει τον ρυθμό μετάδοσης μιας TCP σύνδεσης. Είναι καλό να επιλεχθεί ένας αλγόριθμος παρόμοιος με τον αλγόριθμο τον οποίο χρησιμοποιεί το TCP για λόγους δικαιοσύνης στη χρήση των δικτυακών πόρων (όπως παράδειγμα το εύρος ζώνης) ειδικά κατά τις περιόδους συμφόρησης στο δίκτυο. Εφαρμογές με «κακή» συμπεριφορά κατά τις περιόδους συμφόρησης επιτείνουν το πρόβλημα της συμφόρησης και οδηγούν σε μη δίκαια κατανομή των δικτυακών πόρων στους χρήστες του δικτύου. Σε ένα δίκτυο το οποίο βρίσκεται σε κατάσταση συμφόρησης, όλες οι εφαρμογές θα πρέπει να μειώσουν το ρυθμό μετάδοσης δεδομένων και να μοιραστούν μεταξύ τους, τους δικτυακούς πόρους δίκαια. Ο AIMD

μηχανισμός είναι παρόμοιος με τον αλγόριθμο ανανέωσης του ρυθμού μετάδοσης δεδομένων ο οποίος χρησιμοποιείται σε πολλές εφαρμογές στο Διαδίκτυο σήμερα: Όταν η εφαρμογή αντιληφθεί διαθέσιμο εύρος ζώνης στο δίκτυο αυξάνει τον ρυθμό μετάδοσης δεδομένων με το να προσθέτει ένα παράγοντα στον ρυθμό μετάδοσης δεδομένων (probing). Στην περίπτωση συμφόρησης στο δίκτυο, η εφαρμογή μειώνει τον ρυθμό μετάδοσης δεδομένων με το να πολλαπλασιάσει τον ρυθμό μετάδοσης δεδομένων με ένα παράγοντα μικρότερο του 1 (back off). Εάν η εφαρμογή δεν αντιμετωπίζει απώλειες πακέτων ή ο ρυθμός απωλειών των πακέτων είναι μικρός η εφαρμογή διατηρεί σταθερό τον ρυθμό μετάδοσης δεδομένων.

Όταν ο αποστολέας εκτιμά τον νέο ρυθμό μετάδοσης δεδομένων r έχει τις τρεις παρακάτω εναλλακτικές δυνατότητες:

1. Να αυξήσει το ρυθμό μετάδοσης δεδομένων προσθέτοντας ένα παράγοντα, $R_{increase}$ (οπότε προκύπτει ο ρυθμός μετάδοσης r_{incr}).

2. Να διατηρήσει τον προηγούμενο ρυθμό μετάδοσης δεδομένων (οπότε προκύπτει ο ρυθμός μετάδοσης r_{stay}).

3. Να μειώσει το ρυθμό μετάδοσης δεδομένων με το να πολλαπλασιάσει τον ρυθμό μετάδοσης δεδομένων r με ένα παράγοντα μικρότερο του 1, $R_{decrease}$ (οπότε προκύπτει ο ρυθμός μετάδοσης r_{dcr}).

Στη συνέχεια, ο αλγόριθμος ανανέωσης του ρυθμού μετάδοσης δεδομένων επιλέγει το νέο ρυθμό μετάδοσης δεδομένων r από τα $\{ r_{incr}, r_{stay}, r_{dcr} \}$ ο οποίος παρέχει τη μεγαλύτερη συνολικά ικανοποίηση στην ομάδα των παραληπτών, που σημαίνει ο ρυθμός μετάδοσης δεδομένων r από τα $\{ r_{incr}, r_{stay}, r_{dcr} \}$ ο οποίος έχει τη μεγαλύτερη τιμή της συνάρτησης IRF. Επιπλέον, ο αλγόριθμος ανανέωσης του ρυθμού μετάδοσης δεδομένων ανανεώνει τις παλιές τιμές των επιθυμητών ρυθμών μετάδοσης δεδομένων για όλους τους παραλήπτες προκειμένου ο αλγόριθμος ανάλυσης των πληροφοριών του δικτύου να είναι ενήμερος για τον τρέχον ρυθμό μετάδοσης των δεδομένων. Ακολουθεί η σύνοψη της λειτουργίας του αλγόριθμου ανανέωσης του ρυθμού μετάδοσης δεδομένων:

$$r_{incr} = r_{old} + R_{increase}$$

$$r_{stay} = r_{old}$$

$$r_{dcr} = r_{old} * R_{decrease}$$

$$r_{new} = MaxIRF_{r=r_{incr}, r_{stay}, r_{dcr}}[IRF(r)]$$

$$receiver - i_{i=1..n} : r_{i-old} = r_{new}$$

$$r_{old} = r_{new}$$

Όπου r_{new} είναι ο νέος ρυθμός μετάδοσης δεδομένων του αποστολέα και r_{old} είναι ο προηγούμενος ρυθμός μετάδοσης δεδομένων του αποστολέα. Επιπλέον ο νέος ρυθμός μετάδοσης r_{new} δεν μπορεί να είναι μεγαλύτερος από την τιμή $r_{\max}$ και μικρότερος από την τιμή $r_{\min}$. Οι τιμές των $r_{\max}$ και $r_{\min}$ εξαρτώνται από τον τύπο του δικτύου και τον τύπο της εφαρμογής.

Ο αλγόριθμος ανανέωσης του ρυθμού μετάδοσης δεδομένων δε λαμβάνει υπόψη άμεσα τις τρέχουσες δικτυακές συνθήκες κατά τον υπολογισμό του νέου ρυθμού μετάδοσης δεδομένων r_{new} από τον αποστολέα. Οι τρέχουσες δικτυακές συνθήκες λαμβάνονται υπόψη από τον αλγόριθμο ανάλυσης των πληροφοριών του δικτύου κατά τον υπολογισμό του επιθυμητού ρυθμού μετάδοσης δεδομένων r_i των παραληπτών. Λόγω του γεγονότος ότι οι επιθυμητοί ρυθμοί μετάδοσης δεδομένων r_i των παραληπτών εμπλέκονται στον υπολογισμό της τιμής της συνάρτησης IRF , ο αλγόριθμος ανανέωσης του ρυθμού μετάδοσης δεδομένων λαμβάνει έμμεσα υπόψη του τους επιθυμητούς ρυθμούς μετάδοσης δεδομένων r_i των παραληπτών.

Με την παραπάνω διαδικασία ο ρυθμός μετάδοσης δεδομένων του αποστολέα τίθεται συνεχώς στην τιμή η οποία ικανοποιεί περισσότερο την ομάδα των παραληπτών με τις τρέχουσες δικτυακές συνθήκες.

3.2.2.2 Συμπεράσματα

Σε αυτήν την ενότητα, παρουσιάσαμε ένα μηχανισμό για την multicast μετάδοση δεδομένων με τη χρήση μιας multicast ροής δεδομένων σε μια ετερογενή ομάδα παραληπτών. Παρουσιάστηκε ο σχεδιασμός ενός μηχανισμού για την παρακολούθηση των δικτυακών συνθηκών και την εκτίμηση του κατάλληλου ρυθμού μετάδοσης δεδομένων για τη μετάδοση πολυμεσικής πληροφορίας προκειμένου να εξυπηρετείται δίκαια μια ομάδα από παραλήπτες της πολυμεσικής πληροφορίας. Σημαντικό

χαρακτηριστικό του μηχανισμού ο οποίος αναλύθηκε, είναι το γεγονός όχι χρησιμοποιεί μια συνάρτηση δικαιοσύνης η οποία μετρά την ικανοποίηση κάθε χρήστη. Με τη χρήση αυτής της συνάρτησης αντιμετωπίζεται το βασικό μειονέκτημα της multicast μετάδοσης με την χρήση μιας multicast ροής δεδομένων. Ο προτεινόμενος μηχανισμός χρησιμοποιεί τα πρωτόκολλα RTP / RTCP για τη μετάδοση της πολυμεσικής πληροφορίας.

3.2.3 Multicast μετάδοση πολυμεσικών δεδομένων με χρήση της simulcast τεχνικής

Σε αυτήν την ενότητα παρουσιάζεται ένας μηχανισμός για τη multicast μετάδοση πολυμέσων με τη χρήση πολλαπλών multicast ροών δεδομένων (τεχνική simulcast) [14].

Ο μηχανισμός ο οποίος προτείνεται σε αυτό το κεφάλαιο βασίζεται στη multicast μετάδοση πολυμέσων με τη χρήση του RTP / RTCP και όπως αναφέραμε χρησιμοποιεί την τεχνική simulcast. Οι βασικοί στόχοι τους οποίους καλείται να πετύχει ο προτεινόμενος μηχανισμός είναι κάθε παραλήπτης να μπορεί να λαμβάνει τη βέλτιστη ποιότητα πολυμέσων που είναι ικανός να δεχτεί και η δημιουργούμενη multicast ροή δεδομένων να μην αποτελεί περιορισμό για τις υπόλοιπες ροές οι οποίες μεταδίδονται στο δίκτυο.

Για να επιτύχουμε τον πρώτο στόχο ο προτεινόμενος μηχανισμός δημιουργεί *n* διαφορετικές multicast ροές δεδομένων (στις περισσότερες περιπτώσεις ένας μικρός αριθμός διαφορετικών ροών δεδομένων είναι αρκετός - συνήθως τρεις ή τέσσερις ροές), κάθε μία με τα δικά της όρια στο εύρος ζώνης που μπορεί να καταναλώσει. Όλες οι ροές δεδομένων μεταφέρουν την ίδια πολυμεσική πληροφορία, κάθε μία όμως κωδικοποιημένη σε διαφορετική ποιότητα. Οι παραλήπτες λαμβάνουν την κατάλληλη ροή δεδομένων ανάλογα με την κατάσταση του μονοπατιού που τους ενώνει με την πηγή και τις επεξεργαστικές δυνατότητες τους. Αν ένας παραλήπτης διαπιστώσει ότι η ροή δεδομένων την οποία λαμβάνει δεν είναι κατάλληλη πλέον, ο υλοποιημένος μηχανισμός δίνει στον παραλήπτη τη δυνατότητα να μετακινηθεί σε άλλη ροή δεδομένων.

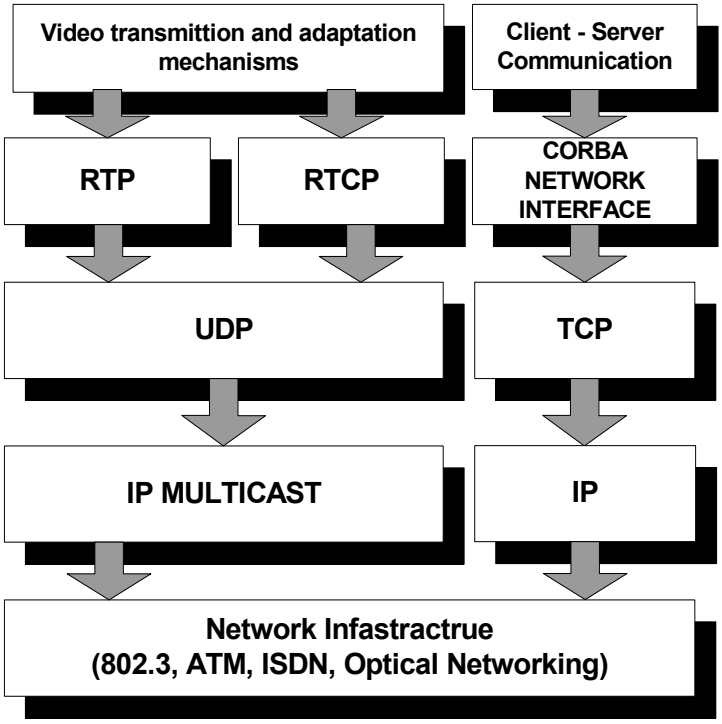
Για να επιτύχουμε το δεύτερο στόχο, χρησιμοποιούμε ένα αλγόριθμο προσθετικής αύξησης και πολλαπλασιαστικής μείωσης (Additive Increase Multiplicative Decrease - AIMD) για την προσαρμογή του ρυθμού μετάδοσης κάθε ροής δεδομένων. Ο

αλγόριθμος προσαρμογής μεταβάλλει το ρυθμό μετάδοσης κάθε ροής δεδομένων λαμβάνοντας υπόψη τον αριθμό των παραληπτών που βρίσκονται σε κατάσταση συμφόρησης (congested) ή μη φόρτου (unloaded). Επιπρόσθετα, αν οι δυνατότητες ενός παραλήπτη δεν συμβαδίζουν με τη ροή δεδομένων που λαμβάνει, μετακινείται σε άλλη ροή δεδομένων με χαμηλότερο ή υψηλότερο εύρος ζώνης.

3.2.3.1 Περιγραφή του προτεινόμενου μηχανισμού

Η στοίβα πρωτοκόλλων του προτεινόμενου μηχανισμού

Ο μηχανισμός οποίος προτείνεται βασίζεται στην multicast μετάδοση πολυμέσων. Η multicast μετάδοση δεδομένων πάνω από IP δίκτυα είναι ουσιαστικά η μοναδική επιλογή για τη μετάδοση πολυμεσικών δεδομένων ταυτόχρονα σε μεγάλο αριθμό παραληπτών πάνω από το Διαδίκτυο. Στο επίπεδο συνόδου, σύμφωνα με το μοντέλο αναφοράς OSI, γίνεται χρήση των RTP / RTCP πρωτοκόλλων. Τα πρωτόκολλα RTP / RTCP χρησιμοποιούνται τόσο από τις H.323 εφαρμογές όσο και από το MBONE, και θεωρείται ευρέως ως το de facto πρότυπο για τη μετάδοση πολυμέσων πάνω από το Διαδίκτυο. Τα πρωτόκολλα RTP / RTCP χρησιμοποιούνται επίσης λόγω των δυνατοτήτων τους για την παροχή πληροφοριών ανάδρασης με την χρήση των RTCP αναφορών. Ταυτόχρονα με την RTP / RTCP επικοινωνία εγκαθίσταται και μία unicast σύνδεση μεταξύ αποστολέα και παραλήπτη. Η σύνδεση αυτή παρέχει την απαραίτητη πληροφορία στον παραλήπτη σχετικά με την multicast IP διεύθυνση και τη θύρα (port) που χρησιμοποιείται, έτσι ώστε να μπορεί ο παραλήπτης να λάβει μέρος στη multicast μετάδοση των πολυμέσων. Όλη η επικοινωνία ελέγχου ανάμεσα στον αποστολέα και τους παραλήπτες γίνεται με τη χρήση της CORBA (Common Object Request Broker Architecture) τεχνολογίας. Εξαιτίας των διαφορών στην ποιότητα των πολυμέσων που μπορούν να διαχειριστούν οι διάφοροι παραλήπτες, ο αποστολέας μεταδίδει έναν μικρό αριθμό διαφορετικών multicast ροών δεδομένων, κάθε μία με τα δικά της μέγιστα και ελάχιστα όρια στο εύρος ζώνης, χωρίς επικάλυψη στη ζώνη του εύρους ζώνης που κάθε ροή δεδομένων καλύπτει. Οι ξεχωριστές αυτές ροές δεδομένων είναι, κατά μία έννοια, τα δοχεία (buckets) στα οποία οι παραλήπτες θα διαμοιραστούν, ανάλογα με τις δυνατότητές τους. Παρακάτω φαίνεται η στοίβα πρωτοκόλλων του μηχανισμού.



Σχήμα 3.7 – Στοίβα πρωτοκόλλων

Ο ρυθμός μετάδοσης σε κάθε ροή δεδομένων προσαρμόζεται μέσα στα όρια της ροής δεδομένων ανάλογα με τις δυνατότητες και την κατάσταση των παραληπτών που λαμβάνουν αυτή την ροή δεδομένων. Η στοίβα των πρωτοκόλλων που χρησιμοποιείται από τον προτεινόμενο μηχανισμό παρουσιάζεται στο παραπάνω σχήμα.

Οι βασικοί στόχοι του μηχανισμού ο οποίος προτείνεται είναι:

- Ένας μηχανισμός αρκετά ευέλικτος και γρήγορος, ο οποίος θα μπορεί να αντιδρά γρήγορα στις μεταβολές της κατάστασης του δικτύου.
- Ο μηχανισμός να προσαρμόζει το ρυθμό μετάδοσης με τέτοιο τρόπο ώστε να διατηρεί φιλική συμπεριφορά απέναντι στις άλλες TCP και UDP ροές δεδομένων που χρησιμοποιούν το δίκτυο.
- Να πετυχαίνει τη μέγιστη δυνατή δικαιοσύνη για κάθε παραλήπτη βάσει της ιδέας ότι δικαιοσύνη για έναν παραλήπτη σημαίνει να του δίνεται ελαφρά χειρότερη ποιότητα πολυμέσων από αυτή που θα λάμβανε εάν ήταν ο μόνος παραλήπτης της πολυμεσικής πληροφορίας.

Η λειτουργία του αποστολέα

Ο αποστολέας είναι μοναδικός και υπεύθυνος για:

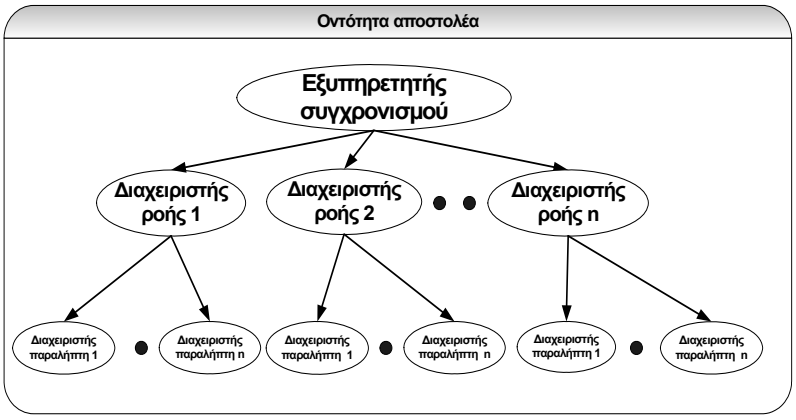
- Τη δημιουργία n διαφορετικών multicast ροών δεδομένων.

▪ Τον καθορισμό των ορίων στο εύρος ζώνης που θα χρησιμοποιεί η κάθε μία ροή δεδομένων.

▪ Την αναζήτηση τυχών παραληπτών που δεν τυγχάνουν δίκαιης μεταχείρισης.

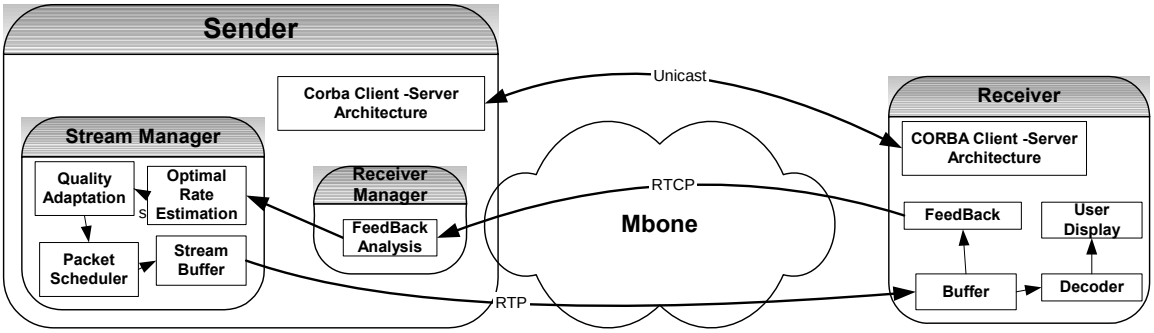
▪ Την παροχή των μηχανισμών ώστε οι παραλήπτες να μπορούν να αλλάξουν τη ροή δεδομένων από την οποία λαμβάνουν δεδομένα όταν θεωρούν ότι θα έπρεπε να βρίσκονται σε μία ροή δεδομένων πλησιέστερη στις δυνατότητές τους.

Πριν την έναρξη της μετάδοσης ο αποστολέας αλληλεπιδρά με το διαχειριστή του συστήματος μέσω ενός γραφικού περιβάλλοντος και ο χρήστης θέτει τις επιθυμητές τιμές στις βασικές παραμέτρους του μηχανισμού. Οι παράμετροι αυτοί περιλαμβάνουν τον αριθμό των ροών δεδομένων που θα δημιουργηθούν, τα όρια στο εύρος ζώνης της κάθε μιας, τις multicast IP διευθύνσεις, τις θύρες δεδομένων και ελέγχου, και τα πολυμέσα τα οποία θα μεταδοθούν.



Σχήμα 3.8 – Αρχιτεκτονική αποστολέα

Στο σχήμα αυτό, φαίνεται η οργάνωση και η αρχιτεκτονική του αποστολέα. Ο αποστολέας δημιουργεί *n* διαφορετικούς διαχειριστές ροών δεδομένων (stream managers). Σε κάθε διαχειριστή ροής δεδομένων ανατίθεται ένας μη καθορισμένος αριθμός διαχειριστών παραληπτών (receiver managers). Κάθε διαχειριστής παραλήπτη αντιπροσωπεύει ένα μοναδικό παραλήπτη που λαμβάνει τη ροή δεδομένων που ελέγχει ο συγκεκριμένος διαχειριστής ροής δεδομένων. Στο παρακάτω σχήμα φαίνεται η λειτουργία του προτεινόμενου μηχανισμού.



Σχήμα 3.9 – Λειτουργία του προτεινόμενου μηχανισμού

Αρχείο πολυμέσων

Οι σκληροί δίσκοι όπου αποθηκεύονται τα αρχεία με τα πολυμεσικά δεδομένων. Η εφαρμογή η οποία υλοποιήθηκε υποστηρίζει τρεις κωδικοποιήσεις βίντεο, συγκεκριμένα H.263, MPEG και JPEG.

Εξυπηρετητής συγχρονισμού (synchronisation server)

Υπεύθυνος για τη διαχείριση, τον συγχρονισμό και την επικοινωνία μεταξύ διαχειριστών ροών δεδομένων. Όταν ένας διαχειριστής ροής δεδομένων θέλει να ξαναρχίσει τη μετάδοση των πολυμέσων που σταμάτησε είτε γιατί όλοι του οι παραλήπτες μετακινήθηκαν σε άλλες ροές δεδομένων είτε γιατί σταμάτησαν να λαμβάνουν πολυμέσα, πρέπει να γνωρίζει που βρίσκεται η μετάδοση των πολυμέσων στις υπόλοιπες ροές δεδομένων. Αυτή την πληροφορία τη ζητά από τον εξυπηρετητή συγχρονισμού.

Διαχειριστής ροής δεδομένων (stream manager)

Ένας διαχειριστής ροής δεδομένων δημιουργείται από την εφαρμογή του αποστολέα. Είναι υπεύθυνος για τη διαχείριση και την επίβλεψη μίας από τις *n* διαφορετικές multicast ροές δεδομένων που δημιουργούνται στο ξεκίνημα της εφαρμογής του αποστολέα. Περιλαμβάνει επίσης όλους τους εσωτερικούς στη ροή δεδομένων μηχανισμούς που προσαρμόζουν το ρυθμό μετάδοσης. Ένας διαχειριστής ροής δεδομένων χρησιμοποιεί την πληροφορία που παρέχουν οι διαχειριστές παραληπτών για να προσαρμόσει το ρυθμό μετάδοσης της ροής δεδομένων την οποία διαχειρίζεται και να επιτύχει τα βέλτιστα συνολικά αποτελέσματα. Ένας διαχειριστής ροής

δεδομένων μεταδίδει δεδομένα μόνο όταν ένας τουλάχιστον παραλήπτης λαμβάνει την multicast ροή δεδομένων την οποία διαχειρίζεται. Ο διαχειριστής ροής δεδομένων είναι υπεύθυνος για τα παρακάτω:

Εκτίμηση βέλτιστου ρυθμού μετάδοσης (optimal rate estimation)

Ο μηχανισμός ανάλυσης των πληροφοριών ανάδρασης (που περιγράφεται παρακάτω με λεπτομέρεια) εκτελείται σε κάθε οντότητα παραλήπτη και επεξεργάζεται τις RTCP αναφορές από τον παραλήπτη και κατατάσσει τον παραλήπτη σε μία από τις τρεις παρακάτω κατηγορίες: κατάσταση συμφόρησης (congestion), κατάσταση φόρτου (load) ή κατάσταση μη φόρτου (unload). Για την εκτίμηση του βέλτιστου ρυθμού μετάδοσης, ο διαχειριστής ροής περιοδικά συγκεντρώνει τις καταστάσεις που του αναφέρουν όλοι οι διαχειριστές παραληπτών που ανήκουν στον συγκεκριμένο διαχειριστή ροής. Αυτό γίνεται στο τέλος μιας συγκεκριμένης, καθορισμένης χρονικής περιόδου, την οποία ονομάζουμε epoch. Κατόπιν χρησιμοποιείται ένας αλγόριθμος, που περιγράφεται σε ακόλουθη παράγραφο, ο οποίος προσπαθεί να βελτιώσει την δικαιοσύνη μεταξύ των παραληπτών με το να καθορίσει αν ένας χαμηλότερος ή ένας υψηλότερος ρυθμός μετάδοσης είναι καταλληλότερος. Όποτε ένας παραλήπτης δεν μπορεί να ικανοποιηθεί από μία ροή δεδομένων (εξαιτίας του ότι οι περισσότεροι από τους άλλους παραλήπτες της ίδιας ροής έχουν πολύ ψηλότερες ή πολύ χαμηλότερες δυνατότητες λήψης δεδομένων), ο διαχειριστής ροής πληροφορεί τον παραλήπτη ότι θα πρέπει να μετακινηθεί σε μία ροή δεδομένων χαμηλότερου ή υψηλότερου ρυθμού μετάδοσης δεδομένων.

Προσαρμογή ποιότητας (quality adaptation)

Ο διαχειριστής ροής δεδομένων είναι υπεύθυνος για την προσαρμογή της ποιότητας των πολυμέσων τα οποία μεταδίδονται. Η συμπεριφορά του (το εάν θα αυξήσει ή θα μειώσει τον ρυθμό μετάδοσης δεδομένων) καθορίζεται από την εκτίμηση του βέλτιστου ρυθμού μετάδοσης. Εξαιτίας του ότι ο νέος ρυθμός μετάδοσης εξαρτάται από την τρέχουσα τιμή του, η προσαρμογή της ποιότητας υπολογίζει την τρέχουσα τιμή του ρυθμού μετάδοσης διαιρώντας τον αριθμό των bytes που μεταδόθηκαν από το τέλος του προηγούμενου epoch μέχρι το τέλος του τρέχοντος epoch, με τη χρονική περίοδο που αντιπροσωπεύει ένα epoch. Η άλλη εναλλακτική λύση θα ήταν να μην γίνεται κανένας υπολογισμός και να υποθέτουμε ότι ο τρέχον ρυθμός μετάδοσης είναι

αυτός που ο διαχειριστής ροής δεδομένων προσπάθησε να θέσει στο τέλος του προηγούμενου epoch. Η δεύτερη προσέγγιση έχει όμως το σημαντικό μειονέκτημα ότι δεν υπάρχει κάποια εγγύηση πως ο επιδιωκόμενος ρυθμός πράγματι επετεύχθη από τον κωδικοποιητή και στάλθηκε στο δίκτυο. Στην πραγματικότητα η πράξη, κατά την διάρκεια των πειραμάτων που πραγματοποιήσαμε με την εφαρμογή την οποία υλοποιήσαμε, έδειξε ότι πολλές φορές ο πραγματικός ρυθμός μετάδοσης που επετεύχθη αποκλίνει σημαντικά από τον ρυθμό μετάδοσης που ο διαχειριστής ροής προσπάθησε να επιτύχει. Το εάν θα υπάρχει μεγάλη διαφορά μεταξύ του ρυθμού μετάδοσης που προτείνεται και του πραγματικού, εξαρτάται κυρίως από τον κωδικοποιητή και την κωδικοποίηση των πολυμέσων, καθώς και τις (επεξεργαστικές) δυνατότητες του υπολογιστή στο οποίο εκτελείται η εφαρμογή του αποστολέα.

Προγραμματισμός πακέτων / μνήμη προσωρινής αποθήκευσης (packet scheduler / data buffer)

Ο διαχειριστής ροής δεδομένων μεταδίδει όλη την εξερχόμενη πληροφορία στο δίκτυο. Η δουλειά του είναι να ενθυλακώνει τα δεδομένα σε RTP πακέτα. Υπάρχει επίσης μια μνήμη προσωρινής αποθήκευσης, η οποία χρησιμοποιείται για να εξομαλύνει τυχόν προβλήματα κατά τη μετάδοση των πολυμέσων στο δίκτυο.

Διαχειριστής παραλήπτη (receiver manager)

Αντιστοιχεί σε έναν μοναδικό παραλήπτη και επεξεργάζεται τις RTCP αναφορές που παράγει ο παραλήπτης και μπορεί να θεωρηθεί ως ο αντιπρόσωπος του παραλήπτη στην πλευρά του αποστολέα. Μπορεί να αλληλεπιδρά μόνο με έναν διαχειριστή ροής δεδομένων σε μία δεδομένη χρονική στιγμή, τον διαχειριστή ροής δεδομένων που ελέγχει τη ροή δεδομένων από την οποία ο παραλήπτης λαμβάνει την πολυμεσική πληροφορία. Είναι υπεύθυνος για τις παρακάτω λειτουργίες:

Ανάλυση των πληροφοριών ανάδρασης (feedback analysis)

Ο διαχειριστής παραλήπτη λαμβάνει τις RTCP αναφορές από τον παραλήπτη και τις επεξεργάζεται βασισμένος στην πληροφορία για το ρυθμό απώλειας πακέτων (packet loss rate) και την διακύμανση καθυστέρησης (delay jitter) και κάνει μία εκτίμηση της κατάστασης του παραλήπτη, βασισμένος στην τρέχουσα και σε προηγούμενες αναφορές που έχει μεταδώσει ο παραλήπτης. Η ακριβής λειτουργία του αλγορίθμου περιγράφεται σε παρακάτω παράγραφο.

Η λειτουργία του παραλήπτη

Η αρχιτεκτονική του παραλήπτη αποτελείται από τα ακόλουθα συστατικά (modules) :

Μνήμη προσωρινής αποθήκευσης (buffer)

Τα πολυμεσικά δεδομένα που λαμβάνονται αρχικά αποθηκεύονται στην μνήμη προσωρινής αποθήκευσης και η παρουσίαση τους δεν ξεκινά αν δεν υπάρχει αρκετή ποσότητα δεδομένων αποθηκευμένη στην μνήμη προσωρινής αποθήκευσης. Για να επιτευχθεί ομαλή παρουσίαση στον χρήστη, η χωρητικότητα της μνήμης προσωρινής αποθήκευσης πρέπει να ξεπερνά την μέγιστη διακύμανση καθυστέρησης (delay jitter) κατά τη διάρκεια της μετάδοσης.

Παροχή πληροφοριών ανάδρασης (feedback)

Ο παραλήπτης παρέχει τις απαραίτητες πληροφορίες για την ανάλυση των πληροφοριών ανάδρασης στον αποστολέα έτσι ώστε να μπορέσει αυτός να εκτιμήσει την κατάσταση του παραλήπτη. Οι πληροφορίες αυτές μεταδίδονται με RTCP αναφορές οι οποίες περιλαμβάνουν πληροφορίες για το ρυθμό απώλειας των πακέτων (packet loss rate) και την διακύμανση καθυστέρησης (delay jitter).

Αποκωδικοποιητής (decoder)

Ο παραλήπτης λαμβάνει τα πακέτα δεδομένων από τον την μνήμη προσωρινής αποθήκευσης ως είσοδο, τα αποκωδικοποιεί και τα εξάγει κατάλληλα για παρουσίαση. Η ποιότητα των πολυμέσων επηρεάζεται από το ρυθμό απώλειας των πακέτων (packet loss rate) και τη διακύμανση της καθυστέρησης (delay jitter). Η παρουσίαση ίσως χρειαστεί και να σταματήσει εάν τα δεδομένα στη μνήμη προσωρινής αποθήκευσης πέσουν κάτω από το απαιτούμενο ελάχιστο όριο.

Απεικόνιση στον χρήστη (user display)

Συνήθως τα πολυμεσικά δεδομένα παρουσιάζονται στο χρήστη στην οθόνη και τα ηχεία ενός υπολογιστή.

Εκτίμηση του ρυθμού μετάδοσης δεδομένων για κάθε multicast ροή δεδομένων

Ο αποστολέας αρχικά μεταδίδει έναν αριθμό από multicast ροές δεδομένων. Ο αριθμός αυτός εξαρτάται από τον αριθμό των παραληπτών που αναμένεται να λάβουν τα πολυμεσικά δεδομένα, από τον αποστολέα και από τις επεξεργαστικές δυνατότητες του υπολογιστή στο οποίο εκτελείται η εφαρμογή του αποστολέα.

Κάθε διαχειριστής ροής δεδομένων πρέπει να κάνει τη δική του επεξεργασία στα πολυμεσικά δεδομένα έτσι ώστε να τα μεταδίδει στον καθορισμένο ρυθμό μετάδοσης, και έτσι μεγάλος αριθμός ροών δεδομένων σημαίνει μεγάλο επεξεργαστικό φόρτο για τον αποστολέα. Μία λύση σε αυτό το πρόβλημα, εκτός από το να χρησιμοποιείται μικρός αριθμός ροών δεδομένων, θα μπορούσε να είναι η αποθήκευση των πολυμέσων σε διάφορους (ή όλους) τους απαραίτητους τύπους κωδικοποίησης έτσι ώστε κάθε ροή δεδομένων να χρησιμοποιεί το δικό της, τοπικά αποθηκευμένο, αρχείο. Αυτό φυσικά έχει το μειονέκτημα της αυξημένης ανάγκης για χώρο στο δίσκο. Εφόσον ένας μικρός αριθμός ροών δεδομένων θεωρείται ικανοποιητικός, η υλοποίηση εκτελεί την κωδικοποίηση των πολυμέσων κατά την μετάδοση τους («on the fly»). Μία ροή δεδομένων την οποία δεν λαμβάνει κανένας παραλήπτης είναι έτοιμη για μετάδοση αλλά παραμένει ανενεργή.

Όταν ένας παραλήπτης θελήσει να αρχίσει να λαμβάνει τα πολυμεσικά δεδομένα, ζητά από το αποστολέα τις multicast διευθύνσεις των ροών δεδομένων που μεταδίδονται. Στην εφαρμογή η οποία υλοποιήθηκε όλη η επικοινωνία ανάμεσα στον αποστολέα και τους παραλήπτες γίνεται με τη χρήση της CORBA (Common Object Request Broker Architecture) τεχνολογίας. Εφόσον ο παραλήπτης μπορεί να έχει κάποια τοπική γνώση της ποιότητας της σύνδεσης του, μπορεί να κάνει μια αρχική εκτίμηση της ροής δεδομένων στην οποία ο παραλήπτης ίσως ταιριάζει καλύτερα και του επιτρέπεται να εισέλθει σε οποιαδήποτε ροή δεδομένων ο παραλήπτης επιλέξει. Εάν δεν γίνει κάποια επιλογή, τότε ο παραλήπτης συμμετέχει στην ροή δεδομένων χαμηλότερης ποιότητας. Μία υπερεκτίμηση αναμένεται να έχει μικρές αρνητικές επιπτώσεις, γιατί ο παραλήπτης θα μετακινηθεί σύντομα σε πιο κατάλληλη ροή δεδομένων.

Όταν ένας παραλήπτης θέλει να αρχίσει να λαμβάνει μια multicast ροή δεδομένων ενημερώνει τον αποστολέα να αρχίσει τη μετάδοσή της, αν η συγκεκριμένη ροή δεδομένων δε μεταδίδεται ήδη. Ένας αποκλειστικός διαχειριστής παραλήπτη δημιουργείται για να αντιπροσωπεύει τον παραλήπτη στη μεριά του αποστολέα. Οι RTCP αναφορές του παραλήπτη μεταδίδονται στον κατάλληλο διαχειριστή παραλήπτη. Για μεγαλύτερη σαφήνεια, γίνεται μια διάκριση ανάμεσα σε δύο καταστάσεις, που μπορούν και οι δύο να πάρουν τις τιμές συμφόρηση (congestion), φόρτος (load) ή μη φόρτος (unload): ονομάζουμε το πρώτο είδος «μη επεξεργασμένη κατάσταση» (unprocessed state) και το δεύτερο είδος «επεξεργασμένη κατάσταση» (processed state). Η μη επεξεργασμένη κατάσταση προκύπτει άμεσα από τις φιλτραρισμένες τιμές

του ρυθμού απώλειας των πακέτων και της διακύμανσης καθυστέρησης σύμφωνα με τους ακόλουθους κανόνες:

$$if\,(LR_{new} \geq LR_c) \rightarrow \text{unprocessed state} = \text{congestion}$$

$$if\,(LR_u < LR_{new} < LR_c) \rightarrow \text{unprocessed state} = \text{load}$$

$$if\,(LR_{new} \leq LR_u) \rightarrow \text{unprocessed state} = \text{unload}$$

$$if\,(J_{new} > \gamma * J_{old}) \rightarrow u\text{nprocessed state} = \text{congestion}$$

Η κατάσταση η οποία θα χρησιμοποιηθεί για την εκτίμηση του βέλτιστου ρυθμού μετάδοσης από το διαχειριστή ροής δεδομένων ονομάζεται επεξεργασμένη κατάσταση. Υπολογίζεται λαμβάνοντας υπόψη τις τελευταίες n μη επεξεργασμένες καταστάσεις, οι οποίες κρατούνται σε μια μνήμη προσωρινής αποθήκευσης μεγέθους n στον διαχειριστή παραλήπτη. Αυτός ο μηχανισμός συνεισφέρει στην συντηρητική συμπεριφορά κατά την εκτίμηση του βέλτιστου ρυθμού μετάδοσης. Μία μη επεξεργασμένη κατάσταση που είναι σε κατάσταση συμφόρησης δε σημαίνει αναγκαστικά ότι και η επεξεργασμένη κατάσταση θα είναι κατάσταση συμφόρησης, ειδικά εάν η πλειοψηφία των προηγούμενων μη επεξεργασμένων καταστάσεων ήταν καταστάσεις μη φόρτου. Ο τρόπος με τον οποίο υπολογίζεται η επεξεργασμένη κατάσταση παρουσιάζεται παρακάτω:

Αρχικά εισάγεται μία νέα μεταβλητή, η μεταβλητή μη επεξεργασμένης κατάστασης (Unprocessed State Variable - USV), η οποία παίρνει μία νέα τιμή για κάθε μη επεξεργασμένη κατάσταση όπως φαίνεται παρακάτω:

$$\text{if (unprocessed state}_i == \text{congestion) then USV}_i = -1$$

$$\text{if (unprocessed state}_i == \text{load) then USV}_i = 0$$

$$\text{if (unprocessed state}_i == \text{unload) then USV}_i = 1$$

Η επεξεργασμένη κατάσταση καθορίζεται σύμφωνα με την παρακάτω εξίσωση:

$$f(i) = \text{state}_i * w_i + \text{state}_{i-1} * w_{i-1} + \ldots + \text{state}_{i-n+2} * w_{i-n+2} + \text{state}_{i-n+1} * w_{i-n+1}$$

όπου $w_i, \ldots, w_{i-n+1}$ είναι βάρη. Στη συνέχεια η επεξεργασμένη κατάσταση υπολογίζεται ως εξής:

$$\text{if } (f(i) < 0) \text{ then processed state}_i = \text{congestion}$$
$$\text{if } (f(i) == 0) \text{ then processed state}_i = \text{load}$$
$$\text{if } (f(i) > 0) \text{ then processed state}_i = \text{unload}$$

Η ανανέωση των πληροφοριών στους διαχειριστές παραληπτών γίνεται ασύγχρονα, κάθε φορά που λαμβάνουν μία RTCP αναφορά. Οι διαχειριστές ροών δεδομένων ανανεώνουν τους ρυθμούς μετάδοσης των ροών τους σύγχρονα και για αυτό ο χρόνος στη λειτουργία του συστήματος χωρίζεται σε καθορισμένα χρονικά διαστήματα τα οποία ονομάζουμε epochs. Στο τέλος ενός epoch, κάθε διαχειριστής ροής δεδομένων συγκεντρώνει τις καταστάσεις όλων των διαχειριστών παραληπτών που αντιστοιχούν σε έναν παραλήπτη που λαμβάνει αυτήν τη ροή δεδομένων και προσαρμόζει την ποιότητα των πολυμέσων καθορίζοντας την βελτίωση ή τη υποβάθμιση της ποιότητας των πολυμέσων σε αυτήν τη ροή δεδομένων. Εάν θα υπάρξει βελτίωση ή υποβάθμιση καθορίζεται ως εξής: Εάν όλοι οι παραλήπτες είναι σε κατάσταση μη φόρτου, η ποιότητα των πολυμέσων βελτιώνεται. Εάν περισσότεροι από ένα συγκεκριμένο όριο παραληπτών είναι σε κατάσταση συμφόρησης, η ποιότητα των πολυμέσων υποβαθμίζεται.

Ο νέος ρυθμός μετάδοσης υπολογίζεται χρησιμοποιώντας έναν αλγόριθμο προσθετικής αύξησης και πολλαπλασιαστικής μείωσης (Additive Increase Multiplicative Decrease - AIMD) αλγόριθμο παρόμοιο με τον αλγόριθμο που χρησιμοποιεί το TCP. Η αύξηση επιτυγχάνεται προσθέτοντας ένα μικρό ποσό στον προηγούμενο ρυθμό μετάδοσης, και είναι επομένως αρκετά συντηρητική στην κατανάλωση εύρους ζώνης, ενώ η μείωση επιτυγχάνεται πολλαπλασιάζοντας τον προηγούμενο ρυθμό μετάδοσης με έναν αριθμό στην περιοχή 0…1 (συνήθως γύρω στο 0.75) και έτσι ο αλγόριθμος είναι περισσότερο επιθετικός όταν προσπαθεί να αντιδράσει σε συμφόρηση.

Καθορισμός των αλλαγών ροών δεδομένων από τους παραλήπτες

Υπάρχουν τρεις περιπτώσεις που θα προκαλέσουν τη μετάβαση ενός παραλήπτη προς μία άλλη ροή δεδομένων:

1. Εάν η ροή δεδομένων την οποία ο παραλήπτης λαμβάνει έχει ήδη φτάσει τον ελάχιστο της ρυθμό μετάδοσης και ο παραλήπτης βρίσκεται ακόμα σε κατάσταση συμφόρησης τότε ο παραλήπτης σταματά να λαμβάνει αυτήν τη ροή δεδομένων και προσχωρεί στην σύνοδο μίας ροής χαμηλότερης ποιότητας (αν μια τέτοια ροή δεδομένων υπάρχει).

2. Εάν η ροή δεδομένων την οποία ο παραλήπτης λαμβάνει έχει ήδη φτάσει τον μέγιστο της ρυθμό μετάδοσης και ο παραλήπτης βρίσκεται ακόμα σε κατάσταση μη φόρτου τότε ο παραλήπτης σταματά να λαμβάνει αυτήν τη ροή δεδομένων και προσχωρεί στην σύνοδο μίας ροής υψηλότερης ποιότητας (αν μια τέτοια ροή δεδομένων υπάρχει).

3. Η τρίτη περίπτωση εμφανίζεται όταν ένας παραλήπτης συνυπάρχει σε μία ροή δεδομένων με παραλήπτες χαμηλών δυνατοτήτων αλλά ο ίδιος είναι ικανός να δεχτεί πολυμέσα υψηλότερης ποιότητας, και δεν έχει καταφέρει να βελτιώσει την ποιότητα των πολυμέσων αυτής της ροής εξαιτίας των άλλων παραληπτών. Ο χρησιμοποιούμενος μηχανισμός στοχεύει στο να κάνει το πρωτόκολλο πιο συντηρητικό και μετράει τις συνεχείς φορές που ένας παραλήπτης ήταν σε κατάσταση μη φόρτου αλλά απέτυχε να βελτιώσει την ποιότητα των πολυμέσων. Όταν ο αριθμός των φορών αυτών ξεπεράσει ένα ορισμένο όριο, υποθέτουμε ότι ο παραλήπτης έχει όντως μεγαλύτερες δυνατότητες και τον μετακινούμε σε ροή δεδομένων καλύτερης ποιότητας. Η μετάβαση από τη μία ροή δεδομένων στην άλλη σημαίνει επίσης ότι το διαχειριστής παραλήπτη που αντιστοιχεί σε αυτόν τον παραλήπτη θα αλληλεπιδρά τώρα με το νέο διαχειριστή ροής δεδομένων.

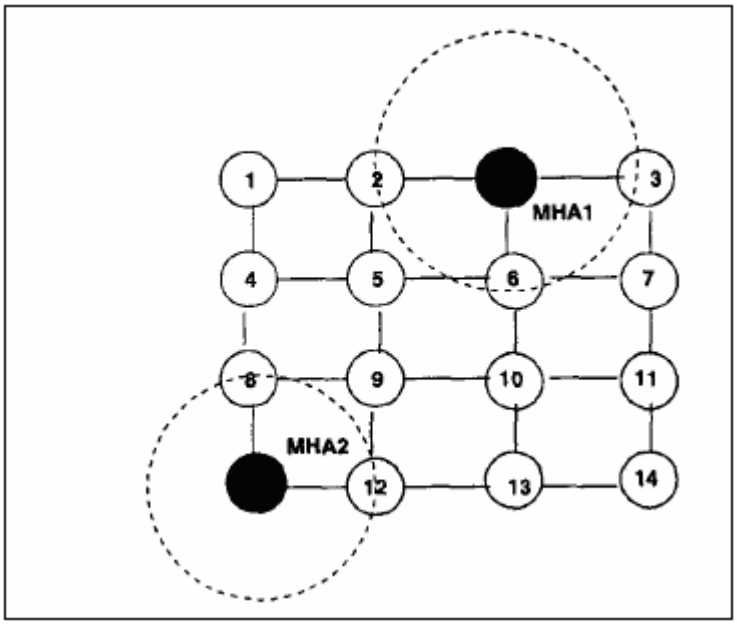
Για να συμβεί μία μετάβαση ενός παραλήπτη, ένας επιπρόσθετος κανόνας επιβάλλεται σε όλες τις περιπτώσεις: Ο παραλήπτης πρέπει να έχει στείλει έναν ελάχιστο αριθμό αναφορών από τη στιγμή που άρχισε να λαμβάνει τη ροή δεδομένων από την οποία θέλει να αποχωρήσει. Ο κανόνας αυτός προσπαθεί να αποφύγει τις πολύ συχνές μεταβάσεις που προκαλούν άχρηστο επεξεργαστικό φόρτο και σπαταλούν εύρος ζώνης. Ο παραλήπτης αναγκάζεται να μείνει σε μία ροή δεδομένων για τουλάχιστον μία συγκεκριμένη χρονική περίοδο (το χρόνο που χρειάζεται για να στείλει τον ελάχιστο αριθμό RTCP αναφορών) και έτσι εάν παρόλα αυτά αποφασίσει να αλλάξει ροή δεδομένων μετά το πέρας αυτής της χρονικής περιόδου, είμαστε σίγουροι ότι αυτή είναι μία δικαιολογημένη απόφαση.

ΚΕΦΑΛΑΙΟ 4

Προτεινόμενες λύσεις που ικανοποιούν τις απαιτήσεις μας και τελική πρόταση λύσης

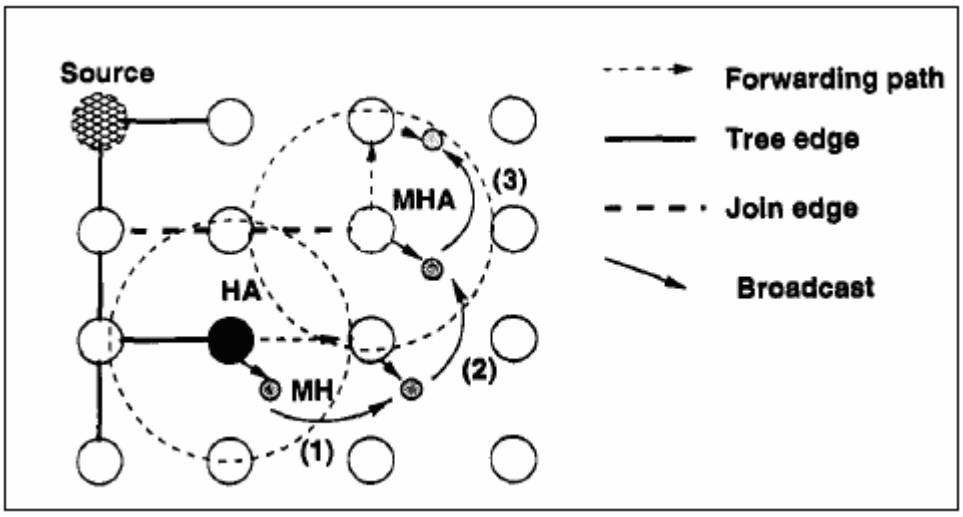
4.1 RBMoM

Υβριδικό πρωτόκολλο των remote subscription και bi-directional tunnelling [7]. Σύμφωνα με το εύρος υπηρεσιών του RBMoM πρωτοκόλλου, η ανταλλαγή μεταξύ του μήκους του tunnelling μονοπατιού και της συχνότητας του multicast tree reconstruction έχει να δείξει τα καλύτερα αποτελέσματα όταν multicast δεδομένα παραδίδονται στο κοντινότερο tunnelling μονοπάτι από MHA σε FA χωρίς να καταβάλλεται το υψηλό κόστος του multicast reconstruction tree. Το RBMoM πρωτόκολλο αποφασίζει την τιμή του service range σύμφωνα με τα handoff rates των κινητών hosts και τον αριθμό των μελών του multicast group. Οι δύο αυτές παράμετροι όμως δεν μπορούν να αποτελούν κριτήριο για καθορισμό μιας βέλτιστης τιμής του πεδίου υπηρεσιών καθώς σε πραγματικές συνθήκες αλλάζουν δυναμικά. Για την επίλυση αυτού του προβλήματος προτείνεται ένας αλγόριθμος μείωσης του κόστους του multicast tree reconstruction και παράδοσης multicast data στο κοντινότερο μονοπάτι tunnelling από MHA σε FA. Προτείνεται, για το σκοπό αυτό, μια ειδική παράμετρος συστήματος η οποία ονομάζεται tunnelling service range. Το tunnelling πεδίο υπηρεσίας ικανοποιεί το όριο του end-to-end delay και το delay variation, και παίρνει το μέγιστο tunnelling πεδίο υπηρεσίας από το MHA. Το κόστος του multicast tree reconstruction μειώνεται κατά ένα σημαντικό βαθμό. Οι κινητοί hosts αποτελούν ληφθέντα δεδομένα multicast από το παρόν MHA μέχρι να τεθούν εκτός του πεδίου υπηρεσιών. Αν το service range έχει μεγάλη τιμή, τότε το μήκος του μονοπατιού tunnelling από το MHA γίνεται μακρύ. Για να μειωθεί το μήκος αυτό, όταν ένας κινητός host κινείται σε ξένο δίκτυο, τα multicast δεδομένα παραδίδονται από το κοντινότερο MHA ψάχνοντας τον MHA πίνακα του FA.



Σχήμα 4.1 – Τοπολογία για το RBMoM

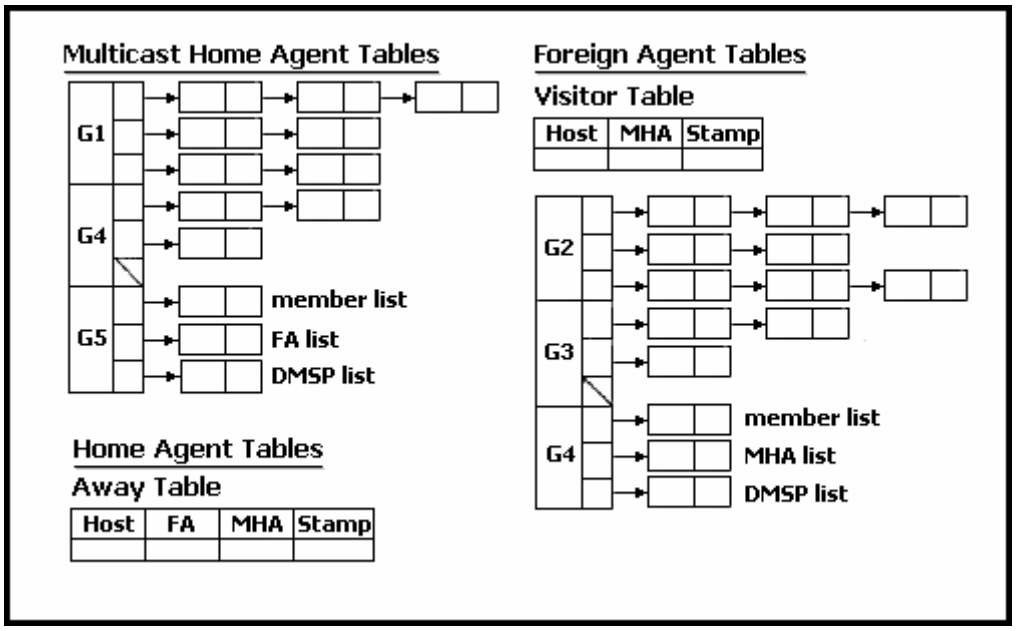
Στο παραπάνω σχήμα βλέπουμε ένα παράδειγμα τοπολογίας στο οποίο υπάρχουν 14 χρήστες κινητοί οι οποίοι ελέγχονται από τον αντίστοιχο Ατζέντη τους (MHA). Παρατηρούμε ότι ο κάθε MHA έχει ένα συγκεκριμένο πεδίο στο οποίο ελέγχει τους χρήστες που εμπίπτουν σε αυτόν.



Σχήμα 4.2 – Μετακίνηση του κινητού χρήστη

Στο παραπάνω σχήμα βλέπουμε τη διαδικασία στην οποία ένας κινητός χρήστης ενώ αρχικά βρίσκεται σε ένα σημείο όπου υπεύθυνος είναι ο MHA, στη συνέχεια

μετακινείται σε ένα άλλο σημείο το οποίο είναι εκτός του πεδίου ελέγχου του αρχικού του MHA, και αυτό που πρέπει να γίνει, είναι ο MHA' που είναι ο υπεύθυνος του πεδίου στο οποίο εισέρχεται ο κινητός χρήστης, να έχει τις πληροφορίες του νέου χρήστη έτσι ώστε να καταλάβει ότι ένα νέο μέλος έχει ενταχθεί στην ακτίνα την οποία ελέγχει.



Σχήμα 4.3 – Δομές δεδομένων για κάθε χρήστη

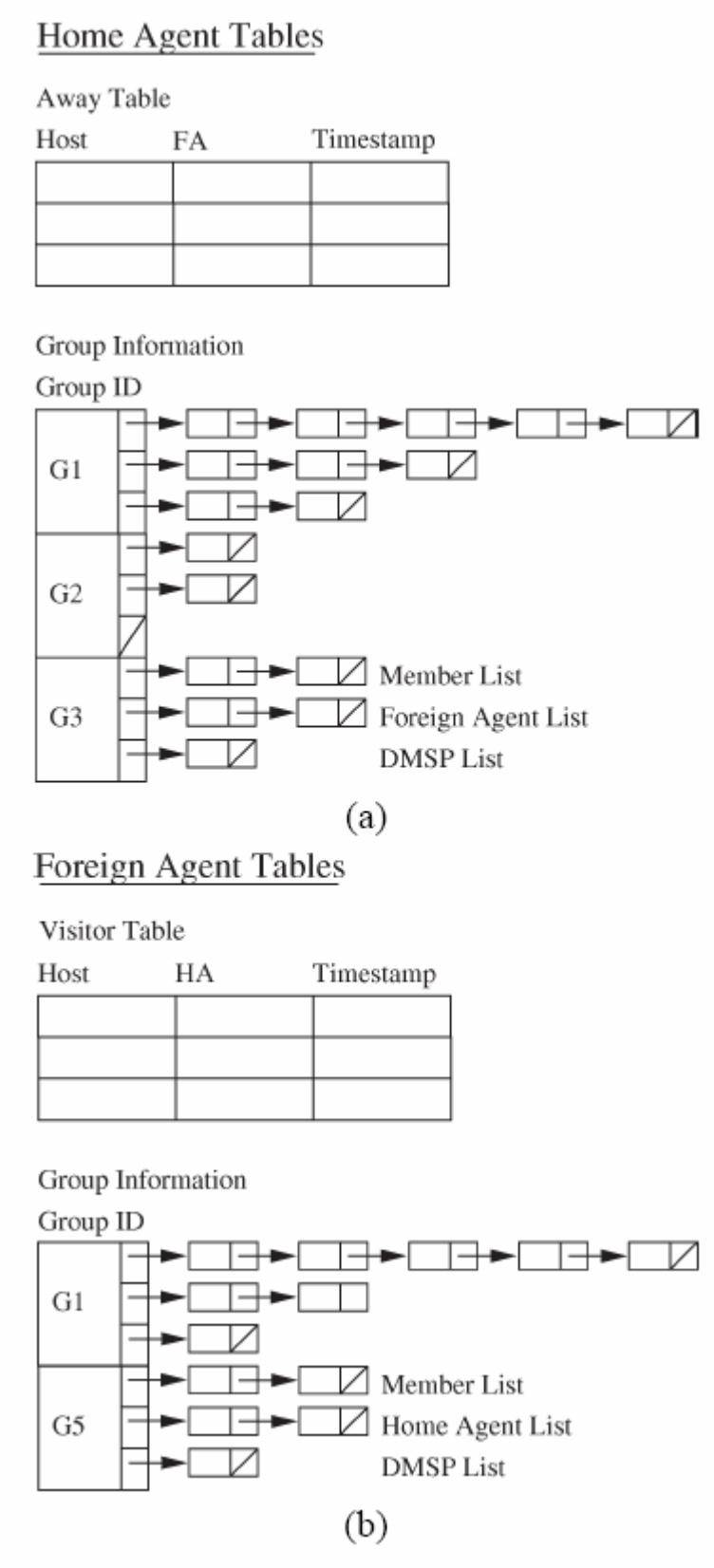
Στο τελευταίο σχήμα βλέπουμε τη διαδικασία λειτουργίας του πρωτοκόλλου RBMoM. Αρχικά, ο κινητός χρήστης είναι στο δικό του δίκτυο και ο προσωρινός Multicast home agent είναι ο HA. Μετά το βήμα 1 όπως φαίνεται παραπάνω, ο κινητός χρήστης μετακινείται σε ένα ξένο δίκτυο. Το ξένο δίκτυο εμπίπτει στο πεδίο υπηρεσίας του MH. Υποθέτουμε ότι ο ο ξένος ατζέντης (FA) συναντά τον HA ως τον DMSP. Στο παρόν στάδιο, ο κινητός χρήστης παραλαμβάνει το Multicast datagram από τον multicast home agent του. Ομως, στο βήμα 2, ο κινητός χρήστης εισέρχεται σε άλλο δίκτυο το οποίο είναι εκτός του πεδίου υπηρεσίας του HA του. Έτσι, ο τρέχοντας FA γίνεται ο multicast home agent και εισέρχεται στο Multicast group. Ο προηγούμενος multicast home agent πρέπει να φύγει από το συγκεκριμένο multicast group σε περίπτωση που όλοι οι κινητοί του χρήστες έχουν βγει εκτός του πεδίου υπηρεσίας του. Άρα από δω και πέρα, ο κινητός χρήστης θα παραλαμβάνει τα multicast datagrams μέσω του MHA. Στο βήμα 3, ο κινητός χρήστης εισέρχεται σε ένα άλλο δίκτυο στο οποίο όμως εξακολουθεί να εξυπηρετείται από τον τρέχοντα MHA.

4.2 MoM

Ένας άλλος μηχανισμός ο οποίος μοιάζει με τον RBMoM, είναι το πρωτόκολλο MoM (Mobile Multicast), ο οποίος μπορεί να χρησιμοποιηθεί για μεταφορά multicast δεδομένων σε κινητούς χρήστες [6]. Παρακάτω ακολουθεί η περιγραφή του πρωτοκόλλου αυτού.

Το πρωτόκολλο MoM βασίζεται στους Home Agents (HAs) για να μεταφέρουν τα multicast πακέτα στους κινητούς χρήστες μέσω του Mobile IP tunnel και του ξένου ατζέντη (FA). Αφού ο HA μπορεί να εξυπηρετεί κινητούς χρήστες σε πολλούς FAs που επιθυμούν να παραλάβουν datagrams που προορίζονται για ένα group G, ο HA προωθεί ένα αντίγραφο σε κάθε αντίστοιχο Mobile IP tunnel. Στο σημείο αυτό, τονίζουμε ότι ο HA δε χρειάζεται να προωθήσει ένα ξεχωριστό αντίγραφο για κάθε κινητό χρήστη που εξυπηρετεί, αλλά ένα μόνο αντίγραφο για κάθε ξένο δίκτυο στο οποίο βρίσκονται οι κινητοί χρήστες. Link-level multicast χρησιμοποιείται από τον FA σε κάθε ξένο δίκτυο έτσι ώστε να ολοκληρώσει τη μεταφορά των πακέτων. Επίσης, ο FA επιλέγει έναν Home Agent ως τον **D**esignated **M**ulticast **S**ervice **P**rovider (DMSP) για κάποιο multicast group.

4.2.1 Δομές Δεδομένων του πρωτοκόλλου



Σχήμα 4.4 – Δομές δεδομένων για το MoM

Το παραπάνω σχήμα παρουσιάζει τις δομές δεδομένων που χρειάζονται για την περιγραφή του πρωτοκόλλου MoM. Ο κάθε HA πρέπει να διατηρεί μία away λίστα έτσι ώστε να κρατεί πληροφορίες για το ποιοι κινητοί χρήστες είναι μακρυά, σε ποιον FA ανήκουν και πότε θα λήξουν οι δεσμοί τους. Παρόμοια, ο κάθε FA διατηρεί μια visitor λίστα, έτσι ώστε να κρατεί πληροφορίες για το ποιοι κινητοί χρήστες βρίσκονται την τρέχουσα στιγμή στο LAN του, από ποιον HA ήρθε ο κάθε κινητός χρήστης, και πότε θα λήξουν οι δεσμοί τους. Επιπρόσθετα, το πρωτόκολλο αυτό απαιτεί group membership πληροφορίες για τους απομακρυσμένους και τους επισκέπτες κινητούς χρήστες. Αυτές οι Group πληροφορίες μπορούν να βρίσκονται στους Home και Foreign Agents. Στην εφαρμογή αυτή, βρίσκονται στους Home και Foreign Agents. Κατά συνέπεια, ο κάθε Home Agent διατηρεί για το κάθε multicast group που γνωρίζει:

1. Μία λίστα με τους απομακρυσμένους κινητούς χρήστες που ανήκουν στο Group του.
2. Μία λίστα με τους FAs στους οποίους βρίσκονται οι απομακρυσμένοι κινητοί χρήστες του group.
3. Μία λίστα με τους FAs στους οποίους ο HA έχει DSMP ευθύνες.

Παρόμοια, κάθε Foreign Agent διατηρεί για κάθε group τα ακόλουθα:

1. Μία λίστα με τους επισκέπτες κινητούς χρήστες που είναι μέλη του multicast group.
2. Μία λίστα με τους HAs στους οποίους ανήκουν αυτοί οι επισκέπτες κινητοί χρήστες του group.
3. Μία λίστα με τους HAs που εξυπηρετούν ως DMSPs για το συγκεκριμένο group.

Λεπτομέρειες του πρωτοκόλλου

Τα γεγονότα που πρέπει να χειρίζεται το πρωτόκολλο MoM, είναι οι αφίξεις των χρηστών, οι αναχωρήσεις των χρηστών, τα timeouts, οι μετακινήσεις των DMSP και οι αφίξεις των multicast μηνυμάτων.

Τονίζεται ότι όταν ένας κινητός χρήστης φεύγει από το home δίκτυό του με σκοπό να πάει σε ένα ξένο δίκτυο, δεν υπάρχει ειδοποίηση για το γεγονός αυτό. Όταν ένας κινητός χρήστης φτάσει σε ένα ξένο δίκτυο πρέπει να εκτελεστεί ο παρακάτω αλγόριθμος.

Όταν ένας κινητός χρήστης φτάσει σε ένα ξένο δίκτυο:

1. Εγγράφεται στον Foreign Agent (FA).
   1.1 Δημιουργεί ο FA είσοδο στον πίνακα επισκεπτών για τον κινητό χρήστη.

1.2 Εισάγει ο FA το όνομα του χρήστη, τις πληροφορίες του HA του και ενεργοποιεί το χρονόμετρο.

1.3 Ειδοποιεί τα υπόλοιπα μέλη στο group που ανήκει ο κινητός χρήστης.

Για κάθε multicast group στο οποίο έχει εισέλθει ο κινητός χρήστης:

1.3.1 Δημιουργείται είσοδος στον group information πίνακα αν χρειάζεται.

1.3.2 Προστίθεται ο κινητός χρήστης στη λίστα των μελών του group G.

1.3.3 Αν αυτός είναι ο πρώτος κινητός χρήστης από αυτόν τον HA σε αυτόν τον FA, τότε προστίθεται ο HA του κινητού χρήστη στη λίστα πληροφοριών για το group G, αλλιώς αυξάνεται ο μετρητής για τον HA του κινητού χρήστη.

1.3.4 Επιλέγεται ένας DMSP για αυτό το group από τη λίστα των Home Agents.

1.3.5 Αν ο επιλεχθής DMSP διαφέρει από τον προηγούμενο DMSP, τότε εκτελείται η αλλαγή του DMSP.

2. Εγγράφεται με τον HA.

2.1 Δημιουργείται ή ενημερώνεται η είσοδος στον away πίνακα για τον κινητό χρήστη.

2.2 Κρατείται ο προηγούμενος FA αν υπάρχει.

2.3 Εισάγεται το όνομα του χρήστη, οι πληροφορίες του FA, και ενεργοποιείται το χρονόμετρο.

2.4 Ειδοποιείται ο Home Agent των κινητών χρηστών που ανήκουν στο group.

Για κάθε multicast group στο οποίο έχει εισέλθει ο κινητός χρήστης:

2.4.1 Δημιουργείται είσοδος στον group information πίνακα αν χρειάζεται.

2.4.2 Προστίθεται ο κινητός χρήστης στη λίστα των μελών του group G αν χρειάζεται.

2.4.3 Αν αυτός είναι ο πρώτος κινητός χρήστης από αυτόν τον HA σε αυτόν τον FA, τότε προστίθεται ο FA του κινητού χρήστη στη λίστα πληροφοριών για το group G, αλλιώς αυξάνεται ο μετρητής του FA για τον κινητό χρήστη.

2.4.4 Αν ο νέος FA του κινητού χρήστη διαφέρει από τον προηγούμενο FA, τότε μειώνεται ο μετρητής του προηγούμενου FA και αφαιρείται ο προηγούμενος FA από τη λίστα, αν ο μετρητής πάρει την τιμή μηδεν.

2.4.5 Ενημερώνεται / κρατείται DMSP κατάσταση (NAI/OXI) για τον HA του group G στον FA (και του προηγούμενου FA αν χρειάζεται).

Όταν ένας κινητός χρήστης επιστρέφει στο home δίκτυό του:

1. Ενημερώνεται ο HA.

1.1 Διαγράφεται η είσοδος στον away πίνακα του MH, σημειώνοντας τον προηγούμενο FA.

1.2 Για κάθε multicast group G στο οποίο εισέρχεται ο κινητός χρήστης:

1.2.1 Διαγράφεται ο κινητός χρήστης από τη λίστα μελών του G.

1.2.2 Μειώνεται ο μετρητής του προηγούμενου FA του κινητού χρήστη, αφαιρώντας τον προηγούμενο FA από τη λίστα των FAs αν ο μετρητής πάρει την τιμή μηδέν, και διαγράφεται επίσης από τη λίστα DMSP, αν χρειάζεται.

Όταν λήγει ο χρόνος στον οποίο ο κινητός χρήστης βρίσκεται σε ξένο δίκτυο:

1. Διαγράφεται η είσοδος του κινητού χρήστη από τη λίστα επισκεπτών, σημειώνοντας τον HA.

2. Για κάθε multicast group G στο οποίο εισέρχεται ο κινητός χρήστης:

2.1 Διαγράφεται ο κινητός χρήστης από τη λίστα μελών του G.

2.2 Μειώνεται ο μετρητής του HA του κινητού χρήστη, και αφαιρείται ο HA από τη λίστα των HA αν ο μετρητής πάρει την τιμή μηδέν, και διαγράφεται ο HA από την DMSP λίστα, αν χρειάζεται.

2.3 Επιλέγεται ένας DMSP από τη λίστα των Home Agents για το συγκεκριμένο group.

2.4 Αν ο επιλεχθής DMSP διαφέρει από τον προηγούμενο DMSP, τότε εκτελείται η αλλαγή του DMSP.

Όταν ένα unicast πακέτο για τον κινητό χρήστη φτάσει στον HA του:

1. Κοίταξε τις πληροφορίες του FA στον away πίνακα για τον κινητό χρήστη.

2. Ενθυλάκωσε το πακέτο και στείλε το (tunnel) στον FA.

Όταν ένα multicast πακέτο για το group G φτάσει στον HA:

1. Προώθησε το multicast πακέτο στα τοπικά μέλη του group.

2. Κοίταξε τις πληροφορίες των μελών για τα απομακρυσμένα μέλη του συγκεκριμένου Group.

3. Ενθυλάκωσε το πακέτο και προώθησέ το σε κάθε FA για τον οποίο ο HA είναι ο DMSP για το group G.

Όταν ένα tunnelled πακέτο φτάσει στον FA από τον HA:

1. Decapsulate το πακέτο.
2. Αν το πακέτο είναι unicast για έναν κινητό χρήστη, τότε προωθείται μόνο στο χρήστη αυτον.
3. Αν το πακέτο είναι multicast πακέτο για το group G, τότε έλεγξε για τα τοπκά μέλη και προώθησέ το χρησιμοποιώντας link-level multicast αν βέβαια βρεθούν τοπικά μέλη.

Τα παραπάνω βήματα, ενημερώνουν τη λίστα επισκεπτών στον FA και τη λίστα απομακρυσμένων στον HA και ενημερώνουν τις πληροφορίες για τα μέλη του group σε κάθε ατζέντη, έτσι ώστε η κατάσταση του DMSP που καθιρίζεται από τον FA, να είναι γνωστή και στον HA. Η απόφαση για τον DMSP παίρνεται ανεξάρτητα από τον κάθε FA για κάθε group G. Όταν ένας κινητός χρήστης φεύγει από το ξένο δίκτυο δε χρειάζεται να γίνει κάποια ενημέρωση, διότι οι ενημερώσεις των καταστάσεων γίνονται όταν λήξει το χρονόμετρο.

4.3 Τελική πρόταση λύσης

4.3.1 SSM (Source Specific Multicast)

Το πρωτόκολλο αυτό έχει σχεδιαστεί για να υποστηρίζει multicast μετάδοση δεδομένων σε κινητά IPv6 δίκτυα. Έχοντας το συγκρίνει με τα πρωτόκολλα που αναφέρθηκαν στο προηγούμενο κεφάλαιο, το πρωτόκολλο αυτό είναι πιο απλό και η πολυπλοκότητά του είναι χαμηλής τάξεως [17]. Επομένως, το SSM είναι το πρωτόκολλο που θα χρησιμοποιηθεί στην εργασία αυτή και θα αξιολογηθεί μέσω προσομοίωσης στο περιβάλλον NS-2. Ακολουθεί η περιγραφή του πρωτοκόλλου αυτού και στη συνέχεια η σχηματική αναπαράσταση μεταφοράς δεδομένων βασισμένη στο πρωτόκολλο SSM.

Το πρωτόκολλο SSM θεωρείται απλό σε σύγκριση με τα υπόλοιπα πρωτόκολλα που υποστηρίζουν multicast σε IPv6 δίκτυα, διότι διαμορφώνει εύκολα τους αποστολείς, έχει σφαιρική εμβέλεια και δε χρειάζονται από πριν διακανονισμοί για τους παροχείς δικτύου (RP configuration). Η μεταφορά των datagrams βασίζεται σε λογικά κανάλια. Ένα (S,G) κανάλι περιέχει datagrams με την IP διεύθυνση του αποστολέα S και την multicast διεύθυνση της ομάδας G η οποία αποτελεί την IP διεύθυνση προορισμού. Οι χρήστες παραλαμβάνουν δεδομένα αν γίνουν μέλη του συγκεκριμένου καναλιού. Στην Ipv6 επικεφαλίδα, στο πεδίο διεύθυνσης του αποστολέα, αναγράφεται ο ιδιοκτήτης της multicast διεύθυνσης, και έχει σφαιρική εμβέλεια. Οι SSM εφαρμογές πρέπει να

γνωρίζουν ότι υπάρχει το κανάλι προτού γίνουν μέλη σε αυτό. Η ανακάλυψη του καναλιού είναι ευθύνη των εφαρμογών και μπορεί να γίνει μέσω ιστοσελίδων. Η ικανότητα καθορισμού της διεύθυνσης (S,G) ενός SSM καναλιού παρέχεται από το Multicast Listener Discovery Version 2 (MLDv2). Υποστηρίζει φιλτράρισμα του αποστολέα (source filtering) ή την ικανότητα ενός χρήστη να εκφράσει ενδιαφέρον στην παραλαβή πακέτων δεδομένων τα οποία αποστέλλονται μόνο από συγκεκριμένους αποστολείς. MLDv2 εφαρμόζεται μέσω ενός IPv6 δρομολογητή ο οποίος ανακαλύπτει την παρουσία των multicast listeners στις απευθείας ενωμένες συνδέσεις του. Επίσης, ρόλος του έιναι να ανακαλύπτει τις multicast διευθύνσεις που οι γειτονικοί κόμβοι ενδιαφέρονται. Το SSM αποτρέπει το πρόβλημα της σφαιρικής δέσμευσης των multicast διευθύνσεων και κάνει κάθε αποστολέα ανεξάρτητα υπεύθυνο για την επίλυση των διεθύνσεων που συγκρούονται στα διάφορα (S,G) κανάλια που δημιουργεί. Όταν ο παραλήπτης εγγραφεί σε ένα (S,G) κανάλι, παραλαμβάνει δεδομένα που στέλνονται μόνο από τον αποστολέα S, και όταν ο αποστολέας S επιλέγει ένα κανάλι (S,G) για να στείλει τα δεδομένα του, αυτόματα εξασφαλίζεται ότι κανένας άλλος αποστολέας δεν μπορεί να στείλει παράλληλα δεδομένα στο ίδιο κανάλι. Το SSM απαιτεί μόνο source-based forwarding trees. Άρα δε χρειάζεται η υποδομή του PM – SM για R – P shared trees ούτε το πρωτόκολλο MSDP. Η πολυπλοκότητα της υποδομής του SSM είναι χαμηλή, γι αυτό και το SSM είναι εφαρμόσιμο και πολύ απλό.

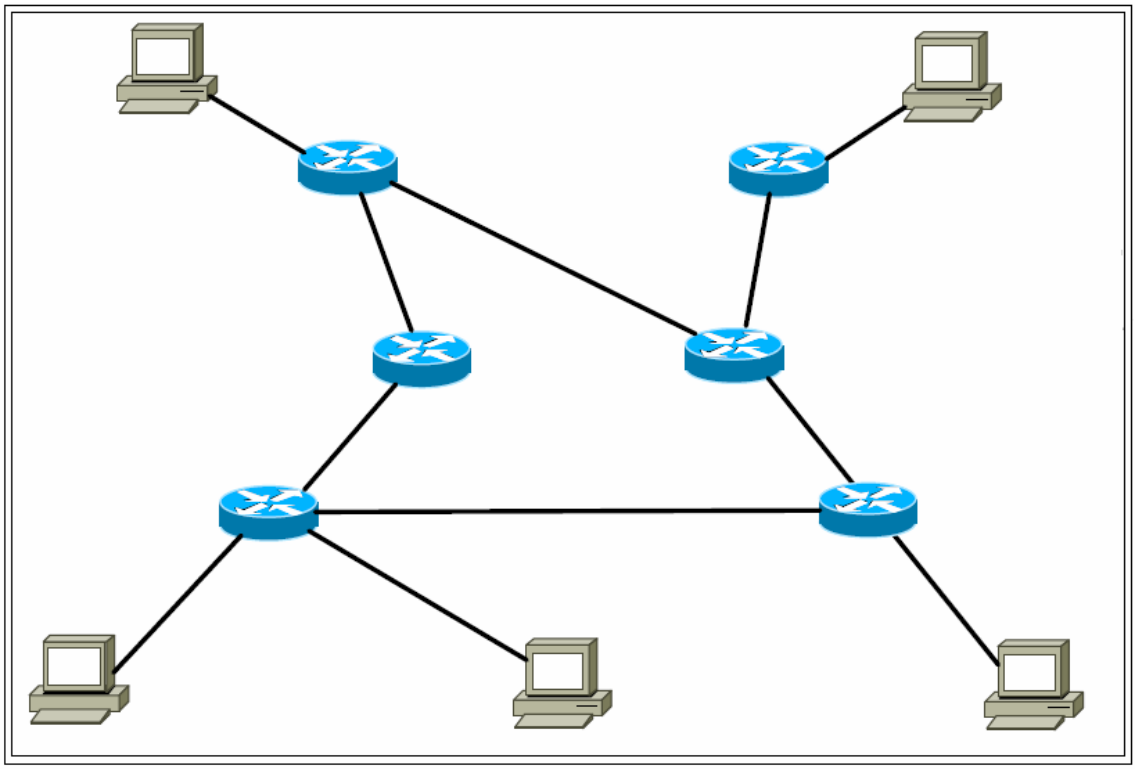
Η IPv6 διεύθυνση προορισμού έχει δεσμευθεί από την IANA και έχει τη μορφή FF3x::/96, όπως καθορίζεται στο REC3306. Μία νόμιμη διεύθυνση δημιουργείται από τα 96 prefix bits ενωμένα με τη μόνιμη ταυτότητα του group. Οι μόνιμες αναγνωρίσεις των ομάδων κυμαίνονται στο πεδίο FF3x::0000:0000 μέχρι FF3x::3fff:ffff. Οι διευθύνσεις FF3x::4000:0000 μέχρι FF3x::7fff:ffff έχουν δεσμευθεί για μελλοντικούς σκοπούς.

Το SSM είναι ένα πρωτόκολλο που αρμόζει στις απαιτήσεις της παρούσας εργασίας μιας και υποστηρίζει multicast μετάδοση τόσο σε κινητούς όσο και σε σταθερούς χρήστες. Όσον αφορά τη μετάδοση πολυμεσικού περιεχομένου και συγκεκριμένα απλού και ιεραρχικού Video, το πρωτόκολλο αυτό έχει τη δυνατότητα να υποστηρίξει μετάδοση απλού Video και το αξιολογήσαμε στο περιβάλλον NS-2 και αποδείχτηκε πολύ αποδοτικό. Παράλληλα, σχετικά με το πρωτόκολλο ADIVIS που υποστηρίζει unicast μετάδοση ιεραρχικού Video σε κινητούς χρήστες, ανάλογα με τη θεωρία που

γνωρίζουμε, θεωρούμε ότι με κάποιες μικρές τροποποιήσεις θα είναι εφικτό να υποστηρίξει και multicast μετάδοση.

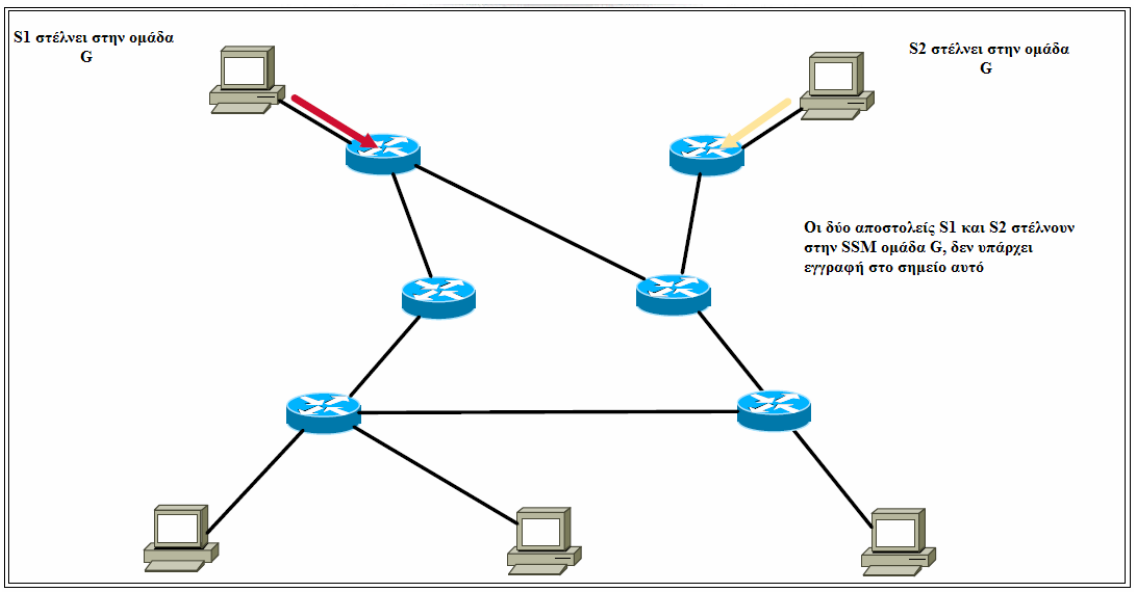
Σχηματική αναπαράσταση μεταφοράς δεδομένων στο πρωτόκολλο SSM

Βήμα 1
Στο βήμα αυτό βλέπουμε την τοπολογία η οποία αποτελείται από πέντε χρήστες οι οποίοι είναι συνδεδεμένοι άμεσα με τέσσερις δρομολογητές και αυτοί με τη σειρά τους ενώνονται με άλλους ενδιάμεσους δρομολογητές.



Σχήμα 4.5 – Αρχική τοπολογία SSM

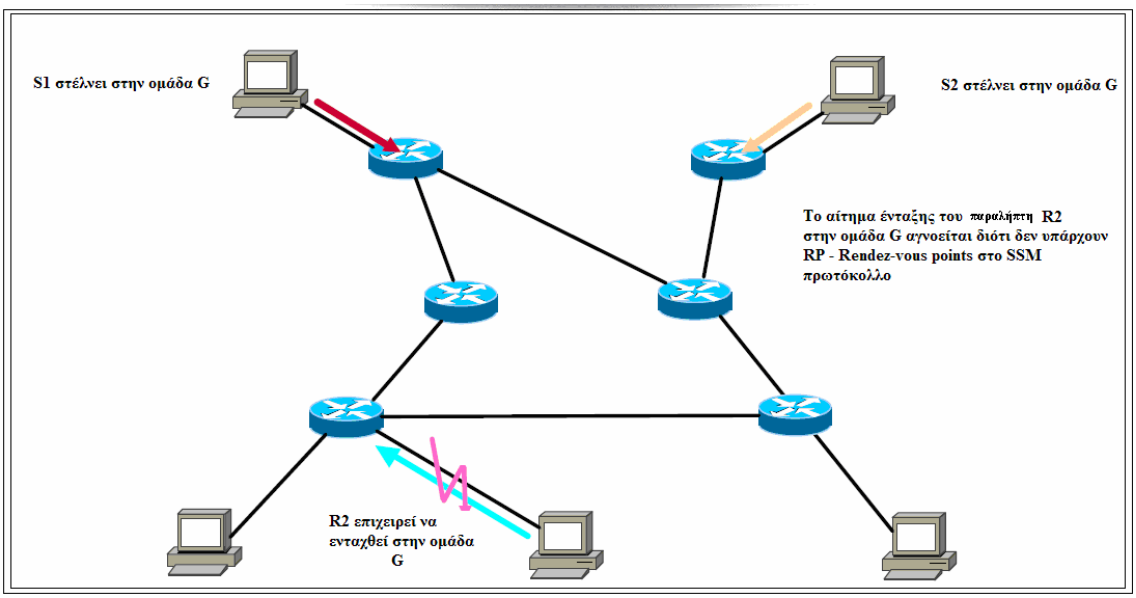
Βήμα 2
Στο βήμα αυτό καθορίζονται τα δύο ξεχωριστά κανάλια, (S1,G) και (S2,G). Το πρώτο κανάλι αποτελείται από τη unicast διεύθυνση του χρήστη S1 και τη multicast διεύθυνση της ομάδας G, ενώ το δεύτερο κανάλι αποτελείται από τη unicast διεύθυνση του χρήστη S2 και τη multicast διεύθυνση της ομάδας G. Και οι δύο αποστολείς στέλνουν δεδομένα στην ίδια ομάδα.



Σχήμα 4.6 – Καθορισμός καναλιών

Βήμα 3

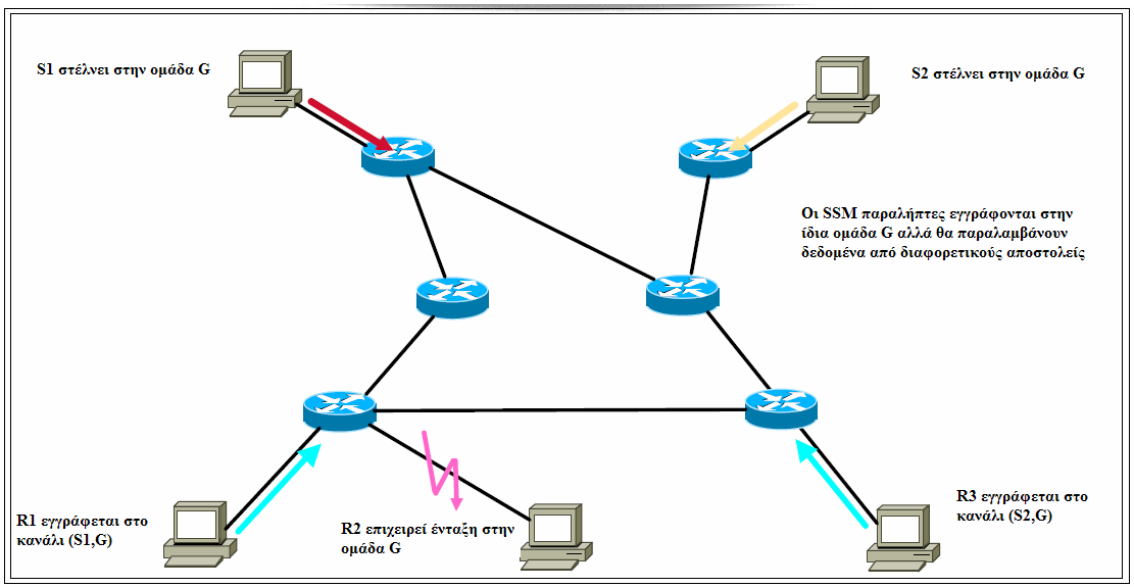
Στο τρίτο βήμα ο παραλήπτης R2 επιχειρεί να ενταχθεί σε ένα από τα δύο κανάλια, όμως καθορίζει μόνο τη multicast διεύθυνση και όχι παράλληλα και τη διεύθυνση ενός από τους δύο αποστολείς, και για αυτό το λόγο απορρίπτεται.



Σχήμα 4.7 – Απόρριψη παραλήπτη R2 να ενταχθεί σε κανάλι

Βήμα 4

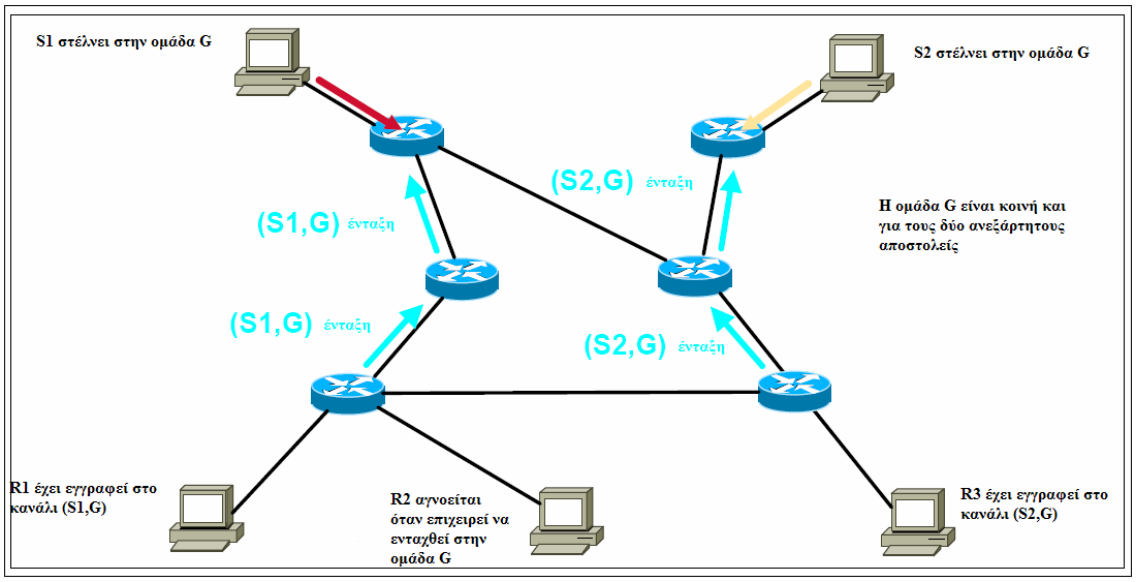
Στο τέταρτο βήμα οι παραλήπτες R1 και R3 επιτυχώς εγγράφονται στην ομάδα G όμως σε διαφορετικό αποστολέα ο καθένας.



Σχήμα 4.8 - Εγγραφή R1 και R3 στην ομάδα

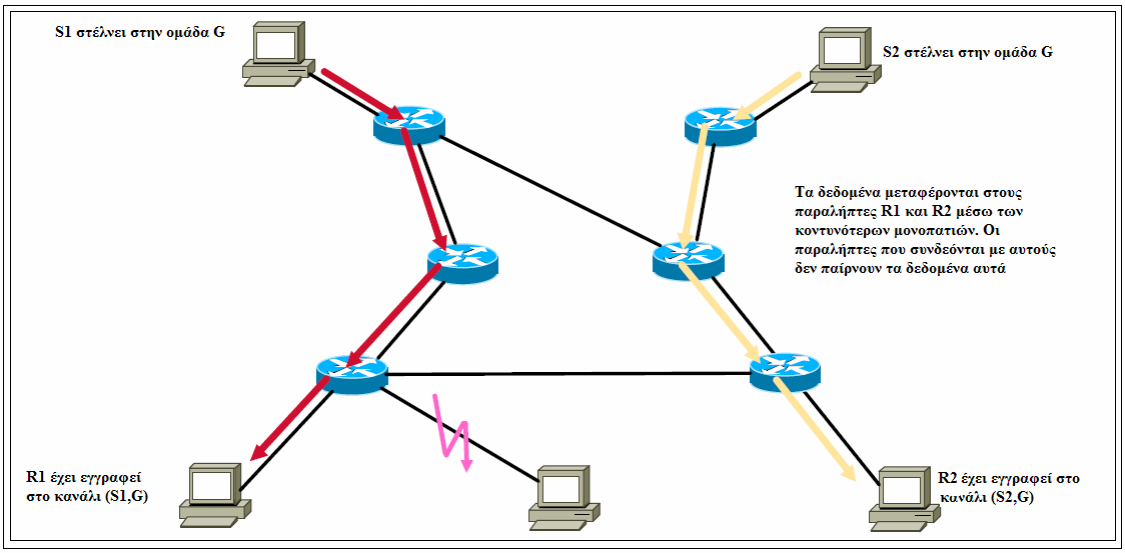
Βήμα 5

Οι παραλήπτες R1 και R3 εγγράφονται ο κάθε ένας σε διαφορετικό κανάλι, όμως η ομάδα στην οποία εγγράφονται είναι κοινή και για τους δύο.



Σχήμα 4.9 – Εγγραφή R1 και R3 στην ίδια ομάδα

Βήμα 6
Στο τελευταίο βήμα τα δεδομένα μεταφέρονται στους παραλήπτες R1 και R3 από το κανάλι (S1,G) και (S2,G) αντίστοιχα.



Σχήμα 4.10 – Μεταφορά δεδομένων

4.3.2 Προσομοίωση στο περιβάλλον NS-2

Χρησιμοποιώντας το περιβάλλον **NS-2** δημιουργήσαμε μία τοπολογία στην οποία τρέξαμε διάφορα σενάρια για να παρατηρήσουμε τη συμπεριφορά του δικτύου. Συγκεκριμένα, ελέγξαμε έξι σενάρια τα οποία περιγράφονται αναλυτικά πιο κάτω:

Σενάριο 1: Simple video μετάδοση με 2 multiple unicast users.
Σενάριο 2: Simple video μετάδοση με 10 multiple unicast users.
Σενάριο 3: ADIVIS (Scalable Video) μετάδοση με 2 multiple unicast users.
Σενάριο 4: ADIVIS (Scalable Video) μετάδοση με 10 multiple unicast users.
Σενάριο 5: Simple video μετάδοση με 2 multicast users (πρωτόκολλο *S*ource *S*pecific *M*ulticast).
Σενάριο 6: Simple video μετάδοση με 10 multicast users (πρωτόκολλο *S*ource *S*pecific *M*ulticast).

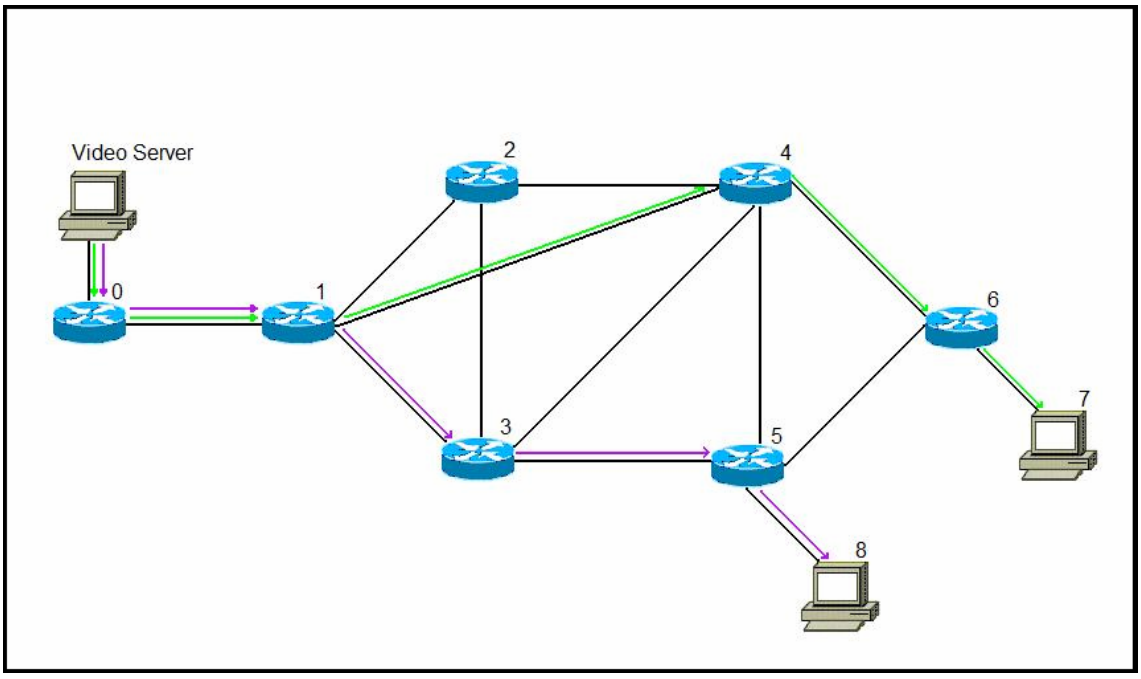
Και για τα έξι σενάρια που δημιουργήσαμε, πήραμε μετρήσεις όσον αφορά την καθυστέρηση ανάμεσα στα δύο άκρα (αποστολέα και παραλήπτη), το εύρος ζώνης, την

καθυστέρηση διακύμανσης και την απώλεια των πακέτων. Οι μετρήσεις αυτές παρουσιάζονται μέσω γραφικών παραστάσεων και αναλύονται παρακάτω για το κάθε σενάριο ξεχωριστά.

4.3.2.1 Σενάριο 1: Simple video transfer with 2 multiple unicast users

Για το σενάριο αυτό, δημιουργήσαμε μια τοπολογία η οποία αποτελείται από από τέσσερις κεντρικούς δρομολογητές, έναν εξυπηρετητή Video, και τέσσερις ακρινούς δρομολογητές. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία αυτή η οποία είναι και τοπολογία πάνω στην οποία θα τρέξουμε όλα τα σενάρια. Στο συγκεκριμένο σενάριο υπάρχουν δύο χρήστες οι οποίοι παίρνουν δεδομένα από τον εξυπηρετητή Video. Στο σενάριο αυτό το Video που μεταφέρεται στους χρήστες είναι απλό και δε χωρίζεται σε επίπεδα.

Τοπολογία



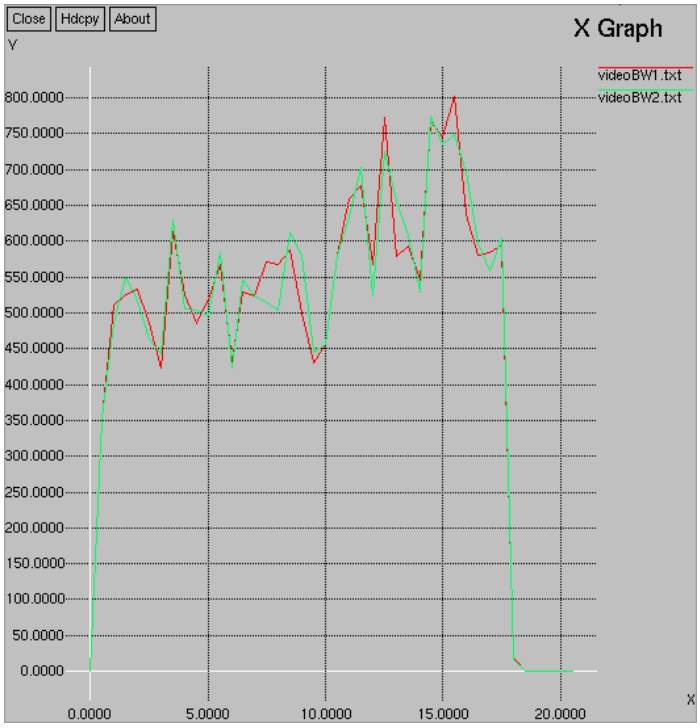
Σχήμα 4.11 – Τοπολογία για σενάριο 1

Αποτελέσματα

	Απώλεια Πακέτων (%)	**Καθυστέρηση δύο άκρων (δευτερόλεπτα)**	**Διακύμανση Καθυστέρησης (δευτερολεπτα)**	**Μέσος Όρος Εύρους Ζώνης (Kbps)**
Παραλήπτης 1	0	0.07	0.01	588
Παραλήπτης 2	0	0.09	0.02	587.5

Και για τους δύο παραλήπτες δεν υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής είναι αρκετό για να καλύψει και τις δύο ροές δεδομένων που θα φεύγουν από τον αποστολέα. Η καθυστέρηση δύο άκρων για τον πρώτο παραλήπτη κυμαίνεται στα 0.07 δευτερόλεπτα και για τον δεύτερο παραλήπτη κυμαίνεται στα 0.09 δευτερόλεπτα. Η διακύμανση καθυστέρησης στον πρώτο παραλήπτη είναι 0.01 δευτερόλεπτα και στον δεύτερο παραλήπτη είναι 0.02 δευτερόλεπτα. Ο μέσος όρος εύρους ζώνης για τον πρώτο παραλήπτη είναι 588 Kbps και για τον δεύτερο παραλήπτη είναι 587.5 Kbps που είναι σχεδόν ακριβώς το ίδιο οπότε γίνεται δίκαιη κατανομή του μεγέθους της σύνδεσης και στους δύο παραλήπτες.

Γραφική παράσταση – Εύρος Ζώνης



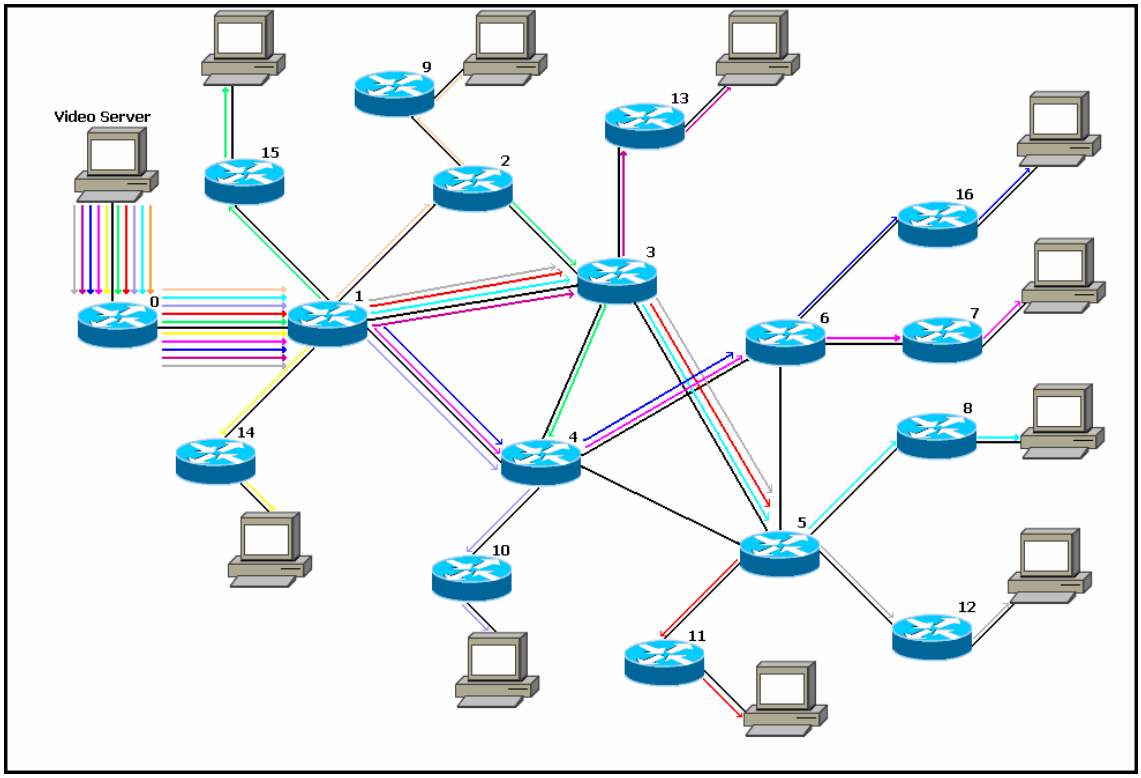
Γράφημα 4.1 – Εύρος ζώνης για τους 2 παραλήπτες

Στην παραπάνω γραφική παράσταση παρατηρούμε ότι η προσομοίωσή μας διαρκεί 21 δευτερόλεπτα. Το εύρος ζώνης κυμαίνεται περίπου στα ίδια επίπεδα τόσο για τον πρώτο παραλήπτη όσο και για τον δεύτερο παραλήπτη, και συγκεκριμένα για τον πρώτο παραλήπτη το εύρος ζώνης κυμαίνεται από 400 μέχρι 800 Kbps ενώ για τον δεύτερο παραλήπτη κυμαίνεται από 400 μέχρι 760 Kbps. Αυτό σημαίνει ότι γίνεται δίκαιη κατανομή του εύρους ζώνης και για τους δύο παραλήπτες.

4.3.2.2 Σενάριο 2: Simple video transfer με 10 multiple unicast users

Για το σενάριο αυτό, δημιουργήσαμε μια τοπολογία η οποία αποτελείται από από τέσσερις κεντρικούς δρομολογητές, έναν εξυπηρετητή Video, και δώδεκα ακρινούς δρομολογητές. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία αυτή η οποία είναι και τοπολογία πάνω στην οποία θα τρέξουμε όλα τα σενάρια με δέκα χρήστες. Στο συγκεκριμένο σενάριο υπάρχουν δέκα χρήστες οι οποίοι παίρνουν δεδομένα από τον εξυπηρετητή Video. Στο σενάριο αυτό το Video που μεταφέρεται στους χρήστες είναι απλό και δε χωρίζεται σε επίπεδα.

Τοπολογία



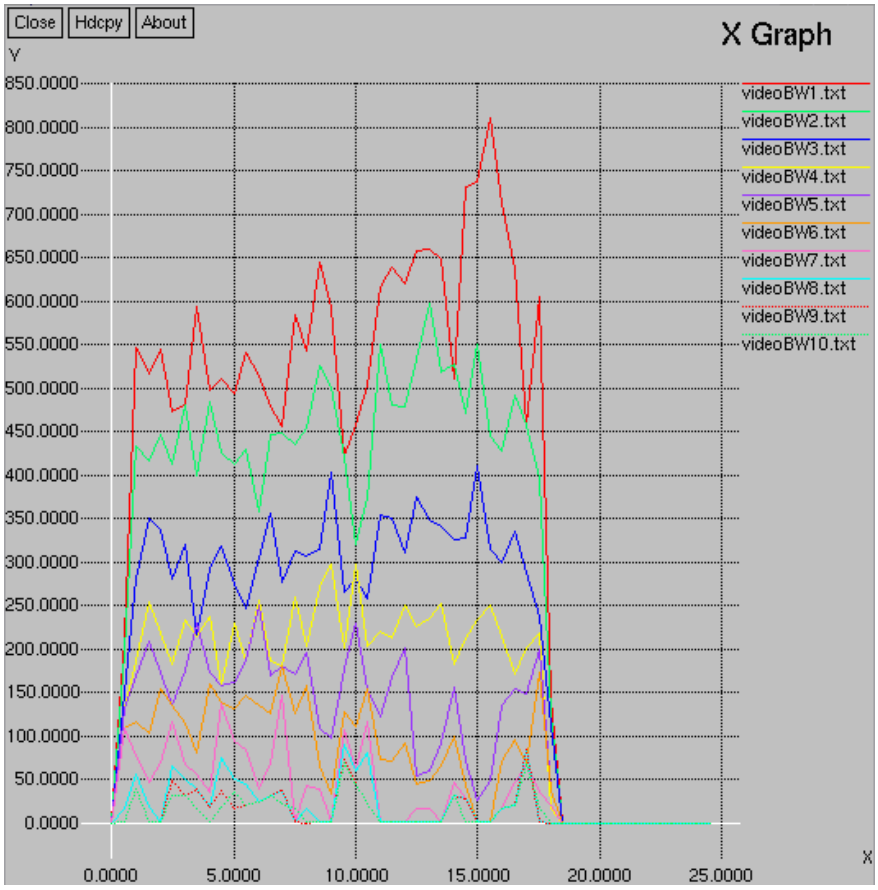
Σχήμα 4.12 – Τοπολογία για σενάριο 2

Αποτελέσματα

	Απώλεια Πακέτων (%)	Καθυστέρηση δύο άκρων (δευτερόλεπτα)	Διακύμανση Καθυστέρησης (δευτερόλεπτα)	Μέσος Όρος Εύρους Ζώνης (Kbps)
Παραλήπτης 1	0.3	0.17	0.02	586.4
Παραλήπτης 2	18.5	0.18	0.02	470.3
Παραλήπτης 3	41.3	0.18	0.02	322.9
Παραλήπτης 4	56.9	0.18	0.02	228.3
Παραλήπτης 5	67.9	0.19	0.02	158
Παραλήπτης 6	77.6	0.19	0.02	107.1
Παραλήπτης 7	87.1	0.19	0.02	54.2
Παραλήπτης 8	91.9	0.17	0.02	29
Παραλήπτης 9	93.7	0.15	0.02	20.4
Παραλήπτης 10	94.3	0.15	0.02	17.9

Και στους δέκα παραλήπτες υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής δεν είναι αρκετό για να καλύψει και τις δέκα ροές δεδομένων που θα φεύγουν από τον αποστολέα. Όσο αυξάνεται ο αριθμός των παραληπτών τόσο αυξάνεται και το ποσοστό απώλειας πακέτων. Η καθυστέρηση δύο άκρων και για τους δέκα παραλήπτες κυμαίνεται από 0.15 μέχρι 0.19 δευτερόλεπτα. Η διακύμανση καθυστέρησης είναι 0.02 δευτερόλεπτα και για τους δέκα παραλήπτες. Ο μέσος όρος εύρους ζώνης δεν είναι ο ίδος για όλους τους παραλήπτες οπότε δεν γίνεται δίκαιη κατανομή του μεγέθους της σύνδεσης σε όλους τους παραλήπτες.

Γραφική Παράσταση – Εύρος Ζώνης



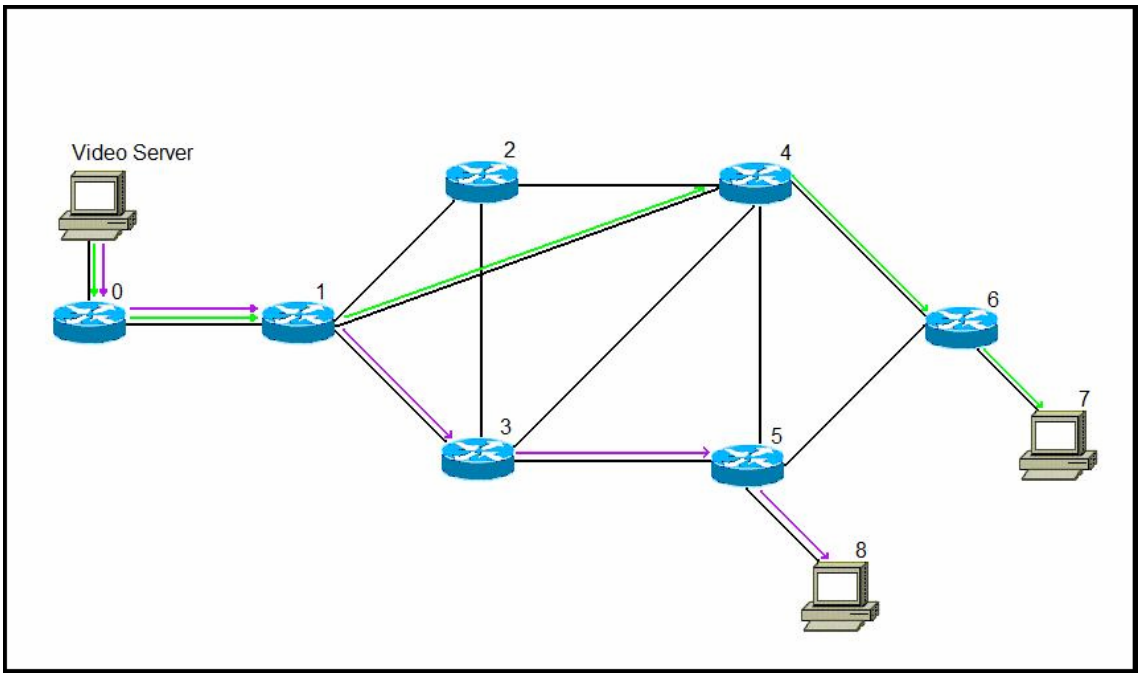
Γράφημα 4.2 – Εύρος ζώνης για τους 10 παραλήπτες

Στην παραπάνω γραφική παράσταση η προσομοίωσή μας διαρκεί 24 δευτερόλεπτα. Παρατηρούμε ότι σε γενικές γραμμές δεν γίνεται δίκαιη κατανομή του εύρους ζώνης σε όλους τους παραλήπτες. Συγκεκριμένα, ο πρώτος παραλήπτης έχει το μεγαλύτερο εύρος ζώνης μιας και φτάνει στα 800 Kbps. Για τους υπόλοιπους εννέα παραλήπτες, το εύρος ζώνης είναι σε χαμηλότερα επίπεδα και καθώς φτάνουμε στον τελευταίο παραλήπτη το εύρος ζώνης είναι ακόμη πιο χαμηλό, και συγκεκριμένα κάτω από 50 Kbps . Αυτό συμβαίνει διότι η γραμμή δεν είναι αρκετή για να εξυπηρετήσει δίκαια όλους τους παραλήπτες.

4.3.2.3 Σενάριο 3: ADIVIS (Scalable Video) with 2 multiple unicast users

Για το σενάριο αυτό [18], δημιουργήσαμε μια τοπολογία η οποία αποτελείται από από τέσσερις κεντρικούς δρομολογητές, έναν εξυπηρετητή Video, και τέσσερις ακρινούς δρομολογητές. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία αυτή. Στο συγκεκριμένο σενάριο υπάρχουν δύο χρήστες οι οποίοι ενώνονται και παραλαμβάνουν δεδομένα από τον εξυπηρετητή Video. Στο σενάριο αυτό το Video που μεταφέρεται στους χρήστες είναι χωρισμένο σε επίπεδα.

Τοπολογία



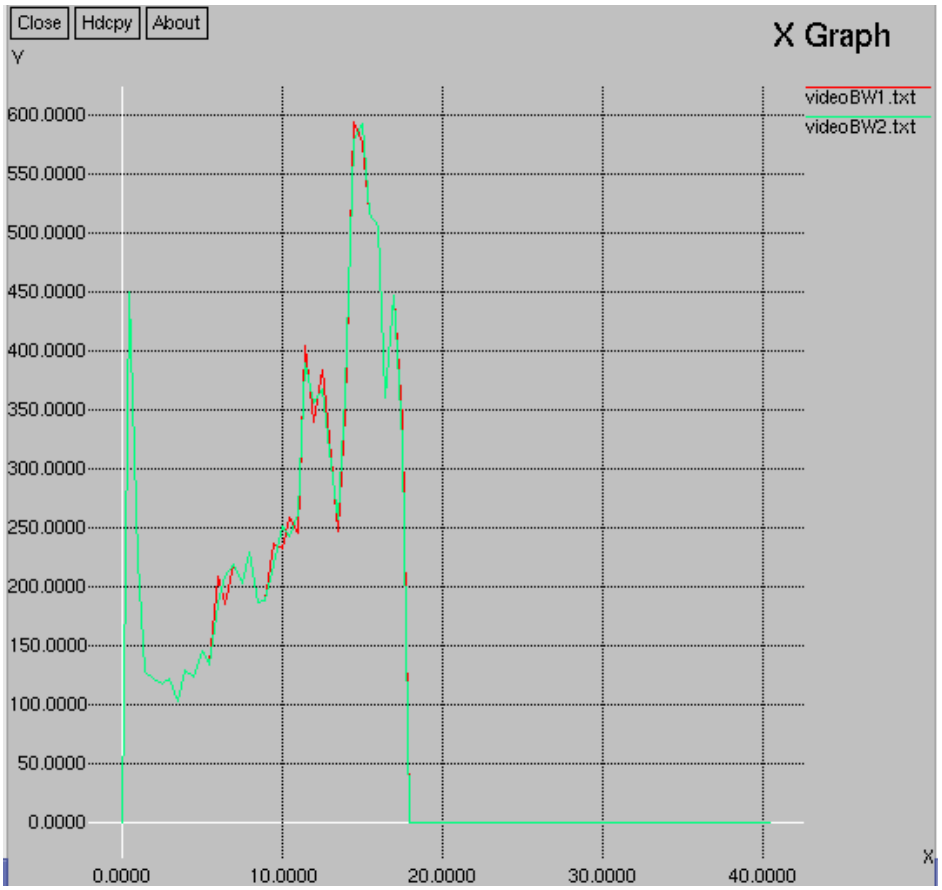
Σχήμα 4.13 - Τοπολογία για σενάριο 3

Αποτελέσματα

	Απώλεια Πακέτων (%)	**Καθυστέρηση δύο άκρων (δευτερόλεπτα)**	**Διακύμανση Καθυστέρησης (δευτερολεπτα)**	**Μέσος Όρος Εύρους Ζώνης (Kbps)**
Παραλήπτης 1	0	0.06	0.01	286.2
Παραλήπτης 2	0	0.07	0.01	285.7

Και για τους δύο παραλήπτες δεν υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής είναι αρκετό για να καλύψει και τις δύο ροές δεδομένων που θα φεύγουν από τον αποστολέα. Η καθυστέρηση δύο άκρων για τον πρώτο παραλήπτη κυμαίνεται στα 0.06 δευτερόλεπτα και για τον δεύτερο παραλήπτη κυμαίνεται στα 0.07 δευτερόλεπτα επίσης. Η διακύμανση καθυστέρησης και για τους δύο παραλήπτες είναι 0.01 δευτερόλεπτα. Ο μέσος όρος εύρους ζώνης είναι σχεδόν ο ίδιος και για τους δύο παραλήπτες οπότε γίνεται δίκαιη κατανομή του μεγέθους της σύνδεσης και στους δύο παραλήπτες.

Γραφική παράσταση – Εύρος Ζώνης



Γράφημα 4.3 – Εύρος ζώνης για τους 2 παραλήπτες

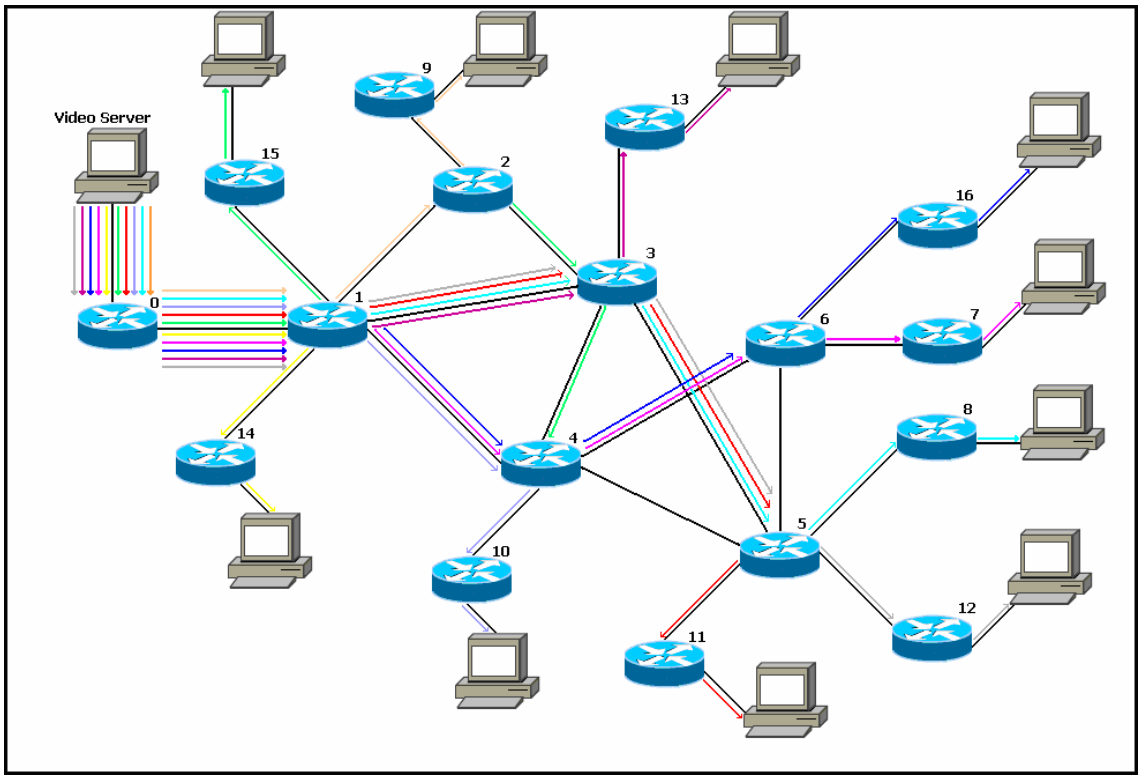
Στην παραπάνω γραφική παράσταση παρατηρούμε ότι η προσομοίωσή μας διαρκεί 40 δευτερόλεπτα. Το εύρος ζώνης κυμαίνεται περίπου στα ίδια επίπεδα τόσο για τον πρώτο παραλήπτη όσο και για τον δεύτερο παραλήπτη, και συγκεκριμένα για τον πρώτο παραλήπτη το εύρος ζώνης κυμαίνεται από 120 μέχρι 600 Kbps και για τον δεύτερο

παραλήπτη κυμαίνεται στα ίδια επίπεδα. Αυτό σημαίνει ότι γίνεται δίκαιη κατανομή του εύρους ζώνης και για τους δύο παραλήπτες.

4.3.2.4 Σενάριο 4: ADIVIS (Scalable Video) with 10 multiple unicast users

Για το σενάριο αυτό, δημιουργήσαμε μια τοπολογία η οποία αποτελείται από από τέσσερις κεντρικούς δρομολογητές, έναν εξυπηρετητή Video, και δώδεκα ακρινούς δρομολογητές. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία αυτή. Στο συγκεκριμένο σενάριο υπάρχουν δέκα χρήστες οι οποίοι ενώνονται και παραλαμβάνουν δεδομένα από τον εξυπηρετητή Video. Στο σενάριο αυτό το Video που μεταφέρεται στους χρήστες είναι χωρισμένο σε επίπεδα.

Τοπολογία



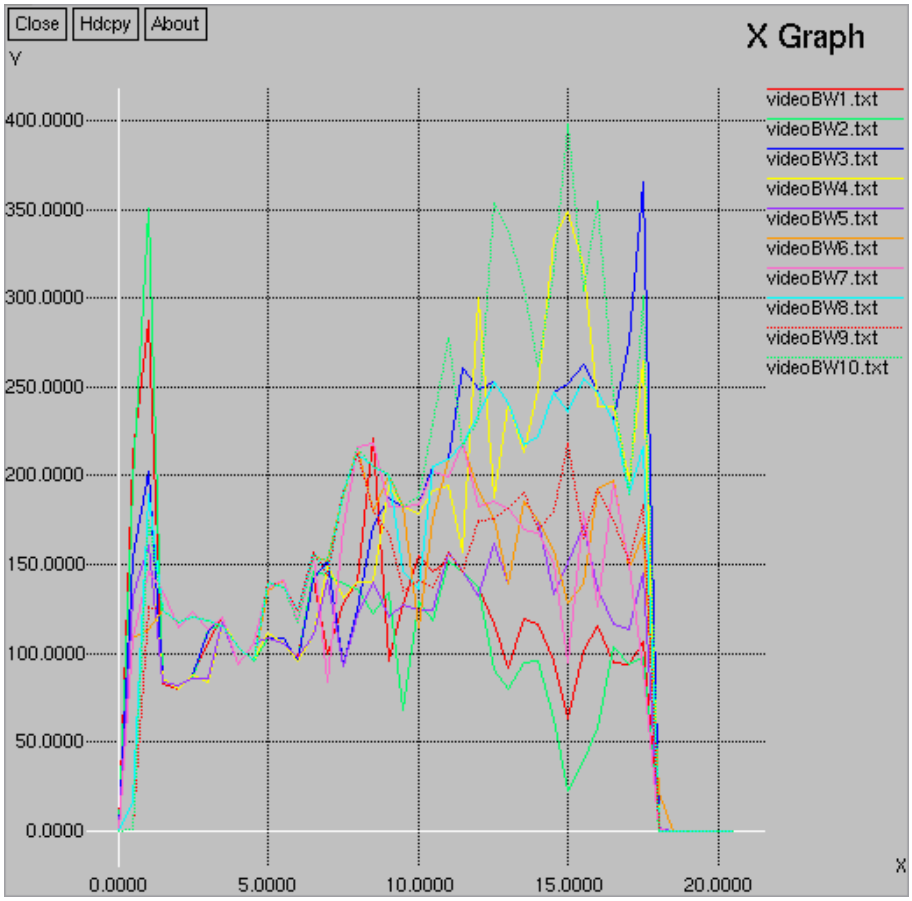
Σχήμα 4.14 - Τοπολογία για σενάριο 4

Αποτελέσματα

	Απώλεια Πακέτων (%)	**Καθυστέρηση δύο άκρων (δευτερόλεπτα)**	**Διακύμανση Καθυστέρησης (δευτερολεπτα)**	**Μέσος Όρος Εύρους Ζώνης (Kbps)**
Παραλήπτης 1	7.85	0.11	0.05	130.5
Παραλήπτης 2	8.81	0.11	0.05	120.4
Παραλήπτης 3	3.80	0.10	0.04	190.2
Παραλήπτης 4	5.66	0.09	0.04	185.3
Παραλήπτης 5	7.22	0.11	0.04	134.3
Παραλήπτης 6	8.57	0.10	0.05	163.5
Παραλήπτης 7	10.27	0.10	0.05	161.1
Παραλήπτης 8	6.38	0.08	0.04	186.8
Παραλήπτης 9	7.98	0.08	0.04	158.4
Παραλήπτης 10	7.96	0.10	0.05	215

Και στους δέκα παραλήπτες υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής δεν είναι αρκετό για να καλύψει και τις δέκα ροές δεδομένων που θα φεύγουν από τον αποστολέα. Η καθυστέρηση δύο άκρων και για τους δέκα παραλήπτες κυμαίνεται από 0.08 μέχρι 0.11 δευτερόλεπτα. Η διακύμανση καθυστέρησης κυμαίνεται από 0.04 μέχρι 0.05 δευτερόλεπτα και για τους δέκα παραλήπτες. Ο μέσος όρος εύρους ζώνης δεν είναι ο ίδος για όλους τους παραλήπτες οπότε δεν γίνεται δίκαιη κατανομή του μεγέθους της σύνδεσης σε όλους τους παραλήπτες.

Γραφική Παράσταση – Εύρος Ζώνης



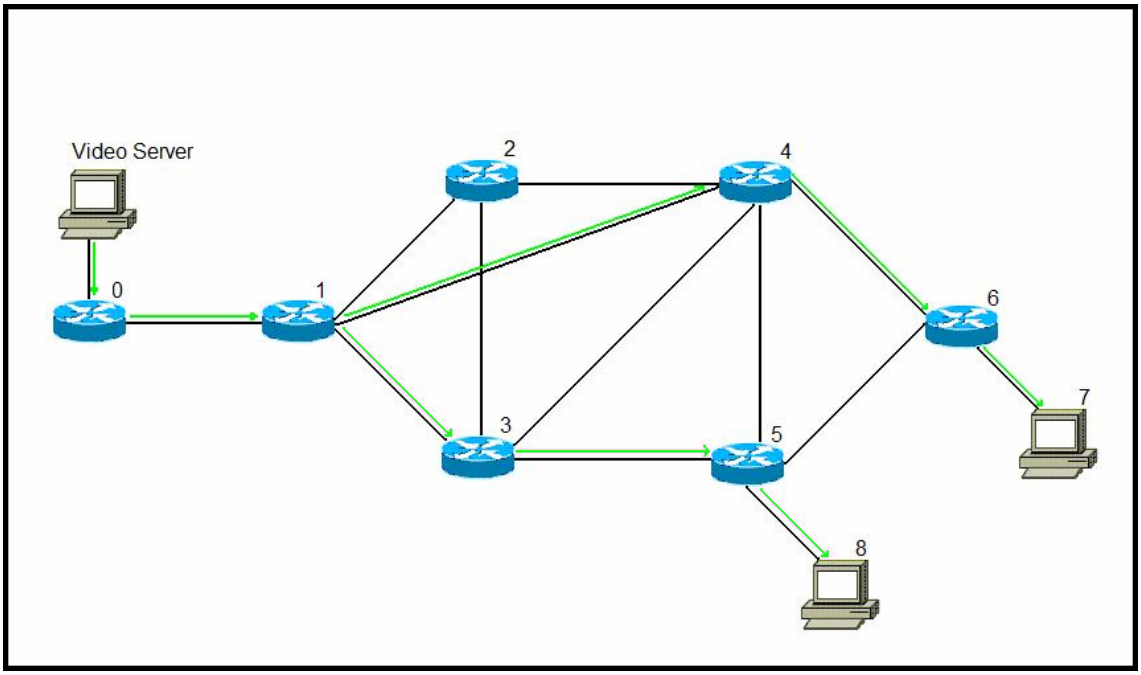
Γράφημα 4.4 – Εύρος ζώνης για τους 10 παραλήπτες

Στην παραπάνω γραφική παράσταση η προσομοίωσή μας διαρκεί 21 δευτερόλεπτα. Παρατηρούμε ότι σε γενικές γραμμές δεν γίνεται δίκαιη κατανομή του εύρους ζώνης σε όλους τους παραλήπτες. Συγκεκριμένα, ο πρώτος παραλήπτης έχει το μεγαλύτερο έυρος ζώνης μιας και φτάνει στα 400 Kbps. Για τους υπόλοιπους εννέα παραλήπτες, το εύρος ζώνης είναι σε χαμηλότερα επίπεδα και καθώς φτάνουμε στον τελευταίο παραλήπτη το εύρος ζώνης είναι ακόμη πιο χαμηλό. Αυτό συμβαίνει διότι η γραμμή δεν είναι αρκετή για να εξυπηρετήσει δίκαια όλους τους παραλήπτες.

4.3.2.5 Σενάριο 5: Simple video transfer with 2 multicast users (πρωτόκολλο Source Specific Multicast)

Για το σενάριο αυτό, δημιουργήσαμε μια τοπολογία η οποία αποτελείται από από τέσσερις κεντρικούς δρομολογητές, έναν εξυπηρετητή Video, και τέσσερις ακρινούς δρομολογητές. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία αυτή. Στο συγκεκριμένο σενάριο υπάρχουν δύο χρήστες οι οποίοι ενώνονται και παραλαμβάνουν δεδομένα από τον εξυπηρετητή Video. Στο σενάριο αυτό το Video που μεταφέρεται στους χρήστες είναι απλό και για την Multicast μετάδοση χρησιμοποιείται το πρωτόκολλο SSM όπως αναφέρεται παραπάνω.

Τοπολογία



Σχήμα 4.15 - Τοπολογία για σενάριο 5

Αποτελέσματα

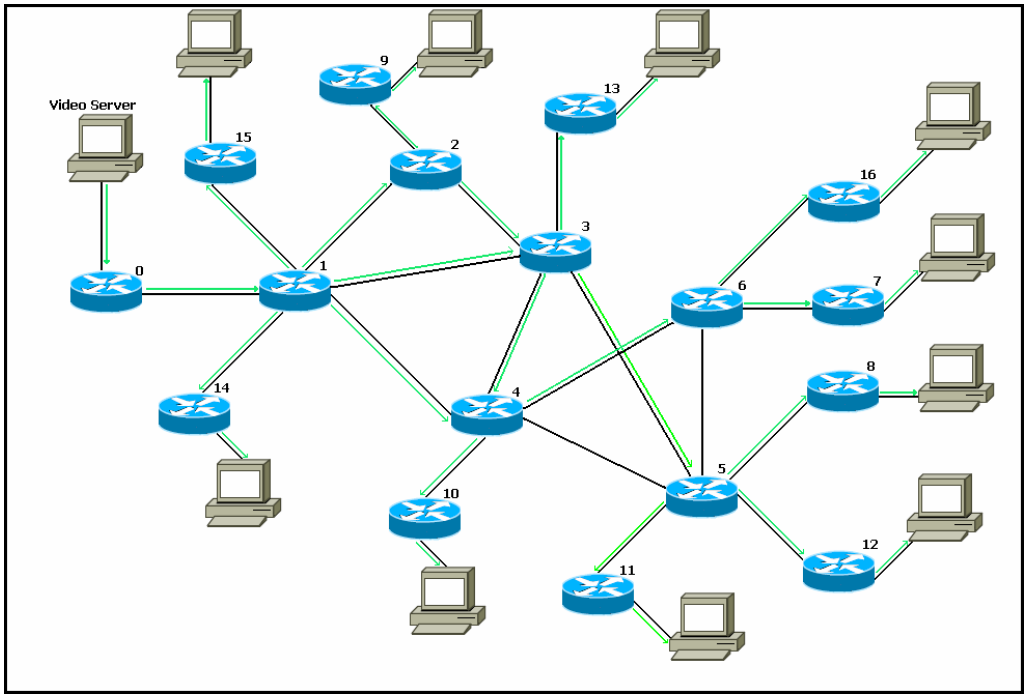
	Απώλεια Πακέτων (%)	**Καθυστέρηση δύο άκρων (δευτερόλεπτα)**	**Διακύμανση Καθυστέρησης (δευτερολεπτα)**	**Εύρος Ζώνης (Kbps)**
Παραλήπτης 1	0	0.06	0.04	588
Παραλήπτης 2	0	0.06	0.04	588

Και για τους δύο παραλήπτες δεν υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής είναι αρκετό για να καλύψει τη μια ροή δεδομένων που θα φεύγει από τον αποστολέα. Η καθυστέρηση δύο άκρων για τον πρώτο παραλήπτη κυμαίνεται στα 0.6 δευτερόλεπτα και για τον δεύτερο παραλήπτη κυμαίνεται στα 0.6 δευτερόλεπτα επίσης. Η διακύμανση καθυστέρησης και για τους δύο παραλήπτες είναι 0.04 δευτερόλεπτα. Το εύρος ζώνης και για τους δύο παραλήπτες είναι 588 Kbps. Λόγω του ότι χρησιμοποιούμε multicast μετάδοση, μόνο μία ροή δεδομένων φεύγει από τον αποστολέα και για αυτό το λόγο δεν υπάρχει συμφόρηση στο δίκτυο.

4.3.2.6 Σενάριο 6: Simple video transfer with 10 multicast users (πρωτόκολλο Source Specific Multicast)

Για το σενάριο αυτό, δημιουργήσαμε μια τοπολογία η οποία αποτελείται από από τέσσερις κεντρικούς δρομολογητές, έναν εξυπηρετητή Video, και δώδεκα ακρινούς δρομολογητές. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία αυτή. Στο συγκεκριμένο σενάριο υπάρχουν δέκα χρήστες οι οποίοι ενώνονται και παραλαμβάνουν δεδομένα από τον εξυπηρετητή Video. Στο σενάριο αυτό το Video που μεταφέρεται στους χρήστες είναι απλό και για την Multicast μετάδοση χρησιμοποιείται το πρωτόκολλο SSM όπως αναφέρεται παραπάνω.

Τοπολογία



Σχήμα 4.16 - Τοπολογία για σενάριο 6

Αποτελέσματα

	Απώλεια Πακέτων (%)	Καθυστέρηση δύο άκρων (δευτερόλεπτα)	Διακύμανση Καθυστέρησης (δευτερολεπτα)	Εύρος Ζώνης (Kbps)
Παραλήπτης 1	0	0.06	0.04	584
Παραλήπτης 2	0	0.06	0.04	581
Παραλήπτης 3	0	0.05	0.03	581
Παραλήπτης 4	0	0.05	0.03	581
Παραλήπτης 5	0	0.06	0.04	581
Παραλήπτης 6	0	0.06	0.04	581
Παραλήπτης 7	0	0.05	0.03	581
Παραλήπτης 8	0	0.03	0.03	581
Παραλήπτης 9	0	0.03	0.04	581
Παραλήπτης 10	0	0.06	0.04	583

Και για τους δέκα παραλήπτες δεν υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής είναι αρκετό για να καλύψει τη μια ροή δεδομένων που θα φεύγει από τον αποστολέα. Η καθυστέρηση δύο άκρων ξεκινά από 0.03 δευτερόλεπτα και φτάνει στα 0.06. Η διακύμανση καθυστέρησης και για τους δέκα παραλήπτες είναι από 0.03 δευτερόλεπτα μέχρι 0.04 δευτερόλεπτα. Το εύρος ζώνης για τον πρώτο παραλήπτη είναι 584 Kbps, για τους επόμενους οκτώ παραλήπτες είναι 581 Kbps και για τον τελευταίο παραλήπτη είναι 583 Kbps. Λόγω του ότι χρησιμοποιούμε multicast μετάδοση, μόνο μία ροή δεδομένων φεύγει από τον αποστολέα και για αυτό το λόγο δεν υπάρχει συμφόρηση στο δίκτυο.

ΚΕΦΑΛΑΙΟ 5

Συμπεράσματα και μελλοντική εργασία

5.1 Συμπεράσματα

Εκτελώντας τα πιο πάνω σενάρια που αφορούσαν unicast και multicast μετάδοση πολυμεσικού περιεχομένου και συγκεκριμένα απλού και ιεραρχικού Video, παρατηρούμε ότι στην περίπτωση που έχουμε αξιολόγηση unicast μετάδοσης απλού Video, όταν οι παραλήπτες είναι μόνο δύο, δεν προκαλείται συμφόρηση στο δίκτυο διότι το μέγεθος της γραμμής που είναι 2Mb είναι αρκετό για να εξυπηρετήσει και τις δύο ροές δεδομένων που δημιουργεί ο αποστολέας για να στείλει το απλό Video και στους δύο παραλήπτες. Στην περίπτωση όμως που το σενάριο αλλάζει και οι παραλήπτες αυξάνονται, και συγκεκριμένα γίνονται δέκα, παρατηρούμε ότι για όλους τους παραλήπτες υπάρχει απώλεια πακέτων διότι το μέγεθος της γραμμής σε αυτήν την περίπτωση δεν είναι αρκετό για να εξυπηρετηθούν όλοι οι παραλήπτες. Έχουμε δέκα διαφορετικές ροές που δημιουργεί ο αποστολέας και για το λόγο αυτό στο δίκτυο προκαλείται συμφόρηση. Το ίδιο συμβαίνει και με την περίπτωση που οι χρήστες παραλαμβάνουν Video που είναι χωρισμένο σε επίπεδα (ADIVIS), διότι και πάλι προκαλείται συμφόρηση του δικτύου μιας και δημιουργούνται δέκα διαφορετικές ροές για να εξυπηρετηθούν και οι δέκα παραλήπτες. Κατά συνέπεια, καταλήγουμε στο συμπέρασμα ότι μέσω της unicast μετάδοσης δεδομένων, δεν υπάρχει ορθή εκμετάλλευση των δικτυακών πόρων. Παρουσιάστηκε λοιπόν η ανάγκη δημιουργίας σεναρίου που να υποστηρίζει multicast μετάδοση πολυμεσικού περιεχομένου έτσι ώστε να γίνεται σωστή εκμετάλλευση των πόρων του δικτύου και να μην υπάρχει συμφόρηση στο δίκτυο. Χρησιμοποιώντας το πρωτόκολλο SSM (Source Specific Multicast), πετύχαμε αποδοτικά σενάρια όσον αφορά την εκατοστιαία απώλεια πακέτων, το εύρος ζώνης και τις καθυστερήσεις στο δίκτυο.

5.2 Μελλοντική Εργασία

Τα σενάρια τα οποία έχουμε τρέξει αφορούν χρήστες οι οποίοι είναι ακίνητοι. Τόσο στην περίπτωση της multicast όσο και στην περίπτωση της multiple unicast μετάδοσης, οι χρήστες που παραλαμβάνουν πολυμεσικό περιεχόμενο από τον εξυπηρετητή, δε μετακινούνται καθόλη τη διάρκεια της προσομοίωσης. Ως μελλοντική εργασία, θα μπορούσε να εξεταστεί κατά πόσο το πρωτόκολλο SSM μπορεί να χρησιμοποιηθεί

αποδοτικά και για κινητούς χρήστες σε MIPv6 δίκτυα. Παράλληλα, θα μπορούσε να εξεταστεί αν το πρωτόκολλο ADIVIS μπορεί να λειτουργήσει και με multicast μετάδοση παρά με μόνο multiple unicast μετάδοση που έχει αξιολογηθεί μέχρι στιγμής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Jianping Pan, Jon W. Mark, Sherman X. Shen, "TCP Performance and Behaviors with Local Retransmissions", The Journal of Supercomputing

[2] Nicholas J. Ploplys, Andrew G. Alleyne, "UDP Network Communications for Distributed Wireless Control", Proceedings of 2003 IEEE Conference on Decision and Control, December 2003 (CDC'03)

[3] Sangheon Pack,Yanghee Choi, "A Study on Performance of Hierarchical Mobile IPv6 in IP-based Cellular Networks", IP Based Cellular Network Conference 2003 (IPCN 2003) Paris, France

[4] Johnson, D.B., Perkins, C.E., Arko, J. 2003, "Mobility Support in IPv6", IETF work in process, expires 21 July 2003

[5] Jiunn-Ru Lai, Wanjiun Liao, Ming-Yu Jiang, Chien-An Ke, "Mobile Multicast with Routing Optimization for Recipient Mobility", IEEE Trans. Consumer Electronics,vol. 47, no. 1, pp. 199-206,2001 (EI, SCI)

[6] Carey L. Williamson, Tim G. Harrison, Wayne L. Mackrell, Richard B. Bunt, "Performance evaluation of the MoM mobile multicast protocol", ACM/Baltzer Journal on Mobile Networks and Applications (MONET), Vol. 3, pp. 189-201, 1998

[7] Liao Yong, Sun Limin, "Wu Zhimei, Multicast for Mobile Hosts in IP Networks", Document No. 69 11-15 November 2002 28th Conference (Manila)

[8] Jin Park and Young-Joo Suh, "A Timer-Based Mobile Multicast Routing Protocol in Mobile Networks", Computer Communications Journal 2003, 2004-07-23

[9] Nishizawa A, Mizoguchi M and Hattori T, "Wireless Multicast System With Connection And Connectionless Modes", Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th

[10] Changdong Liu and Myung J. Lee, "Core-Manager Based Scalable Multicast Routing", ICC98, June 1998, Atlanta

[11] C.R. Lin and K.-M. Wang, "Scalable multicast protocolin IP-based mobile networks", ACM Wireless Networks, 8:27-36, 2002

[12] Jiunn-Ru Lai, Wanjiun Liao, Ming-Yu Jiang, and Chien-An Ke, "Mobile multicast with routing optimization for recipient mobility," Proc. of IEEE ICC 2001

[13] H. Zheng and J. Boyce, "Packet Coding Schemes for MPEG Video over Internet and Wireless Networks", Proc. of IEEE Wireless and Communications and Networking Conference(WCNC), Chicago, IL. Sept. 2000, no.1, pp. 191-195

[14] Christos Bouras, Apostolos Gkamas, Dimitris Primpas, Kostas Stamos, "Multicast of Multimedia Data", International Journal of Communications Systems, Wiley InterScience, Volume 16, Issue 2, pp. 225 – 248

[15] Abderrahim Benslimane, "Multimedia Multicast in Mobile Computing", Published in Microelectronics Systems Education Journal, 2000

[16] Ing-Chau Chang, Kuo-Shun Huang, Chaoyang University of Technology

Taichung County, "Synchronized Multimedia Multicast on Mobile IP Networks", http://ieeexplore.ieee.org/iel5/8564/27114/01204438.pdf

[17] Gopi Kurup Ahmet Sekercioglu, "Source Specific Multicast (SSM) for MIPv6: A survey of current state of standardisation and Research", ECSE Research Forum, Feb 2004

[18] P. Antoniou, A. Pitsillides, and V. Vassiliou, "Adaptive Feedback Algorithm for Internet Video Streaming based on Fuzzy Rate Control, " Submitted to the IEEE Symposium on Computers and Communications (ISCC'07), Aveiro, Portugal, July 1-4, 2007.

ΠΑΡΑΡΤΗΜΑ Α

Α.1 Κώδικας που χρησιμοποιήθηκε για προσομοίωση σεναρίων

Α.1.1 Σενάριο 2 - Simple video μετάδοση με 10 multiple unicast users

```
set ns [new Simulator]

set f [open out_multiple_unicast_ten.txt w]

set videoBW1 [open videoBW1.txt w]
set videoBW2 [open videoBW2.txt w]
set videoBW3 [open videoBW3.txt w]
set videoBW4 [open videoBW4.txt w]
set videoBW5 [open videoBW5.txt w]
set videoBW6 [open videoBW6.txt w]
set videoBW7 [open videoBW7.txt w]
set videoBW8 [open videoBW8.txt w]
set videoBW9 [open videoBW9.txt w]
set videoBW10 [open videoBW10.txt w]

$ns trace-all $f
$ns namtrace-all [open out_multiple_unicast_ten.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# nod is the number of nodes
set num_wired_nodes 17

# create multicast capable nodes;
```

```
for {set i 1} {$i <= $num_wired_nodes} {incr i} {
 set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(7) 2Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 2Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 2Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 2Mb 10ms DropTail

$ns duplex-link $n(10) $n(3) 2Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(12) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(13) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(14) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(15) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(16) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(17) $n(7) 2Mb 10ms DropTail



set max_fragmented_size 1000
set udp1 [new Agent/myUDP]
$ns attach-agent $n(1) $udp1
$udp1 set packetSize_ 1028
```

```
$udp1 set_filename sd_be

set udp2 [new Agent/myUDP]
$ns attach-agent $n(1) $udp2
$udp2 set packetSize_ 1028
$udp2 set_filename sd_be

set udp3 [new Agent/myUDP]
$ns attach-agent $n(1) $udp3
$udp3 set packetSize_ 1028
$udp3 set_filename sd_be

set udp4 [new Agent/myUDP]
$ns attach-agent $n(1) $udp4
$udp4 set packetSize_ 1028
$udp4 set_filename sd_be

set udp5 [new Agent/myUDP]
$ns attach-agent $n(1) $udp5
$udp5 set packetSize_ 1028
$udp5 set_filename sd_be

set udp6 [new Agent/myUDP]
$ns attach-agent $n(1) $udp6
$udp6 set packetSize_ 1028
$udp6 set_filename sd_be

set udp7 [new Agent/myUDP]
$ns attach-agent $n(1) $udp7
$udp7 set packetSize_ 1028
$udp7 set_filename sd_be

set udp8 [new Agent/myUDP]
$ns attach-agent $n(1) $udp8
```

```
$udp8 set packetSize_ 1028
$udp8 set_filename sd_be

set udp9 [new Agent/myUDP]
$ns attach-agent $n(1) $udp9
$udp9 set packetSize_ 1028
$udp9 set_filename sd_be

set udp10 [new Agent/myUDP]
$ns attach-agent $n(1) $udp10
$udp10 set packetSize_ 1028
$udp10 set_filename sd_be

# ---------------------
# SIMPLE VIDEO TRAFFIC |
# ---------------------
set original_file_name st
set trace_file_name video1.dat
set original_file_id [open $original_file_name r]
set trace_file_id [open $trace_file_name w]

set frame_count 0
set last_time 0

while {[eof $original_file_id] == 0} {
   gets $original_file_id current_line

   scan $current_line "%d%s%d%s%s%s%d%s" no_ frametype_ length_ tmp1_ tmp2_
tmp3_ tmp4_ tmp5_
   #puts "$no_ $frametype_ $length_ $tmp1_ $tmp2_ $tmp3_ $tmp4_ $tmp5_"

   # 30 frames/sec. if one want to generate 30 frames/sec, one can use set time [expr
1000*1000/30]
   set time [expr 1000 * 1000/25]
```

```
    if { $frametype_ == "I" } {
        set type_v 1
    }

    if { $frametype_ == "P" } {
        set type_v 2
    }

    if { $frametype_ == "B" } {
        set type_v 3
    }

    if { $frametype_ == "H" } {
        set type_v 1
    }

    puts  $trace_file_id "$time $length_ $type_v $max_fragmented_size"
    incr frame_count
}

close $original_file_id
close $trace_file_id
set end_sim_time [expr 1.0 * 1000/25 * ($frame_count + 1)  / 1000]

set trace_file [new Tracefile]
$trace_file filename $trace_file_name
set video1 [new Application/Traffic/myTrace2]
set video2 [new Application/Traffic/myTrace2]
set video3 [new Application/Traffic/myTrace2]
set video4 [new Application/Traffic/myTrace2]
set video5 [new Application/Traffic/myTrace2]
set video6 [new Application/Traffic/myTrace2]
set video7 [new Application/Traffic/myTrace2]
```

```
set video8 [new Application/Traffic/myTrace2]
set video9 [new Application/Traffic/myTrace2]
set video10 [new Application/Traffic/myTrace2]



$video1 attach-agent $udp1
$video2 attach-agent $udp2
$video3 attach-agent $udp3
$video4 attach-agent $udp4
$video5 attach-agent $udp5
$video6 attach-agent $udp6
$video7 attach-agent $udp7
$video8 attach-agent $udp8
$video9 attach-agent $udp9
$video10 attach-agent $udp10

$video1 attach-tracefile $trace_file
$video2 attach-tracefile $trace_file
$video3 attach-tracefile $trace_file
$video4 attach-tracefile $trace_file
$video5 attach-tracefile $trace_file
$video6 attach-tracefile $trace_file
$video7 attach-tracefile $trace_file
$video8 attach-tracefile $trace_file
$video9 attach-tracefile $trace_file
$video10 attach-tracefile $trace_file

# END OF VIDEO TRAFFIC

# create receiver agents
set rcvr1 [new Agent/myUdpSink2]
$rcvr1 set_trace_filename rd_be delay1.txt
```

```
set rcvr2 [new Agent/myUdpSink2]
$rcvr2 set_trace_filename rd_be delay2.txt

set rcvr3 [new Agent/myUdpSink2]
$rcvr3 set_trace_filename rd_be delay3.txt

set rcvr4 [new Agent/myUdpSink2]
$rcvr4 set_trace_filename rd_be delay4.txt

set rcvr5 [new Agent/myUdpSink2]
$rcvr5 set_trace_filename rd_be delay5.txt

set rcvr6 [new Agent/myUdpSink2]
$rcvr6 set_trace_filename rd_be delay6.txt

set rcvr7 [new Agent/myUdpSink2]
$rcvr7 set_trace_filename rd_be delay7.txt

set rcvr8 [new Agent/myUdpSink2]
$rcvr8 set_trace_filename rd_be delay8.txt

set rcvr9 [new Agent/myUdpSink2]
$rcvr9 set_trace_filename rd_be delay9.txt

set rcvr10 [new Agent/myUdpSink2]
$rcvr10 set_trace_filename rd_be delay10.txt




$ns attach-agent $n(8) $rcvr1
$ns attach-agent $n(9) $rcvr2
$ns attach-agent $n(10) $rcvr3
$ns attach-agent $n(11) $rcvr4
$ns attach-agent $n(12) $rcvr5
```

```
$ns attach-agent $n(13) $rcvr6
$ns attach-agent $n(14) $rcvr7
$ns attach-agent $n(15) $rcvr8
$ns attach-agent $n(16) $rcvr9
$ns attach-agent $n(17) $rcvr10



$ns at 0.0 "record_video_bw"
# joining and leaving the group;

$ns connect $rcvr1 $udp1
$ns connect $rcvr2 $udp2
$ns connect $rcvr3 $udp3
$ns connect $rcvr4 $udp4
$ns connect $rcvr5 $udp5
$ns connect $rcvr6 $udp6
$ns connect $rcvr7 $udp7
$ns connect $rcvr8 $udp8
$ns connect $rcvr9 $udp9
$ns connect $rcvr10 $udp10

$ns at 0.1 "$video1 start"
$ns at 0.1 "$video2 start"
$ns at 0.1 "$video3 start"
$ns at 0.1 "$video4 start"
$ns at 0.1 "$video5 start"
$ns at 0.1 "$video6 start"
$ns at 0.1 "$video7 start"
$ns at 0.1 "$video8 start"
$ns at 0.1 "$video9 start"
$ns at 0.1 "$video10 start"

$ns at 20.0 "$video1 stop"
```

```
$ns at 20.0 "$video2 stop"
$ns at 20.0 "$video3 stop"
$ns at 20.0 "$video4 stop"
$ns at 20.0 "$video5 stop"
$ns at 20.0 "$video6 stop"
$ns at 20.0 "$video7 stop"
$ns at 20.0 "$video8 stop"
$ns at 20.0 "$video9 stop"
$ns at 20.0 "$video10 stop"


$ns at 21.0 "$rcvr1 closefile"
$ns at 21.0 "$rcvr2 closefile"
$ns at 21.0 "$rcvr3 closefile"
$ns at 21.0 "$rcvr4 closefile"
$ns at 21.0 "$rcvr5 closefile"
$ns at 21.0 "$rcvr6 closefile"
$ns at 21.0 "$rcvr7 closefile"
$ns at 21.0 "$rcvr8 closefile"
$ns at 21.0 "$rcvr9 closefile"
$ns at 21.0 "$rcvr10 closefile"
$ns at 25.0 "record_bw_loss_delay"
$ns at 25.0 "finish"

proc finish {} {
    global ns f videoBW1 videoBW2 videoBW3  videoBW4 videoBW5 videoBW6
videoBW7 videoBW8 videoBW9 videoBW10
    $ns flush-trace
      close $f
      close $videoBW1
      close $videoBW2
    close $videoBW3
      close $videoBW4
      close $videoBW5
```

```
        close $videoBW6
        close $videoBW7
        close $videoBW8
        close $videoBW9
        close $videoBW10
    exec ../../../../xgraph-12.1/xgraph videoBW1.txt videoBW2.txt videoBW3.txt videoBW4.txt videoBW5.txt videoBW6.txt videoBW7.txt videoBW8.txt videoBW9.txt videoBW10.txt &
    exit 0
}

set total_bw1_ 0
set total_bw2_ 0
set total_bw3_ 0
set total_bw4_ 0
set total_bw5_ 0
set total_bw6_ 0
set total_bw7_ 0
set total_bw8_ 0
set total_bw9_ 0
set total_bw10_ 0

proc record_video_bw {} {
    global videoBW1 videoBW2 videoBW2 videoBW3 videoBW4 videoBW5 videoBW6 videoBW7 videoBW8 videoBW9 videoBW10 rcvr1 rcvr2 rcvr3 rcvr4 rcvr5 rcvr6 rcvr7 rcvr8 rcvr9 rcvr10 total_bw1_ total_bw2_ total_bw3_ total_bw4_ total_bw5_ total_bw6_ total_bw7_ total_bw8_ total_bw9_ total_bw10_

        #Get an instance of the simulator
    set ns [Simulator instance]
    #Set the time after which the procedure should be called again
    set timez 0.5
    #How many bytes have been received by the traffic sessions?
    set bw1 [$rcvr1 set bytes_]
```

```
    set total_bw1_ [expr $total_bw1_ + $bw1]
set bw2 [$rcvr2 set bytes_]
    set total_bw2_ [expr $total_bw2_ + $bw2]
set bw3 [$rcvr3 set bytes_]
    set total_bw3_ [expr $total_bw3_ + $bw3]
set bw4 [$rcvr4 set bytes_]
    set total_bw4_ [expr $total_bw4_ + $bw4]
set bw5 [$rcvr5 set bytes_]
    set total_bw5_ [expr $total_bw5_ + $bw5]
set bw6 [$rcvr6 set bytes_]
    set total_bw6_ [expr $total_bw6_ + $bw6]
set bw7 [$rcvr7 set bytes_]
    set total_bw7_ [expr $total_bw7_ + $bw7]
set bw8 [$rcvr8 set bytes_]
    set total_bw8_ [expr $total_bw8_ + $bw8]
set bw9 [$rcvr9 set bytes_]
    set total_bw9_ [expr $total_bw9_ + $bw9]
set bw10 [$rcvr10 set bytes_]
    set total_bw10_ [expr $total_bw10_ + $bw10]


#Get the current time
set now [$ns now]
#Calculate the bandwidth (in KBit/s) and write it to the files
puts $videoBW1 "$now  [expr $bw1/$timez*8/1000]"
puts $videoBW2 "$now  [expr $bw2/$timez*8/1000]"
puts $videoBW3 "$now  [expr $bw3/$timez*8/1000]"
puts $videoBW4 "$now  [expr $bw4/$timez*8/1000]"
puts $videoBW5 "$now  [expr $bw5/$timez*8/1000]"
puts $videoBW6 "$now  [expr $bw6/$timez*8/1000]"
puts $videoBW7 "$now  [expr $bw7/$timez*8/1000]"
puts $videoBW8 "$now  [expr $bw8/$timez*8/1000]"
puts $videoBW9 "$now  [expr $bw9/$timez*8/1000]"
puts $videoBW10 "$now  [expr $bw10/$timez*8/1000]"
```

```
        #Reset the bytes_ values on the traffic sessions
        $rcvr1 set bytes_ 0
        $rcvr2 set bytes_ 0
        $rcvr3 set bytes_ 0
        $rcvr4 set bytes_ 0
        $rcvr5 set bytes_ 0
        $rcvr6 set bytes_ 0
        $rcvr7 set bytes_ 0
        $rcvr8 set bytes_ 0
        $rcvr9 set bytes_ 0
        $rcvr10 set bytes_ 0

        #Re-schedule the procedure
        $ns at [expr $now+$timez] "record_video_bw"
}

proc record_bw_loss_delay {} {
        global rcvr1 rcvr2 rcvr3 rcvr4 rcvr5 rcvr6 rcvr7 rcvr8 rcvr9 rcvr10  total_bw1_
total_bw2_  total_bw3_  total_bw4_  total_bw5_  total_bw6_  total_bw7_  total_bw8_
total_bw9_ total_bw10_
puts "MEAN BANDWIDTH FROM VIDEO CLIENT 1: [expr $total_bw1_ "
      puts "MEAN BANDWIDTH FROM VIDEO CLIENT 2: [expr $total_bw2_ "
        set loss_1 [$rcvr1 set loss_]
        set loss_2 [$rcvr2 set loss_]
        set loss_3 [$rcvr3 set loss_]
        set loss_4 [$rcvr4 set loss_]
        set loss_5 [$rcvr5 set loss_]
        set loss_6 [$rcvr6 set loss_]
        set loss_7 [$rcvr7 set loss_]
        set loss_8 [$rcvr8 set loss_]
        set loss_9 [$rcvr9 set loss_]
        set loss_10 [$rcvr10 set loss_]
```

```
set mean_delay_1 [$rcvr1 set mean_delay_]
set mean_delay_2 [$rcvr2 set mean_delay_]
set mean_delay_3 [$rcvr3 set mean_delay_]
   set mean_delay_4 [$rcvr4 set mean_delay_]
set mean_delay_5 [$rcvr5 set mean_delay_]
set mean_delay_6 [$rcvr6 set mean_delay_]
set mean_delay_7 [$rcvr7 set mean_delay_]
set mean_delay_8 [$rcvr8 set mean_delay_]
set mean_delay_9 [$rcvr9 set mean_delay_]
set mean_delay_10 [$rcvr10 set mean_delay_]
```

```
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 1: [expr $loss_1 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 2: [expr $loss_2 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 3: [expr $loss_3 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 4: [expr $loss_4 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 5: [expr $loss_5 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 6: [expr $loss_6 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 7: [expr $loss_7 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 8: [expr $loss_8 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 9: [expr $loss_9 *
100]%"
puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 10: [expr
$loss_10 * 100]%"
```

```
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 1: $mean_delay_1 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 2: $mean_delay_2 sec(s)"
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 3: $mean_delay_3 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 4: $mean_delay_4 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 5: $mean_delay_5 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 6: $mean_delay_6 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 7: $mean_delay_7 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 8: $mean_delay_8 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 9: $mean_delay_9 sec(s)"
        puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 10: $mean_delay_10 sec(s)"



    set count_1 0
    set std_1 0.0
    set original_file_id_1 [open "delay1.txt" r]

    while {[eof $original_file_id_1] == 0} {

        gets $original_file_id_1 current_line_1
        scan $current_line_1 "%f" delay_1
        set std_1 [expr $std_1 + pow([expr $delay_1 - $mean_delay_1],2)]
        incr count_1
    }
```

```
set std_1 [expr sqrt([expr $std_1 / $count_1])]
close $original_file_id_1
puts "END-TO-END JITTER FROM VIDEO CLIENT 1: $std_1 sec(s)"

set count_2 0
set std_2 0.0
set original_file_id_2 [open "delay2.txt" r]

while {[eof $original_file_id_2] == 0} {

        gets $original_file_id_2 current_line_2
        scan $current_line_2 "%f" delay_2
        set std_2 [expr $std_2 + pow([expr $delay_2 - $mean_delay_2],2)]
        incr count_2
}
set std_2 [expr sqrt([expr $std_2 / $count_2])]
close $original_file_id_2
puts "END-TO-END JITTER FROM VIDEO CLIENT 2: $std_2 sec(s)"

set count_3 0
set std_3 0.0
set original_file_id_3 [open "delay3.txt" r]

while {[eof $original_file_id_3] == 0} {

        gets $original_file_id_3 current_line_3
        scan $current_line_3 "%f" delay_3
        set std_3 [expr $std_3 + pow([expr $delay_3 - $mean_delay_3],2)]
        incr count_3
}
set std_3 [expr sqrt([expr $std_3 / $count_3])]
close $original_file_id_3
puts "END-TO-END JITTER FROM VIDEO CLIENT 3: $std_3 sec(s)"
```

```
set count_4 0
set std_4 0.0
set original_file_id_4 [open "delay4.txt" r]


while {[eof $original_file_id_4] == 0} {

        gets $original_file_id_4 current_line_4
        scan $current_line_4 "%f" delay_4
        set std_4 [expr $std_4 + pow([expr $delay_4 - $mean_delay_4],2)]
        incr count_4
}
set std_4 [expr sqrt([expr $std_4 / $count_4])]
close $original_file_id_4
puts "END-TO-END JITTER FROM VIDEO CLIENT 4: $std_4 sec(s)"



set count_5 0
set std_5 0.0
set original_file_id_5 [open "delay5.txt" r]


while {[eof $original_file_id_5] == 0} {

        gets $original_file_id_5 current_line_5
        scan $current_line_5 "%f" delay_5
        set std_5 [expr $std_5 + pow([expr $delay_5 - $mean_delay_5],2)]
        incr count_5
}
set std_5 [expr sqrt([expr $std_5 / $count_5])]
close $original_file_id_5
puts "END-TO-END JITTER FROM VIDEO CLIENT 5: $std_5 sec(s)"
```

```
set count_6 0
set std_6 0.0
set original_file_id_6 [open "delay6.txt" r]


while {[eof $original_file_id_6] == 0} {
        gets $original_file_id_6 current_line_6
       scan $current_line_3 "%f" delay_6
      set std_6 [expr $std_6 + pow([expr $delay_6 - $mean_delay_6],2)]
       incr count_6
}
set std_6 [expr sqrt([expr $std_6 / $count_6])]
close $original_file_id_6
puts "END-TO-END JITTER FROM VIDEO CLIENT 6: $std_6 sec(s)"

set count_8 0
set std_8 0.0
set original_file_id_8 [open "delay8.txt" r]



while {[eof $original_file_id_8] == 0} {


       gets $original_file_id_8 current_line_8
        scan $current_line_8 "%f" delay_8
        set std_8 [expr $std_8 + pow([expr $delay_8 - $mean_delay_8],2)]
        incr count_8
}
set std_8 [expr sqrt([expr $std_8 / $count_8])]
close $original_file_id_8
```

```
puts "END-TO-END JITTER FROM VIDEO CLIENT 8: $std_8 sec(s)"

set count_7 0
set std_7 0.0
set original_file_id_7 [open "delay7.txt" r]



while {[eof $original_file_id_7] == 0} {

        gets $original_file_id_7 current_line_7
        scan $current_line_7 "%f" delay_7
        set std_7 [expr $std_7 + pow([expr $delay_7 - $mean_delay_7],2)]
        incr count_7
}
set std_7 [expr sqrt([expr $std_7 / $count_7])]
close $original_file_id_7
puts "END-TO-END JITTER FROM VIDEO CLIENT 7: $std_7 sec(s)"



set count_9 0
set std_9 0.0
set original_file_id_9 [open "delay9.txt" r]



while {[eof $original_file_id_9] == 0} {

        gets $original_file_id_9 current_line_9
        scan $current_line_9 "%f" delay_9
        set std_9 [expr $std_9 + pow([expr $delay_9 - $mean_delay_9],2)]
        incr count_9
}
set std_9 [expr sqrt([expr $std_9 / $count_9])]
close $original_file_id_9
puts "END-TO-END JITTER FROM VIDEO CLIENT 9: $std_9 sec(s)"
```

```
	set count_10 0
	set std_10 0.0
	set original_file_id_10 [open "delay10.txt" r]


	while {[eof $original_file_id_10] == 0} {

		gets $original_file_id_10 current_line_10
		scan $current_line_10 "%f" delay_10
		set std_10 [expr $std_10 + pow([expr $delay_10 - $mean_delay_10],2)]
		incr count_10
	}
	set std_10 [expr sqrt([expr $std_10 / $count_10])]
	close $original_file_id_10
	puts "END-TO-END JITTER FROM VIDEO CLIENT 10: $std_10 sec(s)"

}

$ns run
```

A.1.2 Σενάριο 4 - ADIVIS (Scalable Video) μετάδοση με 10 multiple unicast users

```
set ns [new Simulator]

set opt(nl) 8

set f [open out_adivis_ten.txt w]

set videoBW1 [open videoBW1.txt w]
set videoBW2 [open videoBW2.txt w]
set videoBW3 [open videoBW3.txt w]
set videoBW4 [open videoBW4.txt w]
set videoBW5 [open videoBW5.txt w]
```

```
set videoBW6 [open videoBW6.txt w]
set videoBW7 [open videoBW7.txt w]
set videoBW8 [open videoBW8.txt w]
set videoBW9 [open videoBW9.txt w]
set videoBW10 [open videoBW10.txt w]
set metricsfile_1 "metrics_1"
set metricsfile_2 "metrics_2"
set metricsfile_3 "metrics_3"
set metricsfile_4 "metrics_4"
set metricsfile_5 "metrics_5"
set metricsfile_6 "metrics_6"
set metricsfile_7 "metrics_7"
set metricsfile_8 "metrics_8"
set metricsfile_9 "metrics_9"
set metricsfile_10 "metrics_10"

$ns trace-all $f
$ns namtrace-all [open out_adivis_ten.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# nod is the number of nodes
set num_wired_nodes 17

# create multicast capable nodes;
for {set i 1} {$i <= $num_wired_nodes} {incr i} {
 set n($i) [$ns node]
}

#Create links between the nodes
```

```
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(7) 2Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 2Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 2Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 2Mb 10ms DropTail

$ns duplex-link $n(10) $n(3) 20Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 20Mb 10ms DropTail
$ns duplex-link $n(12) $n(6) 20Mb 10ms DropTail
$ns duplex-link $n(13) $n(6) 20Mb 10ms DropTail
$ns duplex-link $n(14) $n(4) 20Mb 10ms DropTail
$ns duplex-link $n(15) $n(2) 20Mb 10ms DropTail
$ns duplex-link $n(16) $n(2) 20Mb 10ms DropTail
$ns duplex-link $n(17) $n(7) 20Mb 10ms DropTail



# ---------------------
# SIMPLE VIDEO TRAFFIC |
# ---------------------
set session_0 [new Session/RTP]
set session_1 [new Session/RTP]
$session_0 session_bw 1000kb/s $opt(nl) ;#session bandwidth and number of layers
$session_1 session_bw 1000kb/s $opt(nl)
$session_0 attach-node $n(1)
$session_0 set factorC_ 0.988
$session_1 attach-node $n(8)
```

```
$session_0 sender-filename send_(1)
$session_1 receiver-filename recv_(1)
$session_0 psnr-filename psnrinfo_(1)
$session_0 connect $session_1

set session_2 [new Session/RTP]
set session_3 [new Session/RTP]
$session_2 session_bw 1000kb/s $opt(nl)
$session_3 session_bw 1000kb/s $opt(nl)
$session_2 attach-node $n(1)
$session_2 set factorC_ 0.988
$session_3 attach-node $n(9)
$session_2 sender-filename send_(2)
$session_3 receiver-filename recv_(2)
$session_2 psnr-filename psnrinfo_(2)
$session_2 connect $session_3

set session_4 [new Session/RTP]
set session_5 [new Session/RTP]
$session_4 session_bw 1000kb/s $opt(nl)
$session_5 session_bw 1000kb/s $opt(nl)
$session_4 attach-node $n(1)
$session_4 set factorC_ 0.988
$session_5 attach-node $n(10)
$session_4 sender-filename send_(3)
$session_5 receiver-filename recv_(3)
$session_4 psnr-filename psnrinfo_(3)
$session_4 connect $session_5

set session_6 [new Session/RTP]
set session_7 [new Session/RTP]
$session_6 session_bw 1000kb/s $opt(nl)
$session_7 session_bw 1000kb/s $opt(nl)
$session_6 attach-node $n(1)
```

```
$session_6 set factorC_ 0.988
$session_7 attach-node $n(11)
$session_6 sender-filename send_(4)
$session_7 receiver-filename recv_(4)
$session_6 psnr-filename psnrinfo_(4)
$session_6 connect $session_7

set session_8 [new Session/RTP]
set session_9 [new Session/RTP]
$session_8 session_bw 1000kb/s $opt(nl)
$session_9 session_bw 1000kb/s $opt(nl)
$session_8 attach-node $n(1)
$session_8 set factorC_ 0.988
$session_9 attach-node $n(12)
$session_8 sender-filename send_(5)
$session_9 receiver-filename recv_(5)
$session_8 psnr-filename psnrinfo_(5)
$session_8 connect $session_9

set session_10 [new Session/RTP]
set session_11 [new Session/RTP]
$session_10 session_bw 1000kb/s $opt(nl)
$session_11 session_bw 1000kb/s $opt(nl)
$session_10 attach-node $n(1)
$session_10 set factorC_ 0.988
$session_11 attach-node $n(13)
$session_10 sender-filename send_(6)
$session_11 receiver-filename recv_(6)
$session_10 psnr-filename psnrinfo_(6)
$session_10 connect $session_11

set session_12 [new Session/RTP]
set session_13 [new Session/RTP]
$session_12 session_bw 1000kb/s $opt(nl)
```

```
$session_13 session_bw 1000kb/s $opt(nl)
$session_12 attach-node $n(1)
$session_12 set factorC_ 0.988
$session_13 attach-node $n(14)
$session_12 sender-filename send_(7)
$session_13 receiver-filename recv_(7)
$session_12 psnr-filename psnrinfo_(7)
$session_12 connect $session_13

set session_14 [new Session/RTP]
set session_15 [new Session/RTP]
$session_14 session_bw 1000kb/s $opt(nl)
$session_15 session_bw 1000kb/s $opt(nl)
$session_14 attach-node $n(1)
$session_14 set factorC_ 0.988
$session_15 attach-node $n(15)
$session_14 sender-filename send_(8)
$session_15 receiver-filename recv_(8)
$session_14 psnr-filename psnrinfo_(8)
$session_14 connect $session_15

set session_16 [new Session/RTP]
set session_17 [new Session/RTP]
$session_16 session_bw 1000kb/s $opt(nl)
$session_17 session_bw 1000kb/s $opt(nl)
$session_16 attach-node $n(1)
$session_16 set factorC_ 0.988
$session_17 attach-node $n(16)
$session_16 sender-filename send_(9)
$session_17 receiver-filename recv_(9)
$session_16 psnr-filename psnrinfo_(9)
$session_16 connect $session_17

set session_18 [new Session/RTP]
```

```
set session_19 [new Session/RTP]
$session_18 session_bw 1000kb/s $opt(nl)
$session_19 session_bw 1000kb/s $opt(nl)
$session_18 attach-node $n(1)
$session_18 set factorC_ 0.988
$session_19 attach-node $n(17)
$session_18 sender-filename send_(10)
$session_19 receiver-filename recv_(10)
$session_18 psnr-filename psnrinfo_(10)
$session_18 connect $session_19

##############################
# Extract Video Information #
##############################
set max_fragmented_size 1000
for {set j 0} {$j < $opt(nl)} {incr j} {
        set index [expr $j * [expr 8 / $opt(nl)]]
        set original_file_name($j) layer_($index)
        set trace_file_name($j) layer_($j).dat
        set original_file_id($j) [open $original_file_name($j) r]
        set trace_file_id($j) [open $trace_file_name($j) w]
        set frame_count 0
        set last_time 0

        while {[eof $original_file_id($j)] == 0} {
                gets $original_file_id($j) current_line

                scan $current_line "%d%s%d%s%s%s%d%s" no_ frametype_ length_ tmp1_
tmp2_ tmp3_ tmp4_ tmp5_
                #puts "$no_ $frametype_ $length_ $tmp1_ $tmp2_ $tmp3_ $tmp4_ $tmp5_"

                # 25 frames/sec. if one wants to generate 30 frames/sec, one can use set time
[expr 1000*1000/30]
                set time [expr 1000 * 1000/25]
```

```
        if { $frametype_ == "I" } {
            set type_v 1
        }

        if { $frametype_ == "P" } {
            set type_v 2
        }

        if { $frametype_ == "B" } {
            set type_v 3
        }

        if { $frametype_ == "H" } {
            set type_v 1
        }

        puts  $trace_file_id($j) "$time $length_ $type_v $max_fragmented_size"
        incr frame_count
}

close $original_file_id($j)
close $trace_file_id($j)
set end_sim_time [expr 1.0 * 1000/25 * ($frame_count + 1)  / 1000]
#puts "$end_sim_time"
set trace_file($j) [new Tracefile]
$trace_file($j) filename $trace_file_name($j)
set video_1($j) [new Application/Traffic/videoTrace]
set video_2($j) [new Application/Traffic/videoTrace]
set video_3($j) [new Application/Traffic/videoTrace]
set video_4($j) [new Application/Traffic/videoTrace]
set video_5($j) [new Application/Traffic/videoTrace]
set video_6($j) [new Application/Traffic/videoTrace]
set video_7($j) [new Application/Traffic/videoTrace]
```

```
        set video_8($j) [new Application/Traffic/videoTrace]
        set video_9($j) [new Application/Traffic/videoTrace]
        set video_10($j) [new Application/Traffic/videoTrace]
        $session_0 attach-trace $video_1($j)
        $session_2 attach-trace $video_2($j)
        $session_4 attach-trace $video_3($j)
        $session_6 attach-trace $video_4($j)
        $session_8 attach-trace $video_5($j)
        $session_10 attach-trace $video_6($j)
        $session_12 attach-trace $video_7($j)
        $session_14 attach-trace $video_8($j)
        $session_16 attach-trace $video_9($j)
        $session_18 attach-trace $video_10($j)
        $video_1($j) attach-tracefile $trace_file($j)
        $video_2($j) attach-tracefile $trace_file($j)
        $video_3($j) attach-tracefile $trace_file($j)
        $video_4($j) attach-tracefile $trace_file($j)
        $video_5($j) attach-tracefile $trace_file($j)
        $video_6($j) attach-tracefile $trace_file($j)
        $video_7($j) attach-tracefile $trace_file($j)
        $video_8($j) attach-tracefile $trace_file($j)
        $video_9($j) attach-tracefile $trace_file($j)
        $video_10($j) attach-tracefile $trace_file($j)
}


# END OF VIDEO TRAFFIC

$ns at 0.0 "record_video_bw"
$ns at 0.0 "$session_0 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_1 start"
$ns at 0.0 "$session_1 set_metrics_filename $metricsfile_1"
#-------------------------------------------------------
$ns at 0.0 "$session_2 start" ;#starts the VideoRTCPAgent
```

```
$ns at 0.0 "$session_3 start"
$ns at 0.0 "$session_3 set_metrics_filename $metricsfile_2"
#-------------------------------------------------------
$ns at 0.0 "$session_4 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_5 start"
$ns at 0.0 "$session_5 set_metrics_filename $metricsfile_3"
#-------------------------------------------------------
$ns at 0.0 "$session_6 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_7 start"
$ns at 0.0 "$session_7 set_metrics_filename $metricsfile_4"
#-------------------------------------------------------
$ns at 0.0 "$session_8 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_9 start"
$ns at 0.0 "$session_9 set_metrics_filename $metricsfile_5"
#-------------------------------------------------------
$ns at 0.0 "$session_10 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_11 start"
$ns at 0.0 "$session_11 set_metrics_filename $metricsfile_6"
#-------------------------------------------------------
$ns at 0.0 "$session_12 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_13 start"
$ns at 0.0 "$session_13 set_metrics_filename $metricsfile_7"
#-------------------------------------------------------
$ns at 0.0 "$session_14 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_15 start"
$ns at 0.0 "$session_15 set_metrics_filename $metricsfile_8"
#-------------------------------------------------------
$ns at 0.0 "$session_16 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_17 start"
$ns at 0.0 "$session_17 set_metrics_filename $metricsfile_9"
#-------------------------------------------------------
$ns at 0.0 "$session_18 start" ;#starts the VideoRTCPAgent
$ns at 0.0 "$session_19 start"
$ns at 0.0 "$session_19 set_metrics_filename $metricsfile_10"
```

```
#-------------------------------------------------------
$ns at 0.1 "$session_0 transmit-video"          ;#it prepares the video sender to send -->
the video stream must be initiated seperately
$ns at 0.1 "$session_2 transmit-video"
$ns at 0.1 "$session_4 transmit-video"
$ns at 0.1 "$session_6 transmit-video"
$ns at 0.1 "$session_8 transmit-video"
$ns at 0.1 "$session_10 transmit-video"
$ns at 0.1 "$session_12 transmit-video"
$ns at 0.1 "$session_14 transmit-video"
$ns at 0.1 "$session_16 transmit-video"
$ns at 0.1 "$session_18 transmit-video"
$ns at 0.1 "$video_1(0) start"                ;#video stream initiates
$ns at 0.1 "$video_2(0) start"
$ns at 0.1 "$video_3(0) start"
$ns at 0.1 "$video_4(0) start"
$ns at 0.1 "$video_5(0) start"
$ns at 0.1 "$video_6(0) start"
$ns at 0.1 "$video_7(0) start"
$ns at 0.1 "$video_8(0) start"
$ns at 0.1 "$video_9(0) start"
$ns at 0.1 "$video_10(0) start"

$ns at 20.0 "$session_0 stop"
$ns at 21.0 "$session_1 close_tracefile"
$ns at 21.0 "$session_1 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_2 stop"
$ns at 21.0 "$session_3 close_tracefile"
$ns at 21.0 "$session_3 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_4 stop"
$ns at 21.0 "$session_5 close_tracefile"
$ns at 21.0 "$session_5 close_metricsfile"
```

```
#---------------------------------------
$ns at 20.0 "$session_6 stop"
$ns at 21.0 "$session_7 close_tracefile"
$ns at 21.0 "$session_7 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_8 stop"
$ns at 21.0 "$session_9 close_tracefile"
$ns at 21.0 "$session_9 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_10 stop"
$ns at 21.0 "$session_11 close_tracefile"
$ns at 21.0 "$session_11 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_12 stop"
$ns at 21.0 "$session_13 close_tracefile"
$ns at 21.0 "$session_13 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_14 stop"
$ns at 21.0 "$session_15 close_tracefile"
$ns at 21.0 "$session_15 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_16 stop"
$ns at 21.0 "$session_17 close_tracefile"
$ns at 21.0 "$session_17 close_metricsfile"
#---------------------------------------
$ns at 20.0 "$session_18 stop"
$ns at 21.0 "$session_19 close_tracefile"
$ns at 21.0 "$session_19 close_metricsfile"
#---------------------------------------
$ns at 21.0 "record_loss"
$ns at 21.0 "evaluate_metrics"
$ns at 21.0 "finish"

proc finish {} {
```

```
    global ns f videoBW1 videoBW2 videoBW3  videoBW4 videoBW5 videoBW6 videoBW7 videoBW8 videoBW9 videoBW10
    $ns flush-trace
       close $f
       close $videoBW1
       close $videoBW2
    close $videoBW3
       close $videoBW4
       close $videoBW5
       close $videoBW6
       close $videoBW7
       close $videoBW8
       close $videoBW9
       close $videoBW10
    exec ../../../../xgraph-12.1/xgraph videoBW1.txt videoBW2.txt videoBW3.txt videoBW4.txt videoBW5.txt videoBW6.txt videoBW7.txt videoBW8.txt videoBW9.txt videoBW10.txt &
    exit 0
}

proc record_video_bw {} {
    global session_1 session_3 session_5 session_7 session_9 session_11 session_13 session_15 session_17 session_19 videoBW1 videoBW2 videoBW3 videoBW4 videoBW5 videoBW6 videoBW7 videoBW8 videoBW9 videoBW10
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Set the time after which the procedure should be called again
    set time 0.5
    #How many bytes have been received by the traffic sessions?
    set bw1 [$session_1 set bytes_]
    set bw2 [$session_3 set bytes_]
    set bw3 [$session_5 set bytes_]
    set bw4 [$session_7 set bytes_]
    set bw5 [$session_9 set bytes_]
```

```
        set bw6 [$session_11 set bytes_]
        set bw7 [$session_13 set bytes_]
        set bw8 [$session_15 set bytes_]
        set bw9 [$session_17 set bytes_]
        set bw10 [$session_19 set bytes_]
        #Get the current time
        set now [$ns now]
        #Calculate the bandwidth (in KBit/s) and write it to the files
        puts $videoBW1 "$now  [expr $bw1/$time*8/1000]"
        puts $videoBW2 "$now  [expr $bw2/$time*8/1000]"
        puts $videoBW3 "$now  [expr $bw3/$time*8/1000]"
        puts $videoBW4 "$now  [expr $bw4/$time*8/1000]"
        puts $videoBW5 "$now  [expr $bw5/$time*8/1000]"
        puts $videoBW6 "$now  [expr $bw6/$time*8/1000]"
        puts $videoBW7 "$now  [expr $bw7/$time*8/1000]"
        puts $videoBW8 "$now  [expr $bw8/$time*8/1000]"
        puts $videoBW9 "$now  [expr $bw9/$time*8/1000]"
        puts $videoBW10 "$now  [expr $bw10/$time*8/1000]"
        #Reset the bytes_ values on the traffic sessions
        $session_1 set bytes_ 0
        $session_3 set bytes_ 0
        $session_5 set bytes_ 0
        $session_7 set bytes_ 0
        $session_9 set bytes_ 0
        $session_11 set bytes_ 0
        $session_13 set bytes_ 0
        $session_15 set bytes_ 0
        $session_17 set bytes_ 0
        $session_19 set bytes_ 0
        #Re-schedule the procedure
        $ns at [expr $now+$time] "record_video_bw"
}

proc record_loss {} {
```

```
    global session_0 session_2 session_4 session_6 session_8 session_10 session_12 session_14 session_16 session_18
    set loss_1 [$session_0 set loss_]
    set loss_2 [$session_2 set loss_]
    set loss_3 [$session_4 set loss_]
    set loss_4 [$session_6 set loss_]
    set loss_5 [$session_8 set loss_]
    set loss_6 [$session_10 set loss_]
    set loss_7 [$session_12 set loss_]
    set loss_8 [$session_14 set loss_]
    set loss_9 [$session_16 set loss_]
    set loss_10 [$session_18 set loss_]
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 1: [expr $loss_1 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 2: [expr $loss_2 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 3: [expr $loss_3 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 4: [expr $loss_4 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 5: [expr $loss_5 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 6: [expr $loss_6 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 7: [expr $loss_7 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 8: [expr $loss_8 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 9: [expr $loss_9 * 100]%"
    puts "PACKET LOSS PERCENTAGE FROM VIDEO CLIENT 10: [expr $loss_10 * 100]%"
}
```

```
proc evaluate_metrics {} {
    global metricsfile_1 metricsfile_2 metricsfile_3 metricsfile_4 metricsfile_5
metricsfile_6 metricsfile_7 metricsfile_8 metricsfile_9 metricsfile_10
    set count_1 0
    set mean_1 0.0
    set std_1 0.0
    set original_file_id_1 [open $metricsfile_1 r]
    while {[eof $original_file_id_1] == 0} {

        gets $original_file_id_1 current_line_1
        scan $current_line_1 "%d%f" seq_1 delay_1
        set mean_1 [expr $mean_1 + $delay_1]

        incr count_1
    }
    set mean_1 [expr $mean_1 / $count_1]
    close $original_file_id_1
    set original_file_id_1 [open $metricsfile_1 r]

    while {[eof $original_file_id_1] == 0} {

        gets $original_file_id_1 current_line_1
        scan $current_line_1 "%d%f" seq_1 delay_1
        set std_1 [expr $std_1 + pow([expr $delay_1 - $mean_1],2)]
    }
    set std_1 [expr sqrt([expr $std_1 / $count_1])]
    close $original_file_id_1
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 1: $mean_1 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 1: $std_1 sec(s)"

   set count_2 0
   set mean_2 0.0
   set std_2 0.0
   set original_file_id_2 [open $metricsfile_2 r]
```

```
    while {[eof $original_file_id_2] == 0} {

        gets $original_file_id_2 current_line_2
        scan $current_line_2 "%d%f" seq_2 delay_2
        set mean_2 [expr $mean_2 + $delay_2]

        incr count_2
    }
    set mean_2 [expr $mean_2 / $count_2]
    close $original_file_id_2
    set original_file_id_2 [open $metricsfile_2 r]
    while {[eof $original_file_id_2] == 0} {

        gets $original_file_id_2 current_line_2
        scan $current_line_2 "%d%f" seq_2 delay_2
        set std_2 [expr $std_2 + pow([expr $delay_2 - $mean_2],2)]
    }
    set std_2 [expr sqrt([expr $std_2 / $count_2])]
    close $original_file_id_2
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 2: $mean_2 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 2: $std_2 sec(s)"


set count_3 0
set mean_3 0.0
set std_3 0.0
set original_file_id_3 [open $metricsfile_3 r]
while {[eof $original_file_id_3] == 0} {

   gets $original_file_id_3 current_line_3
   scan $current_line_3 "%d%f" seq_3 delay_3
   set mean_3 [expr $mean_3 + $delay_3]

   incr count_3
```

```
}
set mean_3 [expr $mean_3 / $count_3]
close $original_file_id_3
set original_file_id_3 [open $metricsfile_3 r]
while {[eof $original_file_id_3] == 0} {

   gets $original_file_id_3 current_line_3
   scan $current_line_3 "%d%f" seq_3 delay_3
   set std_3 [expr $std_3 + pow([expr $delay_3 - $mean_3],2)]
}
set std_3 [expr sqrt([expr $std_3 / $count_3])]
close $original_file_id_3
puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 3: $mean_3 sec(s)"
puts "END-TO-END JITTER FROM VIDEO CLIENT 3: $std_3 sec(s)"



set count_4 0
      set mean_4 0.0
      set std_4 0.0
      set original_file_id_4 [open $metricsfile_4 r]
      while {[eof $original_file_id_4] == 0} {

            gets $original_file_id_4 current_line_4
            scan $current_line_4 "%d%f" seq_4 delay_4
            set mean_4 [expr $mean_4 + $delay_4]

            incr count_4
      }
      set mean_4 [expr $mean_4 / $count_4]
      close $original_file_id_4
      set original_file_id_4 [open $metricsfile_4 r]
      while {[eof $original_file_id_4] == 0} {

            gets $original_file_id_4 current_line_4
```

```
        scan $current_line_4 "%d%f" seq_4 delay_4
        set std_4 [expr $std_4 + pow([expr $delay_4 - $mean_4],2)]
    }
    set std_4 [expr sqrt([expr $std_4 / $count_4])]
    close $original_file_id_4
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 4: $mean_4 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 4: $std_4 sec(s)"



set count_5 0
    set mean_5 0.0
    set std_5 0.0
    set original_file_id_5 [open $metricsfile_5 r]
    while {[eof $original_file_id_5] == 0} {

        gets $original_file_id_5 current_line_5
        scan $current_line_5 "%d%f" seq_5 delay_5
        set mean_5 [expr $mean_5 + $delay_5]

        incr count_5
    }
    set mean_5 [expr $mean_5 / $count_5]
    close $original_file_id_5
    set original_file_id_5 [open $metricsfile_5 r]
    while {[eof $original_file_id_5] == 0} {

        gets $original_file_id_5 current_line_5
        scan $current_line_5 "%d%f" seq_5 delay_5
        set std_5 [expr $std_5 + pow([expr $delay_5 - $mean_5],2)]
    }
    set std_5 [expr sqrt([expr $std_5 / $count_5])]
    close $original_file_id_5
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 5: $mean_5 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 5: $std_5 sec(s)"
```

```
set count_6 0
    set mean_6 0.0
    set std_6 0.0
    set original_file_id_6 [open $metricsfile_6 r]
    while {[eof $original_file_id_6] == 0} {

        gets $original_file_id_6 current_line_6
        scan $current_line_6 "%d%f" seq_6 delay_6
        set mean_6 [expr $mean_6 + $delay_6]

        incr count_6
    }
    set mean_6 [expr $mean_6 / $count_6]
    close $original_file_id_6
    set original_file_id_6 [open $metricsfile_6 r]
    while {[eof $original_file_id_6] == 0} {

        gets $original_file_id_6 current_line_6
        scan $current_line_6 "%d%f" seq_6 delay_6
        set std_6 [expr $std_6 + pow([expr $delay_6 - $mean_6],2)]
    }
    set std_6 [expr sqrt([expr $std_6 / $count_6])]
    close $original_file_id_6
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 6: $mean_6 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 6: $std_6 sec(s)"


set count_7 0
    set mean_7 0.0
    set std_7 0.0
    set original_file_id_7 [open $metricsfile_7 r]
    while {[eof $original_file_id_7] == 0} {
```

```
        gets $original_file_id_7 current_line_7
        scan $current_line_7 "%d%f" seq_7 delay_7
        set mean_7 [expr $mean_7 + $delay_7]

        incr count_7
    }
    set mean_7 [expr $mean_7 / $count_7]
    close $original_file_id_7
    set original_file_id_7 [open $metricsfile_7 r]
    while {[eof $original_file_id_7] == 0} {

        gets $original_file_id_7 current_line_7
        scan $current_line_7 "%d%f" seq_7 delay_7
        set std_7 [expr $std_7 + pow([expr $delay_7 - $mean_7],2)]
    }
    set std_7 [expr sqrt([expr $std_7 / $count_7])]
    close $original_file_id_7
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 7: $mean_7 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 7: $std_7 sec(s)"



set count_8 0
    set mean_8 0.0
    set std_8 0.0
    set original_file_id_8 [open $metricsfile_8 r]
    while {[eof $original_file_id_8] == 0} {

        gets $original_file_id_8 current_line_8
        scan $current_line_8 "%d%f" seq_8 delay_8
        set mean_8 [expr $mean_8 + $delay_8]

        incr count_8
    }
```

```
    set mean_8 [expr $mean_8 / $count_8]
    close $original_file_id_8
    set original_file_id_8 [open $metricsfile_8 r]
    while {[eof $original_file_id_8] == 0} {

        gets $original_file_id_8 current_line_8
        scan $current_line_8 "%d%f" seq_8 delay_8
        set std_8 [expr $std_8 + pow([expr $delay_8 - $mean_8],2)]
    }
    set std_8 [expr sqrt([expr $std_8 / $count_8])]
    close $original_file_id_8
    puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 8: $mean_8 sec(s)"
    puts "END-TO-END JITTER FROM VIDEO CLIENT 8: $std_8 sec(s)"


set count_9 0
set mean_9 0.0
set std_9 0.0
set original_file_id_9 [open $metricsfile_9 r]
while {[eof $original_file_id_9] == 0} {

   gets $original_file_id_9 current_line_9
   scan $current_line_9 "%d%f" seq_9 delay_9
   set mean_9 [expr $mean_9 + $delay_9]

   incr count_9
}
set mean_9 [expr $mean_9 / $count_9]
close $original_file_id_9
set original_file_id_9 [open $metricsfile_9 r]
while {[eof $original_file_id_9] == 0} {

   gets $original_file_id_9 current_line_9
   scan $current_line_9 "%d%f" seq_9 delay_9
```

```
    set std_9 [expr $std_9 + pow([expr $delay_9 - $mean_9],2)]
}
set std_9 [expr sqrt([expr $std_9 / $count_9])]
close $original_file_id_9
puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 9: $mean_9 sec(s)"
puts "END-TO-END JITTER FROM VIDEO CLIENT 9: $std_9 sec(s)"



set count_10 0
set mean_10 0.0
set std_10 0.0
set original_file_id_10 [open $metricsfile_10 r]
while {[eof $original_file_id_10] == 0} {

    gets $original_file_id_10 current_line_10
    scan $current_line_10 "%d%f" seq_10 delay_10
    set mean_10 [expr $mean_10 + $delay_10]

    incr count_10
}
set mean_10 [expr $mean_10 / $count_10]
close $original_file_id_10
set original_file_id_10 [open $metricsfile_10 r]
while {[eof $original_file_id_10] == 0} {

    gets $original_file_id_10 current_line_10
    scan $current_line_10 "%d%f" seq_10 delay_10
    set std_10 [expr $std_10 + pow([expr $delay_10 - $mean_10],2)]
}
set std_10 [expr sqrt([expr $std_10 / $count_10])]
close $original_file_id_10
puts "MEAN END-TO-END DELAY FROM VIDEO CLIENT 10: $mean_10 sec(s)"
puts "END-TO-END JITTER FROM VIDEO CLIENT 10: $std_10 sec(s)"
}
```

```
$ns run
```

A.1.3 Σενάριο 6 - Simple video μετάδοση με 10 multicast users (πρωτόκολλο Source Specific Multicast)

```
set ns [new Simulator]
$ns multicast

set f [open out_ssm_simple_ten.txt w]
set videoBW1 [open videoBW1_ssm.txt w]
set videoBW2 [open videoBW2_ssm.txt w]
set videoBW3 [open videoBW3_ssm.txt w]
set videoBW4 [open videoBW4_ssm.txt w]
set videoBW5 [open videoBW5_ssm.txt w]
set videoBW6 [open videoBW6_ssm.txt w]
set videoBW7 [open videoBW7_ssm.txt w]
set videoBW8 [open videoBW8_ssm.txt w]
set videoBW9 [open videoBW9_ssm.txt w]
set videoBW10 [open videoBW10_ssm.txt w]



$ns trace-all $f
$ns namtrace-all [open out_ssm_simple_ten.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# allocate a multicast address;
set group [Node allocaddr]

# nod is the number of nodes
```

```
set num_wired_nodes 17

# create multicast capable nodes;
for {set i 1} {$i <= $num_wired_nodes} {incr i} {
   set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(5) $n(7) 2Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 2Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 2Mb 10ms DropTail
$ns duplex-link $n(6) $n(9) 2Mb 10ms DropTail

$ns duplex-link $n(10) $n(3) 2Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(12) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(13) $n(6) 2Mb 10ms DropTail
$ns duplex-link $n(14) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(15) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(16) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(17) $n(7) 2Mb 10ms DropTail



# configure multicast protocol;
```

```
set mproto SSMMcast

# all nodes will contain multicast protocol agents;
set mrthandle [$ns mrtproto $mproto]

set max_fragmented_size 1000

set udp1 [new Agent/myUDP]
$ns attach-agent $n(1) $udp1
$udp1 set packetSize_ 1028
$udp1 set_filename sd_be
$udp1 set dst_addr_ $group

# ---------------------
# SIMPLE VIDEO TRAFFIC |
# ---------------------
set original_file_name st
set trace_file_name video1.dat
set original_file_id [open $original_file_name r]
set trace_file_id [open $trace_file_name w]

set frame_count 0
set last_time 0

while {[eof $original_file_id] == 0} {
   gets $original_file_id current_line

   scan $current_line "%d%s%d%s%s%s%d%s" no_ frametype_ length_ tmp1_ tmp2_
tmp3_ tmp4_ tmp5_
   #puts "$no_ $frametype_ $length_ $tmp1_ $tmp2_ $tmp3_ $tmp4_ $tmp5_"

   # 30 frames/sec. if one want to generate 30 frames/sec, one can use set time [expr
1000*1000/30]
   set time [expr 1000 * 1000/25]
```

```
    if { $frametype_ == "I" } {
        set type_v 1
    }

    if { $frametype_ == "P" } {
        set type_v 2
    }

    if { $frametype_ == "B" } {
        set type_v 3
    }

    if { $frametype_ == "H" } {
        set type_v 1
    }

    puts  $trace_file_id "$time $length_ $type_v $max_fragmented_size"
    incr frame_count
}

close $original_file_id
close $trace_file_id
set end_sim_time [expr 1.0 * 1000/25 * ($frame_count + 1)  / 1000]
puts "$end_sim_time"

set trace_file [new Tracefile]
$trace_file filename $trace_file_name
set video1 [new Application/Traffic/myTrace2]
$video1 attach-agent $udp1
$video1 attach-tracefile $trace_file
# END OF VIDEO TRAFFIC

# create receiver agents
```

```
set rcvr1 [new Agent/myUdpSink2]
$rcvr1 set_trace_filename rd_be1

set rcvr2 [new Agent/myUdpSink2]
$rcvr2 set_trace_filename rd_be2

set rcvr3 [new Agent/myUdpSink2]
$rcvr3 set_trace_filename rd_be3

set rcvr4 [new Agent/myUdpSink2]
$rcvr4 set_trace_filename rd_be4

set rcvr5 [new Agent/myUdpSink2]
$rcvr5 set_trace_filename rd_be5

set rcvr6 [new Agent/myUdpSink2]
$rcvr6 set_trace_filename rd_be6

set rcvr7 [new Agent/myUdpSink2]
$rcvr7 set_trace_filename rd_be7

set rcvr8 [new Agent/myUdpSink2]
$rcvr8 set_trace_filename rd_be8

set rcvr9 [new Agent/myUdpSink2]
$rcvr9 set_trace_filename rd_be9

set rcvr10 [new Agent/myUdpSink2]
$rcvr10 set_trace_filename rd_be10



$ns attach-agent $n(8) $rcvr1
$ns attach-agent $n(9) $rcvr2
```

```
$ns attach-agent $n(10) $rcvr3
$ns attach-agent $n(11) $rcvr4
$ns attach-agent $n(12) $rcvr5
$ns attach-agent $n(13) $rcvr6
$ns attach-agent $n(14) $rcvr7
$ns attach-agent $n(15) $rcvr8
$ns attach-agent $n(16) $rcvr9
$ns attach-agent $n(17) $rcvr10


# joining and leaving the group;
$ns at  0.1 "$video1 start"
$ns at  0.2 "$n(8) join-group $rcvr1 $group $n(1)"
$ns at  0.3 "$n(9) join-group $rcvr2 $group $n(1)"
$ns at  0.3 "$n(10) join-group $rcvr3 $group $n(1)"
$ns at  0.3 "$n(11) join-group $rcvr4 $group $n(1)"
$ns at  0.3 "$n(12) join-group $rcvr5 $group $n(1)"
$ns at  0.3 "$n(13) join-group $rcvr6 $group $n(1)"
$ns at  0.3 "$n(14) join-group $rcvr7 $group $n(1)"
$ns at  0.3 "$n(15) join-group $rcvr8 $group $n(1)"
$ns at  0.3 "$n(16) join-group $rcvr9 $group $n(1)"
$ns at  0.3 "$n(17) join-group $rcvr10 $group $n(1)"


$ns at  20.5 "$n(9) leave-group $rcvr2 $group $n(1)"
$ns at  20.5 "$n(10) leave-group $rcvr2 $group $n(1)"


$ns at 21.0 "$video1 stop"
$ns at 21.0 "$rcvr1 closefile"
$ns at 21.0 "$rcvr2 closefile"
$ns at 21.0 "$rcvr3 closefile"
$ns at 21.0 "$rcvr4 closefile"
$ns at 21.0 "$rcvr5 closefile"
```

```
$ns at 21.0 "$rcvr6 closefile"
$ns at 21.0 "$rcvr7 closefile"
$ns at 21.0 "$rcvr8 closefile"
$ns at 21.0 "$rcvr9 closefile"
$ns at 21.0 "$rcvr10 closefile"
$ns at 21.0 "finish"

proc finish {} {
    global ns f videoBW1 videoBW2
    $ns flush-trace
    close $f
    close $videoBW1
    close $videoBW2
    exit 0
}
 $ns run
.
.
```