ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συνδιάσκεψη πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας και αναμετάδοση πακέτων βίντεο στο λογισμικό BareSIP

Απόστολος Η. Μόδας

Επιβλέπων : **Αλέξανδρος Ελευθεριάδης,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2015

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συνδιάσκεψη πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας και αναμετάδοση πακέτων βίντεο στο λογισμικό BareSIP

Απόστολος Η. Μόδας

Α.Μ.: 1115200900101

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Αλέξανδρος Ελευθεριάδης,** Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Παρουσιάζουμε το σχεδιασμό, την υλοποίηση και την ενσωμάτωση ενός πολυνηματικού μηχανισμού συνδιάσκεψης πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας, καθώς και ενός μηχανισμού αναμετάδοσης RTP πακέτων βίντεο, στο λογισμικό ανοιχτού κώδικα BareSIP. Επειδή η αρχική σχεδίαση του BareSIP δεν υποστηρίζει την ταυτόχρονη συμμετοχή πολλαπλών χρηστών, για τη δημιουργία του μηχανισμού συνδιάσκεψης χρησιμοποιούμε αρχικά έναν εξυπηρετητή. Ο εξυπηρετητής αυτός, πολυπλέκει τις ροές μέσων των διαφόρων χρηστών με βάση το πεδίο SSRC της επικεφαλίδας των RTP πακέτων και στέλνει μια ροή για κάθε τύπο μέσου στον BareSIP πελάτη. Στον BareSIP πελάτη, προκειμένου να πραγματοποιείται συνδιάσκεψη με υψηλή απόδοση, αναπτύξαμε έναν πολυνηματικό μηχανισμό ο οποίος αποπολυπλέκει την εισερχόμενη ροή μέσων βάσει του πεδίου SSRC του κάθε συμμετέχοντα και αναθέτει την εξυπηρέτηση του καθενός σε αντίστοιχα νήματα που τρέχουν παράλληλα. Ο μηχανισμός αυτός σχεδιάστηκε με τρόπο τέτοιο ώστε να απαιτούνται μόνο δύο θύρες για τις εισερχόμενες ροές μέσων, ενώ για μια ταυτόχρονη συνύπαρξη *N* χρηστών να απαιτείται μόνο ένα στιγμιότυπο του κωδικοποιητή και *N-1* στιγμιότυπα του αποκωδικοποιητή. Επιπλέον, χρησιμοποιώντας την επέκταση κλιμακωτής κωδικοποίησης βίντεο (Scalable Video Coding – SVC) του προτύπου συμπίεσης H.264/AVC, η οποία προσφέρει πολλαπλά επίπεδα πιστότητας μέσω μιας πυραμιδικής ιεραρχίας (ροές δεδομένων επιπέδου βάσης – ροές δεδομένων βελτιωτικών επιπέδων), παρουσιάζουμε ένα μοντέλο επικοινωνίας υψηλής ανθεκτικότητας ανάμεσα στον πομπό και το δέκτη, χρησιμοποιώντας αναμετάδοση των RTP πακέτων βίντεο. Το μοντέλο αυτό φροντίζει για την άμεση αποκατάσταση απολεσθέντων RTP πακέτων βίντεο επιπέδου βάσης, ώστε η ρουτίνα αποκωδικοποίησης των πακέτων βίντεο επιπέδου βάσης να πραγματοποιείται χωρίς διακοπές και καθυστερήσεις. Δημιουργούμε έναν πομπό ο οποίος διατηρεί διπλότυπα RTP πακέτων βίντεο επιπέδου βάσης, προκειμένου να μπορεί να αναμεταδώσει εκείνα τα οποία ζητάει ο δέκτης. Στην πλευρά του δέκτη, υλοποιούμε έναν αλγόριθμο ο οποίος εντοπίζει τα απολεσθέντα πακέτα βάσης και στέλνει αιτήματα αναμετάδοσης μέσω RTCP NACK πακέτων, προκειμένου να ανακτήσει άμεσα όλη την πληροφορία του βασικού επιπέδου και να διατηρήσει την ακεραιότητα της ροής δεδομένων του. Ο αλγόριθμος αυτός, τέλος, φροντίζει να λάβει τις κατάλληλες αποφάσεις ανάλογα με τις επιπτώσεις που θα είχαν οι απώλειες στο σύστημα. Ως προς τις μετρήσεις που πραγματοποιήσαμε, για το μηχανισμό συνδιάσκεψης μελετήσαμε την καθυστέρηση που προκύπτει σε επίπεδο μετάδοσης, κωδικοποίησης και αποκωδικοποίησης ήχου και βίντεο, καθώς αυξάνει ο αριθμός των συμμετεχόντων, αλλά και τις διαφορές που παρουσιάζονται αν τροποποιήσουμε το packet time και την καθυστέρηση στους jitter buffers. Από τις μετρήσεις αυτές δείξαμε ότι η συνολική καθυστέρηση του ήχου παραμένει σχεδόν σταθερή, ανεξάρτητα από την αύξηση των συμμετεχόντων, ενώ αντίστοιχα για το βίντεο παρατηρούμε μια μικρή αύξηση της τάξης των 2-4 msec. Τέλος, για το μηχανισμό αναμετάδοσης πακέτων βίντεο, παρουσιάζουμε και συγκρίνουμε το ποσοστό των ωφέλιμων frames (effective frame rate) και των ωφέλιμων bytes στην έξοδο, τόσο με τη χρήση του μηχανισμού όσο και χωρίς αυτόν και παρατηρούμε ότι η απόδοση με το μηχανισμό αναμετάδοσης είναι σαφώς καλύτερη, ειδικά σε περιπτώσεις υψηλών απωλειών, όπου φαίνεται να είναι 265% καλύτερη από αυτή χωρίς το μηχανισμό.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία Εικόνας, Μετάδοση Ψηφιακού Βίντεο

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: επεξεργασία βίντεο, μετάδοση βίντεο, H.264, σύστημα συνδιάσκεψης με χρήση βίντεο, επικοινωνία μέσω ψηφιακού βίντεο, RTP, BareSIP, SVC, αναμετάδοση βίντεο

ABSTRACT

We present the design, the implementation and the integration of a multi-threaded mechanism for multi-user conference using SRC multiplexing and an RTP video packet retransmission mechanism, in open source software BareSIP. Due to the fact that the default design of BareSIP does not support the simultaneous participation of multiple users, in order to set up the conference mechanism, we initially use a server. This server multiplexes the media streams of the different users based on the SSRC field of the RTP header of the packets and sends one stream for each media type to every BareSIP client. In BareSIP client, in order the conference to take place with high efficiency, we developed a multi-threaded mechanism that demultiplexes the incoming media stream based on the SSRC field of each participant and assigns the service of each one to corresponding threads that are running in parallel. This mechanism was designed in such a way as to require only two ports for the incoming media streams and, for the simultaneous coexistence of *N* users, to require only one encoding and *N-1* decoding instances. Additionally, using the Scalable Video Coding extension of H.264/AVC video compression standard, which offers multiple levels of fidelity through a pyramidal hierarchy (base layer data streams – enhancement layer data streams), we present a communication model of high robustness between the transmitter and the receiver, using RTP video packet retransmission. This model ensures for the immediate restoration of lost base layer RTP video packets, so that the decoding routine of base layer video packets will take place without interruptions and delays. We create a transmitter that maintains duplicates of base layer RTP video packets, in order to be able to retransmit those that are requested by the receiver. At the receiver's side, we implement an algorithm that detects the base layer packets that were lost and requests for their retransmission, using RTCP NACK packets, in order to instantly retrieve all the information of the base layer and to maintain the integrity of its data flow. This algorithm, finally, ensures to take the appropriate decisions depending on the impact that the losses will have in the system. As for the results obtained by the measurements we made, for the conference mechanism we present the delay that results in audio/video relaying, coding and decoding level, as the number of participants is increasing, but also the differences that appear by modifying the packet time and the jitter buffer delay. From these measurements we showed that the total sound delay remains almost constant, regardless of the increase of the number of the participants, while for the total video delay there is a small increase of 2-4 msec. Finally, for the video packet retransmission mechanism, we present and compare the percentage of the useful frames (effective frame rate) and bytes (effective byte rate) that appear in the output, when using and when not using the mechanism, and we show that the performance in the case we are using the retransmission mechanism is clearly better, especially in cases of high losses, which seems to be 265% better than the performance in the case we are not using the retransmission mechanism.

SUBJECT AREA: Image Processing, Digital Video Transmission

KEYWORDS: video processing, video transmission, H.264, video conference system, digital video communication, RTP, BareSIP, SVC, video retransmission

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση αυτής της πτυχιακής υλοποιήθηκε με την υποστήριξη ενός αριθμού ανθρώπων, στους οποίους θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου. Πρώτα απ' όλους, θα ήθελα να ευχαριστήσω των επιβλέποντα της πτυχιακής μου εργασίας, Αναπληρωτή Καθηγητή, κ. Αλέξανδρο Ελευθεριάδη για την εμπιστοσύνη που μου έδειξε, την πολύτιμη βοήθεια και καθοδήγησή του, την παροχή επιστημονικών γνώσεων, καθώς και τις πολύτιμες ώρες που αφιέρωσε καθ' όλη τη διάρκεια της δουλειάς μου. Επίσης, οφείλω ένα πολύ μεγάλο ευχαριστώ στους φίλους και συνεργάτες Ευάγγελο Αλεξίου (Διδακτορικός Φοιτητής) και Λεντιόν Σωτήρη (Επιστημονικός Ερευνητής) για την υπομονή που έδειξαν και την ανεκτίμητη βοήθεια που μου προσέφεραν επί ενάμιση χρόνο για την έγκαιρη και ομαλή ολοκλήρωση της πτυχιακής μου. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου Εύα και Ηλία, τον αδερφό μου Βασίλη, τους φίλους(ες) Μυρτώ, Βλάση, Νίκο, Δημήτρη, Βικτώρια, Θάνο, Βασίλη, Χρήστο και Χάρη, αλλά και όλα τα κοντινά και αγαπημένα μου πρόσωπα, για την ολόψυχη αγάπη και υποστήριξη που έδωσαν όλα αυτά τα χρόνια.

Αποστόλης,

Νοέμβριος 2015

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στην Αθήνα από το Μάιο του 2014 μέχρι και τον Νοέμβριο του 2015. Αποτελεί αναπόσπαστο κομμάτι για την απόκτηση του πτυχίου και διεξήχθη κατά το τελευταίο έτος της φοίτησής μου ως προπτυχιακός φοιτητής στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Η μελέτη και οι μηχανισμοί που υλοποιήθηκαν στην εργασία αυτή, αξιοποιήθηκαν και ενσωματώθηκαν στα ερευνητικά προγράμματα MusiNet: Comprehensive design and implementation of a networked music performance system και SeNSE: Wireless sensor networks for dense sampling and replaying events. Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον επιβλέποντα της πτυχιακής μου, Αναπληρωτή Καθηγητή, κ. Αλέξανδρο Ελευθεριάδη για την ευκαιρία που μου έδωσε να συμμετάσχω ενεργά στα δύο αυτά προγράμματα, καθώς αποτέλεσαν μια πολύ σημαντική πηγή γνώσης και εμπειρίας για μένα.

1. ΕΙΣΑΓΩΓΗ

Η μετάδοση ήχου και βίντεο αποτελεί ένα μεγάλο και αναπόσπαστο κομμάτι της καθημερινότητας του ανθρώπου, καθώς προσφέρει διαφόρων ειδών υπηρεσίες σε πολλούς τομείς όπως στην επικοινωνία, στην ψυχαγωγία, στην ενημέρωση, στην ιατρική κ.ά. Όπως μπορεί να γίνει αντιληπτό, υπάρχουν διάφοροι τρόποι με τους οποίους μπορούμε να μεταδώσουμε δεδομένα μέσων, καθένας εκ των οποίων απαιτεί διαφορετική διαχείριση. Η θεματική περιοχή στην οποία επικεντρωνόμαστε είναι η μετάδοση ήχου και βίντεο σε πραγματικό χρόνου μέσω Διαδικτύου. Στην ενότητα αυτή θα αναφερθούμε σε γενικά χαρακτηριστικά που αφορούν τη μετάδοση δεδομένων μέσων πραγματικού χρόνου μέσω του Διαδικτύου. Επίσης, θα θέσουμε ζητήματα απόδοσης και ποιότητας και θα αναφερθούμε σε ένα σύστημα διαδικτυακής μουσικής επίδοσης, το MusiNet, στο οποίο ενσωματώθηκαν οι μηχανισμοί που υλοποιήθηκαν στα πλαίσια της πτυχιακής εργασίας.

1.1 Μετάδοση ήχου και βίντεο πραγματικού χρόνου μέσω του Διαδικτύου

Για τη μετάδοση ήχου και βίντεο πραγματικού χρόνου μέσω του Διαδικτύου υπάρχουν πολύ υψηλές απαιτήσεις ως προς την καθυστέρηση και την ποιότητα στη λήψη, αλλά εμφανίζονται αρκετές δυσκολίες για την ικανοποίησή τους. Παρόλο που το Διαδίκτυο προσφέρει τη δυνατότητα να μεταφέρουμε δεδομένα σε μακρινές αποστάσεις με μεγάλη ταχύτητα, παρουσιάζει πολλές τυχαίες διακυμάνσεις οι οποίες έχουν άμεση επίπτωση στην καθυστέρηση και στην ποιότητα των δεδομένων που μεταφέρονται. Για το λόγο αυτό απαιτείται ο προσδιορισμός κατάλληλων τεχνικών, προκειμένου να μειωθεί τόσο η καθυστέρηση όσο και η απώλεια πακέτων.

1.1.1 Πρωτόκολλα

Για τη Διαδικτυακή μετάδοση μέσων, χρησιμοποιούνται διάφορα πρωτόκολλα, κάθε ένα από τα οποία χαρακτηρίζεται από το μοντέλο αναφοράς OSI. Για τη μεταφορά δεδομένων βίντεο, το πρωτόκολλο που χρησιμοποιείται κυρίως είναι το UDP (User Datagram Protocol), το οποίο είναι επιπέδου μεταφοράς και «τρέχει» πάνω από το πρωτόκολλο IP (Internet Protocol), το οποίο είναι επιπέδου Διαδικτύου. Στην περίπτωση που μεταδίδουμε βίντεο πραγματικού χρόνου, τότε χρησιμοποιούμε το πρωτόκολλο RTP (Real-time Transport Protocol), το οποίο είναι επιπέδου εφαρμογής και «τρέχει» πάνω από το UDP. Το RTP παρέχει όλους εκείνους τους απαραίτητους μηχανισμούς για τη διαχείριση των δεδομένων μέσων πραγματικού χρόνου, συμπεριλαμβανομένων και των μηχανισμών ελέγχου, οι οποίοι καθορίζονται από το πρωτόκολλο RTCP (Real-time Transport Control Protocol). Στα δικτυακά πρωτόκολλα επικοινωνίας σε πραγματικό χρόνο θα αναφερθούμε αναλυτικότερα στην Ενότητα 2.

1.1.2 Απαιτήσεις ως προς την απόδοση

Στη μελέτη της μετάδοσης μέσων πραγματικού χρόνου πρέπει να έχουμε ως βασικό κριτήριο την απόδοση και την ποιότητα που θέλουμε να πετύχουμε, ώστε ο χρήστης να βιώνει τη μέγιστη εμπειρία. Συνεπώς, πρέπει να επικεντρωνόμαστε σε παράγοντες που συμβάλλουν στη μείωση της καθυστέρησης και των απωλειών δεδομένων, καθώς και στην παροχή βέλτιστης ποιότητας και ανάλυσης. Σχετικά με την καθυστέρηση, ένα υψηλής ταχύτητας δίκτυο είναι απαραίτητο αλλά όχι αρκετό. Χρειάζονται, επιπλέον,

τεχνικές κωδικοποίησης/αποκωδικοποίησης οι οποίες θα μειώνουν την πολυπλοκότητα επεξεργασίας και θα συμπιέζουν την πληροφορία με τρόπο τέτοιο, ώστε να μεταδίδουμε το μικρότερο δυνατό όγκο δεδομένων χωρίς να αλλοιώνεται το περιεχόμενο. Έπειτα, όσον αφορά την απώλεια πακέτων, απαιτούνται μηχανισμοί που θα φροντίζουν την άμεση αποκατάσταση των απολεσθέντων πακέτων, ώστε να αποφευχθεί η παραμόρφωση στη λήψη λόγω απωλειών. Τέλος, η ύπαρξη κατάλληλων κωδικοποιητών/αποκωδικοποιητών χρειάζονται όχι μόνο για τη μείωση της καθυστέρησης, αλλά και για την επίτευξη υψηλής ποιότητας και ανάλυσης.

1.2 MusiNet: ένα Σύστημα Διαδικτυακής Μουσικής Εκτέλεσης

Το MusiNet [1] πρόκειται για ένα σύστημα μουσικής εκτέλεσης σε πραγματικό χρόνο μέσω διαδικτύου (Networked Music Performance – NMP) [2]. Απαρτίζεται από ορισμένα επιμέρους υποσυστήματα τα οποία αποτελούν κομμάτι μελέτης της πτυχιακής εργασίας. Τα σημεία στα οποία εστιάζουμε είναι η ταυτόχρονη ύπαρξη τουλάχιστον δύο πολυνηματικών κόμβων/χρηστών, η ύπαρξη ενός πολυνηματικού εξυπηρετητή που αναμεταδίδει τα πακέτα προς τους κόμβους/χρήστες και η ενσωμάτωση ενός μηχανισμού αναμετάδοσης RTP πακέτων βίντεο. Τα συγκεκριμένα αυτά υποσυστήματα, καθώς και το μεγαλύτερο κομμάτι του MusiNet, υλοποιήθηκαν στο λογισμικό ανοιχτού κώδικα BareSIP.

1.2.1 Στόχοι και απαιτήσεις

Στόχος του συστήματος MusiNet είναι επίτευξη μετάδοσης ήχου και βίντεο σε υψηλή ποιότητα/πιστότητα και με πολύ μικρή καθυστέρηση, σε βαθμό τέτοιο, ώστε δεκάδες μουσικοί να μπορούν να συγχρονιστούν για την πραγματοποίηση μιας μουσικής επίδοσης. Για την εκπλήρωση του στόχου αυτού απαιτείται η ύπαρξη κατάλληλης τοπολογίας δικτύου (εξυπηρετητής, κόμβοι/χρήστες, περιφερειακές συσκευές) για τη μείωση της καθυστέρησης μετάδοσης, καθώς και κατάλληλων αλγορίθμων κωδικοποίησης/αποκωδικοποίησης ήχου και βίντεο.

1.2.2 Το πρότυπο συμπίεσης βίντεο H.264/AVC

Το σύστημα MusiNet χρησιμοποιεί το πρότυπο συμπίεσης βίντεο H.264/AVC για την κωδικοποίηση/αποκωδικοποίηση των δεδομένων βίντεο. Το πρότυπο αυτό ισχυροποιείται ακόμα περισσότερο με τη χρήση της επέκτασης SVC. Η επέκταση αυτή προσφέρει διάφορα επίπεδα πιστότητας μέσω μιας πυραμιδικής ιεραρχίας, ώστε να μπορεί να γίνει άμεση προσαρμογή του σήματος στις συνθήκες του συστήματος, χωρίς ή με ελάχιστη επεξεργασία. Περισσότερα για το πρότυπο αυτό θα αναλύσουμε στην Ενότητα 5.

1.2.3 Ο εξυπηρετητής πολλαπλών χρηστών και η χρήση SSRC πολυπλεξίας

Όπως αναφέραμε, στα διάφορα υποσυστήματα του MusiNet συγκαταλέγεται ένας πολυνηματικός εξυπηρετητής και τουλάχιστον δύο πολυνηματικοί κόμβοι/χρήστες. Ο εξυπηρετητής έχει την ιδιότητα ότι δεν επεξεργάζεται καθόλου τα δεδομένα των πακέτων που φτάνουν σε αυτόν, παρά μόνο τα αντιγράφει και τα μεταβιβάζει στους συμμετέχοντες της συνόδου (relay server). Με τον τρόπο αυτό δεν προσδίδεται

επιπλέον καθυστέρηση λόγω επεξεργασίας και το βάρος της επεξεργασίας αναθέτεται εξ' ολοκλήρου στους τερματικούς κόμβους/χρήστες. Επίσης, για να ικανοποιηθούν οι ανάγκες του συστήματος ως προς την ταυτόχρονη συμμετοχή πολλαπλών χρηστών στην ίδια σύνοδο, πολυπλέκει τις διάφορες ροές μέσων, βάσει του πεδίου Synchronization Source (SSRC) της επικεφαλίδας RTP για τον κάθε χρήστη, και τις μεταβιβάζει στους αντίστοιχους συμμετέχοντες της συνόδου. Το γεγονός αυτό απαιτεί την ύπαρξη τερματικών κόμβων/χρηστών, οι οποίοι θα είναι σε θέση να μπορούν να αποπολυπλέξουν τις διαφορετικές αυτές ροές, προκειμένου να ξεχωρίσουν την πληροφορία που δέχονται από κάθε χρήστη. Περισσότερα για την τεχνική αυτή θα αναλύσουμε στην Ενότητα 4.

2. ΔΙΚΤΥΑΚΑ ΠΡΩΤΟΚΟΛΛΑ ΕΠΙΚΟΙΝΩΝΙΑΣ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ

Για τη Διαδικτυακή μετάδοση μέσων, χρησιμοποιούνται διάφορα πρωτόκολλα, κάθε ένα από τα οποία χαρακτηρίζεται από το μοντέλο αναφοράς OSI. Στο επίπεδο δικτύου, χρησιμοποιείται το πρωτόκολλο IP, το οποίο δεν παρέχει καμία εγγύηση για την παράδοση των πακέτων δεδομένων, καθώς αυτά μπορεί να φτάσουν καθυστερημένα, σε διαφορετική σειρά από εκείνη που μεταδόθηκαν ή να μη φτάσουν καθόλου (να χαθούν στο δίκτυο). Στο επίπεδο μεταφοράς, χρησιμοποιείται κυρίως το πρωτόκολλο UDP [3], το οποίο «τρέχει» πάνω από το IP, δεν εγγυάται τη μετάδοση των πακέτων στον προορισμό του αλλά παρέχει πιο γρήγορη μετάδοση των πακέτων δεδομένων και συνεπώς είναι καταλληλότερο για μετάδοση δεδομένων μέσων, που απαιτείται σταθερή γρήγορη ροή. Στην ενότητα αυτή θα ασχοληθούμε με τα πρωτόκολλα RTP (Real-time Transport Protocol), RTCP (Real-time Transport Control Protocol), SIP (Session Initiation Protocol) και SDP (Session Description Protocol) τα οποία «τρέχουν» πάνω στο UDP και παρέχουν τους κατάλληλους μηχανισμούς για τη δημιουργία και την επεξεργασία συνόδων καθώς και για τη μεταφορά, τη διαχείριση και τον έλεγχο των ροών δεδομένων.

2.1 Το πρωτόκολλο RTP

Όταν μεταδίδουμε δεδομένα μέσων πραγματικού χρόνου είναι απαραίτητη η ύπαρξη ενός πρωτοκόλλου επιπέδου εφαρμογής για την κατάλληλη μετάδοση και διαχείριση των δεδομένων που διακινούνται. Ένα τέτοιο πρωτόκολλο είναι το RTP όπως αυτό ορίζεται από το RFC 3550 [4], τα συσχετιζόμενα με αυτό προφίλ (profiles) και τις μορφές ωφέλιμου φορτίου (payload formats).

Το RTP παρέχει μηχανισμούς για την εξομάλυνση της διακύμανσης της καθυστέρησης (jitter), για την ανίχνευση δεδομένων που φτάνουν σε διαφορετική σειρά από εκείνη που μεταδόθηκαν, για την αποκατάσταση του χρονισμού (time recovering), για την ανίχνευση και τη διόρθωση απωλειών, για τον προσδιορισμό του payload και της εισερχόμενης πηγής, για το συγχρονισμό των μέσων κ.ά.

Ως προς τη διαδικασία μετάδοσης, στη γενική μορφή, η πλευρά του αποστολέα ενθυλακώνει ένα τμήμα δεδομένων μέσων (payload data) μέσα σ' ένα πακέτο του RTP, κατόπιν ενθυλακώνει αυτό το πακέτο σε ένα τμήμα του UDP και τέλος παραδίδει αυτό το τμήμα στο IP. Η πλευρά του παραλήπτη εξάγει το πακέτο του RTP από το τμήμα του UDP και κατόπιν εξάγει τα δεδομένα μέσων από το πακέτο RTP και τα περνά στην εφαρμογή αναπαραγωγής για αποκωδικοποίηση και αναπαραγωγή.

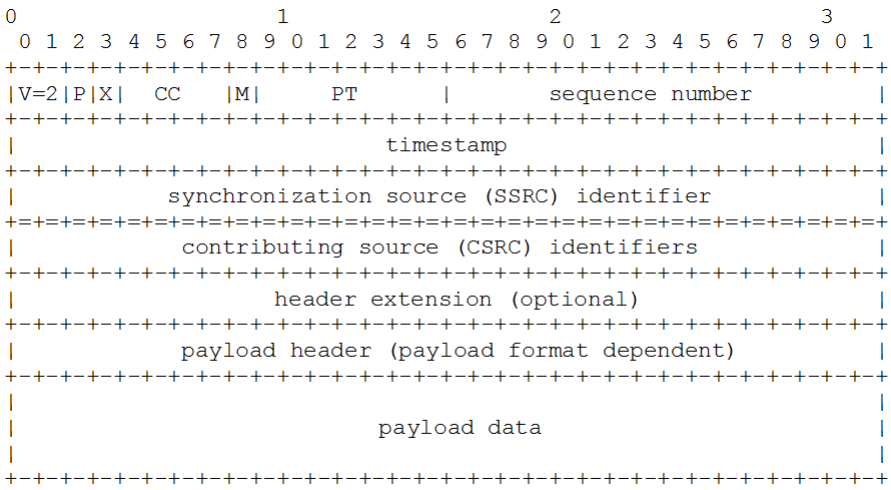
2.1.1 Το RTP πακέτο

Όπως φαίνεται και στην Εικόνα 1, ένα πακέτο RTP αποτελείται από τέσσερα κομμάτια [5]:

1. Την RTP επικεφαλίδα (RTP header)
2. Μια προαιρετική επέκταση της επικεφαλίδας (optional header extension)
3. Μια προαιρετική επικεφαλίδα ωφέλιμου φορτίου (optional payload header) – εξαρτάται από το payload format που χρησιμοποιείται

4. Τα δεδομένα ωφέλιμου φορτίου (payload data)

Παρακάτω ακολουθεί μια σύντομη περιγραφή της RTP επικεφαλίδας καθώς και των δεδομένων ωφέλιμου φορτίου:



Εικόνα 1: Ένα RTP πακέτο και τα πεδία που το αποτελούν

RTP Header [5]

Η επικεφαλίδα συνήθως έχει μήκος 12 octets, αλλά μπορεί να περιέχει μια λίστα από συμβαλλόμενες πηγές (contributing source list), η οποία μπορεί να αυξήσει το μήκος της επικεφαλίδας από 4 μέχρι 60 επιπλέον octets. Τα πεδία της επικεφαλίδας που μας ενδιαφέρουν περισσότερο είναι ο τύπος ωφέλιμου φορτίου (payload type – PT), ο αριθμός ακολουθίας (Sequence number), η χρονοσφραγίδα (Timestamp) και το αναγνωριστικό πηγής συγχρονισμού (Synchronization source identifier – SSRC). Θα αναφέρουμε ενδεικτικά τη σημασία των πεδίων Payload type και Timestamp, καθώς στα πεδία Sequence number και SSRC θα αναφερθούμε αναλυτικότερα παρακάτω.

- Payload Type [6]

  Αποτελείται από 7 bits και προσδιορίζει τους τύπους μέσων (media types) που μεταφέρει το πακέτο, δηλαδή αν τα δεδομένα του πακέτου είναι audio/video, την κωδικοποίησή τους κ.ά. Ο παραλήπτης εξετάζει το payload type προκειμένου να καθορίσει πώς να αξιοποιήσει τα δεδομένα. Ο αριθμός αυτός συσχετίζεται με κάποια συγκεκριμένα, καθορισμένα payload formats, οι προδιαγραφές των οποίων είναι καταγεγραμμένες.

- Timestamp [7]

  Αποτελείται από έναν ακέραιο των 32 bits και προσδιορίζει τη στιγμή δειγματοληψίας του πρώτου octet των δεδομένων μέσων του πακέτου και χρησιμοποιείται για να προγραμματίσει την αναπαραγωγή των δεδομένων. Η αρχική τιμή του timestamp επιλέγεται τυχαία, αυξάνει με ρυθμό που εξαρτάται από τα μέσα που μεταφέρουμε και μηδενίζεται όταν ξεπεράσει μια μέγιστη τιμή. Παρέχει σύγχρονη αναπαραγωγή στον παραλήπτη και μπορεί να χρησιμοποιηθεί για την εξάλειψη της οφειλόμενης στο δίκτυο διακύμανσης της καθυστέρησης κατά τη μετάδοση των πακέτων (jitter).

Payload data [8]

Περιέχει τα δεδομένα μέσων που μεταφέρει το πακέτο, τα οποία είναι κωδικοποιημένα σύμφωνα με το payload format που προσδιορίζεται από το payload type. Στην περίπτωση που μεταφέρουμε δεδομένα βίντεο και χρησιμοποιούμε το πρότυπο κωδικοποίησης H.264/AVC, τότε τα δεδομένα, κατά την πακετοποίηση, χωρίζονται σε Network Abstract Layer (NAL) Units Επιπλέον, ανάλογα με τον τρόπο που γίνεται η πακετοποίηση, τα δεδομένα μπορεί να βρίσκονται ολόκληρα μέσα σε ένα πακέτο (Aggregation packets) ή μπορεί να συνεχίζουν σε επόμενα πακέτα (fragmented packets). Περισσότερα για τα NAL NALUs θα δούμε στην Ενότητα 5.

2.1.2 Τα πεδία SSRC και Sequence Number

Τα πεδία αυτά της επικεφαλίδας είναι εκείνα που μας απασχολούν περισσότερο, καθώς με βάση αυτά λειτουργεί τόσο ο μηχανισμός αναμετάδοσης πακέτων βίντεο, όσο και ο μηχανισμός συνδιάσκεψης.

SSRC [9]

Το πεδίο αυτό προσδιορίζει τους συμμετέχοντες σε μια σύνοδο RTP και είναι ένα εφήμερο, ανά σύνοδο αναγνωριστικό το οποίο αντιστοιχίζεται σε ένα μακρόβιο κανονικό όνομα (Canonical Name – CNAME), μέσω του πρωτοκόλλου RTCP. Πρόκειται για έναν ακέραιο αριθμό μεγέθους 32bits, ο οποίος επιλέγεται τυχαία για κάθε συμμετέχοντα που εισέρχεται σε μια σύνοδο. Κάθε διαφορετική ροή δεδομένων μέσων που ξεκινάει από κάποιο χρήστη ταυτίζεται με ένα διαφορετικό μοναδικό SSRC, δηλαδή, για τον ίδιο χρήστη, το SSRC της ροής βίντεο είναι διαφορετικό από το SSRC της ροής ήχου. Στην περίπτωση που δύο ή περισσότεροι χρήστες τύχει να αποκτήσουν το ίδιο SSRC, τότε μέσω μηχανισμών του RTCP πραγματοποιείται μια άμεση ενημέρωση, προκειμένου όλοι να έχουν μοναδικές διαφορετικές τιμές.

Στην περίπτωση που πραγματοποιείται συνδιάσκεψη πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας, τότε διαφορετικές ροές δεδομένων μέσων φτάνουν στην ίδια θύρα του παραλήπτη. Κατ’ επέκταση, το πεδίο SSRC θα διαφέρει από πακέτο σε πακέτο, καθώς στην ίδια θύρα λαμβάνονται πακέτα από διαφορετικούς χρήστες. Για το λόγο αυτό, θα πρέπει κατά την αρχικοποίηση της συνόδου να υπάρχουν όλες οι απαραίτητες πληροφορίες οι οποίες θα μας επιτρέπουν να αναγνωρίζουμε την προέλευση του πακέτου. Με βάση την τιμή του SSRC που αποκτά κάθε χρήστης για τα δεδομένα ήχου και βίντεο, δημιουργούμε κάποιες δομές οι οποίες συσχετίζουν τα διάφορα SSRC με τους χρήστες. Με τον τρόπο αυτό, όταν φτάνει ένα πακέτο, ελέγχουμε το SSRC, καταλαβαίνουμε αν μεταφέρει δεδομένα ήχου ή βίντεο, προσδιορίζουμε το χρήστη που το έστειλε και συνεπώς μπορούμε να τον εξυπηρετήσουμε ανάλογα. Περισσότερα για τη διαδικασία αυτή θα αναλύσουμε στην Ενότητα 4.

Sequence Number [10]

Το πεδίο αυτό χρησιμοποιείται για τον προσδιορισμό των πακέτων και για να υποδείξει στον παραλήπτη αν έχουν χαθεί πακέτα ή έχουν παραληφθεί σε λανθασμένη σειρά. Δε χρησιμοποιείται για τον προγραμματισμό της αναπαραγωγής αλλά για να επιτρέψει στον παραλήπτη να ανακατασκευάσει την σειρά με την οποία απεστάλησαν τα πακέτα.

Αποτελείται από έναν μη προσημασμένο ακέραιο των 16bits ο οποίος ξεκινάει από μια τυχαία τιμή, αυξάνει κατά ένα για κάθε πακέτο δεδομένων που αποστέλλεται και μηδενίζεται όταν πάρει τη μέγιστη τιμή. Ωστόσο, επειδή τα 16bits δεν είναι αρκετά, η τιμή του πεδίου μηδενίζεται αρκετά συχνά. Για το λόγο αυτό, η εφαρμογή είναι προτιμότερο να χρησιμοποιεί ένα εκτεταμένο sequence number τουλάχιστον 32bits, προκειμένου να προσδιορίζει τα πακέτα εσωτερικά, με τα 16 lower bits να αντιστοιχούν στο sequence number του RTP πακέτου και τα 16 higher bits να είναι ένα πολλαπλάσιο των φορών που το sequence number έφτασε τη μέγιστη τιμή. Η βασική χρήση του sequence number είναι η ανίχνευση απωλειών πακέτων. Λόγω ότι οι αριθμοί αυξάνουν σειριακά κατά ένα, όταν παρουσιαστεί ένα κενό στην λαμβανόμενη ακολουθία, τότε υπάρχει πιθανή απώλεια πακέτων, γεγονός που σημαίνει ότι πρέπει να πραγματοποιηθούν οι κατάλληλες διαδικασίες για την ανάκτηση των δεδομένων που χάθηκαν. Συνεπώς, χρειάζονται οι κατάλληλοι μηχανισμοί για την ανίχνευση των απωλειών, για τον προσδιορισμό των πακέτων που χάθηκαν, για την ενημέρωση του αποστολέα για τα απολεσθέντα πακέτα, για την αναμετάδοση των πακέτων αυτών και, τέλος, για την αξιοποίηση των αναμεταδιδόμενων πακέτων από τον παραλήπτη. Περισσότερα για τη διαδικασία αυτή θα αναλύσουμε στην Ενότητα 5.

2.2 Το πρωτόκολλο RTCP

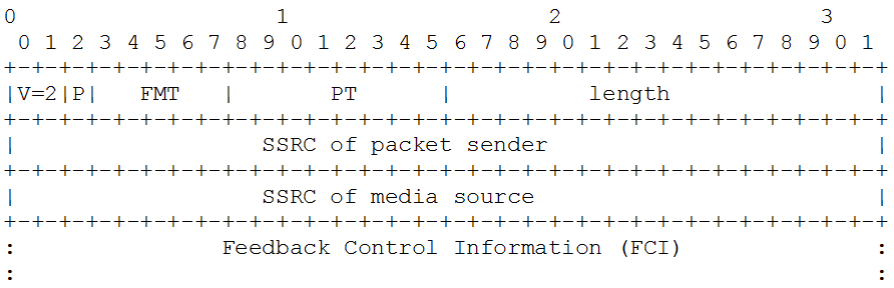
Το RFC 3550 ορίζει επίσης τις προδιαγραφές για το RTCP, ένα πρωτόκολλο το οποίο μπορεί να χρησιμοποιείται από μια δικτυακή εφαρμογή μέσων σε συνδυασμό με το RTP. Το RTCP παρέχει περιοδική αναφορά για την ποιότητα στη λήψη, πληροφορίες για τον προσδιορισμό των συμμετεχόντων και περιγραφής της πηγής, ειδοποιήσεις για αλλαγές στη σύνοδο, πληροφορίες που απαιτούνται για το συγχρονισμό των ροών δεδομένων μέσων, αναφορές για τον αριθμό των πακέτων που στάλθηκαν/παραλήφθηκαν, πληροφορίες για απολεσθέντα RTP πακέτα κ.ά.. Για την επίτευξη των λειτουργιών αυτών χρησιμοποιούνται διάφοροι τύποι RTCP πακέτων.

2.2.1 Το RTCP πακέτο

Τα πακέτα RTCP δεν ενθυλακώνουν τμήμα δεδομένων ήχου ή βίντεο, αλλά περιλαμβάνουν πληροφορίες όπως οι προαναφερθείσες. Υπάρχουν διάφοροι τύποι RTCP πακέτων, αλλά οι βασικές κατηγορίες που ξεχωρίζουν είναι έξι [11].

1. Receiver Report (RR)
2. Sender Report (SR)
3. Source Description (SDES)
4. Membership Management (BYE)
5. Application Defined (APP)
6. Feedback (FB)

Από τις κατηγορίες αυτές, αυτή που μας ενδιαφέρει περισσότερο είναι η κατηγορία των RTCP Feedback packets, όπως αυτή ορίζεται στο RFC 4585 [12], καθώς περιέχει το πακέτο RTCP NACK, το οποίο και χρησιμοποιούμε στην υλοποίησή μας. Η βασική μορφή ενός RTCP FB πακέτου φαίνεται στην Εικόνα 2.



Εικόνα 2: Η βασική μορφή ενός RTCP FB πακέτου

FMT : Feedback Message Type. Έχει μέγεθος 5 bits και προσδιορίζει τον τύπου του FB μηνύματος και ερμηνεύεται σε σχέση με τον τύπο (επίπεδο μεταφοράς, συγκεκριμένο βάσει ωφέλιμου φορτίου, ανάδραση επιπέδου εφαρμογής).

PT : Payload Type. Έχει μέγεθος 8 bits και πρόκειται για τον τύπο RTCP πακέτου που ταυτοποιεί το πακέτο ως ένα RTCP FB μήνυμα. Οι δύο βασικές τιμές που μπορεί να πάρει είναι το 205 (RTPFB), για FB μήνυμα επιπέδου μεταφοράς, και 206 (PSFB), για συγκεκριμένο FB μήνυμα βάσει του ωφέλιμου φορτίου.

SSRC: SSRC of packet sender: Έχει μέγεθος 32 bits και πρόκειται για το SSRC του αποστολέα του πακέτου.

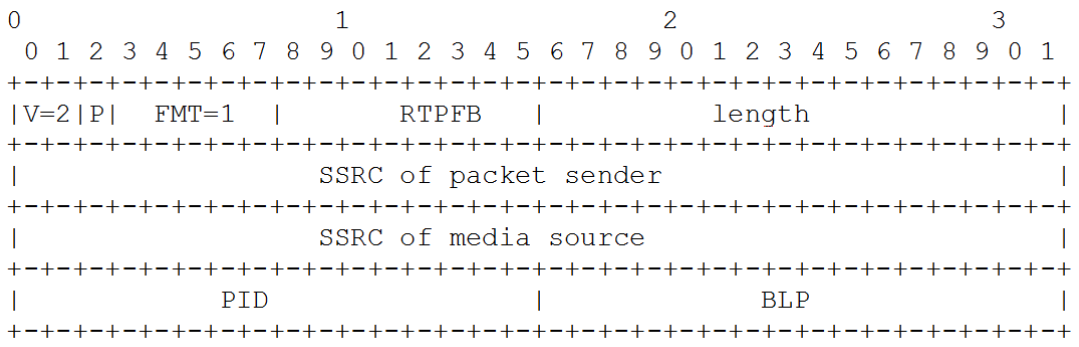
SSRC of media source: Έχει μέγεθος 32 bits και αφορά τον παραλήπτη – προσδιορίζει το SSRC (ήχου ή βίντεο) στο οποίο αναφέρεται αυτό το FB μήνυμα.

FCI : Feedback Control Information. Έχει μέγεθος όσο προσδιορίζει το πεδίο length του πακέτου και πρόκειται για τις πληροφορίες που εμπεριέχονται μέσα στο FB μήνυμα. Τυπικά είναι το μήνυμα.

2.2.2 Το πακέτο RTCP NACK

Το RTCP NACK [12] είναι ένα RTCP FB πακέτο το οποίο χρησιμοποιείται προκειμένου ο παραλήπτης να γνωστοποιήσει στον πομπό τα sequence numbers μιας σειράς πακέτων που χάθηκε – που δεν παραλήφθηκε ποτέ. Ο τύπος RTCP πακέτου είναι επιπέδου μεταφοράς (RTPFB) και συνεπώς το πεδίο PT παίρνει την τιμή 205 (ή RTPFB). Η παράμετρος FMT προσδιορίζει τον τύπο μηνύματος που μεταφέρει το RTCP NACK πακέτο. Στην περίπτωσή μας, θα εστιάσουμε στο Generic NACK FB μήνυμα το οποίο χρησιμοποιούμε στην υλοποίηση του μηχανισμού αναμετάδοσης RTP πακέτων βίντεο.

Η μορφή του Generic NACK φαίνεται στην Εικόνα 3, όπου το πεδίο FMT παίρνει την τιμή 1 και το πεδίο PT την τιμή RTPFB. Το πεδίο FCI αποτελείται από n δυάδες των 16 bits: το PID και το BLP, όπου n ο αριθμός των Generic NACKs που εμπεριέχονται στο πεδίο FCI. Κατ' επέκταση, το πεδίο length θα έχει την τιμή 2+n.



Εικόνα 3: Η μορφή του RTCP Generic NACK

PID : packet ID. Χρησιμοποιείται για να προσδιορίσει ένα χαμένο πακέτο. Η τιμή του PID αναφέρεται στο sequence number του χαμένου πακέτου.

BLP : bitmask of following lost packets. Επιτρέπει την αναφορά απώλειας οποιουδήποτε από τα 16 RTP πακέτα που ακολουθούν το RTP πακέτο που υποδεικνύεται από το PID. Δηλώνοντας το λιγότερο σημαντικό bit του BLP ως bit 1 και πιο σημαντικό bit ως bit 16, τότε το bit i της μάσκας bit τίθεται σε 1 αν ο παραλήπτης δεν έχει λάβει το RTP πακέτο με sequence number ίσο με (PID+i) (modulo 2^{16}) και δηλώνει ότι το πακέτο αυτό έχει χαθεί. Σε αντίθετη περίπτωση, το bit i τίθεται σε 0.

Συνεπώς, με τη χρήση των Generic NACKs, μπορούμε να αναφέρουμε στον αποστολέα ποια πακέτα χάθηκαν μέσα σε διαδοχικές ακολουθίες των 16 πακέτων. Περισσότερα για τη λειτουργία και το χειρισμό του Generic NACK θα αναφέρουμε στην Ενότητα 5.

2.3 Το πρωτόκολλο SIP

Το SIP (Session Initiation Protocol) είναι ένα πρωτόκολλο σηματοδοσίας επιπέδου εφαρμογής, οι προδιαγραφές του οποίου ορίζονται στο RFC 3261 [13]. Βασίζεται σε ανταλλαγές αιτημάτων/αποκρίσεων (request/response transactions) και παρέχει μηχανισμούς για την υλοποίηση κλήσεων μέσω ενός δικτύου βασιζόμενου στο IP, επιτρέπει στον καλούντα να εξακριβώσει την τρέχουσα διεύθυνση IP του καλούμενου και να τον ειδοποιήσει ότι θέλει να εκκινήσει μια κλήση, επιτρέπει στους συμμετέχοντες να συμφωνήσουν για σχήματα κωδικοποίησης μέσων που θα χρησιμοποιούν (μέσω SDP), καθώς και να τερματίζουν τις κλήσεις τους. Επιπλέον, παρέχει μηχανισμούς για διαχείριση των κλήσεων, όπως η προσθήκη νέων ρευμάτων δεδομένων (media streams) κατά τη διάρκεια της κλήσης, η αλλαγή της κωδικοποίησης κατά τη διάρκεια της κλήσης, η πρόσκληση νέων συμμετεχόντων, η μεταβίβαση και η αναμονή κλήσης κ.ά..

2.3.1 Τύποι και δομή μηνυμάτων

Τα SIP μηνύματα χωρίζονται σε SIP αιτήματα (SIP requests) και SIP αποκρίσεις (SIP responses). Τα SIP αιτήματα χαρακτηρίζονται από κάποιο όνομα αιτήματος με πιο γνωστά τα INVITE, ACK, BYE, CANCEL, REGISTER, NOTIFY, MESSAGE και INFO ενώ οι SIP αποκρίσεις χαρακτηρίζονται από κάποιους κωδικούς απόκρισης (response

codes) ακολουθούμενοι από ένα μήνυμα απόκρισης. Υπάρχουν 6 διαφορετικές κατηγορίες απόκρισης που σχετίζονται με το αίτημα και σε κάθε μία από αυτές το πρώτο κωδικό ψηφίο του κωδικού απόκρισης είναι ίδιο:

1. 1XX – Προσωρινές Απαντήσεις (Provisional Responses – 100 Trying, 180 Ringing κ.ά.).
2. 2XX – Επιτυχείς Απαντήσεις (Successful Responses – 200 OK, 202 Accepted κ.ά.).
3. 3XX – Απαντήσεις Ανακατεύθυνσης (Redirection Responses – 300 Multiple Choices, 301 Moved Permanently κ.ά.).
4. 4XX – Απαντήσεις Αποτυχίας Πελάτη (Client failure Responses – 400 Bad Request, 401 Unauthorized κ.ά.).
5. 5XX – Απαντήσεις Αποτυχίας Διακομιστή (Server Failure Responses – 500 Server Internal Error, 501 Not Implemented κ.ά.).
6. 6XX – Καθολικές Απαντήσεις Αποτυχίας (Global Failure Responses – 600 Busy Everywhere, 603 Decline κ.ά.).

Τα SIP μηνύματα είναι σε μορφή αρχείου κειμένου και ακολουθούν μια συγκεκριμένη δομή. Για την καλύτερη κατανόηση της δομής των μηνυμάτων παραθέτουμε ένα κλασικό παράδειγμα επικοινωνίας ανάμεσα στον Bob και την Alice, στο οποίο η Alice επιθυμεί να επικοινωνήσει με τον Bob. Στην περίπτωση αυτή, στέλνει στον Bob ένα μήνυμα INVITE και ο Bob απαντάει με ένα μήνυμα απόκρισης 200 OK. Στα μηνύματα αυτά δεν εμφανίζονται οι SDP πληροφορίες του κάθε συμμετέχοντα, καθώς θα αναφερθούμε ξεχωριστά στο SDP παρακάτω.

Alice

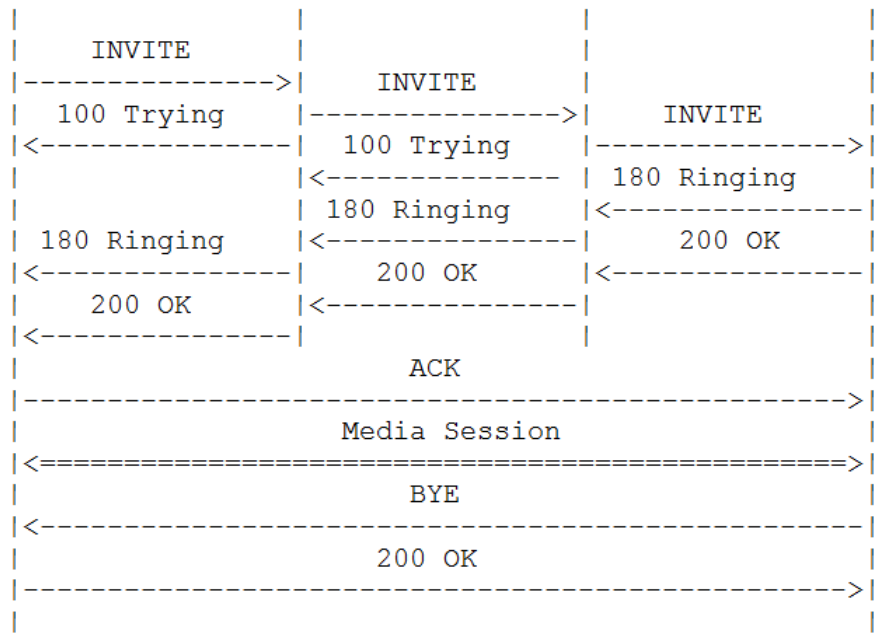
```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@atlanta.com
To: sip:bob@domain.com
Call-ID: a84b4c76e66710@pc33.atlanta.com
Content-Type: application/sdp
Content-Length: 142
```

Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.12
To: sip:alice@atlanta.com>
From: sip:bob@domain.com>
Call-ID: a84b4c76e66710@pc33.atlanta.com
Content-Type: application/sdp
Content-Length: 131
```

2.3.2 Διαδικασία επικοινωνίας

Κατά τη διαδικασία εδραίωσης της επικοινωνίας μεταξύ της Alice και του Bob ανταλλάσσονται διάφορα μηνύματα και ακολουθείται μια συγκεκριμένη ροή για την εγκαθίδρυση της κλήσης. Μια τυπική αναπαράσταση της διαδικασίας, από την εγκαθίδρυση (INVITE – ACK) μέχρι τον τερματισμό (BYE – OK) της κλήσης φαίνεται στην Εικόνα 4.



Εικόνα 4: Τυπική διαδικασία εγκαθίδρυσης και τερματισμού μιας κλήσης μέσω SIP

Στη διαδικασία αυτή, η Alice στέλνει ένα αίτημα INVITE στον Bob και μέχρι το αίτημα να φτάσει στην πλευρά του Bob η Alice λαμβάνει το μήνυμα 100 Trying. Όταν το INVITE φτάσει στην πλευρά του Bob, τότε η Alice λαμβάνει το μήνυμα 180 Ringing, μέχρις ότου ο Bob αποδεχθεί την κλήση, όπου και στέλνει το μήνυμα 200 OK. Όταν η Alice δεχτεί το 200 OK, στέλνει το μήνυμα ACK για να δηλώσει στον Bob ότι όλα είναι έτοιμα για να ξεκινήσει η κλήση τους. Αφού ο Bob λάβει το ACK, τότε ξεκινάει η σύνοδος (Media Session).

Κάποια στιγμή, ο Bob αποφασίζει να τερματίσει την κλήση και στέλνει ένα μήνυμα BYE στην Alice. Όταν η Alice λάβει το BYE στέλνει ένα μήνυμα 200 OK στον Bob και έτσι τερματίζουν και οι δύο τη σύνοδο.

2.4 Το πρωτόκολλο SDP

Κατά την εγκαθίδρυση συνόδων μέσων υπάρχει η απαίτηση να μεταφέρονται στους συμμετέχοντες λεπτομέρειες και μεταδεδομένα σχετικά με τη σύνοδο, όπως για τα μέσα που μεταδίδονται, για τις διευθύνσεις μεταφοράς κ.ά.. Το πρωτόκολλο SDP (Session Description Protocol), όπως αυτό ορίζεται στο RFC 4566 [14], είναι ένα πρωτόκολλο επιπέδου εφαρμογής που παρέχει όλες αυτές τις απαραίτητες πληροφορίες και μπορεί να χρησιμοποιηθεί πάνω από διάφορα πρωτόκολλα μεταφοράς, όπως το RTP, το SIP κ.ά.. Ωστόσο, από μόνο του δεν υποστηρίζει τις διαπραγματεύσεις για το περιεχόμενο της συνόδου ή της κωδικοποίησης των μέσων, αλλά σε συνδυασμό με το πρωτόκολλο SIP μπορεί να επιτελέσει αυτή τη λειτουργία.

2.4.1 SIP και SDP

Για την εγκαθίδρυση μιας κλήσης, οι συμμετέχοντες θα πρέπει να γνωρίζουν τα σχήματα κωδικοποίησης και αποκωδικοποίησης μέσων που χρησιμοποιούν, προκειμένου να συμφωνήσουν ποιες μορφές μπορούν να υποστηρίξουν. Για τον λόγο

αυτό, κατά την ανταλλαγή SIP μηνυμάτων ενθυλακώνουν και μηνύματα SDP. Με τον τρόπο αυτό ξεκινάει μια διαδικασία προσφοράς και απάντησης (offer/answer) στην οποία ο κάθε συμμετέχοντας προτείνει/αντιπροτείνει κάποια σχήματα κωδικοποίησης και αποκωδικοποίησης που υποστηρίζει, προκειμένου να έρθουν σε μια συμφωνία μεταξύ τους για τη σωστή υποστήριξη της συνόδου. Από τη στιγμή που θα συμφωνήσουν, θέτουν σε λειτουργία τους κατάλληλους μηχανισμούς προκειμένου να δημιουργηθούν οι υποδομές για τη σωστή μετάδοση και λήψη δεδομένων μέσων.

Συνεπώς, το προηγούμενο παράδειγμα επικοινωνίας μεταξύ της Alice και του Bob θα πάρει την εξής μορφή:

Alice

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@atlanta.com
To: sip:bob@domain.com
Call-ID: a84b4c76e66710@pc33.atlanta.com
Content-Type: application/sdp
Content-Length: 142

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

Η Alice ενημερώνει τον Bob ότι η διεύθυνση στην οποία θα δέχεται τα δεδομένα μέσων θα είναι η 167.180.112.24 (c=IN IP4 167.180.112.24) και συγκεκριμένα μπορεί να δέχεται δεδομένα ήχου (m=audio) στη θύρα 38060 μέσω RTP/AVP και με payload type 0, δηλαδή με κωδικοποίηση ήχου PCMU. Η διαδικασία αυτή συνεχίζεται με την απάντηση του Bob, ο οποίος με τη σειρά του θα παραθέσει τα δικά του χαρακτηριστικά, και θα ολοκληρωθεί όταν συμφωνήσουν στα σχήματα κωδικοποίησης/αποκωδικοποίησης που θα χρησιμοποιηθούν. Στην περίπτωση που δεν μπορούν να συμφωνηθούν τα σχήματα κωδικοποίησης/αποκωδικοποίησης, τότε είτε ακυρώνεται η συγκεκριμένη ροή δεδομένων μέσων (πχ η ροή δεδομένων βίντεο) ή αν η επικοινωνία θα πραγματοποιούταν μέσω μόνο μιας ροής δεδομένων μέσων (πχ όχι βίντεο αλλά μόνο ήχος), τότε από τη στιγμή που δεν μπορούν να συμφωνήσουν η κλήση ακυρώνεται.

3. ΤΟ ΛΟΓΙΣΜΙΚΟ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ BARESIP

Το BareSIP είναι ένα λογισμικό ανοιχτού κώδικα της Creytiv [15] και αποτελεί το βασικό εργαλείο πάνω στο οποίο υλοποιήθηκε το project MusiNet και οι μηχανισμοί που παρουσιάζουμε στην εργασία αυτή. Μαζί με τις εξαρτήσεις του (dependencies), πρόκειται για ένα λογισμικό ανοιχτού κώδικα συμβατό με τα πρότυπα των γλωσσών C89/C99, το οποίο υιοθετεί μια καλοσχεδιασμένη, αντικειμενοστραφή, σπονδυλωτή δομή που το καθιστά κατάλληλο για επεκτάσεις και τροποποιήσεις.

Πρόκειται για ένα φορητό και σπονδυλωτό SIP πράκτορα χρήστη (user agent) που επιτρέπει μετάδοση ήχου και βίντεο με τη χρήση των βιβλιοθηκών/εργαλειοθηκών της Creytiv, Libre (μια γενική βιβλιοθήκη για την επικοινωνία σε πραγματικό χρόνο με ασύγχρονη I/O υποστήριξη) και Librem (μια φορητή βιβλιοθήκη για επεξεργασία ήχου και βίντεο σε πραγματικό χρόνο). Υποστηρίζει τη πλειοψηφία των πιο σύγχρονων κωδικοποιητών ήχου (Opus, G.711 κ.ά.) και βίντεο (H.264, H.263, MPEG-4 κ.ά.), καθώς και μια πληθώρα από δικτυακά πρωτόκολλα επικοινωνίας και μεταφοράς (TCP/UDP, RTP/RTCP, SIP κ.ά.), παραμένοντας ωστόσο εύρωστο, γρήγορο και με χαμηλή κατανάλωση μνήμης, παρά τη μεγάλη λίστα των λειτουργιών που υποστηρίζει.

Στην καθαρή του μορφή, το BareSIP περιορίζεται σε Peer-to-peer (P2P) συνδέσεις, δηλαδή σε συνόδους αποκλειστικά μεταξύ δύο χρηστών. Κάθε χρήστης πραγματοποιεί τουλάχιστον μία κωδικοποίηση και αποκωδικοποίηση για κάθε μια σύνοδο που ανοίγει και συνεπώς, εάν επιθυμεί να επικοινωνεί με N διαφορετικούς χρήστες, θα πρέπει να πραγματοποιεί συνολικά τουλάχιστον N κωδικοποιήσεις και αποκωδικοποιήσεις. Ωστόσο, δεν υποστηρίζει την ύπαρξη συνδιάσκεψης, δηλαδή διαφορετικοί χρήστες να επικοινωνούν μεταξύ τους στην ίδια σύνοδο, λειτουργία που είναι απαραίτητη για τις ανάγκες μας. Περισσότερα για τη δυνατότητα συνδιάσκεψης αλλά και για τη μείωση της πολυπλοκότητας σε περίπτωση κλιμάκωσης θα αναλύσουμε στην Ενότητα 4.

3.1 Η δομή του λογισμικού

Το λογισμικό χωρίζεται σε τρεις βασικές ενότητες:

1. baresip
   Αποτελεί το βασικό κομμάτι του λογισμικού. Σε ένα αρχείο ρυθμίσεων με όνομα config έχει οριστεί η IP διεύθυνση και η θύρα για τις εισερχόμενες SIP κλήσεις του μηχανήματος, τα πρωτόκολλα επικοινωνίας και μεταφοράς (TCP/UDP, RTP/RTCP), ο ρυθμός κωδικοποίησης ήχου, τα κανάλια, το εύρος ζώνης, οι κωδικοποιητές ήχου και βίντεο που θέλουμε να ενεργοποιήσουμε (opus, H.264 κ.ά.), οι drivers που θέλουμε να ενεργοποιήσουμε κλπ.. Οι συναρτήσεις που είναι υλοποιημένες μέσα στο φάκελο αυτό, δημιουργούν όλους τους απαραίτητους user agents, σύμφωνα με τις παραμέτρους που έχουμε ορίσει στο config (IP address, SIP ports, audio/video codecs, audio/video source, audio/video filters, audio/video drivers κλπ) και φροντίζουν για την εγκαθίδρυση και την διεκπεραίωση των κλήσεων.
2. libre
   Παρέχει όλες εκείνες τις συναρτήσεις που θα μπορούν να διαχειριστούν την επικοινωνία και τα I/O του συστήματος σε πραγματικό χρόνο. Πιο συγκεκριμένα, παρέχει τους μηχανισμούς για τη διαχείριση των SIP αιτημάτων και αποκρίσεων, για την ανταλλαγή SDP πληροφοριών, για τη διαχείριση των πρωτοκόλλων TCP/UDP, RTP/RTCP, για την ανάγνωση από το socket, για την αποθήκευση πληροφορίας στη μνήμη, για τη διαχείριση της μνήμης κλπ.

3. librem
   Παρέχει όλες εκείνες τις συναρτήσεις για τη σωστή λειτουργία των κωδικοποιητών και των αποκωδικοποιητών. Πιο συγκεκριμένα, φροντίζει για την ομαλή μετάβαση από τους drivers στο BareSIP και από το BareSIP στους drivers, για τη δημιουργία όλων των κατάλληλων μηχανισμών για τους κωδικοποιητές/αποκωδικοποιητές, για την εφαρμογή φίλτρων, για τον προσδιορισμό παραμέτρων αναπαραγωγής κλπ.

3.1.1 Εκκίνηση του προγράμματος

Η εκκίνηση του προγράμματος πραγματοποιείται στη συνάρτηση main που βρίσκεται στο φάκελο baresip. Η συνάρτηση αυτή εκτελεί μια σειρά άλλων συναρτήσεων προκειμένου να δημιουργηθούν οι υποδομές και ο βασικός κορμός του προγράμματος. Αρχικά, καλείται η συνάρτηση libre_init που ανήκει στο libre και είναι υπεύθυνη για την ανάθεση τιμών σε μεταβλητές που θα αξιοποιηθούν αργότερα από άλλες συναρτήσεις του libre. Στη συνέχεια, καλείται η conf_configure που ανήκει στο baresip και φροντίζει για τη διαμόρφωση του προγράμματος σύμφωνα με προκαθορισμένες ρυθμίσεις (IP address, SIP ports, audio/video codecs, audio/video source, audio/video filters, audio/video drivers κλπ), τις οποίες και αποθηκεύει σε ένα αρχείο με όνομα config. Σε περίπτωση που το αρχείο αυτό υπάρχει ήδη, τότε το πρόγραμμα διαμορφώνεται σύμφωνα με τις ρυθμίσεις που υπάρχουν μέσα στο αρχείο. Έπειτα, καλείται η συνάρτηση ua_init που ανήκει στο baresip και είναι υπεύθυνη για την δημιουργία όλων εκείνων των παρκτόρων-χρήστη (User-Agents) που αφορούν τα πρωτόκολλα SIP και UDP/TCP. Οι πράκτορες αυτοί είναι τόσο ακροατές (listeners) όσο και χειριστές (handlers) γεγονότων για τα οποία είναι αρμόδιοι. Οι συναρτήσεις που εκτελούν οι User-Agents κατά το listening και το handling γεγονότων SIP βρίσκονται στο libre, ενώ αυτές που αφορούν τα TCP/UDP βρίσκονται στο baresip. Στη συνέχεια, εκτελείται η συνάρτηση conf_modules που ανήκει στο baresip και φροντίζει ώστε να φορτωθούν όλα εκείνα τα modules που είναι ορισμένα μέσα στο αρχείο config, δηλαδή τους drivers και τους κωδικοποιητές που θα χρησιμοποιήσουμε, τεχνικές ICE ή εξυπηρετητές STUN κ.λπ..

Η τελευταία συνάρτηση που καλείται στη main είναι η re_main που ανήκει στο libre. Πρόκειται για το βασικό ατέρμονα βρόχο εκλογής των ασύγχρονων I/O γεγονότων και πρακτικά, όλα ξεκινάνε από το σημείο αυτό. Αρχικά, καθορίζεται η μέθοδος με την οποία θα γίνεται η εκλογή των ασύγχρονων I/O γεγονότων (poll, select κ.λπ.). Έπειτα, δημιουργούμε ένα σύνολο από περιγραφείς αρχείων (file descriptor set) στο οποίο καταχωρούμε όλους τους περιγραφείς αρχείων που έχουμε ανοίξει (sockets κ.λπ. – στο σημείο αυτό τα sockets που έχουμε δημιουργήσει απευθύνονται σε επικοινωνία μέσω UDP για το πρωτόκολλο SIP), καθώς και τους κατάλληλους handlers που αντιστοιχούν στον κάθε ένα. Όταν συμβεί μια αλλαγή σε κάποιο περιγραφέα (ανάγνωση, εγγραφή, σφάλμα) τότε ένα νήμα αναλαμβάνει την εξυπηρέτηση του συγκεκριμένου γεγονότος, καλώντας τον handler που αντιστοιχεί στον περιγραφέα. Για παράδειγμα, αν μας έρθει ένα SIP REQUEST, τότε ο περιγραφέας αρχείου που αντιστοιχεί στο socket που δεχόμαστε SIP αιτήματα θα μεταβεί στην κατάσταση READ, θα ενεργοποιηθεί ένα νήμα προκειμένου να διαβάσει από το socket αυτό και έπειτα θα καλέσει τον handler που αναλογεί στον περιγραφέα, ο οποίος θα φροντίσει για την εξυπηρέτηση του SIP αιτήματος. Αντίστοιχα συμβαίνει και με τους περιγραφείς αρχείων που απευθύνονται σε επικοινωνία RTP/RTCP κ.λπ., οι οποίοι όμως δεν έχουν οριστεί ακόμα σε αυτό το σημείο του προγράμματος αλλά δημιουργούνται με την εγκαθίδρυση μιας κλήσης. Για να τερματίσει η ατέρμονη αυτή διαδικασία θα πρέπει να προκύψει κάποιο σφάλμα ή να

κληθεί η συνάρτηση re_cancel η οποία και θα οδηγήσει στον τερματισμό του προγράμματος.

Συνεπώς, μέσω της main δημιουργήθηκαν οι απαραίτητοι μηχανισμοί και υποδομές και πλέον το σύστημα είναι έτοιμο να δεχθεί και να διεκπεραιώσει SIP κλήσεις για μετάδοση ήχου και βίντεο.

3.1.2 Εγκαθίδρυση κλήσης

Η εγκαθίδρυση μιας κλήσης πραγματοποιείται μέσω του πρωτοκόλλου SIP. Όταν ξεκινάνε οι διαπραγματεύσεις, τόσο στον αποστολέα όσο και στον παραλήπτη δημιουργείται η δομή call, η οποία περιέχει διάφορα πεδία και δομές που χαρακτηρίζουν τη συγκεκριμένη κλήση. Πιο συγκεκριμένα, η δομή call περιέχει τους SIP User-Agents που έχουν δημιουργηθεί, την τοπική SIP διεύθυνση αλλά και του χρήστη με τον οποίο πραγματοποιείται η κλήση, τις SDP πληροφορίες που ανταλλάσσουν, διάφορες μεταβλητές στατιστικών κ.ά., αλλά πιο σημαντικές είναι η δομές audio και video. Οι δομές αυτές παρέχουν όλες εκείνες τις πληροφορίες και τους μηχανισμούς για να μπορέσει να πραγματοποιηθεί η μετάδοση δεδομένων ήχου και βίντεο αντίστοιχα.

Σε πρώτο στάδιο οι συμμετέχοντες θα πρέπει να συμφωνήσουν στα σχήματα κωδικοποίησης ήχου που θα χρησιμοποιήσουν προκειμένου η κλήση να μεταφέρει τουλάχιστον δεδομένα ήχου. Όταν συμφωνήσουν, τότε ο κάθε συμμετέχοντας δημιουργεί δύο νέα sockets, ένα για τη μεταφορά των δεδομένων RTP και ένα για τη μεταφορά δεδομένων RTCP. Οι περιγραφείς αρχείων των sockets αυτών προστίθενται στο σύνολο από περιγραφείς αρχείων της re_main που αναφέραμε προηγουμένως. Έπειτα, δημιουργούν την δομή audio της call, η οποία έχει ως βασικά πεδία μια δομή για την αποστολή ήχου (tx), μια δομή για τη λήψη ήχου (rx) - οι οποίες περιέχουν μηχανισμούς και πληροφορίες για την κωδικοποίηση και αποκωδικοποίηση αντίστοιχα - και τέλος μια δομή stream, η οποία περιέχει όλες εκείνες τις πληροφορίες και τις δομές που χαρακτηρίζουν τις διαδικτυακές ιδιότητες ενός ρεύματος δεδομένων μέσων, όπως έναν ενταμιευτή jitter (jitter_buffer), τα RTP/RTCP sockets, διάφορα στατιστικά στοιχεία για τα RTP/RTCP κ.ά..

Σε δεύτερο στάδιο οι συμμετέχοντες θα επιχειρήσουν να συμφωνήσου στα σχήματα κωδικοποίησης βίντεο, προκειμένου η κλήση να μεταφέρει και δεδομένα βίντεο. Όταν συμφωνήσουν, τότε η διαδικασία που ακολουθείται είναι παρόμοια με αυτή του ήχου. Πιο συγκεκριμένα, ο κάθε συμμετέχοντας δημιουργεί δύο νέα sockets, ένα για τη μεταφορά των δεδομένων RTP και ένα για τη μεταφορά δεδομένων RTCP. Οι περιγραφείς αρχείων των sockets αυτών προστίθενται στο σύνολο από περιγραφείς αρχείων της re_main που αναφέραμε προηγουμένως. Έπειτα, δημιουργούν την δομή video της call, η οποία έχει ως βασικά πεδία μια δομή για την αποστολή βίντεο (vtx), μια δομή για τη λήψη βίντεο (vrx) - οι οποίες περιέχουν μηχανισμούς και πληροφορίες για την κωδικοποίηση και αποκωδικοποίηση αντίστοιχα - και τέλος μια δομή stream, η οποία περιέχει όλες εκείνες τις πληροφορίες και τις δομές που χαρακτηρίζουν τις διαδικτυακές ιδιότητες ενός ρεύματος δεδομένων μέσων, όπως έναν ενταμιευτή jitter (jitter_buffer), τα RTP/RTCP sockets, διάφορα στατιστικά στοιχεία για τα RTP/RTCP κ.ά..

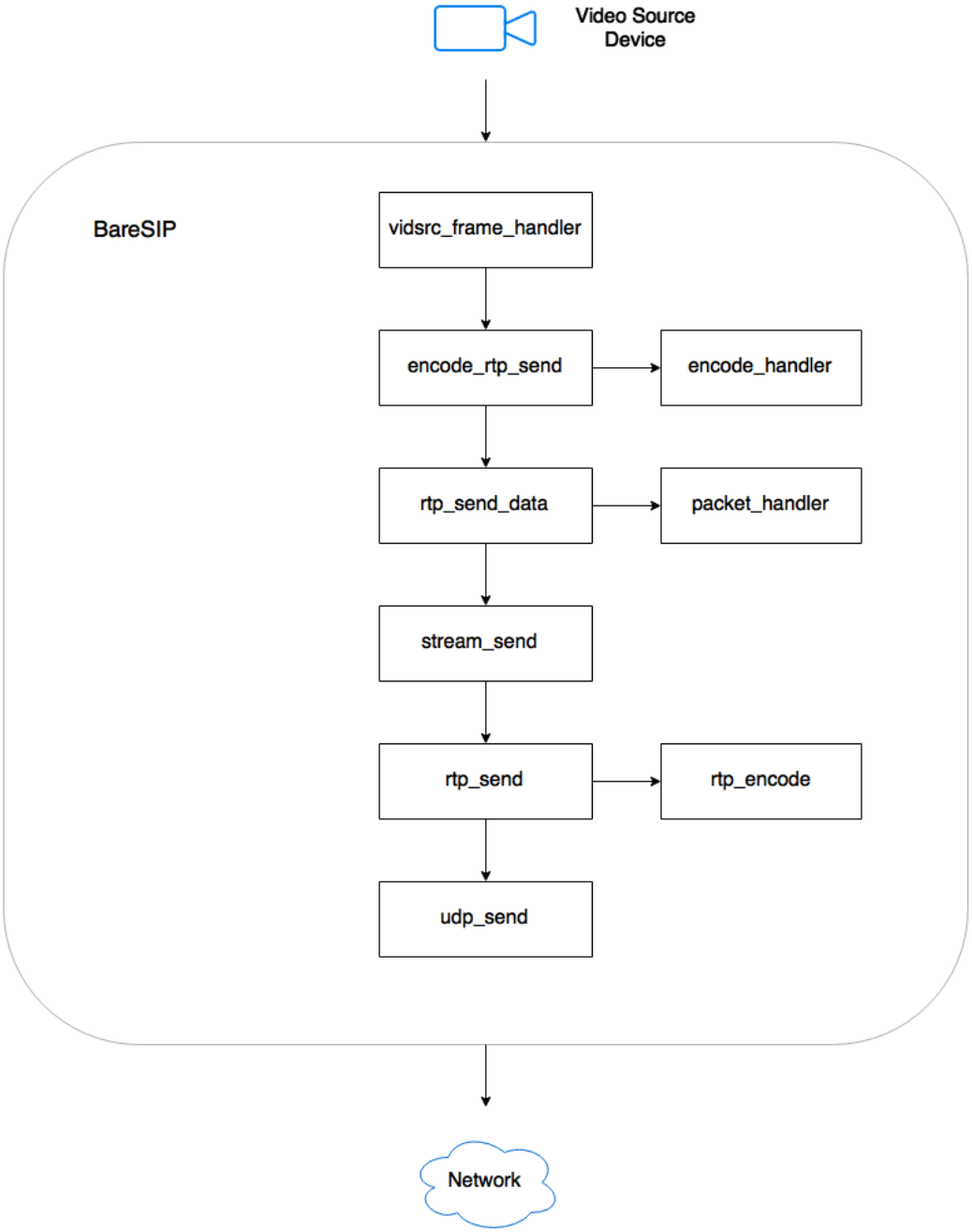
Αφού οι συμμετέχοντες συμφωνήσουν στα παραπάνω και όλοι οι μηχανισμοί για τη μεταφορά δεδομένων μέσων, την κωδικοποίηση, την αποκωδικοποίηση, τη λήψη από το μικρόφωνο και την κάμερα και την αναπαραγωγή στα ηχεία και την οθόνη έχουν δημιουργηθεί, τότε το πρόγραμμα είναι έτοιμο να υποστηρίξει τη μετάδοση μέσων.

Παρακάτω παρουσιάζεται η διαδικασία για την αποστολή και λήψη βίντεο, ενώ για την περίπτωση μετάδοσης ήχου ακολουθείται παρόμοια διαδικασία.

3.1.3 Μετάδοση βίντεο – Αποστολέας

Μια σύντομη και γενική περιγραφή των βημάτων που πραγματοποιούνται στο BareSIP για τη μετάδοση βίντεο, δηλαδή από τη στιγμή που θα γίνει η λήψη μιας εικόνας από την κάμερα μέχρι τη στιγμή που θα μεταδώσουμε τα δεδομένα μέσω του διαδικτύου, παρουσιάζεται στην Εικόνα 5 και είναι η εξής:

1. Αρχικά η συσκευή πηγής βίντεο συλλαμβάνει τις εικόνες, επικοινωνεί με τους αντίστοιχους drivers και modules που έχουμε ορίσει και μέσω αυτών, η πληροφορία συγκεντρώνεται σε πλαίσια (frames) τα οποία αναγνωρίζει το BareSIP.
2. Στη συνέχεια, όταν η πληροφορία αυτή ετοιμαστεί, μέσα στο module καλείται ο vidsrc_frame_handler του BareSIP, ο οποίος δέχεται τα frames από το module, και πλέον η πληροφορία της κάμερας έχει περαστεί στο BareSIP.
3. Το BareSIP με τη σειρά του καλεί την encode_rtp_send, η οποία θα φροντίσει να ετοιμάσει τα δεδομένα για να τοποθετηθούν σε RTP πακέτα. Πιο συγκεκριμένα, καλεί τον encode_handler, ο οποίος κωδικοποιεί τα δεδομένα βίντεο, δημιουργεί τα NALUs και τα πακετοποιεί (RTP payload data) ώστε να μπορούν να τοποθετηθούν σε RTP πακέτα.
4. Στη συνέχεια, λοιπόν, καλείται η rtp_send_data, η οποία θα φροντίσει να δημιουργήσει RTP πακέτα τα οποία θα κουβαλάνε τα δεδομένα κωδικοποιημένου βίντεο. Ειδικότερα, καλεί τον packet_handler, ο οποίος δημιουργεί RTP πακέτα (χωρίς την RTP επικεφαλίδα) και ενθυλακώνει σε αυτά τα payload_data.
5. Έπειτα καλείται η stream_send και η rtp_send, η οποίες θα φροντίσουν να δημιουργήσουν ένα ολοκληρωμένο RTP πακέτο για το συγκεκριμένο stream βίντεο. Καλείται η rtp_encode, η οποία δημιουργεί την RTP επικεφαλίδα, σύμφωνα με τα χαρακτηριστικά του συγκεκριμένου stream βίντεο, και την ενσωματώνει στο RTP πακέτο.
6. Τέλος, το RTP πακέτο ενθυλακώνεται σε ένα UDP πακέτο και μέσω της udp_send αποστέλλεται στο διαδίκτυο.



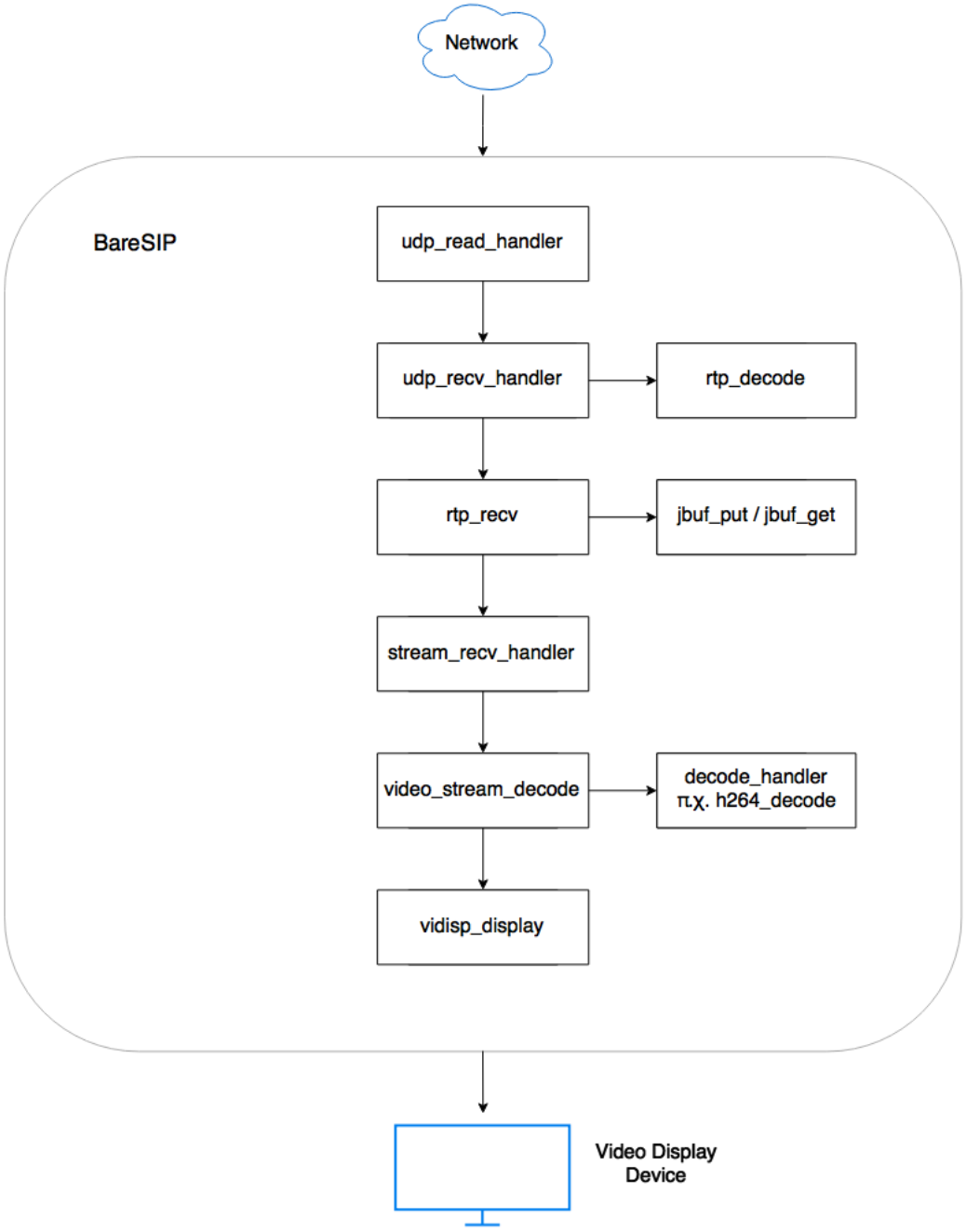
Εικόνα 5: Διαδικασία δημιουργίας και μετάδοσης δεδομένων βίντεο

3.1.4 Μετάδοση βίντεο – Παραλήπτης

Μια σύντομη και γενική περιγραφή των βημάτων που πραγματοποιούνται στο BareSIP για τη λήψη και αναπαραγωγή βίντεο, δηλαδή από τη στιγμή που θα γίνει η λήψη ενός πακέτου βίντεο από το διαδίκτυο μέχρι της στιγμής που θα το αναπαράγουμε, παρουσιάζεται στην Εικόνα 6 και είναι η εξής:

1. Αρχικά από το δίκτυο φτάνει ένα UDP πακέτο και συνεπώς καλείται ο udp_raed_handler ο οποίος είναι υπεύθυνος για να διαβάσει από το socket το πακέτο. Αποθυλακώνει τα δεδομένα που εμπεριέχονται σε αυτό και καλεί τον handler που αντιπροσωπεύει το συγκεκριμένο socket (ανάλογα για το αν στο socket φτάνουν δεδομένα SIP, RTP, RTCP κ.τ.λ.).

2. Ο handler που καλείται στην περίπτωση που παραληφθούν δεδομένα RTP είναι ο udp_recv_handler. Σκοπός του handler αυτού είναι να αποθυλακώσει τα δεδομένα RTP (payload data) από το πακέτο, καθώς και να διακρίνει τα πεδία επικεφαλίδας. Η διαδικασία αυτή πραγματοποιείται καλώντας τη συνάρτηση rtp_decode.
3. Στη συνέχεια, καλείται η συνάρτηση rtp_recv η οποία φροντίζει να ενημερώσει τη δομή stream, που αντιστοιχεί στη συγκεκριμένη ροή βίντεο, αλλά και να την αξιοποιήσει προκειμένου να τοποθετήσει σωστά τα payload data στους jitter buffers. Πιο συγκεκριμένα, καλεί τη συνάρτηση jbuf_put, η οποία τοποθετεί τα payload data στη σωστή θέση του jitter buffer, ελέγχοντας τον αριθμό ακολουθίας (sequence number) του πακέτου που έλαβε. Στη συνέχεια, καλεί την jbuf_get και, αν έχει συγκεντρωθεί ο προκαθορισμένος αριθμός πακέτων στον jitter buffer, τότε παίρνει το πρώτο πακέτο του buffer και το προωθεί για επεξεργασία.
4. Καλείται, λοιπόν, ο stream_recv_handler, ο οποίος θα φροντίσει για το συγκεκριμένο stream τη σωστή εξυπηρέτηση του πακέτου. Καλεί τη video_stream_decode, η οποία ελέγχει ότι τα δεδομένα βίντεο που έχουν ληφθεί μπορούν να αποκωδικοποιηθούν, και έπειτα τον decode_handler, ο οποίος πρακτικά μας οδηγεί στον κατάλληλο αποκωδικοποιητή/module που έχουμε καθορίσει για τη συγκεκριμένη κλήση (π.χ. h264_decode).
5. Σκοπός του decode_handler είναι να αποκωδικοποιήσει τα δεδομένα βίντεο που έχουν ληφθεί και να δημιουργήσει τα αντίστοιχα frames (vidframe) για αναπαραγωγή τα οποία και προσθέτει σε μια λίστα από frames για αναπαραγωγή.
6. Τέλος, όταν ολοκληρωθεί η αποκωδικοποίηση των δεδομένων βίντεο του πακέτου, καλείται η συνάρτηση vidisp_display, η οποία από τη λίστα με τα frames που γεμίζει ο αποκωδικοποιητής, εξετάζει αν μπορεί να πάρει κάποιο frame για να το αναπαράγει και, αν υπάρχει, μέσω του κατάλληλου module και driver τροφοδοτεί τη συσκευή αναπαραγωγής βίντεο.



Εικόνα 6: Διαδικασία λήψης και αναπαραγωγής δεδομένων βίντεο

4. ΣΥΝΔΙΑΣΚΕΨΗ ΠΟΛΛΑΠΛΩΝ ΧΡΗΣΤΩΝ ΣΤΟ BARESIP ΜΕ ΧΡΗΣΗ SSRC ΠΟΛΥΠΛΕΞΙΑΣ

Για να μπορέσουν δύο χρήστες να ξεκινήσουν μια σύνοδο RTP με σύνδεση από τερματικό σε τερματικό (P2P) απαιτούνται κάποιες βασικές υποδομές και προδιαγραφές. Αρχικά, πρέπει να συμφωνήσουν στα μέσα που θα μεταφέρουν, δηλαδή ήχο ή/και βίντεο, να καθορίσουν τα πρωτόκολλα που θα χρησιμοποιήσουν, την IP διεύθυνση και τις θύρες που θα ανοίξουν, καθώς και τα σχήματα κωδικοποίησης που θα εφαρμόσουν. Στη συνέχεια, ο κάθε χρήστης θα πρέπει να δημιουργήσει τις κατάλληλες υποδομές προκειμένου να μπορεί να κωδικοποιεί και να αποκωδικοποιεί τα δεδομένα που στέλνει και λαμβάνει αντίστοιχα, σύμφωνα με τις προδιαγραφές που τέθηκαν.

Όταν θέλουμε να πραγματοποιήσουμε μια P2P σύνοδο RTP, τότε σε πρώτο στάδιο πρέπει να συμφωνήσουν μέσω SIP για τις προδιαγραφές που μόλις αναφέραμε Έπειτα, ο κάθε χρήστης πρέπει να φροντίσει για την κωδικοποίηση και την αποκωδικοποίησης τόσο για τα δεδομένα ήχου, όσο και για τα δεδομένα βίντεο. Δηλαδή, για μια απλή P2P σύνοδο RTP ο κάθε χρήστης πρέπει να φροντίσει για 2 διαφορετικές κωδικοποιήσεις και 2 διαφορετικές αποκωδικοποιήσεις ενώ θα χρησιμοποιεί 2 θύρες, μία για τη μεταφορά δεδομένων ήχου και μία για τη μεταφορά δεδομένων βίντεο. Στην περίπτωση που χρησιμοποιείται και το πρωτόκολλο RTCP, τότε συνολικά θα χρησιμοποιεί 4 διαφορετικές θύρες.

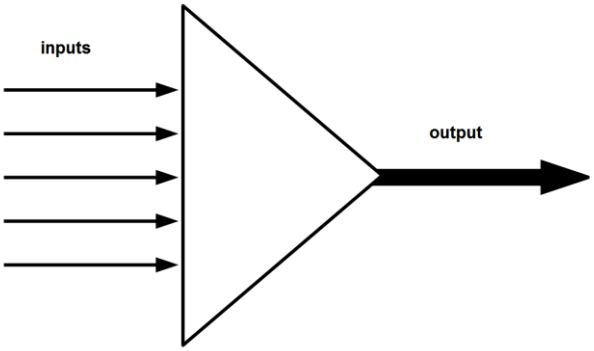
Ωστόσο, σε πολλές περιπτώσεις το πλήθος των χρηστών που θέλουν να επικοινωνήσουν ταυτόχρονα μεταξύ τους είναι μεγαλύτερο από 2, δηλαδή επρόκειτο για μια συνδιάσκεψη πολλαπλών χρηστών. Όταν, λοιπόν, 3 χρήστες επιθυμούν να επικοινωνήσουν ταυτόχρονα μεταξύ τους, ο καθένας από αυτούς θα πρέπει να φροντίσει για 4 διαφορετικές κωδικοποιήσεις και 4 διαφορετικές αποκωδικοποιήσεις, ενώ θα χρησιμοποιεί 8 διαφορετικές θύρες (δεδομένου ότι χρησιμοποιείται και το RTCP). Είναι εύκολο να διακρίνει κανείς ότι σε περίπτωση που θέλουν να επικοινωνήσουν ταυτόχρονα N διαφορετικοί χρήστες μεταξύ τους, τότε ο καθένας θα πραγματοποιεί $2\cdot(N-1)$ κωδικοποιήσεις και $2\cdot(N-1)$ αποκωδικοποιήσεις, ενώ θα χρησιμοποιεί $4\cdot(N-1)$ θύρες.

Προκειμένου οι απαιτήσεις για συνδιάσκεψη πολλαπλών χρηστών να μη γιγαντώνονται όσο αυξάνεται ο αριθμός των συμμετεχόντων, χρησιμοποιούνται διάφορες τεχνικές πολυπλεξίας, προκειμένου να μειωθεί η κατανάλωση τόσο των δικτυακών πόρων όσο και των πόρων του συστήματος, ώστε η μετάδοση να είναι εύκολη, γρήγορη και εύρωστη. Στην ενότητα αυτή θα παρουσιάσουμε τις διάφορες τεχνικές πολυπλεξίας που επικρατούν, καθώς και ένα μοντέλο συνδιάσκεψης πολλαπλών χρηστών. Έπειτα, θα μελετήσουμε τα προτερήματα και τα μειονεκτήματά τους, προκειμένου να καταλήξουμε στις προδιαγραφές και στην παρουσίαση του μηχανισμού συνδιάσκεψης πολλαπλών χρηστών με χρήστη SSRC πολυπλεξίας που υλοποιήσαμε στο BareSIP.

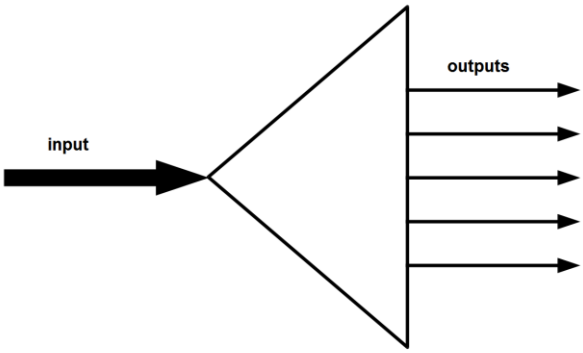
4.1 Βασικές τεχνικές πολυπλεξίας για υλοποίηση συνδιάσκεψης

Γενικά, ως πολυπλεξία (multiplexing) θα μπορούσε να χαρακτηριστεί η μέθοδος κατά την οποία N διαφορετικές είσοδοι πολυπλέκονται με σκοπό να παραχθεί μία κοινή έξοδος. Μια τυπική διαδικασία πολυπλεξίας παρουσιάζεται στην Εικόνα 7. Σε μία συνδιάσκεψη πολλαπλών χρηστών για τη μετάδοση ήχου και βίντεο, η πολυπλεξία χρησιμοποιείται προκειμένου να μπορέσουμε να μεταφέρουμε τις διαφορετικές ροές δεδομένων των χρηστών ως μία κοινή ροή. Για να μπορέσει να εφαρμοστεί απαιτούνται

κάποιοι μηχανισμοί και προδιαγραφές στους τερματικούς κόμβους αλλά και η ύπαρξη ενός ενδιάμεσου κόμβου που θα λειτουργεί ως εξυπηρετητής. Πιο συγκεκριμένα, τα δεδομένα αποστέλλονται στον εξυπηρετητή (και όχι απευθείας στον τερματικό κόμβο) ο οποίος, με τη σειρά του, πολυπλέκει τις εισόδους που δέχεται και την έξοδο που παράγει τη μεταδίδει στους υπόλοιπους κόμβους. Όταν ένας κόμβος δέχεται τα δεδομένα από τον εξυπηρετητή, απαιτείται να έχει έναν αποπολυπλέκτη (demultiplexer) προκειμένου να πραγματοποιήσει την αντίστροφη διαδικασία, δηλαδή την αποπολυπλεξία (demultiplexing), προκειμένου να μπορεί να διακρίνει τα δεδομένα του κάθε χρήστη. Μια τυπική διαδικασία αποπολυπλεξίας φαίνεται στην Εικόνα 8. Στη μετάδοση ήχου και βίντεο υπάρχουν διάφορες τεχνικές πολυπλεξίας, αλλά αυτές που αξίζει να εστιάσουμε είναι η πολυπλεξία RTP συνόδων (Session Multiplexing) [16] και η πολυπλεξία βάσει του SSRC του κάθε χρήστη (SSRC Multiplexing) [16].



Εικόνα 7: Τυπική διαδικασία πολυπλεξίας



Εικόνα 8: Τυπική διαδικασία αποπολυπλεξίας

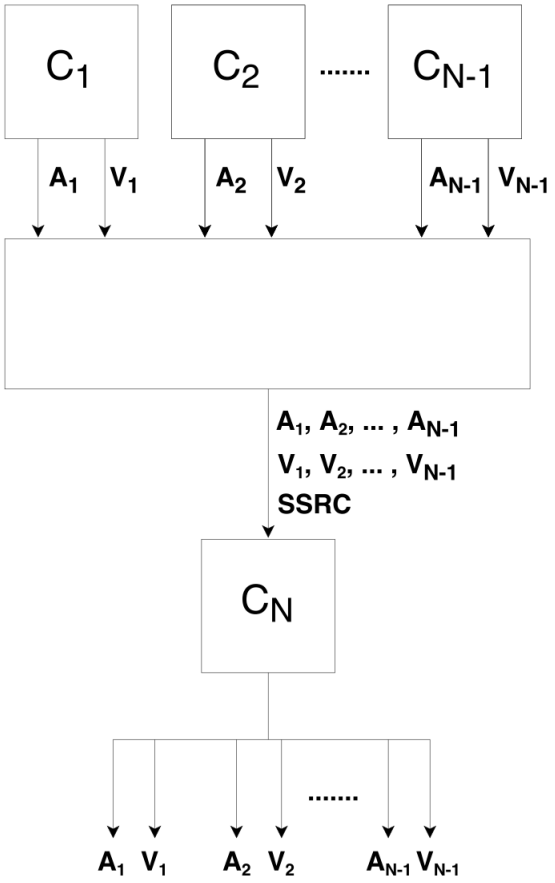
4.1.1 Πολυπλεξία RTP συνόδων

Όταν χρησιμοποιούμε πολυπλεξία RTP συνόδων για την επίτευξη συνδιάσκεψης, τότε διαφορετικές RTP σύνοδοι (π.χ. δύο σύνοδοι ήχου και μια σύνοδος βίντεο) πολυπλέκονται προκειμένου να μεταδοθούν όλες μαζί μέσω μιας μοναδικής RTP συνόδου. Στην περίπτωση λοιπόν όπου N συμμετέχοντες θέλουν να πραγματοποιήσουν μια συνδιάσκεψη μεταξύ τους, τότε οι διαφορετικές RTP σύνοδοι θα πολυπλέκονταν μέσω ενός ενδιάμεσου κόμβου, προκειμένου ο κάθε συμμετέχοντας να λαμβάνει τα δεδομένα μέσω μιας μοναδικής RTP συνόδου, η οποία θα χρησιμοποιεί συγκεκριμένο SSRC για όλα τα πακέτα (Εικόνα 9). Έπειτα, ο κάθε χρήστης θα πρέπει

να αποπολυπλέκει τις διάφορες ροές μέσων βάσει του πεδίου Payload, προκειμένου να αναγνωρίζει και να αποκωδικοποιεί τα δεδομένα βίντεο ή ήχου.

Ωστόσο, δεν είναι θεμιτό να μεταφέρονται διαφορετικές ροές ήχου και βίντεο μέσω μιας μοναδικής RTP συνόδου και να αποπολυπλέκονται βάσει του πεδίου PT, καθώς το να μπλέκονται πακέτα με διαφορετικούς τύπους μέσων, τα οποία χρησιμοποιούν το ίδιο SSRC, μπορεί να εισάγει πολλά προβλήματα [4]:

1. Έστω ότι δύο ρεύματα ήχου μεταφέρονται μέσω της ίδια RTP συνόδου, χρησιμοποιούν το ίδιο SSRC και, έστω, ένα από αυτά αλλάξει σχήμα κωδικοποίησης και συνεπώς και Payload Type. Τότε δε θα υπήρχε κάποιος γενικός τρόπος προκειμένου να αναγνωρίσουμε ποιο ρεύμα άλλαξε το σχήμα κωδικοποίησης.



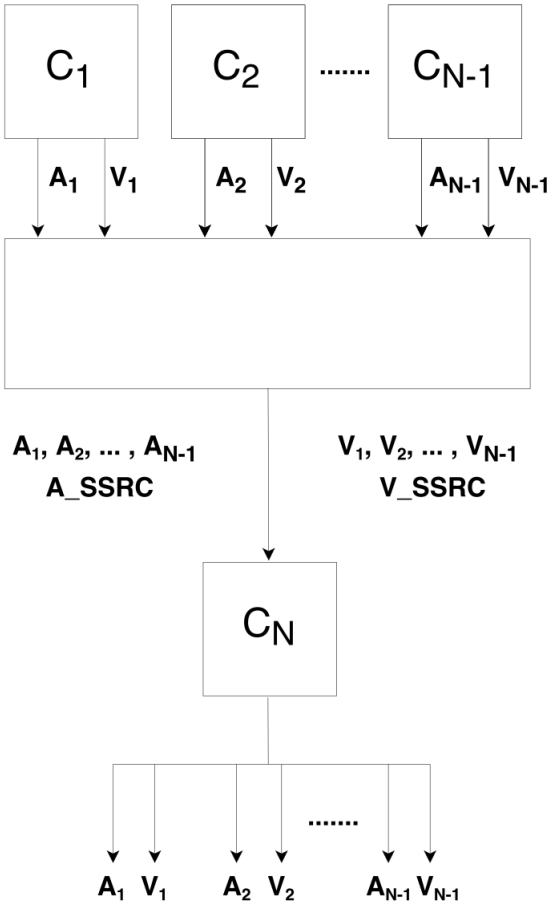
Εικόνα 9: Πολυπλεξία πολλαπλών RTP συνόδων σε μία ενιαία RTP σύνοδο

2. Το SSRC χρησιμοποιείται προκειμένου να προσδιορίσει ένα μοναδικό χώρο χρονισμού και αριθμού ακολουθίας. Η πολυπλεξία πολλαπλών Payload Types θα απαιτούσε διαφορετικούς χώρους χρονισμού, εάν οι ρυθμοί ρολογιού των μέσων διέφεραν, και διαφορετικούς χώρους αριθμού ακολουθίας, προκειμένου να προσδιορίσουμε σε ποιο Payload Type προέκυψε απώλεια πακέτων.
3. Οι RTCP αναφορές αποστολέα και παραλήπτη (SR, RR) μπορούν να περιγράψουν μόνο ένα χώρο χρονισμού και αριθμού ακολουθίας ανά SSRC και δεν εμπεριέχουν το πεδίο Payload Type.

4. Σε περίπτωση που υπάρχει, ένας RTP μείκτης (RTP Mixer) δε θα μπορούσε να συνδυάσει τις ανάμεικτες ροές από ασυμβίβαστους τύπους μέσων σε μία μοναδική ροή.
5. Η μεταφορά πολλαπλών μέσων σε μία RTP σύνοδο αποκλείει τη χρήση διαφορετικών δικτυακών διαδρομών ή την κατανομή των πόρων του δικτύου ανάλογα με την περίπτωση. Επίσης, αποκλείει τη λήψη ενός υποσυνόλου των μέσων, εάν είναι επιθυμητό, για παράδειγμα τη μόνο τη λήψη ήχου εάν το βίντεο υπερβεί το διαθέσιμο εύρος ζώνης και τέλος, αποκλείει την υλοποίηση εφαρμογών στη λήψη που χρησιμοποιούν ξεχωριστές επεξεργασίες για τα διαφορετικά μέσα.

4.1.2 Πολυπλεξία βάσει SSRC

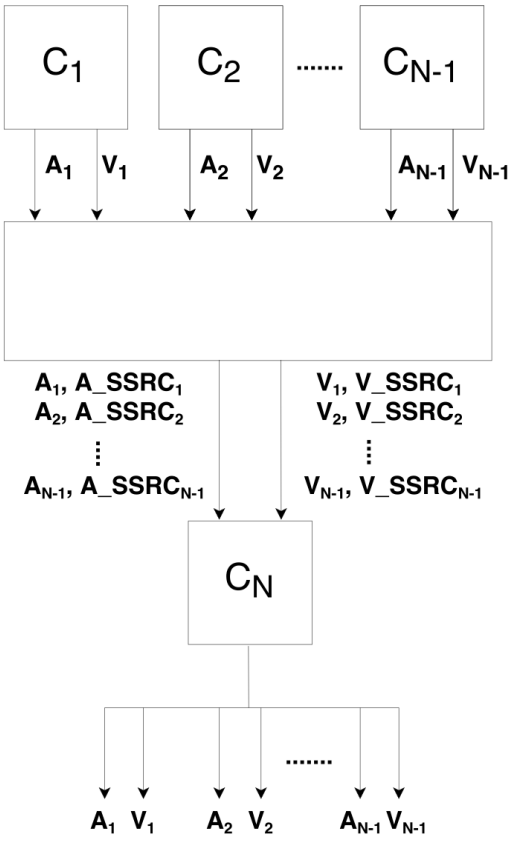
Χρησιμοποιώντας ένα διαφορετικό SSRC για κάθε μέσο, αλλά στέλνοντάς τα στην ίδια RTP σύνοδο (Εικόνα 10) θα βοήθαγε στην αποφυγή των τριών πρώτων προβλημάτων που αναφέρθηκαν προηγουμένως, αλλά όχι και των δύο τελευταίων [4]. Από την άλλη πλευρά, η πολυπλεξία



Εικόνα 10: Πολυπλεξία RTP συνόδων σε μία ενιαία RTP σύνοδο, όπου οι συναφείς πηγές έχουν το ίδιο SSRC

πολλαπλών συναφών πηγών του ίδιου μέσου σε μία RTP σύνοδο, χρησιμοποιώντας διαφορετικές τιμές SSRC, αποτελεί τον κανόνα για συνόδους πολυεκπομπής [4]. Η τεχνική αυτή ονομάζεται πολυπλεξία βάσει SSRC και αποτελεί ιδανική επιλογή για συνδιάσκεψη πολλαπλών χρηστών, καθώς είναι πολύ εύκολη η αποπολυπλεξία και ο εντοπισμός των ροών δεδομένων του κάθε χρήστη. Ειδικότερα, οι συναφείς πηγές πολυπλέκονται σε μία ενιαία RTP σύνοδο και η κάθε μία διατηρεί το δικό της SSRC, δηλαδή, σε μια συνδιάσκεψη πολλαπλών χρηστών, οι πηγές ήχου πολυπλέκονται σε μία RTP σύνοδο ενώ οι πηγές βίντεο σε μία διαφορετική RTP σύνοδο και η κάθε μία σύνοδος θα διατηρεί το SSRC της κάθε πηγής, όπως φαίνεται στην Εικόνα 11.

Η τεχνική αυτή δίνει τη δυνατότητα στον παραλήπτη να μη χρειάζεται να διαχειρίζεται επιπλέον θύρες ενώ προσφέρει χαμηλή χρήση θυρών σε εξυπηρετητές υψηλής πυκνότητας. Επιπλέον, τα προβλήματα που τέθηκαν προηγουμένως παύουν να υφίστανται. Πιο συγκεκριμένα, εάν κάποιο ρεύμα ήχου/βίντεο αλλάξει σχήμα κωδικοποίησης, τότε είναι εύκολο μέσω του πεδίου SSRC να αναγνωρίσουμε ποιο ήταν αυτό, ώστε να το διαχειριστούμε ανάλογα. Επιπρόσθετα, ένας RTP Mixer θα μπορεί να συνδυάσει πολλαπλές πηγές ήχου ή βίντεο και θα τις αντιμετωπίζει όλες με τον ίδιο τρόπο. Επιπλέον, είναι εφικτή διαδικασία αναφοράς απωλειών μέσω RTCP NACK πακέτων και η διαδικασία αναμετάδοσης, κάτι το οποίο δεν μπορούσε να πραγματοποιηθεί με τις προηγούμενες συνθήκες, ενώ ως προς τη γενικότερη λειτουργία του RTCP δεν παρουσιάζονται πλέον ανάλογες δυσκολίες και προβλήματα. Τέλος, αποτελεί βασική τεχνική σε WebRTC εφαρμογές αλλά και ιδανική επιλογή για την αντιμετώπιση ενός βασικού προβλήματος που μας οδήγησε στις τεχνικές πολυπλεξίας, που είναι το κόστος διάσχισης NAT/firewall (NAT/firewall traversal) για κάθε μεμονωμένη ροή RTP.



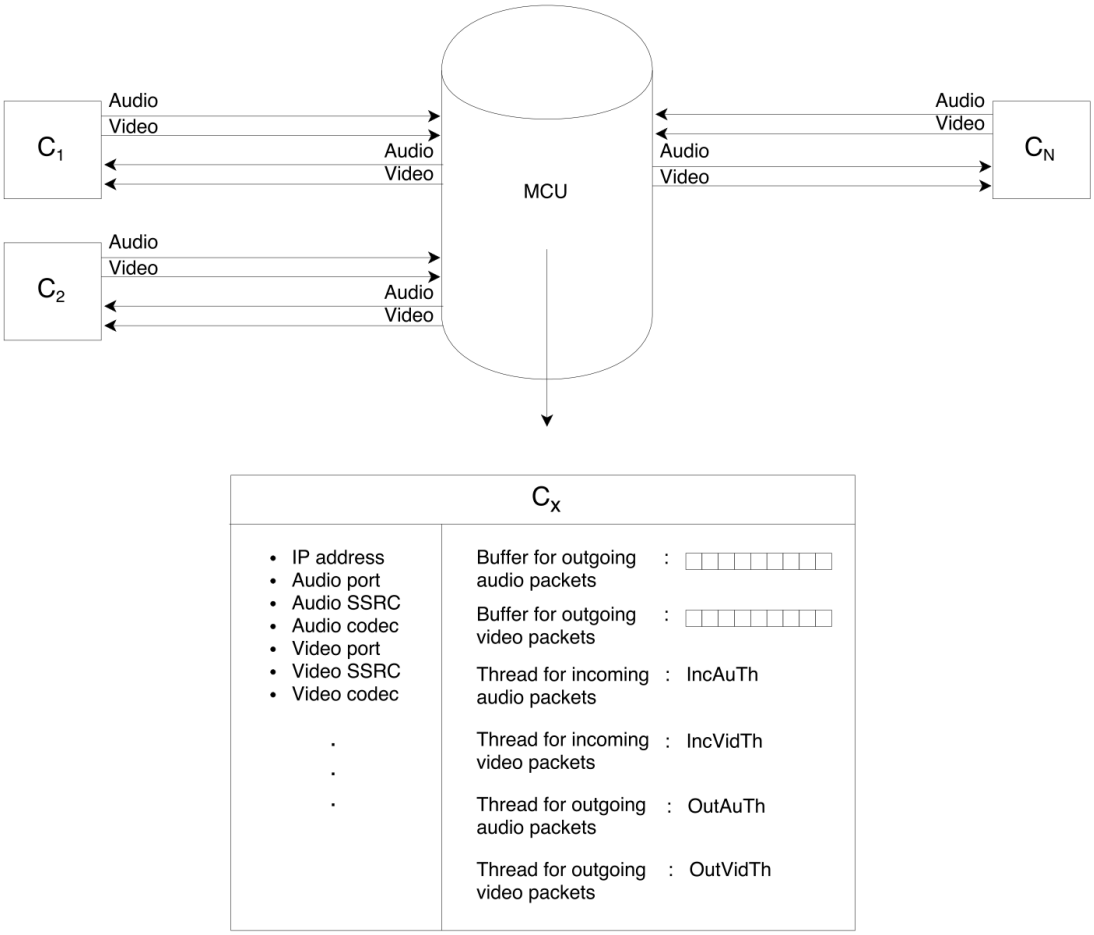
Εικόνα 11: Πολυπλεξία βάσει του SSRC

4.2 Προδιαγραφές συστήματος για συνδιάσκεψη με χρήση SSRC πολυπλεξίας

Προκειμένου ένα σύστημα να υποστηρίζει συνδιάσκεψη πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας, απαιτούνται ορισμένες προδιαγραφές ώστε να γίνεται σωστά τόσο η πολυπλεξία όσο και η αποπολυπλεξία των διαφορετικών ροών. Για το λόγο αυτό απαιτείται η ύπαρξη ενός εξυπηρετητή, ο οποίος θα αναλαμβάνει το ρόλο του πολυπλέκτη, καθώς και τουλάχιστον 2 τερματικών κόμβων/πελατών, όπου ο καθένας θα αναλαμβάνει το ρόλο του αποπολυπλέκτη. Παρακάτω, θα ορίσουμε ένα γενικό μοντέλο συστήματος Ν χρηστών και ενός εξυπηρετητή, θα αναλύσουμε το κάθε κομμάτι του ξεχωριστά, θα ορίσουμε το γενικό τρόπο λειτουργίας του καθενός και θα θέσουμε ζητήματα που αφορούν την πολυπλοκότητα, προκειμένου να καταλήξουμε στις προδιαγραφές του μηχανισμού συνδιάσκεψης που υλοποιήσαμε στο λογισμικό BareSIP.

4.2.1 Εξυπηρετητής

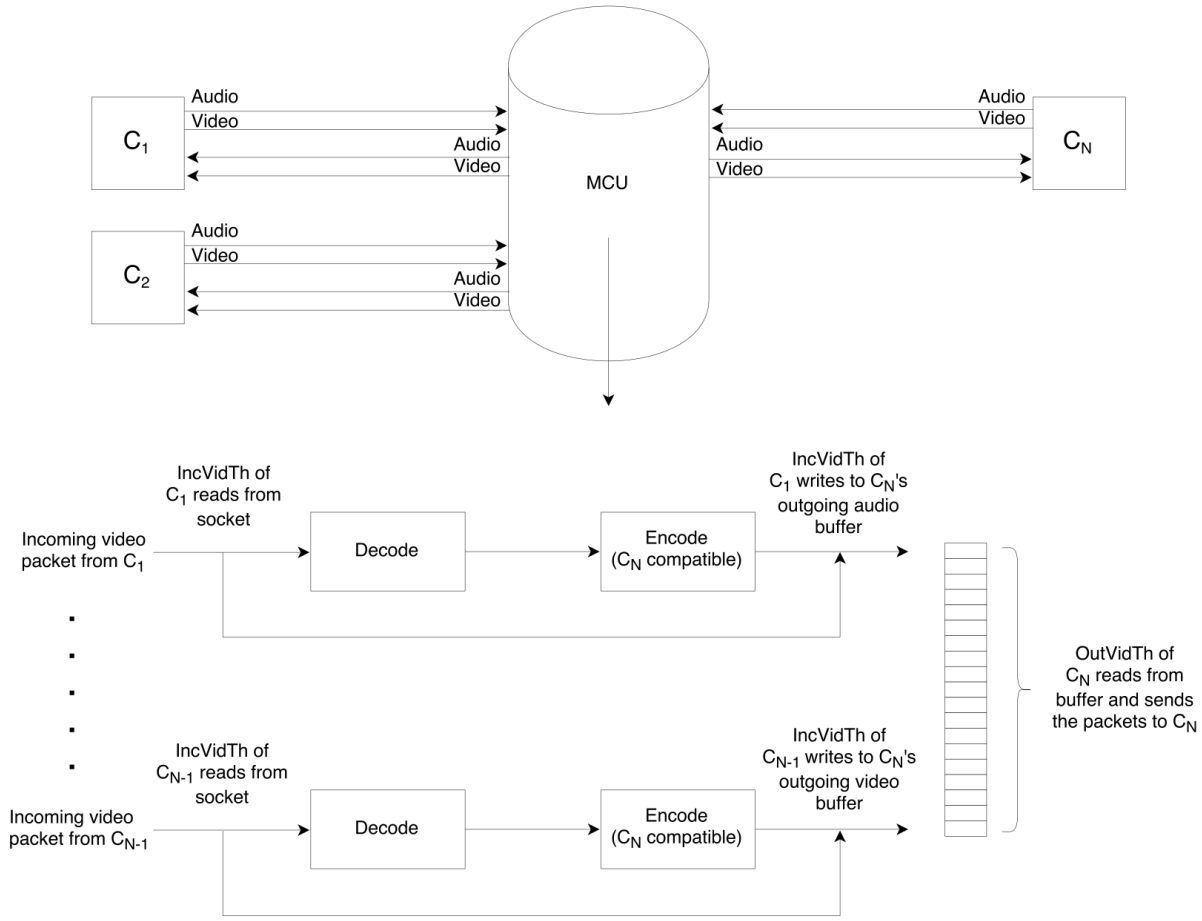
Αρχικά, θα παρουσιάσουμε ένα μοντέλο εξυπηρετητή το οποίο πραγματοποιεί SSRC πολυπλεξία και κωδικοποιεί/αποκωδικοποιεί τα λαμβανόμενα πακέτα (transcoding), όπως συμβαίνει και σε ένα Multipoint Control Unit – MCU [17]. Σε πρώτο στάδιο, η διαδικασία με την οποία γίνεται η εγκαθίδρυση μιας κλήσης δεν πραγματοποιείται πλέον μεταξύ των τερματικών κόμβων, αλλά μέσω του εξυπηρετητή. Πιο συγκεκριμένα, ο κάθε τερματικός κόμβος πραγματοποιεί μια SIP κλήση με τον εξυπηρετητή, γνωστοποιώντας μεταξύ άλλων και τα σχήματα κωδικοποίησης που διαθέτει, ο οποίος, με τη σειρά του, απαντάει στα αιτήματα αυτά με σκοπό να γνωστοποιήσει τα δικά του σχήματα κωδικοποίησης που διαθέτει. Η διαδικασία αυτή συνεχίζεται μέχρι οι τερματικοί κόμβοι να συμφωνήσουν με τον εξυπηρετητή για τα σχήματα κωδικοποίησης ήχου και βίντεο που θα χρησιμοποιήσουν. Αφού συμφωνήσουν, ο εξυπηρετητής ανοίγει 2·Ν θύρες (όπου Ν το πλήθος των τερματικών κόμβων/συμμετεχόντων στη συνδιάσκεψη), μία για τα δεδομένα ήχου και μία για τα δεδομένα βίντεο που θα δέχεται από κάθε χρήστη. Μετά το πέρας της διαδικασίας αυτής, ο εξυπηρετητής δημιουργεί τις κατάλληλες δομές που χαρακτηρίζουν τον κάθε τερματικό κόμβο (IP διεύθυνση, SSRC, τα σχήματα κωδικοποίησης που χρησιμοποιεί για ήχο και βίντεο, κ.ά.), καθώς και ένα νήμα για κάθε εισερχόμενη και εξερχόμενη ροή του κάθε χρήστη, δηλαδή 4·Ν νήματα, όπως φαίνεται και στην Εικόνα 12.



Εικόνα 12: Η δομή που χαρακτηρίζει έναν χρήστη μέσα στον τύπου MCU εξυπηρετητή

Στη συνέχεια, αρχίζει η μεταφορά δεδομένων προς τον εξυπηρετητή, ο οποίος πλέον λαμβάνει N ροές ήχου και N ροές βίντεο. Στο σημείο αυτό ξεκινάει η διαδικασία της πολυπλεξίας, όπου κάθε ένας χρήστης πρέπει να λάβει τα δεδομένα ήχου και βίντεο των υπόλοιπων N – 1 χρηστών. Έστω, λοιπόν ότι θέλουμε να μεταδώσουμε τα πακέτα βίντεο των χρηστών C_1, C_2, … , C_{N-1} στον χρήστη C_N. Το νήμα IncVidTh του χρήστη C_1 θα παραλάβει το πακέτο βίντεο του χρήστη C_1 που μόλις έφτασε και, επειδή ο εξυπηρετητής ακολουθεί τη φιλοσοφία ενός MCU, θα ελέγξει αν οι συγκεκριμένοι χρήστες χρησιμοποιούν τα ίδια σχήματα κωδικοποίησης. Αν δε χρησιμοποιούν τα ίδια σχήματα, αρχικά θα πρέπει να αποκωδικοποιήσει το πακέτο, στη συνέχεια να το κωδικοποιήσει σύμφωνα με το σχήμα κωδικοποίησης του χρήστη C_N, να τοποθετήσει το κατάλληλο PT στην RTP επικεφαλίδα και, τέλος, να το γράψει στο buffer για τα εξερχόμενα πακέτα βίντεο του χρήστη C_N. Αλλιώς, αν οι συγκεκριμένοι χρήστες χρησιμοποιούν τα ίδια σχήματα κωδικοποίησης, τότε το κομμάτι του transcoding παραλείπεται. Αντίστοιχα, τα νήματα των χρηστών C_2, … , C_{N-1} θα εκτελέσουν την ίδια διαδικασία και θα γράψουν τα πακέτα βίντεο στο buffer για τα εξερχόμενα πακέτα βίντεο του χρήστη C_N. Όλα τα πακέτα αυτά, λόγω της πολυπλεξίας βάσει SSRC, θα διατηρήσουν το SSRC της εκάστοτε πηγής. Έπειτα, είναι η σειρά του νήματος OutVidTh του χρήστη C_N να εκτελέσει τη δική του διαδικασία. Πιο συγκεκριμένα, μόλις αντιληφθεί ότι ο buffer για τα εξερχόμενα πακέτα βίντεο του χρήστη C_N δεν είναι άδειος, παίρνει το πρώτο πακέτο από το buffer και το στέλνει στο χρήστη C_N, μετά παίρνει το επόμενο πακέτο από το buffer και το στέλνει στο χρήστη C_N κ.ό.κ.. Ένα στιγμιότυπο της

διαδικασίας αυτής φαίνεται στην Εικόνα 13. Αντίστοιχη διαδικασία θα πραγματοποιηθεί και για τους υπόλοιπους χρήστες, τόσο για τα πακέτα βίντεο όσο και για τα πακέτα ήχου.



Εικόνα 13: Διαδικασία SSRC πολυπλεξίας στον τύπου MCU εξυπηρετητή

4.2.2 Τερματικός κόμβος

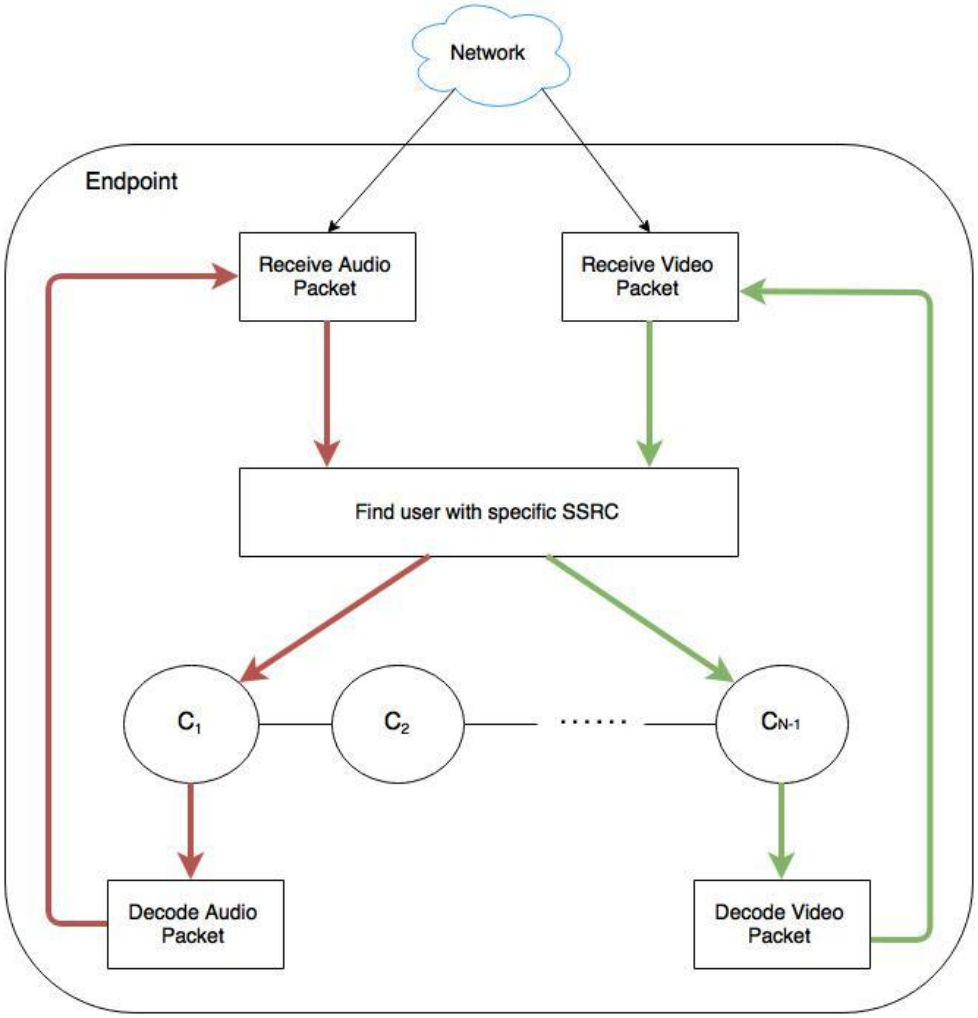
Αρχικά, θα παρουσιάσουμε ένα μοντέλο τερματικού κόμβου, το οποίο υποστηρίζει την ταυτόχρονη συνδιάσκεψη πολλαπλών χρηστών, χρησιμοποιώντας SSRC αποπολυπλεξία, στο οποίο, κάθε μία θύρα που δεσμεύει ικανοποιείται από ένα μοναδικό νήμα και επιπλέον, ο κάθε διαφορετικός χρήστης αντιμετωπίζεται ως μια μεμονωμένη P2P σύνδεση.

Σε πρώτο στάδιο, αφού συμφωνήσει στις διαπραγματεύσεις με τον εξυπηρετητή, θα πρέπει να γνωρίζει (είτε από τον εξυπηρετητή είτε από κάποιον κοινό κανόνα) τον κανόνα συσχέτισης του SSRC των πακέτων ήχου και του SSRC των πακέτων βίντεο του κάθε χρήστη. Για παράδειγμα, ένας κανόνας συσχέτισης για κάποιο χρήστη C_i θα μπορούσε να είναι μια συνάρτηση της μορφής :

$$SSRC\_HXOY(C_i) = i$$
$$SSRC\_BINTEO(C_i) = SSRC\_HXOY(C_i) + 10000 = 10000 + i$$

όπου i ϵ [1, N] και C_i : ο χρήστης i.

Στη συνέχεια θα πρέπει να ανοίξει μια θύρα για την εισερχόμενη ροή ήχου και άλλη μία για την εισερχόμενη ροή βίντεο. Σε κάθε μία από τις θύρες αυτές ανατίθεται ένα μοναδικό νήμα το οποίο θα φροντίζει την εξυπηρέτηση των πακέτων που φτάνουν. Τέλος, δημιουργεί 1 στιγμιότυπο για την κωδικοποίηση δεδομένων ήχου και 1 στιγμιότυπο για κωδικοποίηση δεδομένων βίντεο για αποστολή προς τον εξυπηρετητή.



Εικόνα 14: SSRC αποπολυπλεξία με 1 νήμα ανά τύπο μέσου

Έστω, λοιπόν, ότι σε έναν τερματικό κόμβο φτάνει ένα πακέτο ήχου. Το νήμα που έχει ανατεθεί στη θύρα για τις εισερχόμενες ροές ήχου παίρνει το πακέτο, ελέγχει το SSRC και αναζητά αν έχει έρθει ξανά πακέτο από το χρήστη που περιγράφει το συγκεκριμένο SSRC. Αν δε βρεθεί, τότε δημιουργεί μια δομή που χαρακτηρίζει το συγκεκριμένο χρήστη, η οποία περιέχει πληροφορίες όπως το SSRC ήχου, το SSRC βίντεο, ένα στιγμιότυπο για αποκωδικοποίηση των πακέτων ήχου, ένα στιγμιότυπο αποκωδικοποίησης των πακέτων βίντεο, τους αντίστοιχους jitter buffers κ.τ.λ.. Η δομή αυτή επιτρέπει στον τερματικό κόμβο να αντιμετωπίζει κάθε συμμετέχοντα ως μια μεμονωμένη οντότητα που καθορίζεται από τα SSRC (SSRC αποπολυπλεξία), καθώς περιέχει όλα εκείνα τα στοιχεία τα οποία είναι απαραίτητα για τη σωστή διαχείριση μιας συνόδου RTP μεταξύ δύο χρηστών, ανεξάρτητη από οποιαδήποτε άλλη αντίστοιχη σύνοδο. Αφού δημιουργηθεί η δομή αυτή, το νήμα αναλαμβάνει να ελέγξει την

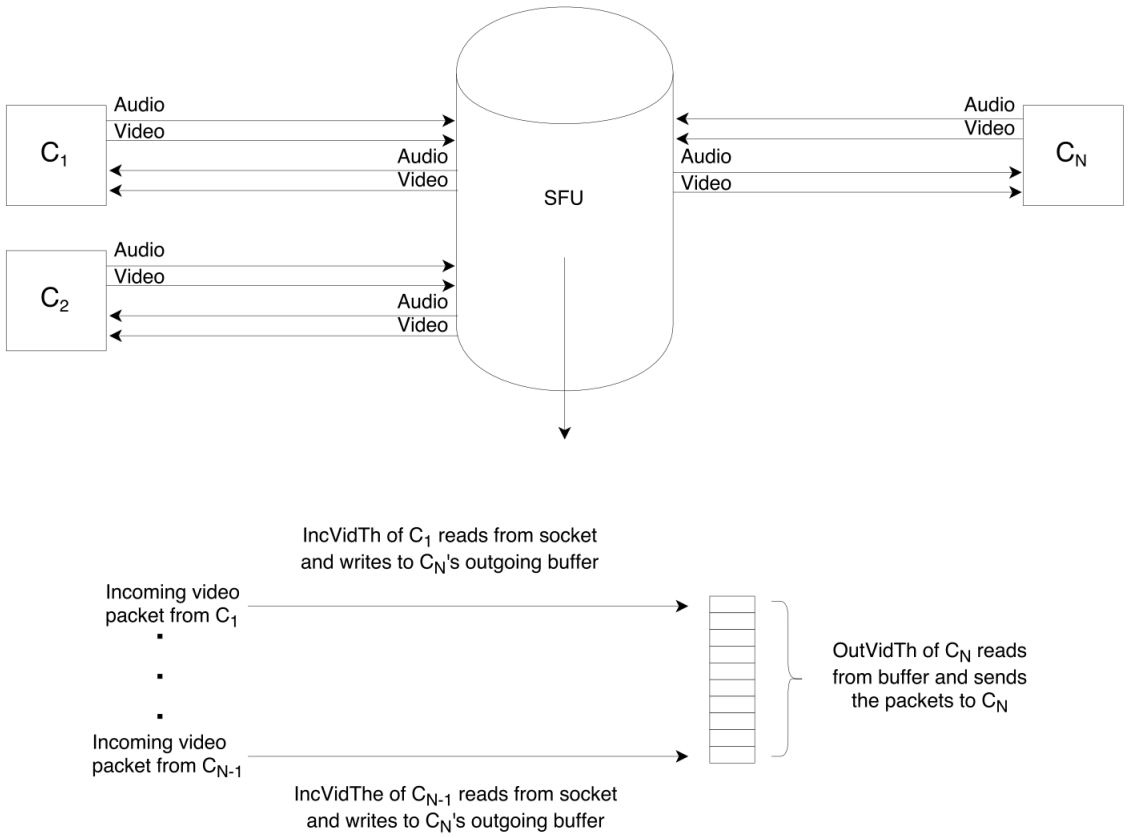
ακεραιότητα του πακέτου, να το τοποθετήσει στον κατάλληλο jitter buffer του χρήστη με το συγκεκριμένο SSRC και, τέλος, να το αποκωδικοποιήσει, όπως θα συνέβαινε σε μια απλή P2P σύνδεση. Όταν το πακέτο αποκωδικοποιηθεί, το νήμα επιστρέφει προκειμένου να πάρει το επόμενο πακέτο ήχου που έχει φτάσει και να εκτελέσει ξανά τη διαδικασία που μόλις περιγράψαμε. Αντίστοιχη διαδικασία θα πραγματοποιηθεί και για τα εισερχόμενα πακέτα βίντεο, όπως φαίνεται και στην Εικόνα 14.

Συνεπώς, στο μοντέλο αυτό ο τερματικός κόμβος δημιουργεί N – 1 στιγμιότυπα για την αποκωδικοποίηση δεδομένων ήχου και N – 1 στιγμιότυπα για την αποκωδικοποίηση δεδομένων βίντεο, καθώς και 2 νήματα (ένα για κάθε τύπο μέσου που λαμβάνουμε), όπου το κάθε ένα οφείλει να εξυπηρετεί N – 1 διαφορετικούς χρήστες.

4.2.3 Μείωση πολυπλοκότητας

Το μοντέλο που περιγράψαμε παραπάνω είναι ικανό για την επίτευξη συνδιάσκεψης πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας, αλλά δεν καλύπτει τις ανάγκες μας και τους στόχους μας ως προς την πολυπλοκότητα, την καθυστέρηση και την κατανάλωση πόρων. Στοχεύουμε σε ένα μοντέλο στο οποίο ο εξυπηρετητής θα αποφεύγει την επεξεργασία και θα φροντίζει για την καλύτερη δυνατή μεταβίβαση των πακέτων προς τους τερματικούς κόμβους, οι οποίοι θα αναλαμβάνουν εξ’ ολοκλήρου το κομμάτι της επεξεργασίας με την καλύτερη δυνατή κατανομή των πόρων τους συστήματος.

Αρχικά, το μοντέλο του εξυπηρετητή που βασίζεται στη φιλοσοφία ενός MCU αυξάνει ιδιαίτερα την πολυπλοκότητα και την καθυστέρηση μετάδοσης, καθώς κάθε πακέτο ήχου ή βίντεο των χρηστών που συμμετέχουν στη συνδιάσκεψη, αποκωδικοποιείται και κωδικοποιείται ξανά πριν την μεταβίβασή του από τον εξυπηρετητή προς τους τερματικούς κόμβους. Προκειμένου να αποφύγουμε το transcoding, στην υλοποίησή μας επιλέγουμε να ακολουθήσουμε τη φιλοσοφία ενός Selective Forward Unit – SFU [18] εξυπηρετητή, ο οποίος, σε αντίθεση με έναν MCU, επιλέγει ποια δεδομένα χρειάζεται να στείλει σε έναν χρήστη (ανάλογα με τις ανάγκες του – αν χρειάζεται μόνο δεδομένα ήχου, μόνο δεδομένα βίντεο ή και τα δύο), τα οποία απλά μεταβιβάζονται χωρίς να επεξεργάζονται ξανά (Relay Server). Για να μπορέσει ο εξυπηρετητής να λειτουργήσει με αυτή τη φιλοσοφία, απαιτείται οι τερματικοί κόμβοι να χρησιμοποιούν τα ίδια σχήματα κωδικοποίησης, ώστε να είναι σε θέση να μπορούν να αποκωδικοποιήσουν τα δεδομένα που λαμβάνουν. Με τον τρόπο αυτό, μειώνουμε αισθητά την καθυστέρηση μετάδοσης λόγω επεξεργασίας στον εξυπηρετητή. Η μορφή του εξυπηρετητή που χρησιμοποιούμε παρουσιάζεται στην Εικόνα 15.



Εικόνα 15: Διαδικασία SSRC πολυπλεξίας στον τύπου SFU εξυπηρετητή

Όπως ήδη αναφέραμε, βασικός μας στόχος είναι το βάρος της επεξεργασίας να ανατεθεί εξ' ολοκλήρου στους τερματικούς κόμβους και όχι στον εξυπηρετητή. Για μια ταυτόχρονη συνύπαρξη N χρηστών στην ίδια συνδιάσκεψη, κάθε τερματικός κόμβος θα χρειαστεί να πραγματοποιεί N – 1 αποκωδικοποιήσεις δεδομένων ήχου και N – 1 αποκωδικοποιήσεις δεδομένων βίντεο. Στο μοντέλο που παρουσιάσαμε, τις N – 1 αποκωδικοποιήσεις για τον εκάστοτε τύπο μέσου τις αναλαμβάνει μόνο ένα νήμα. Είναι εύκολο να αντιληφθεί κανείς πως για μεγάλο N, η εξυπηρέτηση από μόνο ένα νήμα θα οδηγήσει σε προβληματική αποκωδικοποίηση, καθώς η καθυστέρηση εξυπηρέτησης των πακέτων θα είναι αρκετή για την καταστροφή της συνεχόμενης ροής. Για το λόγο αυτό, στην υλοποίησή μας εφαρμόζουμε έναν πολυνηματικό μηχανισμό, ο οποίος φροντίζει για τη σωστή αξιοποίηση και κατανομή των πόρων του συστήματος και προσφέρει τη δυνατότητα στο σύστημα να εξυπηρετεί τους διαφορετικούς χρήστες ως ξεχωριστές/μεμονωμένες οντότητες, προκειμένου να μειώσουμε την καθυστέρηση εξυπηρέτησης των πακέτων. Τα βασικά χαρακτηριστικά καθώς και η αλγοριθμική παρουσίαση αυτού του πολυνηματικού μηχανισμού θα παρουσιαστούν παρακάτω.

4.3 Αλγοριθμική παρουσίαση του μηχανισμού συνδιάσκεψης πολλαπλών χρηστών στο BareSIP

Το λογισμικό ανοιχτού κώδικα BareSIP δεν παρέχει τους κατάλληλους μηχανισμούς και υποδομές για την ταυτόχρονη συμμετοχή πολλαπλών χρηστών, αλλά περιορίζεται σε P2P συνδέσεις μεταξύ δύο χρηστών. Όταν, λοιπόν, 3 χρήστες επιθυμούν να επικοινωνήσουν μεταξύ τους, τότε ο καθένας από αυτούς θα πρέπει να

πραγματοποιήσει μια P2P σύνδεση με τους υπόλοιπους 2, και συνεπώς να δημιουργήσει 2 στιγμιότυπα κωδικοποίησης δεδομένων ήχου, 2 στιγμιότυπα κωδικοποίησης δεδομένων βίντεο, 2 στιγμιότυπα αποκωδικοποίησης δεδομένων ήχου και 2 στιγμιότυπα αποκωδικοποίησης δεδομένων βίντεο, ενώ θα χρησιμοποιεί και 8 διαφορετικές θύρες (δεδομένου ότι χρησιμοποιείται και το RTCP). Συνεπώς, σε περίπτωση που θέλουν να επικοινωνήσουν ταυτόχρονα N διαφορετικοί χρήστες μεταξύ τους, τότε ο καθένας θα χρειάζεται 2·(N – 1) στιγμιότυπα κωδικοποίησης και 2·(N – 1) στιγμιότυπα αποκωδικοποίησης, ενώ θα χρησιμοποιεί 4·(N – 1) θύρες. Όπως γίνεται εύκολα αντιληπτό, για μεγάλο N η διαδικασία αυτή θα έχει υψηλή πολυπλοκότητα και θα γίνεται σπατάλη και υπερβολική κατανάλωση πόρων.

Έστω ότι έχουμε ένα σύστημα συνδιάσκεψης πολλαπλών χρηστών (εξυπηρετητής – τερματικοί κόμβοι) και τροποποιούμε το BareSIP κατάλληλα ώστε να μπορεί να αποπολυπλέξει τις εισερχόμενες ροές δεδομένων. Από προεπιλογή, το BareSIP δημιουργεί ένα νήμα ανά θύρα και κατ' επέκταση, 1 νήμα για την εισερχόμενη ροή δεδομένων ήχου και 1 νήμα για την εισερχόμενη ροή δεδομένων βίντεο. Τότε, η μορφή του τερματικού κόμβου θα είναι αυτή της Εικόνα 14 και συνεπώς, τα δύο αυτά νήματα θα πρέπει να εξυπηρετήσουν N – 1 χρήστες το καθένα. Το γεγονός αυτό θα οδηγήσει σε προβληματική αποκωδικοποίηση, καθώς η καθυστέρηση εξυπηρέτησης των πακέτων θα είναι αρκετή για την καταστροφή της συνεχόμενης ροής

Για την επίλυση του προβλήματος της μεγάλης κατανάλωσης πόρων, που προκύπτει από την αύξηση του αριθμού των συμμετεχόντων, καθώς και της καταστροφικής καθυστέρησης εξυπηρέτησης λόγω της χρήσης ενός μόνο νήματος ανά θύρα, υλοποιήσαμε στο BareSIP έναν πολυνηματικό μηχανισμό συνδιάσκεψης πολλαπλών χρηστών, ο οποίος στηρίζεται στην τεχνική πολυπλεξίας βάσει του SSRC του κάθε συμμετέχοντος, προκειμένου η μετάδοση και η επεξεργασία να είναι εύκολες, γρήγορες και εύρωστες. Όπως θα δούμε και παρακάτω, σε κάθε χρήστη ανατίθενται 2 μοναδικά νήματα, 1 για την εξυπηρέτηση των εισερχόμενων πακέτων ήχου και 1 για την εξυπηρέτηση των εισερχόμενων πακέτων βίντεο. Επιπλέον, από τη στιγμή που ο εξυπηρετητής δεν πραγματοποιεί transcoding, καθώς ακολουθεί τη φιλοσοφία ενός SFU, είναι απαραίτητο οι τερματικοί κόμβοι να χρησιμοποιούν τα ίδια σχήματα κωδικοποίησης.

Τέλος, ως προς την πολυπλεξία βάσει SSRC, ο εξυπηρετητής που χρησιμοποιούμε δίνει ένα χαρακτηριστικό SSRC σε κάθε τύπο μέσου του κάθε χρήστη, χρησιμοποιώντας τη συνάρτηση συσχέτισης:

$$SSRC\_HXOY(C_i) = i$$
$$SSRC\_BINTEO(C_i) = SSRC\_HXOY(C_i) + 10000 = 10000 + i$$

όπου i ϵ [1, N] και C_i : ο χρήστης i.

Για παράδειγμα, ο χρήστης C_1 θα έχει SSRC ήχου ίσο με 1 και SSRC βίντεο ίσο με 10001. Αντίστοιχα, ο χρήστης C_2 θα έχει SSRC ήχου ίσο με 2 και SSRC βίντεο ίσο με 10002 κ.ο.κ.. Με τον τρόπο αυτό, οι χρήστες χαρακτηρίζονται από ένα μοναδικό ζεύγος (SSRC ήχου, SSRC βίντεο), βοηθώντας τους τερματικούς κόμβους να διαχειρίζονται τους διαφορετικούς χρήστες ως μεμονωμένες οντότητες.

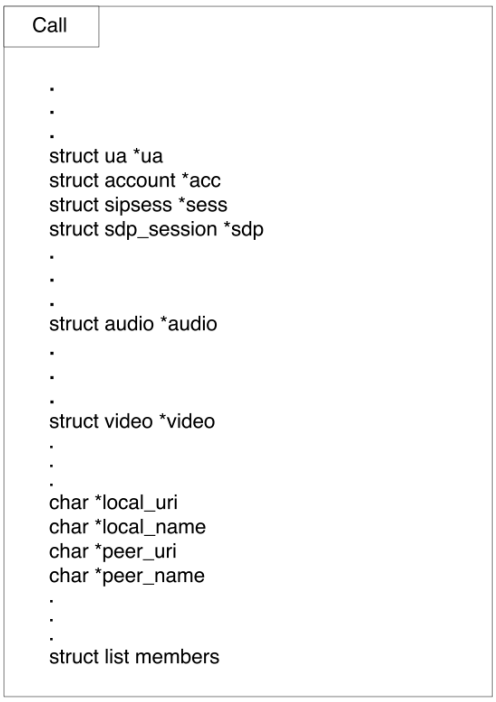
Παρακάτω θα περιγράψουμε αναλυτικά την αλγοριθμική διαδικασία που ακολουθείται στο BareSIP, όπως αυτό τροποποιήθηκε προκειμένου να υποστηρίζει συνδιάσκεψη πολλαπλών χρηστών με χρήση SSRC πολυπλεξίας, από τη στιγμή που ένας χρήστης

επιθυμεί να συμμετάσχει στη συνδιάσκεψη μέχρι τη στιγμή που αποφασίζει να εγκαταλείψει τη συνδιάσκεψη.

4.3.1 Εγκαθίδρυση επικοινωνίας και συμμετοχή στη συνδιάσκεψη

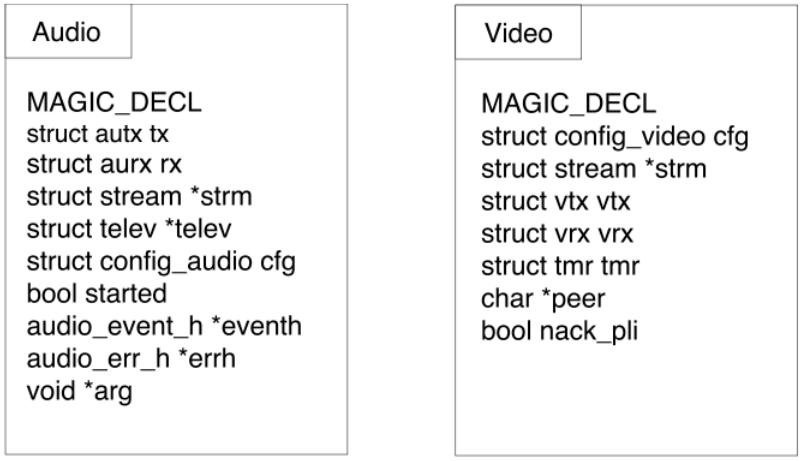
Αρχικά, το BareSIP συμβουλεύεται το αρχείο config για την IP διεύθυνση του μηχανήματος, τη θύρα για τις εισερχόμενες SIP κλήσεις που θα δέχεται, τα πρωτόκολλα επικοινωνίας και μεταφοράς που θα χρησιμοποιήσει (UCP, RTP, RTCP κλπ), το ρυθμό κωδικοποίησης ήχου, τα κανάλια, το εύρος ζώνης, τους κωδικοποιητές ήχου και βίντεο που θέλουμε να ενεργοποιήσουμε κ.ά.. Προκειμένου να τεθεί σε λειτουργία ο μηχανισμός που υλοποιήσαμε, προσθέσαμε στο αρχείο config ένα πεδίο με όνομα ssrc_mux, το οποίο αν έχει την τιμή YES, τότε θα κληθούν οι συναρτήσεις και οι διαδικασίες του μηχανισμού μας. Αλλιώς, αν δηλαδή έχει την τιμή NO, τότε το BareSIP θα λειτουργήσει σύμφωνα με τις προεπιλεγμένες ρυθμίσεις του.

Αφού δημιουργηθούν οι απαραίτητες υποδομές, το σύστημα είναι πλέον έτοιμο να πραγματοποιήσει και να διεκπεραιώσει κλήσεις για τη μεταφορά δεδομένων ήχου και βίντεο. Για να μπορέσει ένας τερματικός κόμβος να συμμετάσχει στη συνδιάσκεψη πρέπει αρχικά να πραγματοποιήσει μια SIP κλήση με τον εξυπηρετητή, προκειμένου να γνωστοποιηθούν οι IP διευθύνσεις τους, τα πρωτόκολλα επικοινωνίας που θα χρησιμοποιήσουν για τη μεταφορά δεδομένων (UDP/RTP/RTCP), τους τύπους μέσων που θα μεταφέρουν (ήχος/βίντεο), τις αντίστοιχες θύρες στις οποίες θα δέχονται τα δεδομένα αυτά κ.τ.λ.. Όταν ολοκληρωθεί η ανταλλαγή SIP μηνυμάτων με τον εξυπηρετητή και πλέον έχουμε όλες τις πληροφορίες που χρειαζόμαστε, τότε το BareSIP δημιουργεί τις κατάλληλες δομές για να στέλνει και να λαμβάνει δεδομένα ήχου και βίντεο. Αρχικά, δεσμεύει μια θύρα για τα εισερχόμενα πακέτα ήχου και μια θύρα για τα εισερχόμενα πακέτα βίντεο. Σε κάθε μία από αυτές τις θύρες αναθέτει από ένα νήμα, το οποίο είναι υπεύθυνο για τη λήψη του πακέτου από το socket, την αποκωδικοποίησή του και τέλος την αναπαραγωγή του. Έπειτα, δημιουργεί τη γενική δομή call (Εικόνα 16), η οποία περιέχει όλες τις πληροφορίες που χαρακτηρίζουν την κλήση, όπως τους SIP User-Agents, τα SIP και SDP sessions, τα τοπικά δικτυακά χαρακτηριστικά (URI), τα δικτυακά χαρακτηριστικά του εξυπηρετητή, διάφορες μετρήσεις, κ.ά.. Στη δομή αυτή, προσθέτουμε μια επιπλέον λίστα με όνομα members, η οποία θα περιέχει όλους τους συμμετέχοντες της συνδιάσκεψης.



Εικόνα 16: Μερικά πεδία της δομής call

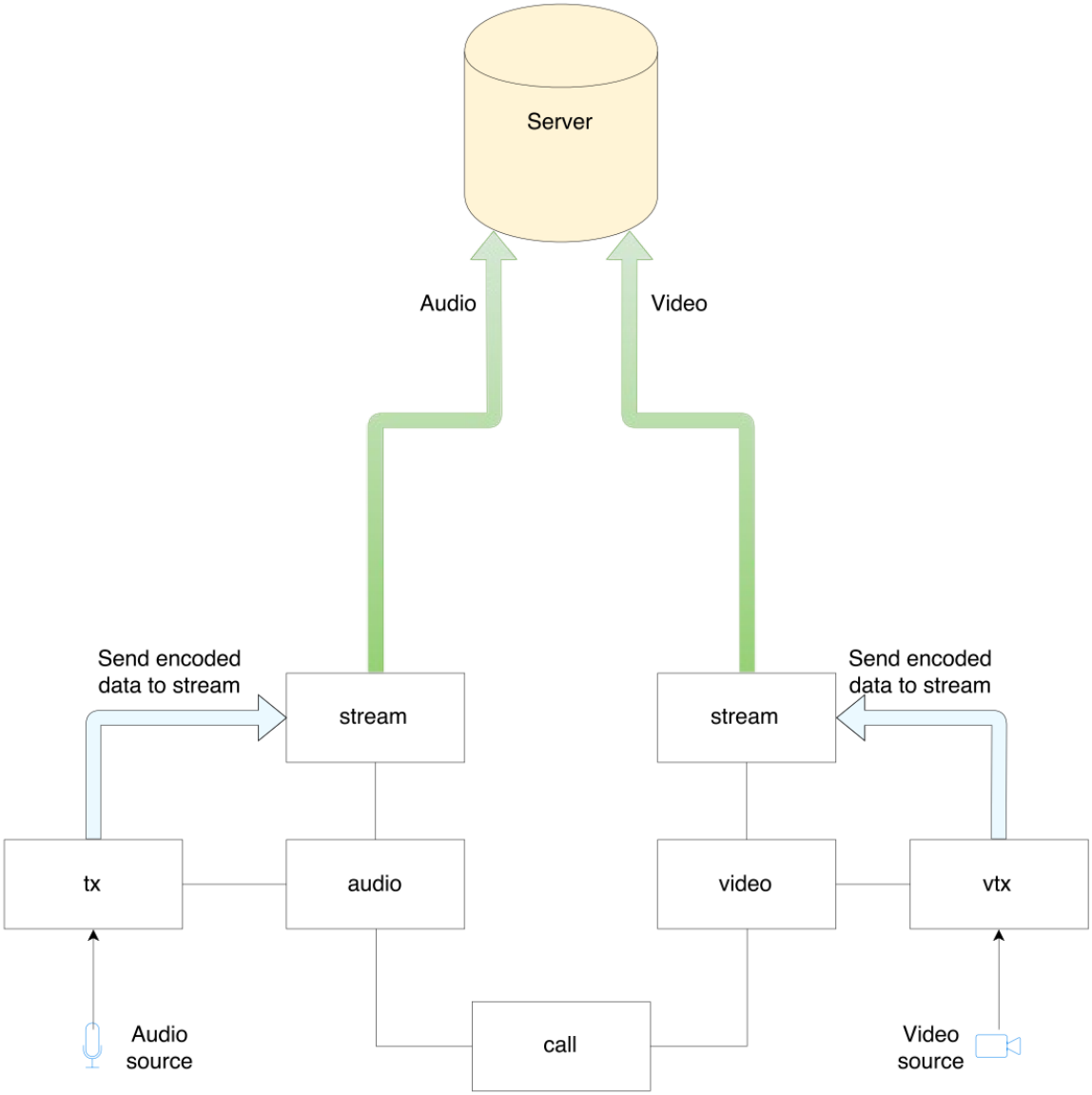
Μεταξύ άλλων, η δομή call περιέχει και τις δομές audio και video, οι οποίες χαρακτηρίζουν πλήρως τα ρεύματα ήχου και βίντεο που στέλνουμε και λαμβάνουμε. Η δομή audio (Εικόνα 17) αποτελείται από τις δομές autx, που αφορά την κωδικοποίηση και την αποστολή δεδομένων ήχου, aurx, που αφορά την αποκωδικοποίησης των δεδομένων ήχου που λαμβάνουμε, stream, που αφορά το ρεύμα δεδομένων ήχου που στέλνουμε και λαμβάνουμε, καθώς και κάποια ακόμα πεδία. Η δομή autx περιέχει τις δομές ausrc_st, που αφορά την πηγή ήχου, aucodec που αφορά τον κωδικοποιητή ήχου που χρησιμοποιούμε, aubuf που περιέχει τα πακέτα ήχου που παράγει ο κωδικοποιητής, mbuf που περιέχει τα RTP πακέτα προς αποστολή κ.ά., καθώς και ένα νήμα το οποίο είναι αυτό που αναλαμβάνει την κωδικοποίηση. Η δομή aurx περιέχει τις δομές auplay_st, που αφορά την αναπαραγωγή ήχου, aucodec που αφορά τον αποκωδικοποιητή ήχου που χρησιμοποιούμε, aubuf που περιέχει τα πακέτα ήχου για αναπαραγωγή κ.ά.. Παρόμοιες πληροφορίες με την audio περιλαμβάνει και η δομή video (vtx, vrx, stream κλπ), καθώς και οι επιμέρους δομές της (vtx: vidcodec, videnc_state, vidframe, mbuf κλπ - vrx: vidcodec, viddec_state, vidisp_st κλπ), όπως φαίνεται στην Εικόνα 17.



Εικόνα 17: Οι δομές audio και video

Όταν το BareSIP δημιουργεί οποιαδήποτε από τις δομές audio ή video, επειδή περιορίζεται σε P2P συνδέσεις μεταξύ δύο χρηστών, δημιουργεί αμέσως ένα stream για την αποστολή και τη λήψη δεδομένων, αλλά και ένα στιγμιότυπο κωδικοποιητή (tx, vtx) και ένα στιγμιότυπο αποκωδικοποιητή για το ίδιο stream. Ωστόσο, στη συνδιάσκεψη πολλαπλών χρηστών δε θα δεχόμαστε δεδομένα μόνο από ένα χρήστη, αλλά από N – 1 διαφορετικούς χρήστες. Συνεπώς πρέπει να διαχωρίσουμε το stream στο οποίο στέλνουμε δεδομένα από το stream που λαμβάνουμε δεδομένα, προκειμένου να μπορούμε να αποκωδικοποιούμε τα δεδομένα του κάθε χρήστη ξεχωριστά. Θέτοντας τη μεταβλητή ssrc_mux σε YES, το BareSIP φροντίζει στο σημείο αυτό να δημιουργήσει μόνο τα στιγμιότυπα των κωδικοποιητών, και όχι και των αποκωδικοποιητών. Επιπλέον, θέτει ως stream για αποστολή δεδομένων ήχου το stream της δομής audio και ως stream για αποστολή δεδομένων βίντεο το stream της δομής video. Έπειτα, αφού δημιουργήσει τις δομές call, audio, video αλλά και τα στιγμιότυπα tx, vtx, ενεργοποιεί τους κωδικοποιητές και ξεκινάει η αποστολή δεδομένων ήχου και βίντεο προς τον εξυπηρετητή, ο οποίος με τη σειρά του τα μεταβιβάζει στους υπόλοιπους συμμετέχοντες της συνδιάσκεψης.

Συνοψίζοντας, στο σημείο αυτό το BareSIP έχει εκκινήσει μια SIP κλήση με τον εξυπηρετητή, έχει δημιουργήσει τη δομή call που χαρακτηρίζει την κλήση και περιέχει μεταξύ άλλων τις δομές audio και video. Σε κάθε μία από αυτές τις δομές (audio, video) έχει αντιστοιχίσει ένα stream για τα κωδικοποιημένα δεδομένα που θα στείλει στον εξυπηρετητή. Επιπλέον, έχει δημιουργήσει 2 στιγμιότυπα κωδικοποιητών (ένα για ήχο και ένα για βίντεο), τα οποία εκτελούνται ανεξάρτητα πάνω σε δικά τους νήματα. Τα στιγμιότυπα αυτά κωδικοποιούν τα δεδομένα από τη συσκευή εισόδου ήχου και τη συσκευή εισόδου βίντεο και τα στέλνουν μέσω του αντίστοιχου stream στον εξυπηρετητή. Τα στιγμιότυπα των αποκωδικοποιητών, όπως θα δούμε αργότερα, δημιουργούνται στο σημείο της αποπολυπλεξίας, όταν δεχθούμε δεδομένα από έναν καινούριο χρήστη. Η κατάσταση στην οποία βρίσκεται αυτή τη στιγμή το BareSIP παρουσιάζεται στην Εικόνα 18.



Εικόνα 18: Η μορφή του BareSIP μετά την εγκαθίδρυση της επικοινωνίας – η διαδικασία λήψης δεδομένων δεν έχει ξεκινήσει ακόμα

4.3.2 Αποπολυπλεξία και πολυνηματισμός

Όπως αναφέραμε, επειδή στη συνδιάσκεψη πολλαπλών χρηστών δε θα δεχόμαστε δεδομένα μόνο από ένα χρήστη, αλλά από N – 1 διαφορετικούς, πρέπει να διαχωρίσουμε το stream στο οποίο στέλνουμε δεδομένα από το stream που λαμβάνουμε δεδομένα, προκειμένου να μπορούμε να αποκωδικοποιήσουμε τα δεδομένα του κάθε χρήστη ξεχωριστά. Συνεπώς, για κάθε χρήστη πρέπει να δημιουργούμε διαφορετικά στιγμιότυπα αποκωδικοποίησης, όπου το κάθε ένα θα χαρακτηρίζεται από το δικό του stream. Επιπλέον, το BareSIP από προεπιλογή δημιουργεί 1 νήμα για την υποδοχή από τη θύρα, την αποκωδικοποίηση και την αναπαραγωγή δεδομένων ήχου και 1 νήμα για την υποδοχή από τη θύρα, την αποκωδικοποίηση και την αναπαραγωγή δεδομένων βίντεο. Ωστόσο, στις 2 αυτές θύρες δε θα δεχόμαστε πλέον δεδομένα από 1 χρήστη αλλά από N – 1. Τη φιλοσοφία αυτή, ότι δηλαδή συγκεκριμένα νήματα εξυπηρετούν τα πακέτα του κάθε χρήστη, θα

πρέπει να τη διατηρήσουμε, προκειμένου η καθυστέρηση εξυπηρέτησης των πακέτων να μην είναι καταστροφική για την αποκωδικοποίηση και την αναπαραγωγή των δεδομένων.

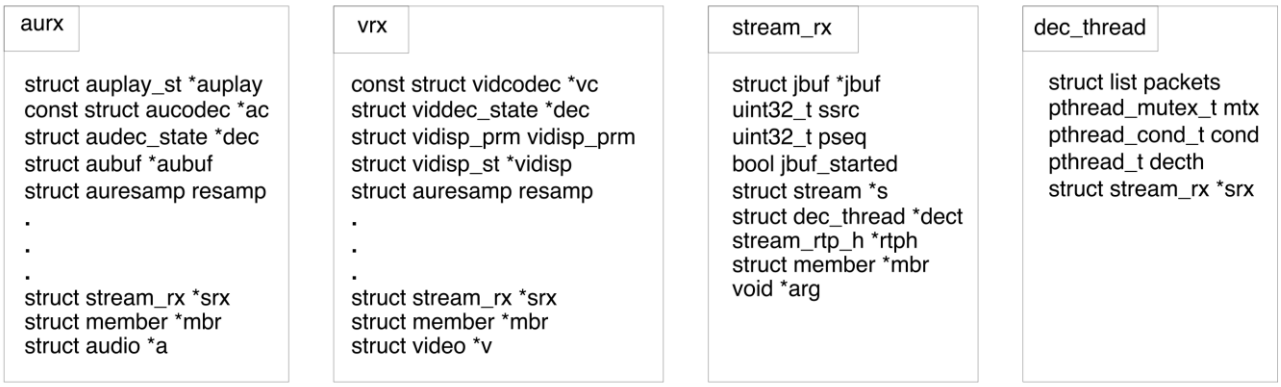
Σε πρώτο στάδιο θα πρέπει να έχουμε μια δομή η οποία θα χαρακτηρίζει πλήρως ένα χρήστη που συμμετέχει στη συνδιάσκεψη ώστε να μπορούμε να τον ξεχωρίζουμε. Επειδή χρησιμοποιούμε SSRC πολυπλεξία, το χαρακτηριστικό κλειδί που θα τον ξεχωρίζει από τους υπόλοιπους συμμετέχοντες είναι το πεδίο SSRC. Συνεπώς, θα πρέπει για κάθε χρήστη να κρατάμε τα ssrc ήχου και βίντεο αλλά και τα στιγμιότυπα αποκωδικοποίησης ήχου και βίντεο που του αντιστοιχούν. Δημιουργούμε, λοιπόν, τη δομή member (Εικόνα 19) η οποία διατηρεί όλες αυτές τις πληροφορίες για κάθε χρήστη που συμμετέχει στη συνδιάσκεψη.

Έπειτα, είπαμε ότι κάθε διαφορετικό στιγμιότυπο αποκωδικοποίησης που δημιουργούμε θα πρέπει να χαρακτηρίζεται από το δικό του stream. Για το λόγο αυτό, στις δομές aurx και vrx

```
Member

uint32_t audio_ssrc
uint32_t video_ssrc
char *uri, *alias
char *pstr

struct {
        /* Audio-related data go here */
        struct aurx *rx
} au

struct {
        /* Video related data go here */
        struct vrx *rx
} vi
```

Εικόνα 19: Η δομή member που χαρακτηρίζει κάθε συμμετέχοντα της συνδιάσκεψης

προσθέτουμε επιπλέον τις δομές stream_rx, member και audio ή video αντίστοιχα. Η δομή member θα δείχνει στο member που αναφέρεται το συγκεκριμένο στιγμιότυπο αποκωδικοποίησης. Αντίστοιχα και οι δομές audio, video θα δείχνουν στις δομές audio ή video που αναφέρεται το συγκεκριμένο στιγμιότυπο αποκωδικοποίησης. Η δομή stream_rx, περιγράφει το stream που λαμβάνουμε δεδομένα. Θα περιέχει τον κατάλληλο jitter buffer (jbuf), το ssrc των πακέτων που εξυπηρετεί, το sequence number του εισερχόμενου RTP πακέτου (pseq), τον αντίστοιχο handler για τη διαχείριση των δεδομένων του πακέτου (rtph) κ.ά.. Βασιζόμενοι στη φιλοσοφία ότι συγκεκριμένα νήματα θα εξυπηρετούν τα πακέτα του κάθε χρήστη, η δομή stream_rx θα περιέχει και τη δομή dec_thread, η οποία θα χαρακτηρίζει ένα νήμα το οποίο είναι υπεύθυνο για την μεταφορά προς αποκωδικοποίηση και αναπαραγωγή των δεδομένων που μεταφέρει το πακέτο. Πιο συγκεκριμένα, η δομή αυτή θα περιέχει μια λίστα RTP πακέτων, απ’ όπου θα διαβάζει τα εισερχόμενα πακέτα, έναν mutex και ένα conditional variable για το συγχρονισμό πάνω στη λίστα, το νήμα για την αποκωδικοποίηση, μια μεταβλητή για να διατηρείται σε επαναληπτική τροχιά και, τέλος, ένα δείκτη προς το stream_rx στο οποίο αναφέρεται. Οι δομές aurx, vrx, stream_rx και dec_thread φαίνονται στην Εικόνα 20.



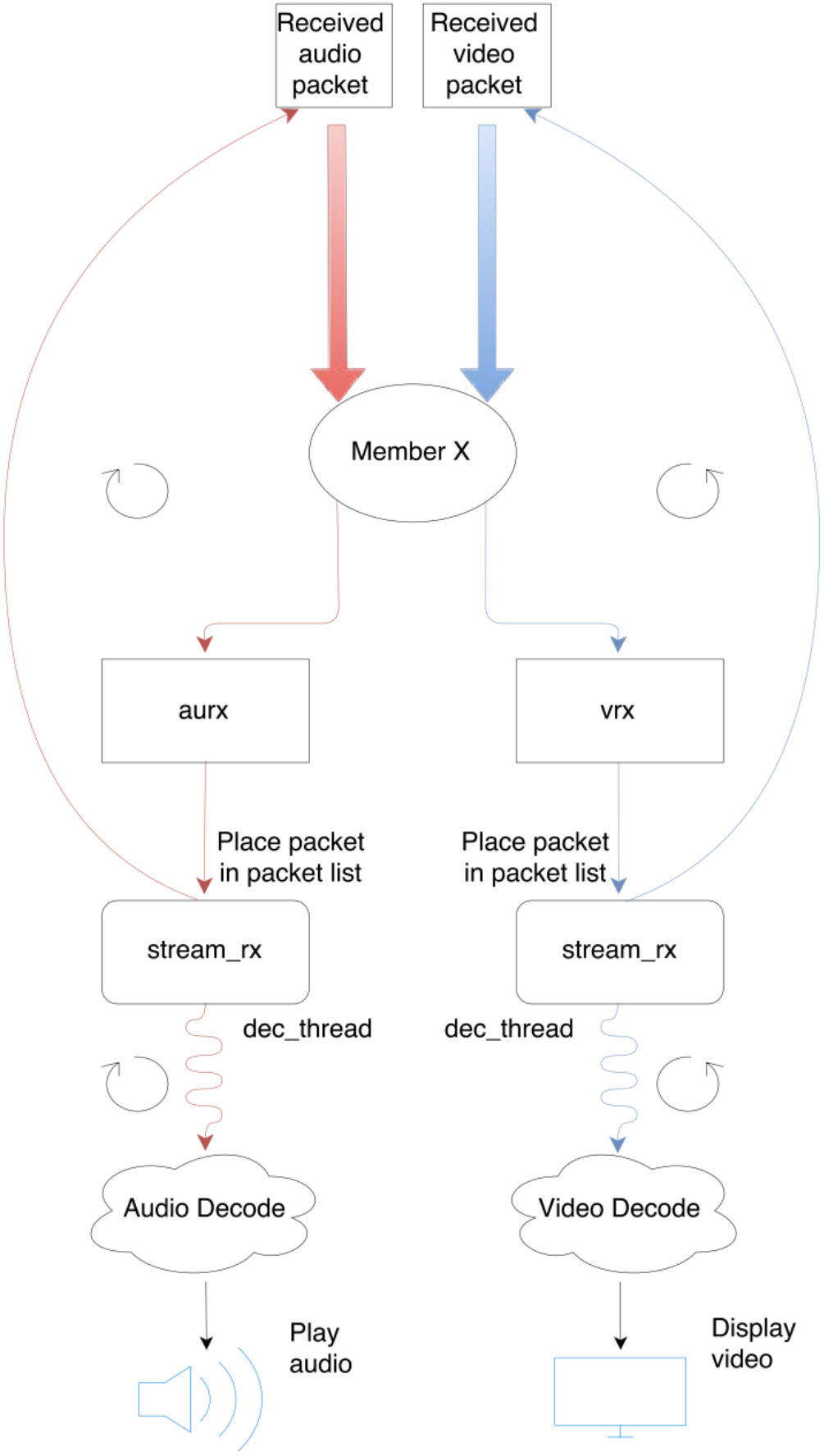
Εικόνα 20: Οι δομές aurx, vrx, stream_rx και dec_thread

Όπως αναφέραμε προηγουμένως, στη δομή call προσθέσαμε μια λίστα από members. Σύμφωνα με τα χαρακτηριστικά του member μπορούμε πλέον να προσδιορίσουμε από ποιο χρήστη προέρχεται το πακέτο που λαμβάνουμε, αναζητώντας στη λίστα members της call το χρήστη που χαρακτηρίζει το ssrc του πακέτου. Ιδανικά, θα θέλαμε την εξυπηρέτηση του κάθε χρήστη να την αντιμετωπίσουμε όπως αντιμετωπίζει το BareSIP μια μεμονωμένη P2P σύνδεση, όπου για κάθε τύπο μέσων που λαμβάνει έχει δημιουργήσει ένα μοναδικό στιγμιότυπο αποκωδικοποίησης, το οποίο θα το διαχειρίζεται ένα επίσης μοναδικό νήμα. Αν, λοιπόν, περιορίσουμε το πρόβλημα στην εξυπηρέτηση ενός μόνο χρήστη και με δεδομένη τη μορφή και τη λειτουργία των δομών aurx, vrx, stream_rx, και dec_thread, τότε το μοντέλο που επιθυμούμε να δημιουργήσουμε είναι αυτό της Εικόνα 21.

Για το μοντέλο αυτό, μελετάμε την άφιξη ενός πακέτου ήχου (κόκκινη διαδρομή): έστω ότι στο BareSIP φτάνει ένα πακέτο ήχου από το χρήστη Χ το οποίο χαρακτηρίζεται από ένα συγκεκριμένο SSRC. Για το χρήστη αυτό, πηγαίνουμε στο στιγμιότυπο αποκωδικοποίησής του aurx, εντοπίζουμε το stream_rx, που αφορά τα πακέτα ήχου που λαμβάνουμε από αυτό το χρήστη, και αντιγράφουμε το εισερχόμενο πακέτο ήχου στην packet list που διαθέτει η δομή dec_thread που του αντιστοιχεί. Αφού ολοκληρώσουμε τη διαδικασία αυτή, τότε ενημερώνουμε ότι τοποθετήσαμε ένα πακέτο στη λίστα και επιστρέφουμε πίσω προκειμένου να δεχτούμε το επόμενο πακέτο ήχου. Το νήμα της δομής dec_thread με τη σειρά του, παρατηρεί ότι η λίστα δεν είναι άδεια, παίρνει ένα πακέτο από αυτή και στη συνέχεια το προωθεί για αποκωδικοποίηση. Αφού γίνει η αποκωδικοποίηση, παίρνει τα αποκωδικοποιημένα δείγματα, τα τοποθετεί στον κατάλληλο buffer και από κει και πέρα είναι δουλεία εσωτερικών handlers να μεταβιβάσου τα δείγματα μέχρι τη συσκευή αναπαραγωγής ήχου. Αντίστοιχη διαδικασία πραγματοποιείται και στην περίπτωση που λάβουμε ένα πακέτο βίντεο (μπλε διαδρομή).

Έχοντας ως γνώμονα το μοντέλο της Εικόνας 21, επεκτείνουμε την υλοποίηση σε επίπεδο Ν χρηστών και φροντίζουμε κατάλληλα, προκειμένου η συνδιάσκεψη πολλαπλών χρηστών στο BareSIP να πραγματοποιείται χωρίς επιπλέον καθυστέρηση εξυπηρέτησης σε σχέση με την προεπιλεγμένη λειτουργία του. Πρακτικά, δημιουργούμε έναν πολυνηματικό μηχανισμό, όπου πλέον τα 2 νήματα που δημιουργεί το BareSIP δεν είναι υπεύθυνα για την υποδοχή των πακέτων από τη θύρα, την αποκωδικοποίηση και την αναπαραγωγή τους, αλλά αναλαμβάνουν το ρόλο του αποπολυπλέκτη βάσει του SSRC. Επιπλέον, για κάθε χρήστη δημιουργούμε 2 ανεξάρτητα στιγμιότυπα κωδικοποίησης, τη διαχείριση των οποίων την αναλαμβάνουν 2 επίσης ανεξάρτητα νήματα. Η μεταφορά των πακέτων από τα νήματα του BareSIP στα νήματα του κάθε

χρήστη γίνεται μέσω μιας λίστας, ο συγχρονισμός της οποίας ακολουθεί το μοντέλο Readers – Writers. Παρακάτω, ακολουθεί η αλγοριθμική αναπαράσταση της διαδικασίας αυτής, από τη στιγμή που φτάνει ένα πακέτο ήχου από τον εξυπηρετητή, μέχρι και την αναπαραγωγή του στην έξοδο (η διαδικασία για τα πακέτα βίντεο είναι αντίστοιχη) :



Εικόνα 21: Διαδικασία εξυπηρέτησης ενός χρήστη X

1. Έστω λοιπόν ότι φτάνει μέσω UDP ένα πακέτο ήχου στο socket. Το νήμα του BareSIP που είναι υπεύθυνο για την υποδοχή των πακέτων ήχου από τη θύρα, την αποκωδικοποίηση και την αναπαραγωγή, αναλαμβάνει να το παραλάβει.
2. Έπειτα, καλεί τον κατάλληλο handler (udp_recv_handler) που αφορά τα RTP πακέτα, προκειμένου να ξεχωρίσει την RTP επικεφαλίδα από το ωφέλιμο φορτίο.
3. Στο επόμενο βήμα, το νήμα αυτό θα έπρεπε να καλέσει τον handler rtp_recv αλλά, λόγω ότι έχουμε θέσει σε YES το ssrc_mux στο αρχείο config, ο handler που καλεί είναι ο rtp_recv_ssrc_mux. Η συνάρτηση rtp_recv_ssrc_mux είναι το πιο κομβικό σημείο της υλο-ποίησής μας, καθώς αποτελεί τον αποπολυπλέκτη των εισερχόμενων ροών και φροντίζει ώστε τα πακέτα να προωθηθούν προς τα στιγμιότυπα αποκωδικοποίησης του κάθε χρήστη σωστά.
4. Σε πρώτο στάδιο, η rtp_recv_ssrc_mux καλεί τη συνάρτηση call_find_member
    - Ελέγχει αν στη λίστα members της call υπάρχει κάποιος χρήστης με ssrc ήχου ίδιο με αυτό του πακέτου ήχου που μόλις έφτασε.
    - Αν ο χρήστης βρεθεί, τότε τον επιστρέφει, αλλιώς επιστρέφει έναν κενό δείκτη (NULL).
5. Αν επιστρέψει NULL, καλείται η συνάρτηση call_create_member
    - Καλεί τη member_alloc
        - Δημιουργεί ένα αντικείμενο τύπου member. Στο αντικείμενο αυτό αναθέτει στο πεδίο του audio_ssrc την τιμή του ssrc που έχει το πακέτο, ενώ στο video_ssrc αναθέτει την τιμή του ssrc που έχει το πακέτο + 10000, σύμφωνα με τη συνάρτηση συσχέτισης που χρησιμοποιεί ο εξυπηρετητής.
        - Έπειτα, καλεί την audio_rx_alloc
            - Δημιουργεί ένα στιγμιότυπο αποκωδικοποίησης ήχου aurx για το member αυτό.
            - Για αυτό το aurx δημιουργεί ένα stream_rx
                - Δημιουργεί ένα αντικείμενο τύπου dec_thread.
                - Αρχικοποιεί τον mutex, την conditional variable και θέτει τη μεταβλητή running σε false.
        - Αφού ολοκληρώσει, με το aurx, καλεί τη video_rx_alloc η οποία κάνει τις αντίστοιχες διαδικασίες για το vrx του χρήστη.
    - Στη συνέχεια, το νέο αυτό member που δημιουργήθηκε το τοποθετεί στη λίστα members της call.
    - Για το member αυτό καλεί την member_stream_start
        - Καλεί την audio_rx_decoder_set
            - Δημιουργεί όλες τις κατάλληλες διαδικασίες για την αποκωδικοποίηση (αποκωδικοποιητής, handlers, παράμετροι κ.λπ.).
            - Καλεί την audio_rx_start
                - Ρυθμίζει τα audio filters.
                - Καλεί τη stream_rx_start, η οποία δημιουργεί το νήμα της δομής dec_thread και του αναθέτει τη συνάρτηση decode_thread,

στην οποία το νήμα περιμένει να γραφτεί ένα πακέτο στη λίστα packets.

- Καλεί τη start_player, η οποία ενεργοποιεί έναν audio player.

- Καλεί τη video_rx_decoder_set, η οποία πραγματοποιεί τις αντίστοιχες διαδικασίες για το vrx του χρήστη.

6. Επιστρέφοντας στην rtp_recv_ssrc_mux, από το member που είτε βρήκαμε εξ' αρχής είτε το δημιουργήσαμε παίρνουμε το stream_rx της δομής aurx (αν το πακέτο ήταν βίντεο θα παίρναμε το stream_rx της δομής vrx).
7. Τέλος, για αυτό το stream_rx, καλεί τη συνάρτηση stream_rx_add_packet. Στο σημείο αυτό αλλάζει η φιλοσοφία του BareSIP, καθώς το νήμα που υποδέχτηκε το πακέτο δεν αναλαμβάνει πλέον την αποκωδικοποίηση και την αναπαραγωγή του, αλλά τις διαδικασίες αυτές τις αναθέτει στο κατάλληλο νήμα της δομής aurx. Συνεπώς, η stream_rx_add_packet
    - Ελέγχει αν το νήμα της dec_thread είναι ενεργοποιημένο.
        - Αν είναι, τότε καλεί την dec_thread_add_packet
            - Καλεί την packet_alloc, η οποία
                - Δημιουργεί ένα packet.
                - Του αναθέτει το ωφέλιμο φορτίο και την RTP επικεφαλίδα.
            - Κάνει lock τον mutex του νήματος.
            - Τοποθετεί στο τέλος της λίστας packets του νήματος το packet.
            - Κάνει unlock τον mutex του νήματος.
            - Στέλνει signal με βάση την conditional variable στο νήμα ότι τοποθετήθηκε ένα πακέτο στη λίστα.
        - Αλλιώς, αν το dec_thread δεν είναι ενεργοποιημένο ή η dec_thread_add_packet αποτύχει, τότε
            - Κάνει επί τόπου συγχρονισμένη/εξαναγκαστική αποκωδικοποίηση του πακέτου.
8. Η συνάρτηση rtp_recv_ssrc_mux επιστρέφει.
9. Το νήμα του BareSIP επιστρέφει προς τα πίσω, προκειμένου να ελέγξει αν έφτασε στο socket κάποιο νέο πακέτο ήχου.

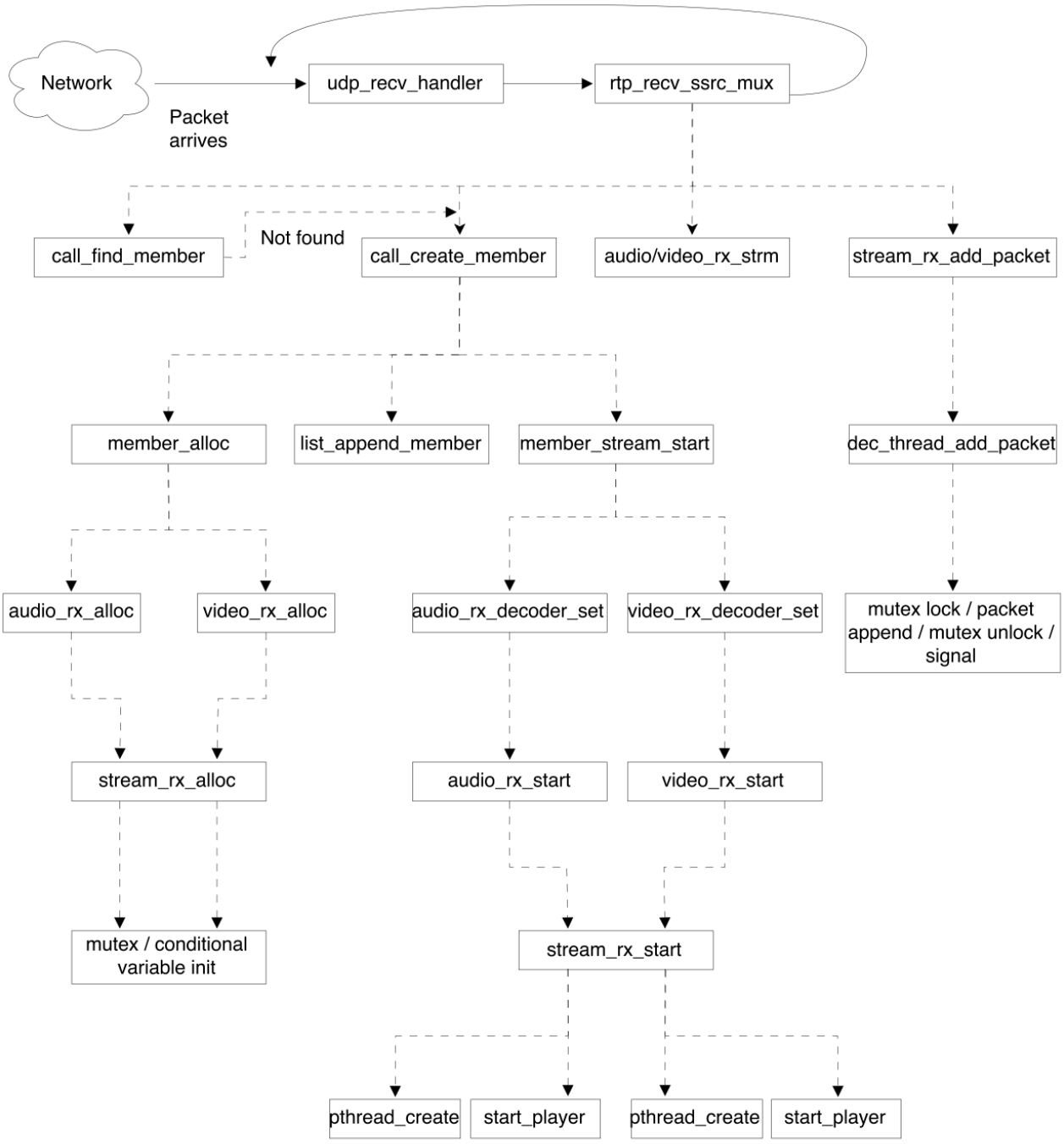
Οι κλήσεις των συναρτήσεων κατά την αποπολυπλεξία πακέτων ήχου και βίντεο φαίνονται στην Εικόνα 22, ενώ η κατάσταση του BareSIP κατά τη διάρκεια της διαδικασίας αυτής φαίνεται στην Εικόνα 23.

Όταν το νήμα του aurx δεχθεί το signal από το νήμα του BareSIP τότε

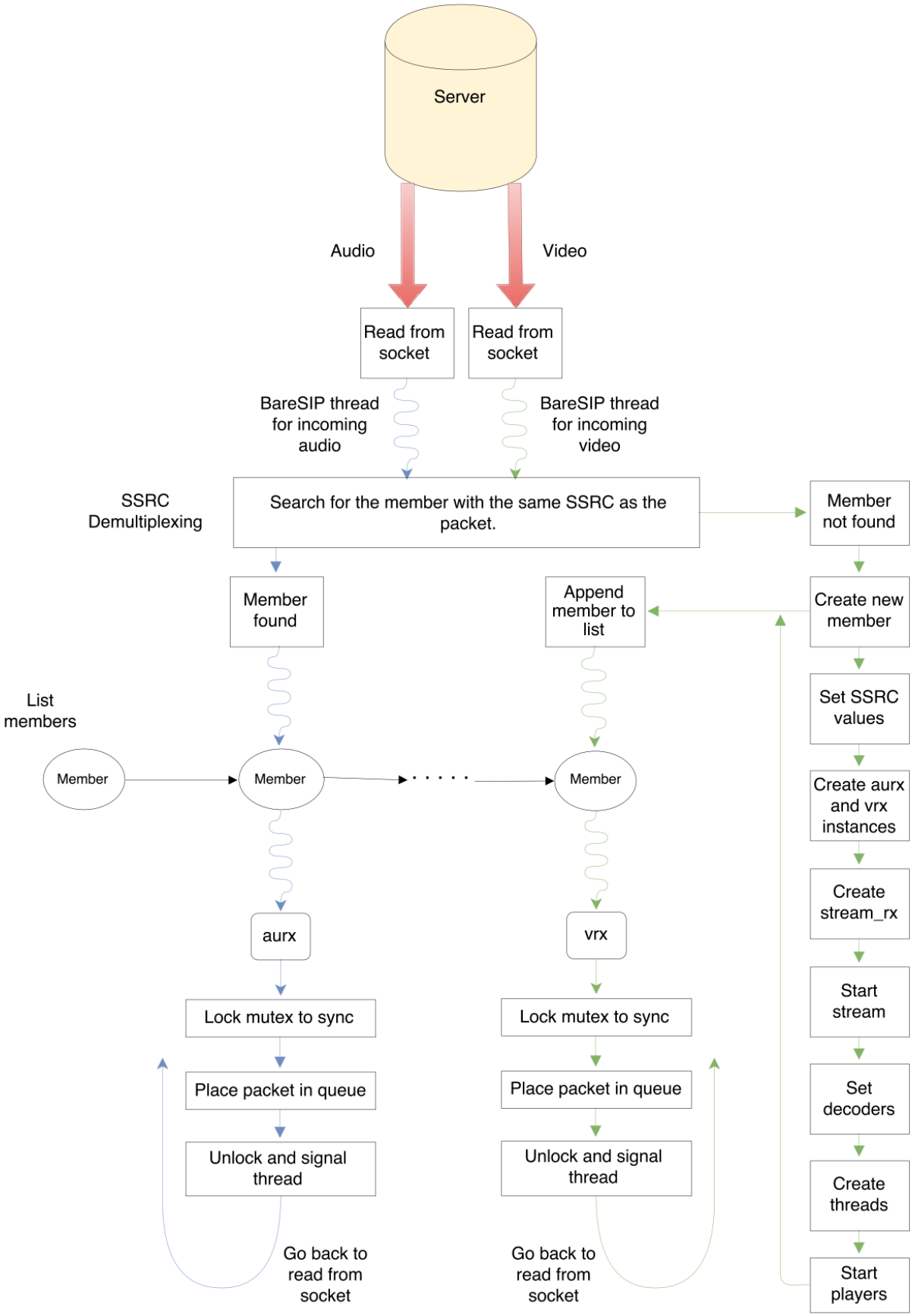
1. Παίρνει από τη λίστα packets το πρώτο packet
2. Καλεί τη συνάρτηση stream_rx_decode
    - Τοποθετεί τα δεδομένα στον jitter buffer στην κατάλληλη θέση.
    - Παίρνει από το jitter buffer τα δεδομένα που βρίσκονται στην αρχή του.

- Αποκωδικοποιεί τα δεδομένα, τα τοποθετεί σε έναν buffer (aubuf) και επιστρέφει για να παραλάβει το επόμενο packet, καθώς από το σημείο αυτό και έπειτα είναι δουλειά εσωτερικών handlers να αναλάβουν την αναπαραγωγή των δεδομένων.

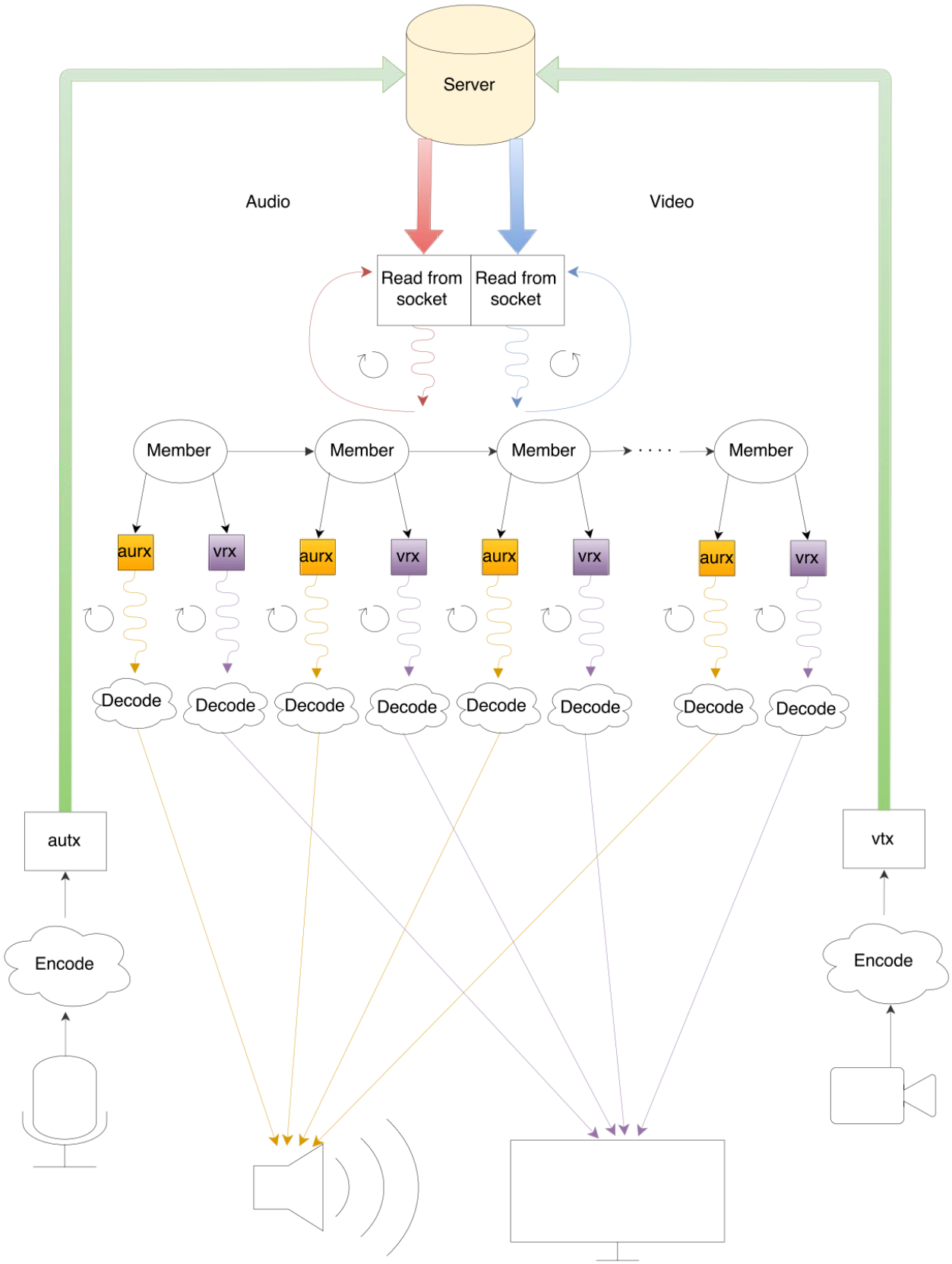
Συνεπώς, η τελική μορφή που παίρνει το BareSIP όταν βρίσκεται σε λειτουργία συνδιάσκεψης παρουσιάζεται στην Εικόνα 24.



Εικόνα 22: Διάγραμμα κλήσης συναρτήσεων κατά την αποπολυπλεξία



Εικόνα 23: SSRC αποπολυπλεξία και ανάθεση της εξυπηρέτησης των πακέτων στα αντίστοιχα νήματα των χρηστών



Εικόνα 24: Η μορφή του BareSIP όταν βρίσκεται σε λειτουργία συνδιάσκεψης

4.3.3 Τερματισμός συνόδου συνδιάσκεψης

Σε περίπτωση που ένας τερματικός κόμβος επιθυμεί να αποχωρήσει από τη συνδιάσκεψη, η διαδικασία που ακολουθείται είναι ιδιαίτερα απλή. Αρχικά στέλνει ένα RTCP BYE μήνυμα προς τον εξυπηρετητή, για να ενημερώσει ότι αποχωρεί από τη συνδιάσκεψη. Στη συνέχεια, για κάθε ένα χρήστη της λίστας members σταματάει τα νήματα για τα πακέτα ήχου και τα πακέτα βίντεο, κλείνει όλα τα στιγμιότυπα αποκωδικοποίησης που έχει ανοιχτά και τέλος καθαρίζει όλους τους πόρους που έχει δεσμεύσει.

Ο εξυπηρετητής, με τη σειρά του, φροντίζει το RTCP BYE μήνυμα να το μεταδώσει σε όλους τους υπόλοιπους N – 1 χρήστες της συνδιάσκεψης και στη συνέχεια καθαρίζει όλους τους πόρους που σχετίζονται με το χρήστη που αποχωρεί (νήματα, buffers κλπ) και κλείνει τις θύρες που είχε ανοίξει.

Τέλος, ο κάθε ένας από τους υπόλοιπους N – 1 τερματικούς κόμβους, όταν δεχτεί το RTCP BYE από τον εξυπηρετητή, διαβάζει μέσα στο μήνυμα τα SSRC του χρήστη που αποχωρεί από τη συνδιάσκεψη, τον εντοπίζει στη λίστα members, σταματάει τα νήματα για τα πακέτα ήχου και τα πακέτα βίντεο του συγκεκριμένου χρήστη, κλείνει όλα τα στιγμιότυπα αποκωδικοποίησης που έχει ανοιχτά για το συγκεκριμένο χρήστη και τέλος καθαρίζει όλους τους αντίστοιχους πόρους που έχει δεσμεύσει.

5. ΕΥΡΩΣΤΗ ΜΕΤΑΔΟΣΗ ΜΕ ΑΝΑΜΕΤΑΔΟΣΗ RTP ΠΑΚΕΤΩΝ ΒΙΝΤΕΟ

Σε εφαρμογές πραγματικού χρόνου είναι ιδιαίτερα σημαντικό να υπάρχει όσο το δυνατόν πιο σταθερή ροή δεδομένων, χωρίς απώλειες ή καταστροφή πακέτων, προκειμένου η εμπειρία στην έξοδο του χρήστη να είναι πλήρως ικανοποιητική. Αντίστοιχα, κατά τη μετάδοση βίντεο σε πραγματικό χρόνο, επιθυμούμε η εικόνα που λαμβάνει ο χρήστης να είναι ακέραια, καθώς η υψηλή καθυστέρηση και οι απώλειες πακέτων μπορούν να οδηγήσουν στην καταστροφή της [19]. Ως προς την καθυστέρηση, πέραν του δικτύου, οφείλουμε να μελετήσουμε τις τεχνικές κωδικοποίησης και συμπίεσης που χρησιμοποιούμε, καθώς μπορούν να συμβάλουν στη μείωση του όγκου της πληροφορίας προκειμένου να χρειάζεται να μεταδώσουμε μικρότερα πακέτα τα οποία φτάνουν πιο γρήγορα και είναι εύκολο να ανακτηθούν σε περίπτωση απώλειας. Έπειτα, ως προς τις απώλειες, οι τεχνικές αποκωδικοποίησης που χρησιμοποιούμε μπορούν να μας προσφέρουν μηχανισμούς, οι οποίοι θα φροντίζουν κατάλληλα ώστε, παρά τις απώλειες πακέτων, είτε να μπορούν να αναπαράγουν εικόνα είτε να φροντίζουν για την άμεση αποκατάστασή τους. Στην ενότητα αυτή θα μελετήσουμε το πρότυπο συμπίεσης βίντεο H.264/AVC, το οποίο με τις τεχνικές που χρησιμοποιεί βελτιώνει την εμπειρία του χρήστη, καθώς και την επέκταση SVC η οποία, μέσω της κλιμακωτής κωδικοποίησης, προσφέρει τη δυνατότητα στο σύστημα να επανέρχεται ύστερα από απώλειες πακέτων. Τέλος, βασιζόμενοι στις δυνατότητες που προσφέρει το SVC, θα παρουσιάσουμε την υλοποίηση ενός μηχανισμού αναμετάδοσης RTP πακέτων βίντεο, ο οποίος φροντίζει για την άμεση αποκατάσταση απολεσθέντων πακέτων, τα οποία είναι απαραίτητα για την ακέραια διατήρηση της ροής δεδομένων βάσης, ώστε η μετάδοση να είναι εύρωστη, αποφεύγοντας την καταστροφή της εικόνας στην έξοδο.

5.1 Το πρότυπο συμπίεσης βίντεο H.264/AVC

Το H.264/AVC (Advanced Video Coding) [20] πρόκειται για ένα πρότυπο συμπίεσης βίντεο το οποίο έχει ένα ευρύ φάσμα εφαρμογών που καλύπτει όλες τις μορφές συμπιεσμένου ψηφιακού βίντεο με σχεδόν χωρίς απώλειες κωδικοποίησης. Η μορφή του ωφέλιμου φορτίου του (payload format) έχει μεγάλη εφαρμογή, καθώς υποστηρίζει από απλές εφαρμογές συνομιλίας χαμηλού ρυθμού bit (bit rate) μέχρι εφαρμογές μετάδοσης βίντεο υψηλού ρυθμού bit. Βασική ιδιαιτερότητα του κωδικοποιητή αποτελεί η πακετοποίηση από ένα ή περισσότερα Network Abstract Layer Units (NALUs), τα οποία παράγονται από τον κωδικοποιητή H.264 σε κάθε RTP ωφέλιμο φορτίο.

5.1.1 Βασικά χαρακτηριστικά

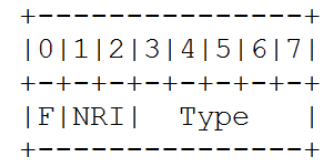
Θα εστιάσουμε σε τρία βασικά χαρακτηριστικά του H.264, όπως αυτά ορίζονται στο RFC 6184 [21]. Αρχικά, υπάρχει ένας σαφής διαχωρισμός του Video Coding Layer (VCL) και του Network Abstract Layer (NAL). Το VCL αντιπροσωπεύει τη λειτουργικότητα του κωδικοποιητή σε επίπεδο επεξεργασίας σήματος, δηλαδή μηχανισμούς όπως οι μετασχηματισμοί, η κβάντιση, η πρόβλεψη αντιστάθμισης της κίνησης κ.ά.. Ο VCL κωδικοποιητής παράγει κομμάτια (slices), δηλαδή μια συμβολοσειρά από bit η οποία περιέχει δεδομένα σε macroblocks, ενώ ο NAL κωδικοποιητής ενθυλακώνει τα slices αυτά σε NALUs, τα οποία είναι κατάλληλα για μεταδόσεις σε δίκτυα που μεταφέρουν πακέτα.

Μία άλλη βασική ιδιότητα του H.264 είναι ο πλήρης διαχωρισμός του χρόνου μετάδοσης, του χρόνου κωδικοποίησης και του χρόνου δειγματοληψίας ή παρουσίασης ενός τμήματος ή μιας ολόκληρης εικόνας. Η διαδικασία αποκωδικοποίησης στον H.264 αγνοεί το χρόνο και γενικά πληροφορίες όπως ο αριθμός των πλαισίων που έχουν παραληφθεί (σε αντίθεση με προηγούμενα πρότυπα συμπίεσης βίντεο). Επίσης, υπάρχουν NALUs που επηρεάζουν πολλές εικόνες και ως εκ τούτου είναι εγγενώς διαχρονικά. Για το λόγο αυτό, η διαχείριση των RTP timestamps, για τα NALUs εκείνα στα οποία ο χρόνος δειγματοληψίας ή παρουσίασης δεν έχει οριστεί, απαιτεί ειδική αντιμετώπιση.

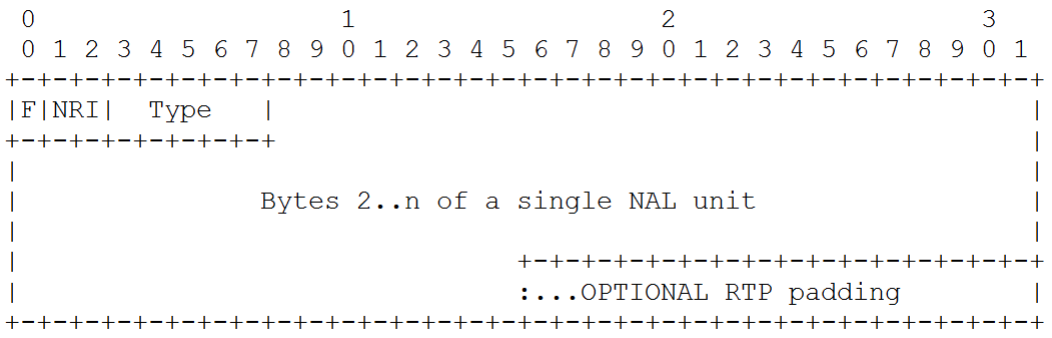
Τέλος, ένα άλλο βασικό σχεδιαστικό χαρακτηριστικό του H.264 είναι ότι δημιουργεί αυτάρκη πακέτα, προκειμένου να καταστήσει διάφορους μηχανισμούς, όπως το να δημιουργούνται διπλότυπα επικεφαλίδων, περιττούς. Αυτό επιτυγχάνεται με το διαχωρισμό των πληροφοριών σχετικών με περισσότερα από ένα slice από τη ροή δεδομένων. Αυτές οι υψηλού επιπέδου μέτα-πληροφορίες θα πρέπει να αποστέλλονται με αξιοπιστία, ασύγχρονα και εκ των προτέρων από τη ροή RTP πακέτων που περιέχει τα πακετοποιημένα slices. Ο συνδυασμός των υψηλού επιπέδου πληροφοριών ονομάζεται σύνολο παραμέτρων (parameter set) και στις προδιαγραφές του H.264 ορίζονται δύο σύνολα παραμέτρων: σύνολα παραμέτρων ακολουθίας και σύνολα παραμέτρων εικόνας. Ένα ενεργό σύνολο παραμέτρων ακολουθίας παραμένει αναλλοίωτο καθ' όλη την ακολουθία κωδικοποιημένου βίντεο και ένα ενεργό σύνολο παραμέτρων εικόνας παραμένει αναλλοίωτο εντός μιας κωδικοποιημένης εικόνας. Τα σύνολα παραμέτρων εικόνας και ακολουθίας περιέχουν πληροφορίες όπως το μέγεθος της εικόνας, προαιρετικά σχήματα κωδικοποίησης που χρησιμοποιούνται και μια συσχέτιση μεταξύ macroblocks και slices.

5.1.2 Πακετοποίηση – NAL UNITS

Όταν χρησιμοποιείται ο κωδικοποιητής H.264, το RTP πακέτο θα αποτελείται από την RTP επικεφαλίδα αλλά στη συνέχεια δε θα τοποθετηθούν τα δεδομένα με τον τρόπο που ορίζεται στο RFC 3550. Αντίθετα, στη θέση τους θα τοποθετηθεί η επικεφαλίδα του NAL Unit (Εικόνα 25) και το ίδιο το NAL Unit. Στην περίπτωση που ένα RTP πακέτο αποτελείται από την RTP επικεφαλίδα, την επικεφαλίδα του NALU και από ένα NALU, τότε έχουμε ένα singe NALU packet, η μορφή του οποίου φαίνεται στην Εικόνα 26.



Εικόνα 25: Επικεφαλίδα του NAL Unit



Εικόνα 26: Single NAL Unit packet

Στην επικεφαλίδα του NAL Unit:

F : Χρησιμοποιείται για να δηλώσουμε αν υπάρχει κάποιο εσφαλμένο bit ή όχι στο περιεχόμενο του NALU.

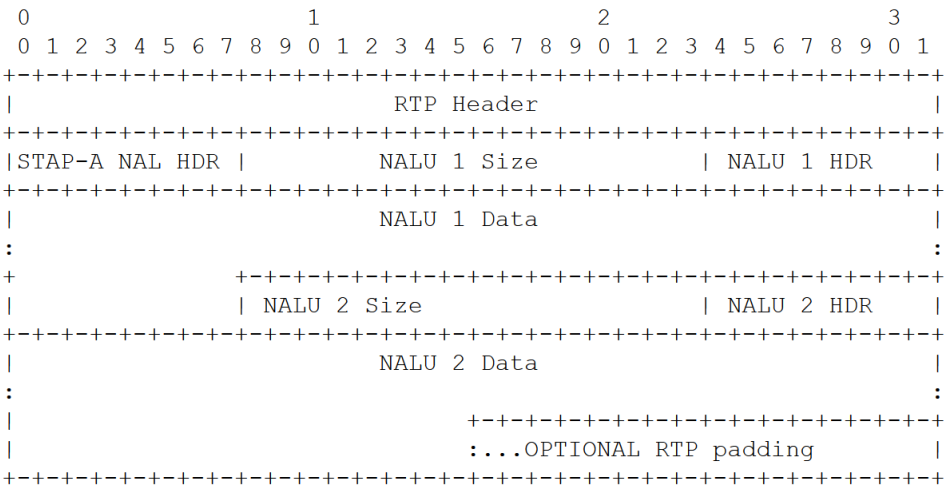
NRI : nal_ref_idc. Χρησιμοποιείται για να δηλώσουμε αν το περιεχόμενο του NAL Unit πρέπει να χρησιμοποιηθεί για να ανακατασκευαστούν εικόνες αναφοράς για την πρόβλεψη μεταξύ των εικόνων. Το πεδίο αυτό σχετίζεται και με το πεδίο Type.

Type : Δηλώνει ένα από τους τύπους NALU της Εικόνα 27.

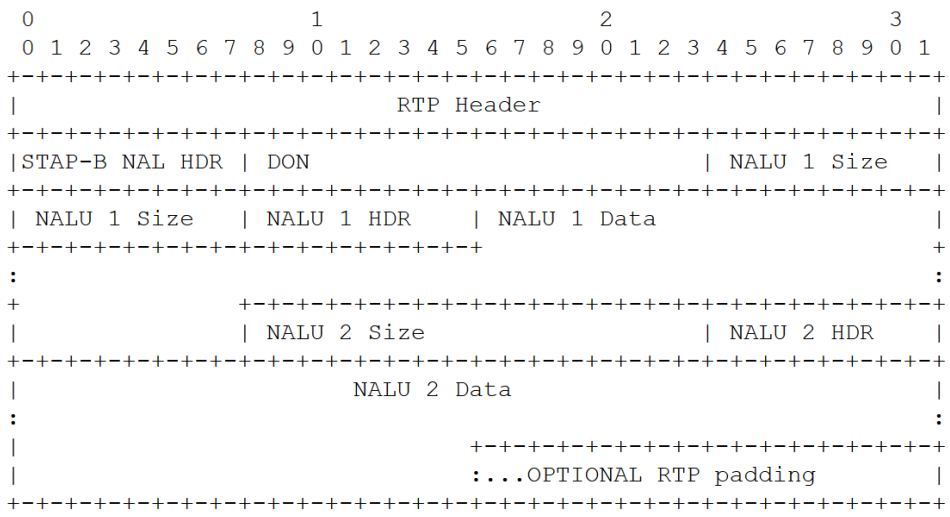
NAL Unit Type	Packet Type	Packet Type Name
0	reserved	
1-23	NAL unit	Single NAL unit packet
24	STAP-A	Single-time aggregation packet
25	STAP-B	Single-time aggregation packet
26	MTAP16	Multi-time aggregation packet
27	MTAP24	Multi-time aggregation packet
28	FU-A	Fragmentation unit
29	FU-B	Fragmentation unit
30-31	reserved	

Εικόνα 27: NAL Unit Types

Από τους τύπους NAL Unit που παρουσιάζονται στην Εικόνα 27, μελετάμε τους τύπους 24 και 25, δηλαδή τους τύπους πακέτων STAP-A και STAP-B. Στην περίπτωση των STAP-A, ένα RTP πακέτο δε θα περιέχει μόνο ένα NALU, αλλά ένα σύνολο από NALUs (Aggregation packet) τα οποία έχουν πανομοιότυπους χρόνους (NALU-times) και πακετοποιούνται με σειρά αποκωδικοποίησης (non-interleaved mode). Η δομή ενός STAP-A πακέτου φαίνεται στην Εικόνα 28. Στην περίπτωση που χρησιμοποιούμε τον τύπο 25, δηλαδή STAP-B πακέτα, ένα RTP πακέτο δε θα περιέχει μόνο ένα NALU, αλλά ένα σύνολο από NALUs (Aggregation packet) τα οποία έχουν πανομοιότυπους χρόνους (NALU-times), αλλά δε πακετοποιούνται με σειρά αποκωδικοποίησης (interleaved mode) και, συνεπώς, θα πρέπει να τοποθετηθεί ένα επιπλέον πεδίο στη STAP-B NALU επικεφαλίδα, το πεδίο DON (Decoding Order Number), το οποίο προσδιορίζει τη σειρά με την οποία πρέπει να αποκωδικοποιηθούν τα NALUs που βρίσκονται μέσα στο πακέτο. Η δομή ενός STAP-B πακέτου φαίνεται στην Εικόνα 29.



Εικόνα 28: Η δομή ενός STAP-A με δύο NALUs



Εικόνα 29: Η δομή ενός STAP-B με δύο NALUs

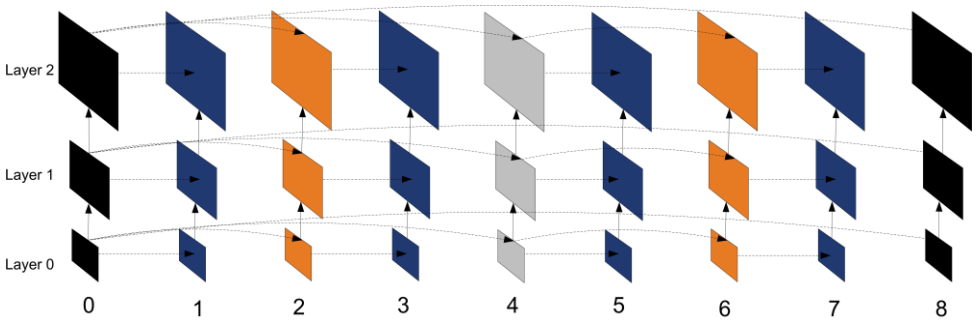
5.2 Κλιμακωτή κωδικοποίηση βίντεο – SVC

Η κλιμακωτή κωδικοποίηση βίντεο (Scalable Video Coding – SVC) [22] πρόκειται για μια επέκταση του προτύπου συμπίεσης H.264/AVC, η οποία καλύπτει ένα εύρος από χαμηλού ρυθμού bit κινητές εφαρμογές μέχρι μετάδοση ψηφιακής τηλεόρασης υψηλής ευκρίνειας (High-Definition Television – HDTV). Εισάγει χαρακτηριστικά κλιμάκωσης (scalability features) στον H.264/AVC, προσαρμόζοντας το σήμα σε διάφορες καταστάσεις συστήματος, με πολύ χαμηλή επεξεργασία. Η προσαρμογή αυτή γίνεται είτε στον πομπό είτε στο δέκτη είτε σε κατάλληλα ενδιάμεσα στοιχεία του δικτύου.

Η επέκταση SVC επιτρέπει την κωδικοποίηση του βίντεο σε έναν αριθμό από επίπεδα (layers) τα οποία μπορούν να αναπαρασταθούν σε πυραμιδική μορφή. Πιο συγκεκριμένα, ο κωδικοποιητής παράγει ένα βασικό επίπεδο (base layer) το οποίο αντιπροσωπεύει την αναπαράσταση του αρχικού σήματος βίντεο σε συγκεκριμένη πιστότητα. Στην συνέχεια, δημιουργείται ένα πλήθος από βελτιωτικά επίπεδα (enhancement layers) τα οποία αυξάνουν την ποιότητα σε τρεις διαστάσεις. Οι διαστάσεις πιστότητας (fidelity dimensions) που προσφέρει είναι χωρικές (spatial –

picture size), ποιοτικές (quality – SNR) και χρονικές (temporal – pictures per second), και οι παράμετροι που προσδιορίζουν αυτές τις διαστάσεις είναι αντίστοιχα οι dependency_id, quality_id, temporal_id.

Η πυραμιδική ιεραρχία, την οποία ακολουθούν τα επίπεδα που παράγει ο κωδικοποιητής, υποδηλώνει ότι ένα συγκεκριμένο επίπεδο, προκειμένου να αποκωδικοποιηθεί, χρειάζεται όλα τα χαμηλότερα επίπεδα στα οποία αναφέρεται. Συνεπώς, κάθε επίπεδο, σε συνδυασμό με τα επίπεδα στα οποία αναφέρεται (ή από τα οποία εξαρτάται), σχηματίζει μια αναπαράσταση του σήματος βίντεο σε συγκεκριμένη χωρικο-χρονική ανάλυση και ποιότητα, όπως φαίνεται και στην Εικόνα 30.



Εικόνα 30: Χωρικο-χρονική αναπαράσταση σήματος βίντεο σε επίπεδα στην επέκταση SVC

Στην Εικόνα 30 φαίνεται ξεκάθαρα η σχέση εξάρτησης ανάμεσα στα διαφορετικά επίπεδα, αλλά αυτό που αξίζει να σημειωθεί είναι η σπουδαιότητα των βασικών επιπέδων. Μπορεί να παρατηρήσει κανείς πως όλα τα βελτιωτικά επίπεδα (Layer 1, Layer 2…) καταλήγουν να εξαρτώνται από κάποιο βασικό επίπεδο (Layer 0). Σε περίπτωση απώλειας, κατά τη μετάδοση, πληροφοριών που αφορούν ένα βελτιωτικό επίπεδο η ποιότητα στο δέκτη δε θα είναι καταστροφική, καθώς η πληροφορία του επιπέδου που χάθηκε μπορεί να ανακτηθεί σε ικανοποιητικό βαθμό από τα επίπεδα στα οποία αναφέρεται. Ωστόσο, σε περίπτωση απώλειας, κατά τη μετάδοση, πληροφοριών που αφορούν κάποιο βασικό επίπεδο, τότε είναι πολύ πιθανό να καταστραφεί η ιεραρχία που φαίνεται στην Εικόνα 30 και κατ' επέκταση η ποιότητα του βίντεο στο δέκτη. Συνεπώς, είναι απαραίτητο να υπάρχουν μηχανισμοί που θα φροντίζουν κατάλληλα ώστε η ρουτίνα αποκωδικοποίησης των πακέτων βίντεο επιπέδου βάσης να πραγματοποιείται χωρίς διακοπές και καθυστερήσεις.

5.2.1 Πακετοποίηση – NAL UNITS

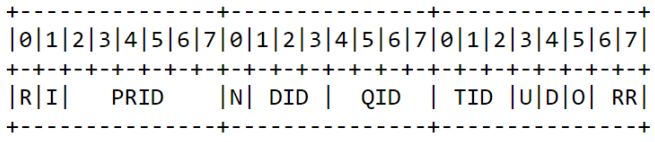
Στο RFC 6190 [23] ορίζονται δύο νέοι τύποι NAL Units: το Scalable Video Coding (SVC) NAL Unit και το Payload Content Scalability Information (PACSI) NAL Unit. Το SVC NALU προσθέτει τρία επιπλέον octets μετά την επικεφαλίδα του NALU, τα οποία φαίνονται στην Εικόνα 31.

DID : Αναφέρεται στη μεταβλητή dependency_id και υποδεικνύει το επίπεδο εξάρτησης της κωδικοποίησης ενός στρώματος από τα ενδιάμεσα στρώματα.

QID : Αναφέρεται στη μεταβλητή quality_id και υποδεικνύει το επίπεδο ποιότητας ενός Medium-grain Scalability (MSG) στρώματος.

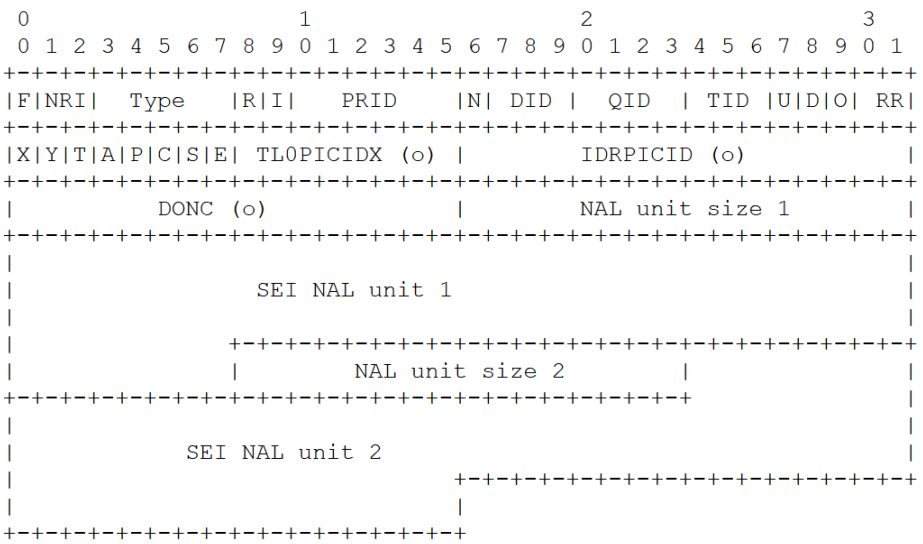
TID : Αναφέρεται στη μεταβλητή temporal_id και υποδεικνύει το χρονικό επίπεδο ενός στρώματος.

Οι τρεις αυτές μεταβλητές αντιπροσωπεύουν τις τρεις διαστάσεις πιστότητας που προσφέρει η επέκταση SVC [24].



Εικόνα 31: Τα επιπλέον πεδία του SVC NAL Unit

Το PACSI NALU, η δομή του οποίου φαίνεται στην Εικόνα 32, αποτελείται από την NALU επικεφαλίδα και στη συνέχεια ακολουθούν τρία επιπλέον octets τα οποία είναι ίδια με το SVC NALU, ένα octet που περιέχει κάποια πεδία σηματοδοσίας (flags), 5 ακόμη προαιρετικά octets και τέλος μηδέν ή περισσότερα SEI NAL Units.



Εικόνα 32: Ένα PACSI NAL Unit με δύο SEI NAL Units

Τα πρώτα 4 bytes του PACSI NALU αποτελούνται από τα πεδία της επικεφαλίδας του NALU της Εικόνα 25 και από τα επιπλέον πεδία του SVC NALU της Εικόνα 31. Παρακάτω ακολουθεί μια επεξήγηση ορισμένων βασικών πεδίων του PACSI NALU:

DID : Το πεδίο DID πρέπει να πάρει την τιμή του ελάχιστου DID των υπολοίπων NAL Units που βρίσκονται μέσα στο Aggregation packet (όταν το PACSI NALU συμπεριλαμβάνεται στο Aggregation packet) ή την τιμή του DID του επόμενου μη-PACSI NALU που επρόκειτο να μεταδοθεί (όταν το PACSI NALU στέλνεται σε Single NALU packet).

QID : Το πεδίο QID πρέπει να πάρει την τιμή του ελάχιστου QID των υπολοίπων NAL Units με το ελάχιστο DID που βρίσκονται μέσα στο Aggregation packet (όταν το PACSI NALU συμπεριλαμβάνεται στο Aggregation packet) ή την

τιμή του QID του επόμενου μη-PACSI NALU που επρόκειτο να μεταδοθεί (όταν το PACSI NALU στέλνεται σε Single NALU packet).

TID : Το πεδίο TID πρέπει να πάρει την τιμή του ελάχιστου TID των υπολοίπων NAL Units με το ελάχιστο DID που βρίσκονται μέσα στο Aggregation packet (όταν το PACSI NALU συμπεριλαμβάνεται στο Aggregation packet) ή την τιμή του TID του επόμενου μη-PACSI NALU που επρόκειτο να μεταδοθεί (όταν το PACSI NALU στέλνεται σε Single NALU packet).

X : Όταν το bit X πάρει την τιμή 1, τότε τα bits A, P, C πρέπει να πάρουν την τιμή 1.

A : Το bit A πρέπει να την τιμή 1 όταν τουλάχιστον ένα από τα υπόλοιπα NALUs στο Aggregation packet ανήκει σε ένα επίπεδο-γέφυρα (anchor layer representation), όταν το PACSI NALU περιλαμβάνεται σε Aggregation packet, ή όταν το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί ανήκει σε ένα επίπεδο-γέφυρα.

P : Το bit P πρέπει να πάρει την τιμή 1 όταν όλα τα υπόλοιπα NALUs σε ένα Aggregation packet έχουν redundant_pic_cnt μεγαλύτερο του 0 (όταν το PACSI NALU συμπεριλαμβάνεται στο Aggregation packet) ή όταν το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί έχει redundant_pic_cnt μεγαλύτερο του 0 (όταν το PACSI NALU στέλνεται σε Single NALU packet).

C : Το bit C πρέπει να πάρει την τιμή 1 όταν τουλάχιστον ένα από τα υπόλοιπα NALUs σε ένα Aggregation packet ανήκει σε ένα ενδο-επίπεδο (intra layer representation), όταν το PACSI NALU περιλαμβάνεται σε Aggregation packet, ή όταν το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί ανήκει σε ένα ενδο-επίπεδο.

Y : Όταν το bit Y πάρει την τιμή 1, τότε τα bits S, E και τα πεδία TL0PICIDX, IDRPICID λαμβάνουν τιμές.

S : Το bit S πρέπει να πάρει την τιμή 1, όταν το πρώτο NALU αμέσως μετά το PACSI NALU σε ένα Aggregation packet είναι το πρώτο VCL NALU, σε σειρά κωδικοποίησης, ενός επιπέδου (όταν το PACSI NALU εμπεριέχεται στο Aggregation packet) ή όταν το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί είναι το πρώτο VCL NALU, σε σειρά κωδικοποίησης, ενός επιπέδου (όταν το PACSI NALU στέλνεται σε Single NALU packet).

E : Το bit E πρέπει να πάρει την τιμή 1, όταν το τελευταίο NALU αμέσως μετά το PACSI NALU σε ένα Aggregation packet είναι το τελευταίο VCL NALU, σε σειρά κωδικοποίησης, ενός επιπέδου (όταν το PACSI NALU εμπεριέχεται στο Aggregation packet) ή όταν το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί είναι το τελευταίο VCL NALU, σε σειρά κωδικοποίησης, ενός επιπέδου (όταν το PACSI NALU στέλνεται σε Single NALU packet).

Σημαντική σημείωση: σε ένα Aggregation packet μπορούμε να προσδιορίσουμε την αρχή και το τέλος της αναπαράστασης ενός επιπέδου, εντοπίζοντας αλλαγές στις τιμές των dependency_id, quality_id και temporal_id που βρίσκονται στις επικεφαλίδες των NALUs, εκτός από τα πρώτα και τελευταία NALUs του πακέτου. Τα bits S, E χρησιμοποιούνται προκειμένου να μας παρέχουν την πληροφορία αυτή τόσο για Aggregation packets όσο και για Single NALUs, προκειμένου να μη χρειαστεί να εξετάσουμε προηγούμενα ή επόμενα πακέτα. Το γεγονός αυτό επιτρέπει στα

MANEs να εντοπίσουν απώλειες τμημάτων (slice loss) και να πραγματοποιήσουν τις ανάλογες ενέργειες, όπως το να ζητήσουν αναμετάδοση όσο το δυνατό γρηγορότερα κ.ά..

TL0 : Το πεδίο αυτό πρέπει να πάρει την τιμή του tl0_dep_rep_idx που παράγει το Temporal Level Zero Dependency Representation Index SEI Message, όπως ορίζεται στο Annex G του H.264 για την αναπαράσταση του επιπέδου που περιέχει το πρώτο NALU που ακολουθεί μετά το PACSI NALU σε ένα Aggregation packet (όταν το PACSI NALU εμπεριέχεται στο Aggregation packet) ή που περιέχει το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί (όταν το PACSI NALU στέλνεται σε Single NALU packet). Την τιμή του TL0, όπως θα δούμε παρακάτω, μπορούμε να την υπολογίσουμε χωρίς τη χρήστη του Temporal Level Zero Dependency Representation Index SEI Message.

IDR : Το πεδίο αυτό πρέπει να πάρει την τιμή του effective_idr_pic_id, όπως ορίζεται στο Annex G του H.264 για την αναπαράσταση του επιπέδου που περιέχει το πρώτο NALU που ακολουθεί μετά το PACSI NALU σε ένα Aggregation packet (όταν το PACSI NALU εμπεριέχεται στο Aggregation packet) ή που περιέχει το επόμενο μη-PACSI NALU που επρόκειτο να μεταδοθεί (όταν το PACSI NALU στέλνεται σε Single NALU packet).

Σημαντική σημείωση: τα πεδία TL0PICIDX και IDRPICID επιτρέπουν στους παραλήπτες και στα MANEs τον εντοπισμό απωλειών στο πιο σημαντικό χρονικό επίπεδο (το επίπεδο με temporal_id ίσο με 0).

Όταν ένα RTP πακέτο μεταφέρει κάποιο PACSI NALU τότε η χρονοσφραγίδα του RTP πακέτου πρέπει να είναι ίση με το NALU-time (δηλαδή τη χρονοσφραγίδα που θα είχε το RTP πακέτο εάν το NALU μεταφερόταν σε δικό του RTP πακέτο) του επόμενου non-PACSI NALU για μετάδοση και το πεδίο Type της NALU επικεφαλίδας πρέπει να είναι 30. Το PACSI NALU μπορεί να τοποθετηθεί είτε σε Single NALU packet είτε σε Aggregation packet, αλλά όχι σε Fragmented packet, και χρησιμοποιείται προκειμένου:

- Να επιτρέψει στα MANEs να αποφασίσουν αν θα προωθήσουν, επεξεργαστούν ή απορρίψουν ένα Aggregation packet.
- Να επιτρέψει την ανάκτηση της σωστής σειράς κωδικοποίησης σε μεταδόσεις πολλαπλών συνόδων (Multi-Session Transmission – MST).
- Να βελτιώσει την ανθεκτικότητα κατά την απώλεια πακέτων, π.χ. με την επανειλημμένη χρήση μηνυμάτων συμπληρωματικής ενισχυτικής πληροφορίας (Supplemental Enhancement Information – SEI Messages), με την παροχή πληροφοριών σχετικά με την αρχή και το τέλος της αναπαράστασης ενός επιπέδου, με των προσδιορισμό δεικτών εξάρτησης ενός επιπέδου από ένα χαμηλότερο κ.λπ..
- Να προσδιορίσει αν τα δεδομένα βίντεο που εμπεριέχονται στο NALU αφορούν το επίπεδο βάσης ή κάποιο βελτιωτικό επίπεδο κ.λπ..

Στην περίπτωση που ένα Aggregation packet περιέχει ένα PACSI NALU, τότε:

- Το PACSI NALU πρέπει να τοποθετηθεί πρώτο.
- Στο πακέτο θα πρέπει να υπάρχει τουλάχιστον ένα NALU μετά το PACSI.
- Τα πεδία της RTP επικεφαλίδες λαμβάνουν τιμές σαν να μην υπήρχε το PACSI μέσα στο πακέτο.

5.2.2 Σηματοδότηση πακέτων βάσης και ενισχυμένων επιπέδων

Μέσω του PACSI NALU μπορούμε εύκολα να αποφανθούμε αν τα δεδομένα που εμπεριέχονται σε ένα πακέτο αφορούν κάποιο επίπεδο βάσης ή κάποιο ενισχυμένο. Πιο συγκεκριμένα, όταν ο τύπος του NALU είναι 30, τότε το πεδίο S μας υποδηλώνει ότι το NALU που ακολουθεί είναι το πρώτο NALU της αναπαράστασης ενός επιπέδου. Αντίστοιχα, το πεδίο E μας υποδηλώνει ότι το NALU που ακολουθεί είναι το τελευταίο της αναπαράστασης ενός επιπέδου. Όταν, λοιπόν, παρατηρήσουμε ότι το S ισούται με 1, καταλαβαίνουμε ότι το NALU που ακολουθεί είναι το πρώτο ενός επιπέδου. Την πληροφορία αυτή θα μπορούσαμε να την αντλήσουμε και από τις τιμές των πεδίων DID, QID, TID, όπου αν έστω και ένα από αυτά διαφέρει από το αντίστοιχο του προηγούμενου NALU (δηλαδή αυτού που είχε E=1), τότε το NALU που ακολουθεί αποτελεί το πρώτο ενός νέο επιπέδου. Αφού, λοιπόν, αναγνωρίσουμε ότι επρόκειτο για ένα νέο επίπεδο, τότε πρέπει να διαπιστώσουμε αν αποτελεί ένα βασικό ή ένα ενισχυμένο επίπεδο. Την πληροφορία αυτή την αντλούμε ξανά από τα πεδία DID, QID και TID.

Στο τμήμα ψευδοκώδικα που ακολουθεί παρουσιάζεται η διαδικασία για το διαχωρισμό πακέτων βάσης και ενισχυμένων επιπέδων:

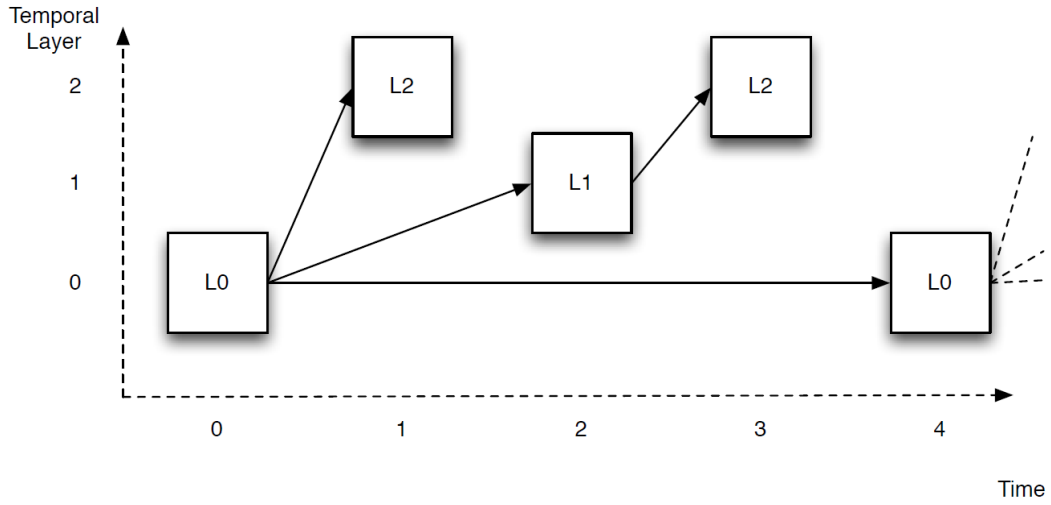
```
if (type == 30)          /* PACSI NALU */
    if (S == 1)          /* First NALU of a layer representation */
        if (DID == QID == TID == 0)      /* Check if base layer */
            base_layer;
        else
            enhancement_layer;
```

Συνεπώς, με τη διαδικασία αυτή σηματοδοτούμε τα πακέτα βάσης και ενισχυμένων επιπέδων, προκειμένου ο παραλήπτης και ο αποκωδικοποιητής να αποφασίσουν πώς θα αξιοποιήσουν τα συγκεκριμένα NALUs. Επίσης, όπως θα δούμε και στο μηχανισμό αναμετάδοσης, η σηματοδότηση των πακέτων βοηθάει τον αποστολέα να εντοπίσει τα πακέτα του πιο σημαντικού χρονικού επιπέδου (TID = 0), προκειμένου να κρατήσει διπλότυπα, ώστε σε περίπτωση απωλειών να τα αναμεταδώσει στον παραλήπτη..

5.2.3 Το πεδίο TL0PICIDX του PACSI NAL Unit

Το πεδίο TL0PICIDX του PACSI NALU πρόκειται για έναν “δείκτη” προς ένα Access Unit το οποίο περιέχει τις πληροφορίες του πιο σημαντικού χρονικού επιπέδου (TID = 0). Το AU αυτό ονομάζεται TL0 AU και τα NALUs που το αποτελούν ονομάζονται TL0 NALUs. Το TL0PICDX πρόκειται για έναν ακέραιο, ο οποίος ξεκινάει από το 0 και αυξάνει κατά 1, κάθε φορά που ο κωδικοποιητής παράγει ένα νέο TL0 AU, ενώ η τιμή του μηδενίζεται κάθε φορά που ο κωδικοποιητής παράγει μια νέα IDR εικόνα. Πιο συγκεκριμένα, έστω ότι από IDR σε IDR ο κωδικοποιητής παράγει 4 TL0 AUs (0,1,2,3). Από τη στιγμή που θα παραχθεί το 0 και μέχρι να παραχθεί το 1, όλα τα Access Units που θα μεσολαβήσουν, συμπεριλαμβανομένου του TL0 Access Unit, θα έχουν TL0PICIDX ίσο με το 0. Έπειτα, όταν παραχθεί το 1 και μέχρι να παραχθεί το 2, όλα τα

AUs θα έχουν TL0PICIDX ίσο με το 1 κ.ό.κ.. Η σχέση ανάμεσα στα TL0 και τα βελτιωτικά AUs φαίνεται στην Εικόνα 33.



Εικόνα 33: Σχέση ανάμεσα στα TL0 και τα βελτιωτικά AUs

Τέλος, μέσω του πεδίου TL0PICIDX, των πεδίων S, E του PACSI αλλά και της SVC επικεφαλίδας είναι δυνατό να ανιχνεύσουμε απώλειες πακέτων που αφορούν τα TL0 AUs. Για παράδειγμα, έστω ότι λαμβάνουμε κάποια NALUs από ένα TL0 AU (TID = 0) με TL0PICIDX ίσο με 1. Για το AU αυτό δε λαμβάνουμε ποτέ κάποια NALUs συμπεριλαμβανομένου και εκείνου που έχει E=1. Όταν φτάσει ένα NALU με TL0PICIDX ίσο με 2, ανατρέχουμε και βλέπουμε ότι το TL0 με TL0PICIDX ίσο με 1 δεν το έχουμε λάβει ολόκληρο. Τότε οφείλουμε να στείλουμε ένα RTCP NACK πακέτο στο οποίο θα ζητάμε τα πακέτα εκείνα που χάθηκαν για το TL0 με TL0PICIDX ίσο με 1, ώστε ο αποστολές να μας τα αναμεταδώσει προκειμένου να διατηρήσουμε ακέραια την αλυσίδα των TL0 και κατ' επέκταση την ακεραιότητα της εικόνας του βίντεο που λαμβάνουμε.

5.3 Απώλειες πακέτων βίντεο

Κατά τη μετάδοση πακέτων μέσω του Διαδικτύου είναι πολύ πιθανό ορισμένα από αυτά να μη φτάσουν ποτέ στον προορισμό τους. Σε εφαρμογές που δεν είναι πραγματικού χρόνου, συνήθως χρησιμοποιείται το πρωτόκολλο TCP, το οποίο παρέχει κατάλληλους μηχανισμούς προκειμένου να αποκατασταθούν οι απώλειες που προέκυψαν. Επίσης, σε αυτές τις εφαρμογές, οι απώλειες πακέτων μπορεί να μην έχουν ιδιαίτερα σημαντικές επιπτώσεις και να μη χρειάζονται ειδικές τεχνικές για την αποκατάστασή τους.

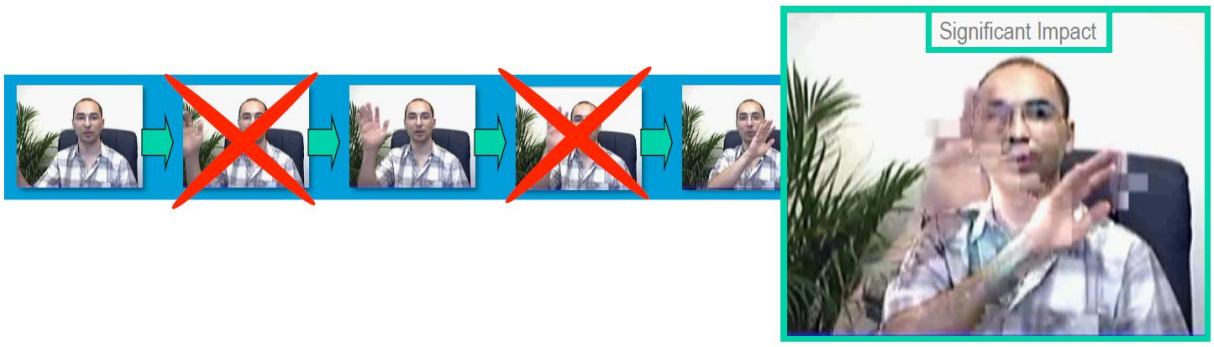
Ωστόσο, σε εφαρμογές πραγματικού χρόνου το πρωτόκολλο που χρησιμοποιείται συνήθως είναι το UDP, καθώς η παραμικρή καθυστέρηση επηρεάζει το αποτέλεσμα στην έξοδο, το οποίο όμως δεν παρέχει τους αντίστοιχους μηχανισμούς του TCP για τις απώλειες πακέτων. Έτσι, όταν μεταδίδουμε δεδομένα ήχου ή βίντεο οι απώλειες πακέτων μπορεί να οδηγήσουν σε καταστροφικές συνέπειες στην έξοδο, με αποτέλεσμα να μειώνεται η εμπειρία του χρήστη. Για παράδειγμα, κατά τη μετάδοση ψηφιακού βίντεο σε πραγματικό χρόνο, οι απώλειες των πακέτων μπορεί να οδηγήσουν σε αλλοίωση της εικόνας ή ακόμη και στο "πάγωμά" της.

Μελετώντας, λοιπόν, την περίπτωση των απωλειών πακέτων στη μετάδοση βίντεο σε πραγματικό χρόνο μέσω RTP, απαιτείται η ύπαρξη τεχνικών και μηχανισμών στο επίπεδο της κωδικοποίησης αλλά και στο επίπεδο της εφαρμογής, όπου μέσω κατάλληλων μηχανισμών αλλά και του RTCP θα φροντίζουμε για την αντιμετώπιση των απωλειών. Παρακάτω, θα αναλύσουμε διάφορες τεχνικές που χρησιμοποιούνται σε επίπεδο κωδικοποίησης (H.264/AVC, H.264/AVC SVC) και θα παρουσιάσουμε ένα μηχανισμό ανίχνευσης και αναμετάδοσης απολεσθέντων RTP πακέτων βίντεο στην περίπτωση που χρησιμοποιείται η επέκταση SVC του προτύπου συμπίεσης H.264/AVC.

5.3.1 Απώλειες πακέτων βίντεο σε απλή σύνοδο RTP

Οι περισσότεροι κωδικοποιητές που χρησιμοποιούνται για την κωδικοποιήση/αποκωδικοποίση δεδομένων βίντεο χρησιμοποιούν ένα επίπεδο κωδικοποίησης πάνω στο οποίο χτίζεται η ακολουθία των δεδομένων που στέλνονται κατά τη μετάδοση. Ορισμένοι εξ' αυτών χρησιμοποιούν και διάφορες τεχνικές συμπίεσης οι οποίες βασίζονται στη μέθοδο της πρόγνωσης, όπου η κωδικοποίηση μιας εικόνας βασίζεται στην κωδικοποίηση μίας ή περισσότερων προηγούμενων εικόνων και αντίστοιχα η αποκωδικοποίησή της βασίζεται στην αποκωδικοποίηση μίας ή περισσότερων προηγούμενων εικόνων. Ένα πρότυπο συμπίεσης το οποίο χρησιμοποιεί τις τεχνικές αυτές είναι το H.264/AVC, το οποίο μεταδίδει τις κωδικοποιημένες εικόνες πάνω σε ένα επίπεδο κωδικοποίησης και η κωδικοποίηση/αποκωδικοποίηση της κάθε μίας βασίζεται σε προηγούμενες. Η ακολουθία των εικόνων πάνω στις οποίες βασίζεται η πρόγνωση, ξεκινάει τη στιγμή που ο κωδικοποιητής παράγει μια IDR εικόνα και τερματίζει αντίστοιχα όταν ο κωδικοποιητής παράγει την επόμενη IDR εικόνα, όπου και ξεκινάει η επόμενη ακολουθία. Συνεπώς, στο διάστημα μεταξύ δύο IDR εικόνων, όλα τα ενδιάμεσα AUs βασίζονται στα προηγούμενα AUs που παράχθηκαν από το ένα IDR μέχρι το επόμενο.

Ωστόσο, το γεγονός πως όλα τα κωδικοποιημένα δεδομένα μεταφέρονται πάνω σε ένα επίπεδο κωδικοποίησης, σε συνδυασμό με τη μέθοδο της πρόγνωσης, μπορεί να δημιουργήσει προβλήματα στην περίπτωση όπου έχουμε απώλειες πακέτων [25]. Πιο συγκεκριμένα, έστω ότι χάνονται πακέτα τα οποία περιέχουν πληροφορία πάνω στην οποία βασίζεται η αποκωδικοποίηση των επόμενων εικόνων. Όταν, λοιπόν, φτάσει η στιγμή για την αποκωδικοποίηση των επόμενων εικόνων τότε η πληροφορία πάνω στην οποία θα βασιστούν για να αποκωδικοποιηθούν επιτυχώς δε θα υπάρχει, με αποτέλεσμα να αποτύχει η σωστή αποκωδικοποίησή τους. Το γεγονός αυτό θα έχει άμεση επίπτωση στην αναπαράσταση της εικόνας στην έξοδο του δέκτη, η οποία θα είναι κατεστραμμένη, όπως φαίνεται στην Εικόνα 34 [25].

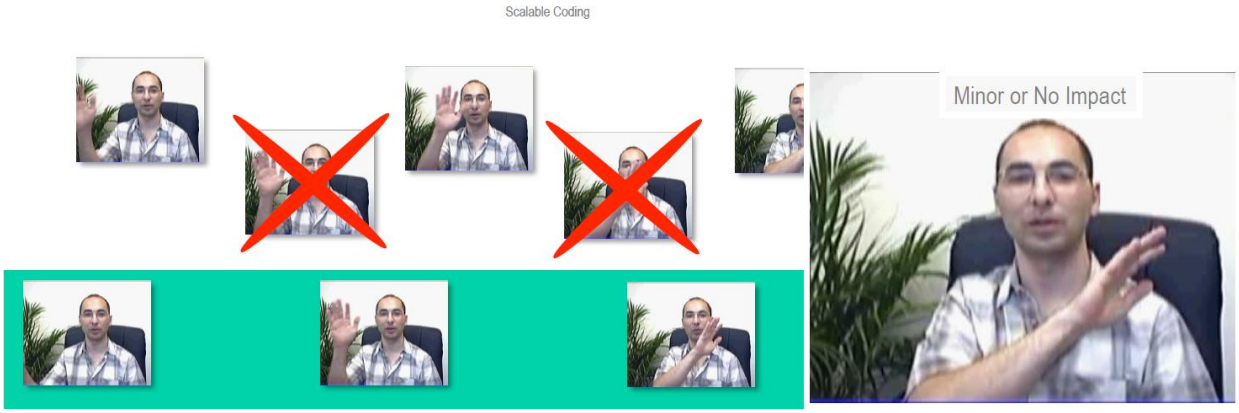


Εικόνα 34: Η παραμόρφωση της εικόνας στην έξοδο όταν έχουμε απώλειες πακέτων στο H.264/AVC

Για να μπορέσουμε να αποκαταστήσουμε την εικόνα στην έξοδο πρέπει είτε να έχουμε ένα μηχανισμό που θα ζητήσει όλα τα χαμένα πακέτα ή, στην περίπτωση που η απώλεια είναι πολύ μεγάλη, να δημιουργήσουμε μια νέα ακολουθία ζητώντας από τον αποστολέα να παράξει μια IDR εικόνα, οι οποίες όμως είναι ιδιαίτερα μεγάλες σε μέγεθος και συνεπώς δε συμφέρει να παράγονται με μεγάλη συχνότητα. Σε κάθε περίπτωση, τόσο η πολυπλοκότητα όσο και το εύρος ζώνης που θα χρειαστεί να καταναλώσουμε είναι ιδιαίτερα μεγάλα.

5.3.2 Απώλειες πακέτων βίντεο σε σύνοδο RTP με χρήση SVC

Τις προδιαγραφές για τη λύση στο πρόβλημα που εμφανίζεται κατά την απώλεια πακέτων σε κωδικοποιητές που χρησιμοποιούν ένα επίπεδο κωδικοποίησης έρχεται να τις προσφέρει το πρότυπο κλιμακωτής κωδικοποίησης SVC. Όπως έχουμε ήδη αναφέρει, το SVC προσφέρει διάφορα κλιμακωτά επίπεδα κωδικοποίησης: το επίπεδο βάσης (base layer) και τα ενισχυμένα επίπεδα (enhancement layers). Μελετώντας την πληροφορία του πιο σημαντικού χρονικού επιπέδου, ορίζουμε ως TL0 AUs εκείνα τα AU τα περιέχουν πληροφορία που αφορά το χρονικό επίπεδο 0 (Temporal Layer 0 – TL0). Το επίπεδο αυτό περιέχει την πιο βασική και στοιχειώδη πληροφορία για το βασικό και για κάθε ενισχυμένο επίπεδο, η οποία είναι ικανή και αναγκαία για την αναπαραγωγή βίντεο στην έξοδο. Πρακτικά περιέχει τη βασική και στοιχειώδη χρονική, χωρική και ποιοτική πληροφορία, πάνω στην οποία βασίζονται τα ενισχυμένα επίπεδα. Τα ενισχυμένα επίπεδα περιέχουν πληροφορία που συμπληρώνει τα TL0, ώστε η εικόνα στην έξοδο να εμπλουτίζεται αντίστοιχα στις τρεις διαστάσεις πιστότητας. Εάν ο κωδικοποιητής δεν παράγει ενισχυμένα χωρικά και ποιοτικά επίπεδα, τότε το TL0 ταυτίζεται με το επίπεδο κωδικοποίησης του H.264/AVC. Μελετώντας λοιπόν την περίπτωση αυτή, αν προκύψουν απώλειες πακέτων σε κάποιο ενισχυμένο χρονικό επίπεδο, στην εικόνα της εξόδου δε θα δημιουργηθεί καταστροφική παραμόρφωση αλλά, στη χειρότερη, η εικόνα στην έξοδο θα είναι απλά χαμηλότερης ποιότητας αλλά εύρωστη, καθώς η βασική πληροφορία θα αντλείται από τα TL0 AUs και θα συμπληρώνεται από όποια πληροφορία μπορούμε να πάρουμε από τα ενισχυμένα χρονικά επίπεδα, όπως φαίνεται στην Εικόνα 35 [25].



Εικόνα 35: Η εικόνα στην έξοδο όταν έχουμε απώλειες πακέτων στα ενισχυμένα χρονικά επίπεδα με τη χρήση SVC

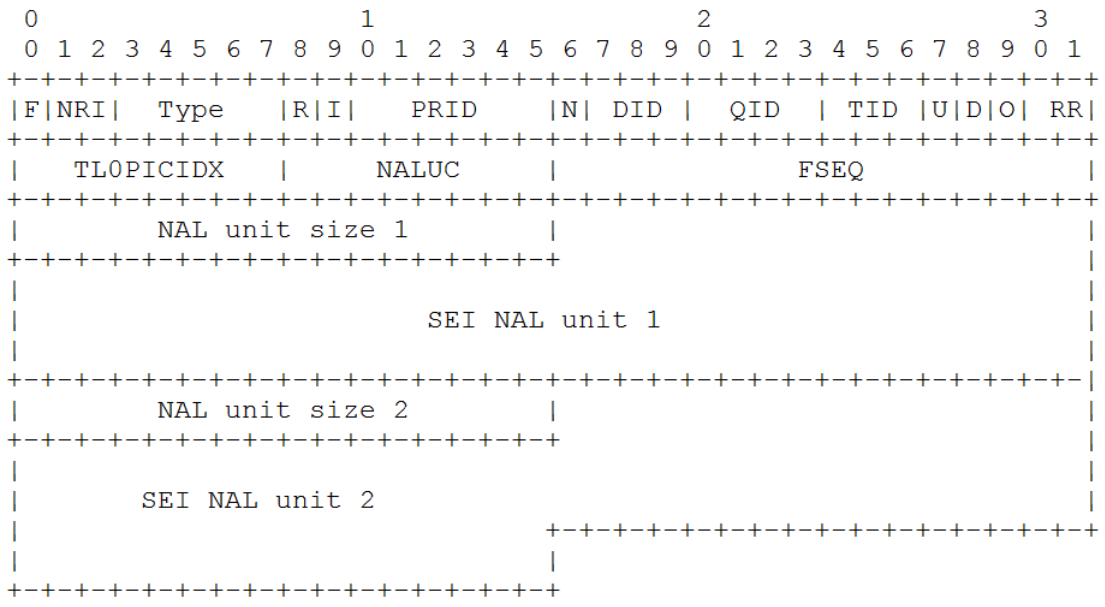
Στο σημείο αυτό είναι προφανές ότι πρέπει να φροντίσουμε τα TL0 να φτάνουν πάντα στον παραλήπτη προκειμένου να εκμεταλλευτούμε τη δυνατότητα αυτή του SVC. Συνεπώς, πρέπει να υπάρχουν οι κατάλληλες υποδομές και μηχανισμοί, προκειμένου τα απολεσθέντα TL0 πακέτα να αναμεταδίδονται έγκαιρα στον παραλήπτη. Παρακάτω θα παρουσιάσουμε ένα σύστημα, όπως αυτό υλοποιήθηκε στο BareSIP, το οποίο θα ανιχνεύει άμεσα τις απώλειες στα TL0 πακέτα και θα φροντίζει για την άμεση αναμετάδοσή τους, ώστε η μετάδοση βίντεο να είναι εύρωστη.

5.4 Σύστημα αναμετάδοσης TL0 πακέτων στο BareSIP

Προκειμένου να αξιοποιήσουμε τα TL0 AUs που παράγει το SVC, υλοποιήσαμε στο BareSIP ένα μηχανισμό υψηλής ανθεκτικότητας ανάμεσα στον πομπό και το δέκτη, χρησιμοποιώντας αναμετάδοση RTP πακέτων βίντεο. Το μοντέλο αυτό φροντίζει για την άμεση αποκατάσταση απολεσθέντων TL0 πακέτων βίντεο, ώστε η αποκωδικοποίησή τους να πραγματοποιείται χωρίς διακοπές ή καθυστερήσεις. Για τη δημιουργία του μηχανισμού χρειάστηκε να τροποποιήσουμε ορισμένες λειτουργίες του BareSIP, να προσθέσουμε ορισμένες καινούριες, αλλά και να κάνουμε ορισμένες συμβάσεις προκειμένου το μοντέλο αυτό να λειτουργήσει σωστά και να μας προσφέρει τις δυνατότητες που επιθυμούμε. Επιπλέον, ως προς την κλιμακωτή κωδικοποίηση, ρυθμίζουμε τον κωδικοποιητή να παράγει μόνο εμπλουτισμένα χρονικά επίπεδα και άρα, τα TL0 AUs θα ταυτίζονται με το επίπεδο κωδικοποίησης του H.264/AVC. Συνεπώς, κατά την αλγοριθμική παρουσίαση του μηχανισμού, όπου αναφέρουμε ενισχυμένα/βελτιωτικά επίπεδα θα αναφερόμαστε μόνο στα χρονικά και όχι στα χωρικά και ποιοτικά.

Στην πλευρά του αποστολέα, σε πρώτο στάδιο δημιουργήσαμε ένα νέου τύπου NALU (type 32), το οποίο το ονομάζουμε Temporal Level 0 Descriptor – TL0D NALU και αποτελεί μια παραλλαγή του PACSI NALU αλλά με επιπλέον πληροφορία σχετικά με τον πρώτο (FSEQ) αριθμό RTP ακολουθίας καθώς και το πλήθος των NALUs (NALUC) που παράγονται για κάποιο TL0 AU. Η μορφή του TL0D φαίνεται στην Εικόνα 36. Επιπλέον, ο κωδικοποιητής τροποποιήθηκε ώστε να δημιουργεί το νέο αυτό NALU με τις απαραίτητες πληροφορίες και να το πακετοποιεί μαζί με ένα παραγόμενο NALU σε ένα STAP-A πακέτο. Συνεπώς, γνωρίζουμε εκ των προτέρων ότι αν ένα TL0 AU αποτελείται από N NALUs, τότε τα NALUs αυτά θα βρίσκονται στα RTP πακέτα με αριθμούς ακολουθίας από FSEQ μέχρι FSEQ + NALUC – 1. Έπειτα, ο αποστολέας

τροποποιήθηκε να κρατάει διπλότυπα για τα TL0 AU που πρόκειται να στείλει, όπου για κάθε TL0 AU διατηρεί μια λίστα με όλα τα RTP πακέτα που το συνθέτουν, καθώς και πληροφορίες όπως την τιμή του TL0PICIDX, τον πρώτο αριθμό ακολουθίας και τον τελευταίο αριθμό ακολουθίας. Τα διπλότυπα αυτά διατηρούνται μέχρι ο κωδικοποιητής να παράξει μια IDR εικόνα, όπου και ξεκινάει εκ νέου η διαδικασία διατήρησης διπλοτύπων για τα TL0 AUs που θα παραχθούν.



Εικόνα 36: Η δομή του TL0D NALU

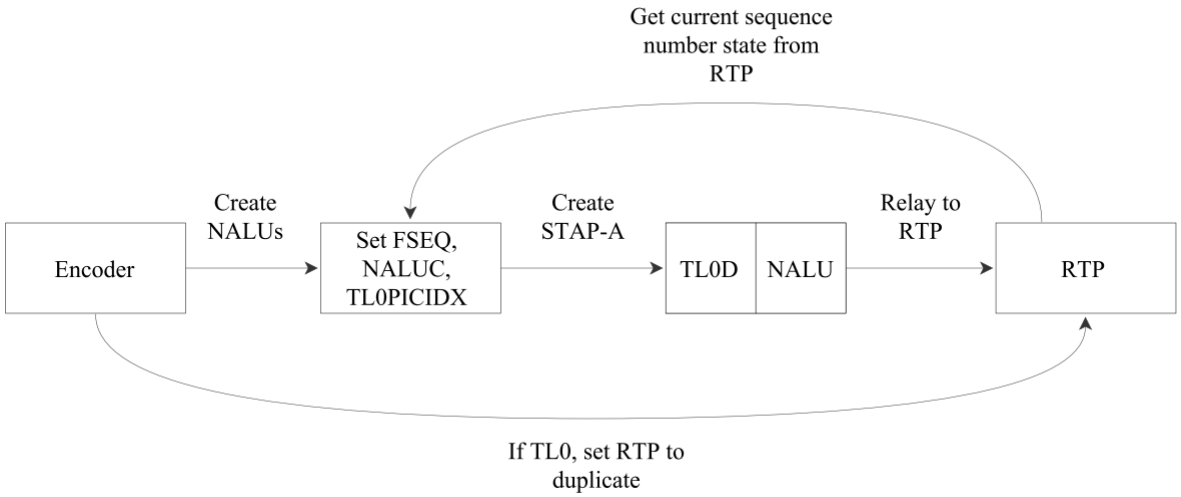
Έπειτα, επειδή επιθυμούμε τα πακέτα που αναμεταδίδουμε να φτάσουν όσο το δυνατό πιο άμεσα στον παραλήπτη, ανοίγουμε μια νέα θύρα στην οποία θα στέλνουμε μόνο τα αναμεταδιδόμενα TL0 πακέτα. Έτσι, όταν στον αποστολέα φτάσει ένα RTCP NACK πακέτο με τα πακέτα που έχει χάσει ο παραλήπτης, τότε από τη λίστα των διπλοτύπων βρίσκουμε τα ζητούμενα πακέτα και τα στέλνουμε στη θύρα αυτή.

Τέλος, τροποποιήσαμε κατάλληλα τον παραλήπτη, ώστε σταδιακά να χτίζει τόσο το κάθε TL0 AU αλλά και τα ενισχυμένα AUs τα οποία έχουν TL0PICIDX που αναφέρεται στο εκάστοτε TL0 AU, ενώ μέσω μιας αλγοριθμικής διαδικασία ανιχνεύουμε τα TL0 πακέτα που χάνονται και φροντίζουμε για την άμεση αποκατάστασή τους. Με την ολοκλήρωση και την αποκωδικοποίηση ενός TL0 AU φροντίζουμε για τη σταδιακή αποκωδικοποίηση και των ενισχυμένων AUs που σχετίζονται με αυτό, ώστε να εκμεταλλευτούμε πλήρως τις δυνατότητες του SVC. Τέλος, δίνουμε μια ανοχή ως προς τις απώλειες που μπορούμε να έχουμε και θέτουμε ως άνω φράγμα την απώλεια ενός μόνο TL0 AU κάθε φορά, καθώς η απώλεια δύο διαδοχικών TL0 AUs θα οδηγήσει σε καταστροφή της εικόνας. Όταν κάτι τέτοιο συμβεί, δηλαδή απώλεια δύο διαδοχικών TL0 AUs, τότε οφείλουμε να ζητήσουμε από τον κωδικοποιητή να παράξει μια IDR εικόνα προκειμένου να χτίσουμε από την αρχή την ακολουθία των TL0 AUs.

Παρακάτω, παρουσιάζεται αναλυτικά η διαδικασία με την οποία λειτουργούν οι δύο πλευρές του μηχανισμού προκειμένου η αποκωδικοποίηση των TL0 πακέτων βίντεο στο δέκτη να πραγματοποιείται χωρίς διακοπές ή καθυστερήσεις.

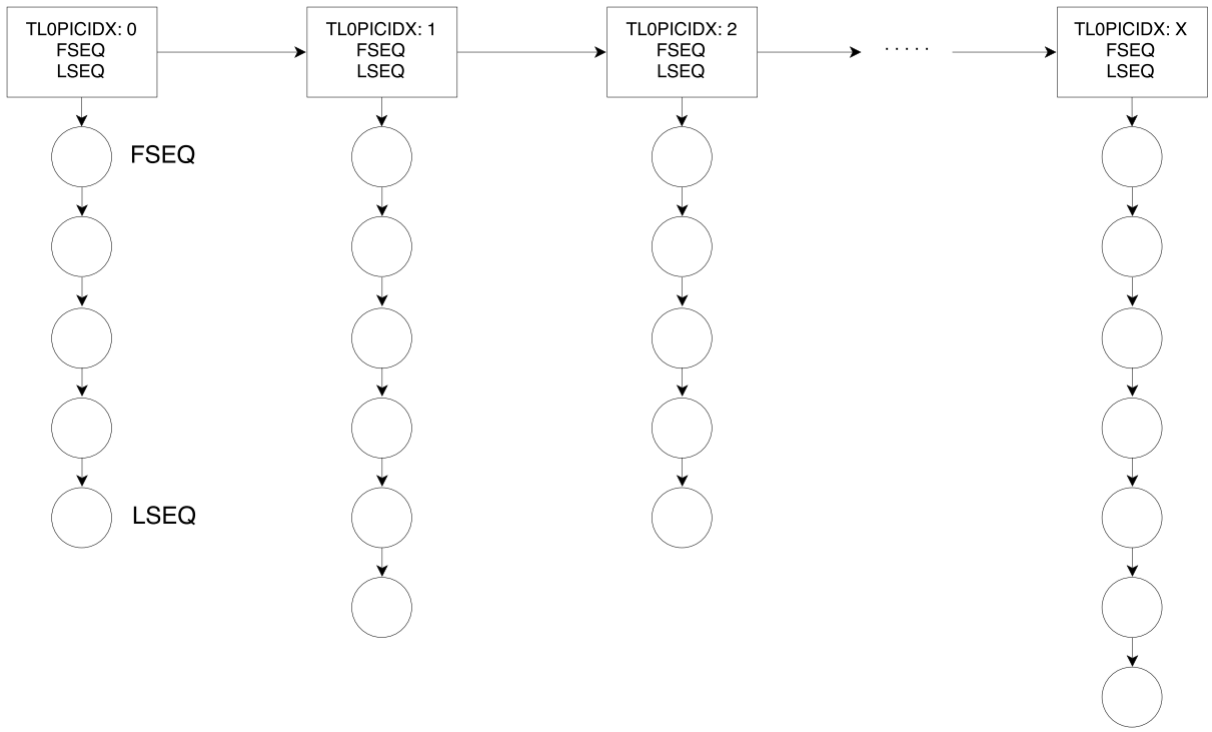
5.4.1 Αλγοριθμική παρουσίαση του συστήματος αναμετάδοσης TL0 πακέτων: Αποστολέας

Στην πλευρά του αποστολέα, σε πρώτο στάδιο ελέγχουμε αν το AU που δημιουργήθηκε είναι TL0. Αν λοιπόν ανιχνεύσουμε ότι πρόκειται για ένα TL0 AU, τότε ενεργοποιούμε μια Boolean μεταβλητή η οποία καθορίζει αν θα πρέπει να κρατήσουμε διπλότυπα για τα πακέτα ή όχι. Στη συνέχεια, επειδή κάθε πακέτο μας θα αποτελείται από ένα TL0D NALU και ένα NALU που παράχθηκε από τον κωδικοποιητή, μπορούμε να προσδιορίσουμε τον αρχικό RTP αριθμό ακολουθίας και το πλήθος των πακέτων που θα στείλουμε. Έτσι, αν το AU που θα στείλουμε είναι TL0, υπολογίζουμε τα πεδία FSEQ και NALUC. Στην περίπτωση που το AU δεν είναι TL0, τότε τα FSEQ και NALUC θα έχουν τις τιμές του TL0 AU από το οποίο εξαρτάται το AU που επρόκειτο να στείλουμε. Έπειτα, σε όλα τα NALUs που στέλνουμε, ανεξάρτητα αν είναι TL0 ή όχι, τοποθετούμε στην αρχή το TL0D NALU (STAP-A πακετοποίηση), προκειμένου να έχουμε πάντα την πληροφορία για το TL0 AU στο οποίο αναφέρονται. Η πληροφορία αυτή, όπως θα εξηγήσουμε στη συνέχεια, είναι απαραίτητη για την ορθή λειτουργία του αλγορίθμου στην πλευρά του παραλήπτη. Τέλος, κάθε ένα STAP-A πακέτο που ετοιμάζουμε το προωθούμε προς το τμήμα που πραγματοποιείται η RTP πακετοποίηση. Η διαδικασία αυτή φαίνεται στην Εικόνα 37.



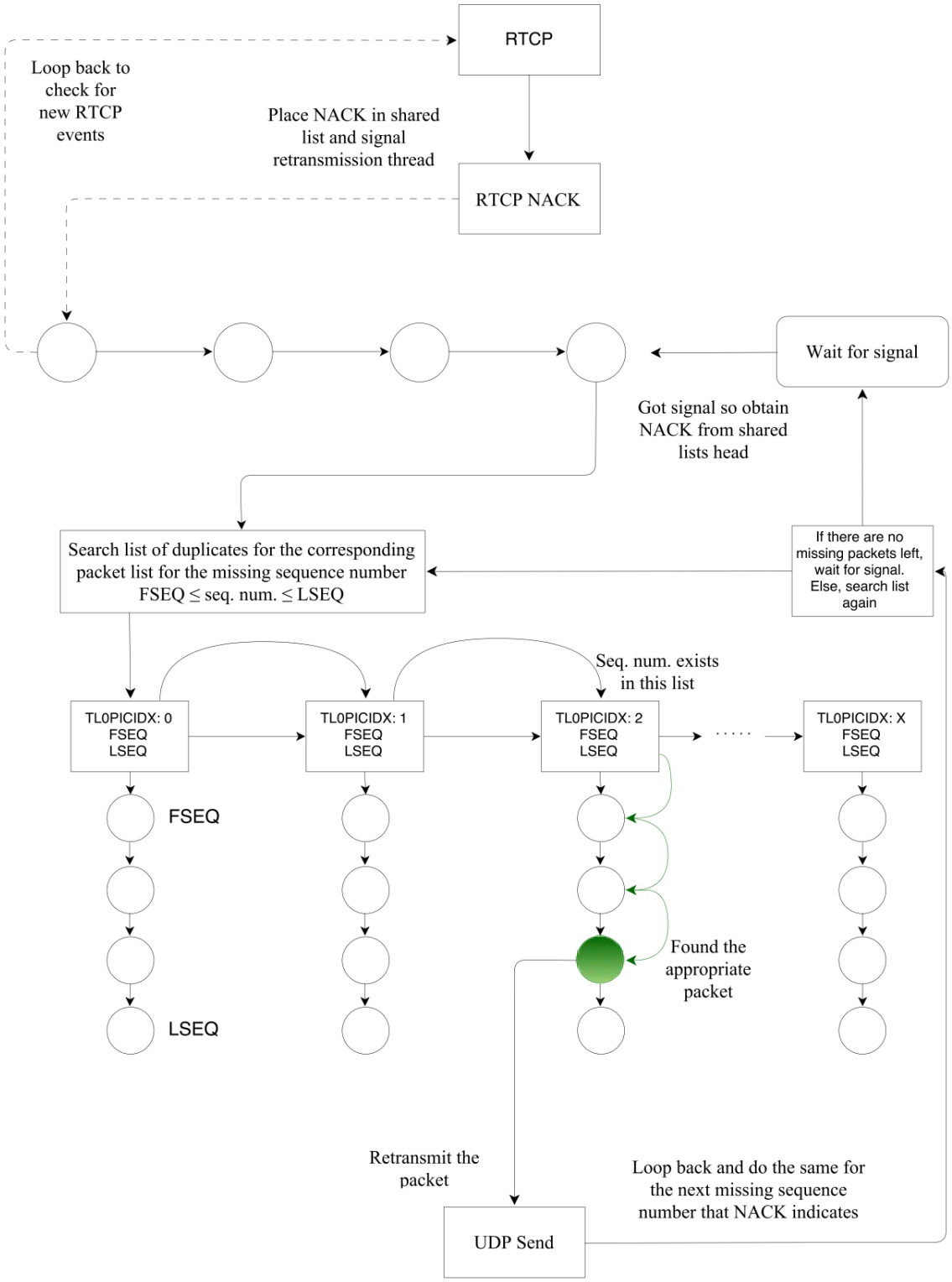
Εικόνα 37: Διαδικασία ανίχνευσης TL0 και πακετοποίησης σε STAP-A

Στο τμήμα της RTP πακετοποίησης, παίρνουμε τα STAP-A πακέτα (payload data) και τους τοποθετούμε την RTP επικεφαλίδα. Για κάθε ένα RTP πακέτο που δημιουργούμε, ελέγχουμε αν η Boolean μεταβλητή έχει οριστεί και, αν έχει οριστεί, τότε πρέπει να κρατήσουμε ένα διπλότυπο του πακέτου πριν την αποστολή του. Η διαδικασία με την οποία κρατάμε τα διπλότυπα είναι η εξής: κάθε ένα TL0 AU χαρακτηρίζεται από το TL0PICIDX και τα RTP πακέτα τα οποία το απαρτίζουν. Συνεπώς, δημιουργούμε μια δομή η οποία περιέχει τα πεδία TL0PICIDX, FSEQ, LSEQ και αποτελεί την κεφαλή μιας λίστας από RTP πακέτα. Στο διάστημα μεταξύ δύο IDR εικόνων, έστω ότι παράγονται X TL0 AUs, τότε θα έχουμε μια λίστα από X κόμβους κεφαλές και κάθε κόμβος θα διατηρεί μια λίστα με RTP πακέτα. Τέλος, κάθε φορά που ο κωδικοποιητής παράγει μια IDR εικόνα πρέπει να καθαρίσουμε τη λίστα και να χτίσουμε την ακολουθία από την αρχή. Η μορφή της λίστας των διπλοτύπων φαίνεται στην Εικόνα 38.



Εικόνα 38: Η μορφή της λίστας των διπλοτύπων

Το τελευταίο κομμάτι που μελετάμε στην πλευρά του αποστολέα είναι το σημείο που δεχόμαστε ένα RTCP NACK πακέτο, το οποίο περιέχει τους αριθμούς ακολουθίας που έχει χάσει ο δέκτης. Για την αναμετάδοση των RTP πακέτων βίντεο, δημιουργούμε ένα νήμα το οποίο θα εντοπίζει από τη λίστα των διπλοτύπων τα χαμένα πακέτα και θα τα αναμεταδίδει. Το νήμα αυτό επικοινωνεί με το νήμα του RTCP μέσω μιας κοινής λίστας. Ο συγχρονισμός μεταξύ τους ακολουθεί το μοντέλο Readers – Writers, όπου έχουμε έναν Reader και έναν Writer. Όταν, λοιπόν, λάβουμε ένα RTCP NACK τότε το νήμα του RTCP κλειδώνει τη λίστα, τοποθετεί τα πεδία fsn και blp στον τελευταίο κόμβο, ξεκλειδώνει τη λίστα και στέλνει ένα σήμα στο νήμα για τις αναμεταδώσεις, προκειμένου να πάει να διαβάσει. Όταν το νήμα λάβει το σήμα, τότε από τη λίστα αντλεί τα πεδία fsn και blp. Έπειτα, πρέπει να εντοπίσει τα πακέτα στη λίστα με τα διπλότυπα και να τα αναμεταδώσει άμεσα. Συνεπώς, σε πρώτο στάδιο παίρνουμε το πεδίο fsn, το οποίο υποδηλώνει τον αριθμό ακολουθίας του πρώτου πακέτου που έχασε ο δέκτης, και ελέγχουμε στους κόμβους κεφαλές μέχρι να βρούμε σε ποια λίστα βρίσκεται το πακέτο. Όπως είπαμε, οι κόμβοι κεφαλές περιέχουν τα πεδία FSEQ, LSEQ και συνεπώς μπορούμε να εντοπίσουμε τον κατάλληλο κόμβο με βάση το κριτήριο FSEQ ≤ fsn ≤ LSEQ. Αφού εντοπίσουμε τον κόμβο, τότε πρέπει να μεταβούμε στη θέση μέσα στη λίστα με τα RTP πακέτα όπου βρίσκεται το πακέτο με αριθμό ακολουθίας fsn. Μόλις το βρούμε τότε το στέλνουμε στη θύρα που έχουμε ανοίξει ειδικά για τα αναμεταδιδόμενα RTP πακέτα βίντεο. Για τα υπόλοιπα χαμένα πακέτα του δέκτη, γνωρίζουμε πως θα βρίσκονται στη συνέχεια της λίστας καθώς κάθε NACK που στέλνει ο δέκτης αφορά κάποιο TL0 AU. Συνεπώς, για τα επόμενα πακέτα αρκεί να δούμε από το πεδίο blp ποια έχουν χαθεί, να μεταβούμε στις κατάλληλες θέσεις μέσα στη λίστα και να τα στείλουμε στη θύρα για τα αναμεταδιδόμενα πακέτα βίντεο. Τέλος, το νήμα επιστρέφει και περιμένει να του έρθει σήμα ώστε να διαβάσει το επόμενο NACK από τη λίστα. Η διαδικασία για την αναμετάδοση των χαμένων TL0 πακέτων φαίνεται στην Εικόνα 39.



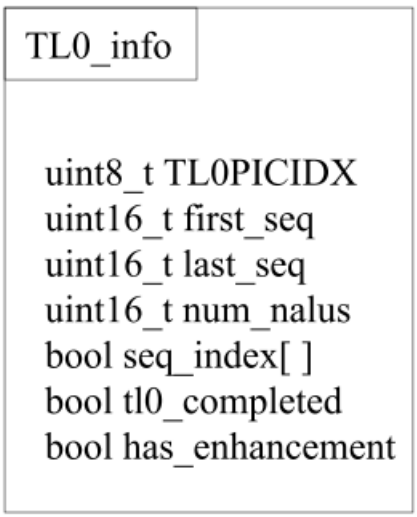
Εικόνα 39: Διαδικασία αναμετάδοσης χαμένων TL0 πακέτων

5.4.2 Αλγοριθμική παρουσίαση του συστήματος αναμετάδοσης TL0 πακέτων: Παραλήπτης

Στην πλευρά του παραλήπτη υλοποιήσαμε ένα μηχανισμό ο οποίος ανιχνεύει απώλειες πακέτων στο επίπεδο TL0 και φροντίζει για την άμεση αποκατάστασή τους, ζητώντας την αναμετάδοσή τους από τον αποστολέα. Ο μηχανισμός αυτός, αξιοποιεί το TL0D NALU που εμπεριέχονται στα STAP-A πακέτα που στέλνουμε, προκειμένου να αντλήσει τις κατάλληλες πληροφορίες που αφορούν τα TL0 AUs και να δημιουργήσει τις απαραίτητες δομές για τη σωστή διαχείρισή τους.

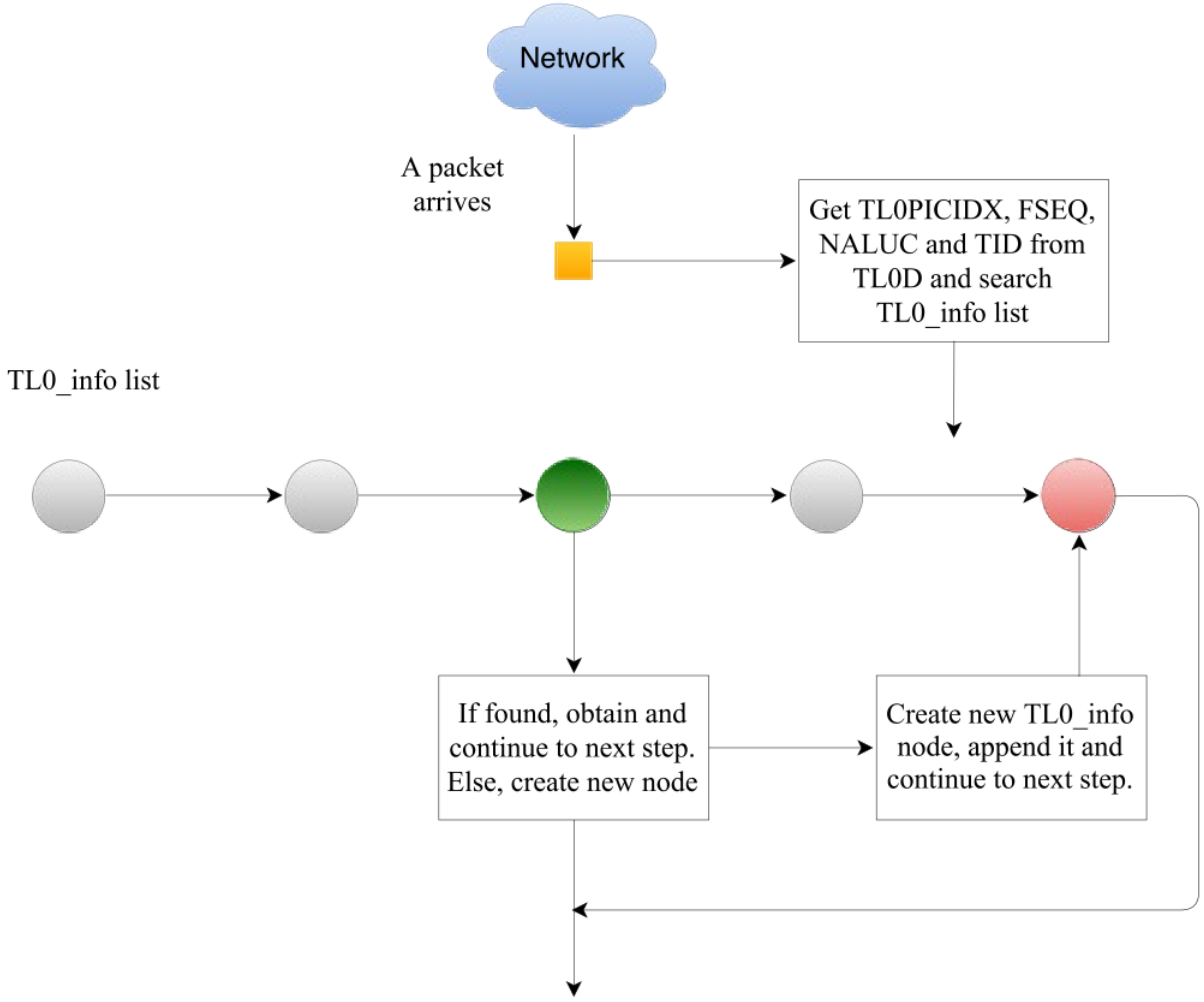
Σε πρώτο στάδιο, δημιουργούμε μια αλγοριθμική διαδικασία ελέγχου, η οποία είναι προσαρμοσμένη στον τρόπο με τον οποίο ο κωδικοποιητής δημιουργεί τις αναφορές για την κωδικοποίηση/αποκωδικοποίηση των frames που παράγονται. Στην περίπτωσή μας που χρησιμοποιούμε τον OpenH264, ο αποκωδικοποιητής δε φροντίζει να αποκωδικοποιήσει μόνο εκείνα τα AUs που μπορεί, με αποτέλεσμα να παρουσιάζει σφάλματα. Για το λόγο αυτό, δημιουργούμε τους κατάλληλους ελέγχους προκειμένου να φροντίζουμε για την ακεραιότητα των AUs που τον τροφοδοτούμε, ώστε η αποκωδικοποίησή τους να είναι εφικτή χωρίς σφάλματα. Σε περίπτωση που κρίνουμε ότι οι απώλειες σε κάποιο επίπεδο θα καταστήσουν αδύνατη την αποκωδικοποίηση του AU από τον αποκωδικοποιητή, αποφασίζουμε να το παρακάμψουμε και να μην το δώσουμε για αποκωδικοποίηση, ώστε να αποφύγουμε πιθανή ανωμαλία στη συμπεριφορά του συστήματος. Όταν κάτι τέτοιο συμβεί, τότε το συγκεκριμένο frame το θεωρούμε χαμένο.

Έπειτα, για κάθε TL0 AU που θα μας έρχεται πρέπει να έχουμε όσο το δυνατό πιο χρήσιμες πληροφορίες που θα το χαρακτηρίζουν. Για το λόγο αυτό, δημιουργούμε τη δομή TL0_info η οποία περιέχει πληροφορίες όπως το TL0PICIDX, το FSEQ, το LSEQ, έναν πίνακα seq_index στον οποίο σημειώνουμε ποια πακέτα έχουμε πάρει ή μας λείπουν, μια Boolean μεταβλητή tl0_completed η οποία μας υποδηλώνει αν έχουμε λάβει ολόκληρο το TL0 AU, καθώς και μια Boolean μεταβλητή has_enhancement η οποία μας υποδηλώνει αν έχει φτάσει πληροφορία από κάποιο βελτιωτικό επίπεδο για το συγκεκριμένο TL0. Η μορφή της δομής αυτής φαίνεται στην Εικόνα 40.



Εικόνα 40: Η δομή TL0_info που χαρακτηρίζει ένα TL0 AU

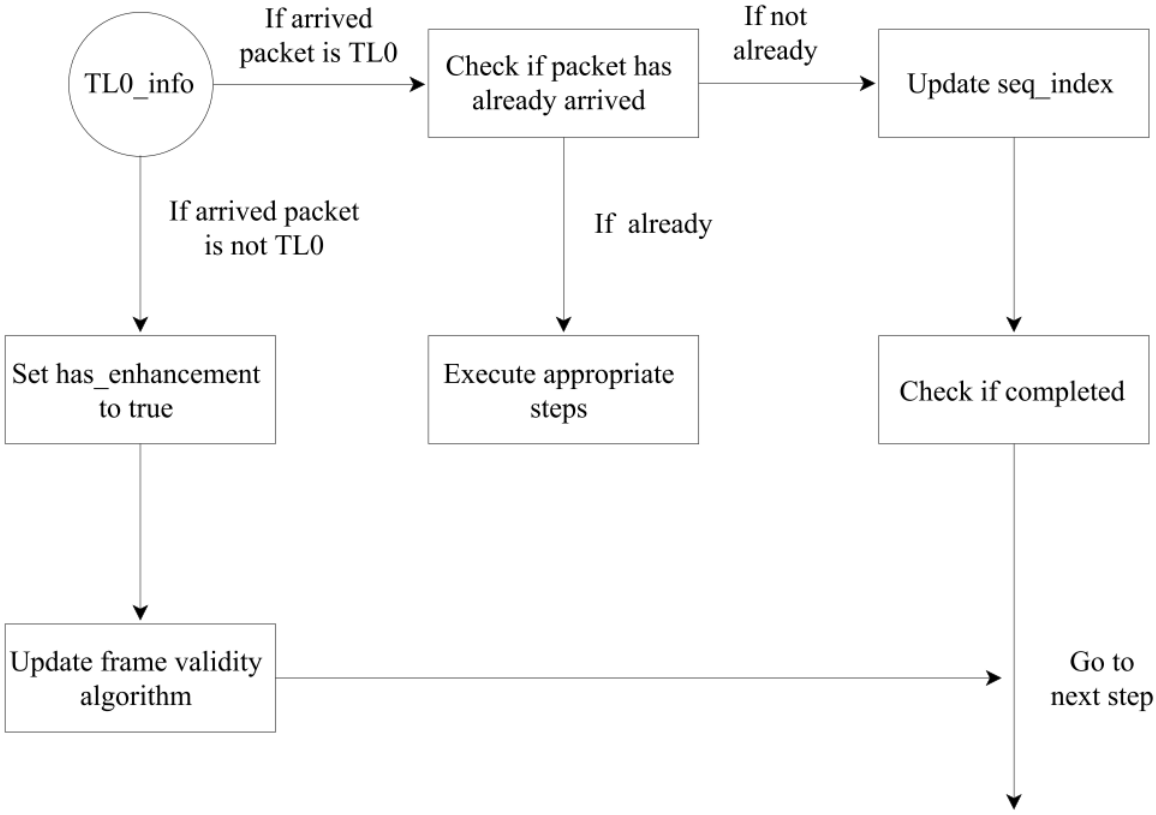
Έστω λοιπόν ότι φτάνει ένα πακέτο βίντεο. Σε πρώτο στάδιο, φροντίζουμε να πάρουμε από το TL0D NALU το πεδίο TL0PICIDX, ώστε να ξέρουμε σε ποιο TL0 αναφέρεται αυτό το NALU, τα πεδία FSEQ και NALUC, ώστε να μπορούμε να προσδιορίσουμε τα RTP πακέτα τα οποία θα μεταφέρουν πληροφορία για το TL0 με TL0PICIDX, και τέλος το πεδίο TID, ώστε να γνωρίζουμε αν το συγκεκριμένο πακέτο αφορά κάποιο TL0 ή κάποιο ενισχυμένο AU. Αφού πάρουμε όλες αυτές τις πληροφορίες, πρέπει να αναζητήσουμε αν έχουμε δημιουργήσει τη δομή TL0_info για το TL0 AU που προσδιορίζεται από το TL0PICDX, να ελέγξουμε δηλαδή αν έχουμε πάρει ξανά πληροφορία για το συγκεκριμένο TL0 AU. Τη δομή αυτή την αναζητούμε σε μια λίστα από TL0_info, η οποία περιέχει πληροφορίες για τα διαδοχικά TL0 AUs που λαμβάνουμε και το πλήθος των κόμβων της είναι όσο και το πλήθος των TL0 που εμφανίζονται μεταξύ δύο IDRs. Αν, λοιπόν στη λίστα αυτή δεν υπάρχει κάποιο TL0_info με TL0PICIDX όπως αυτό του πακέτου που μόλις λάβαμε, τότε πρέπει να δημιουργήσουμε ένα νέο. Αλλιώς, αν βρεθεί στη λίστα, τότε το επιλέγουμε προκειμένου να το επεξεργαστούμε. Η διαδικασία αυτή φαίνεται στην Εικόνα 41.



Εικόνα 41: Διαδικασία εύρεσης και δημιουργίας ενός TL0_info κόμβου

Στη συνέχεια, είτε βρήκαμε είτε δημιουργήσαμε το TL0_info, πρέπει να ενημερώσουμε τα διάφορα πεδία του, ανάλογα με το αν το πακέτο που μόλις λάβαμε αφορά κάποιο

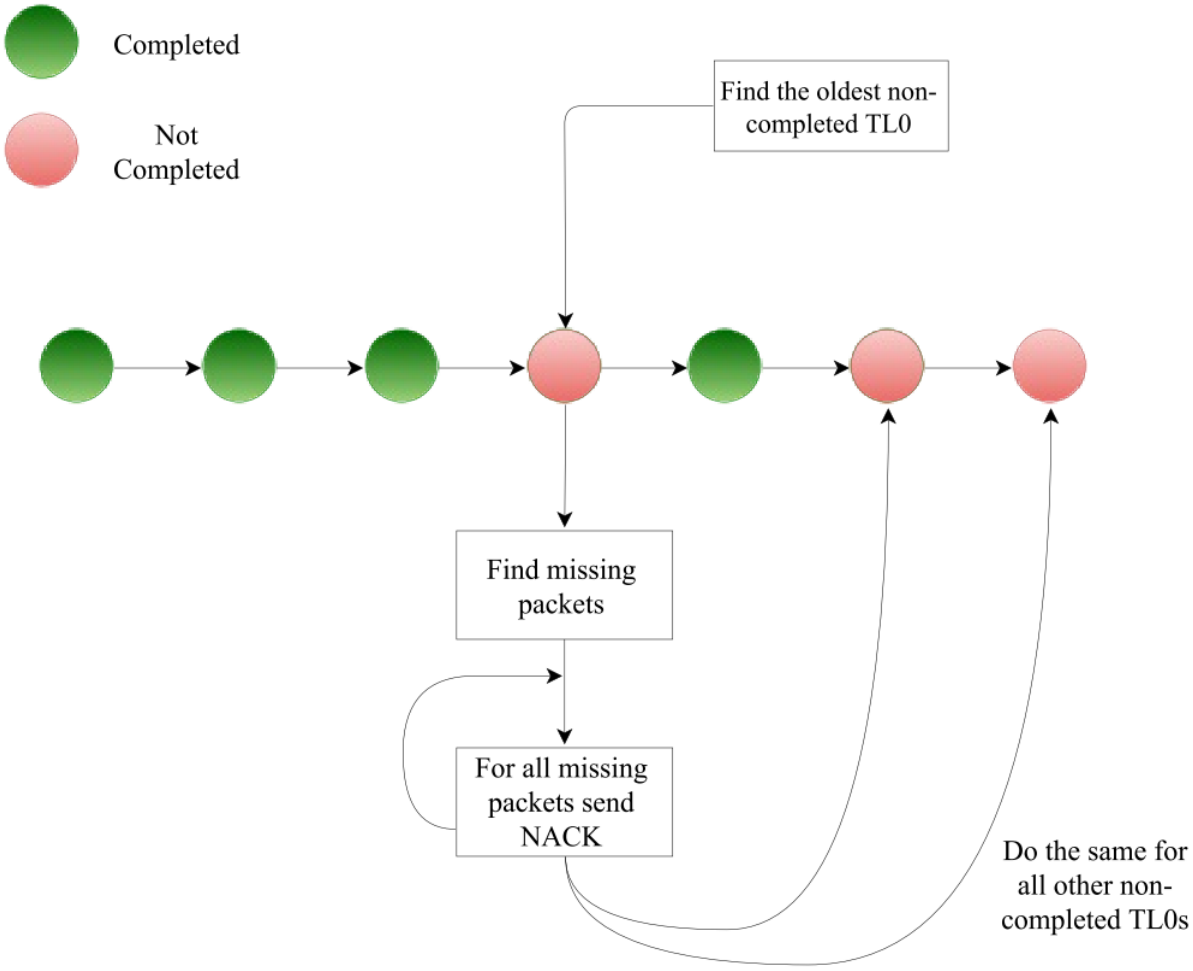
TL0 ή κάποιο ενισχυμένο επίπεδο. Στην περίπτωση που το πακέτο που λάβαμε αφορά κάποιο ενισχυμένο επίπεδο, τότε θέτουμε τη μεταβλητή has_enhancement σε true και ενημερώνουμε τον αλγόριθμο ελέγχου της ακεραιότητας των frames με τις κατάλληλες πληροφορίες. Αλλιώς, αν το πακέτο αυτό αφορά κάποιο TL0 επίπεδο, τότε πρέπει να ελέγξουμε αν το έχουμε λάβει ξανά. Ο λόγος που μπορεί να το έχουμε λάβει ξανά οφείλεται στις αναμεταδόσεις που πραγματοποιεί ο αποστολέας, καθώς, όπως θα δούμε παρακάτω, το ίδιο πακέτο μπορεί να το ζητήσουμε παραπάνω από μία φορές, μιας και είναι πιθανό να χάθηκε όπως συμβαίνει με τα κανονικά μεταδιδόμενα πακέτα. Αν το πακέτο δεν το έχουμε λάβει ξανά, τότε πρέπει να ενημερώσουμε τη δομή TL0_info σύμφωνα με τις πληροφορίες που εμπεριέχονται στο πακέτο που ήρθε. Για την ενημέρωση της δομής, σε πρώτο στάδιο, προσδιορίζουμε τη θέση στον πίνακα seq_index που αντιστοιχεί στον RTP αριθμό ακολουθίας του πακέτου, και τη θέτουμε σε true και αμέσως μετά ελέγχουμε αν όλες του οι θέσεις έχουν τεθεί σε true. Αν όλες οι θέσεις του είναι true, τότε έχουμε λάβει επιτυχώς όλα τα NALUs που αφορούν το συγκεκριμένο TL0 AU και συνεπώς θέτουμε τη μεταβλητή tl0_completed σε true. Η διαδικασία ενημέρωσης της δομής TL0_info φαίνεται στην Εικόνα 42.



Εικόνα 42: Διαδικασία ενημέρωσης του TL0_info

Έπειτα, είτε είναι ένα πακέτο που αφορά κάποιο ενισχυμένο επίπεδο είτε είναι ένα νέο πακέτο που αφορά κάποιο TL0 και ενημερώσαμε τη δομή TL0_info, πρέπει να ελέγξουμε ποια TL0 πακέτα μας λείπουν γενικά και να τα ζητήσουμε από τον αποστολέα. Ο αλγόριθμος που χρησιμοποιούμε ακολουθεί τη λογική του LRU, μιας θα φροντίσουμε πρώτα να ζητήσουμε τα χαμένα πακέτα των πιο παλαιών TL0, καθώς για τη σωστή αποκωδικοποίηση του $TL0_N$ χρειαζόμαστε την αποκωδικοποίηση του $TL0_{N-1}$.

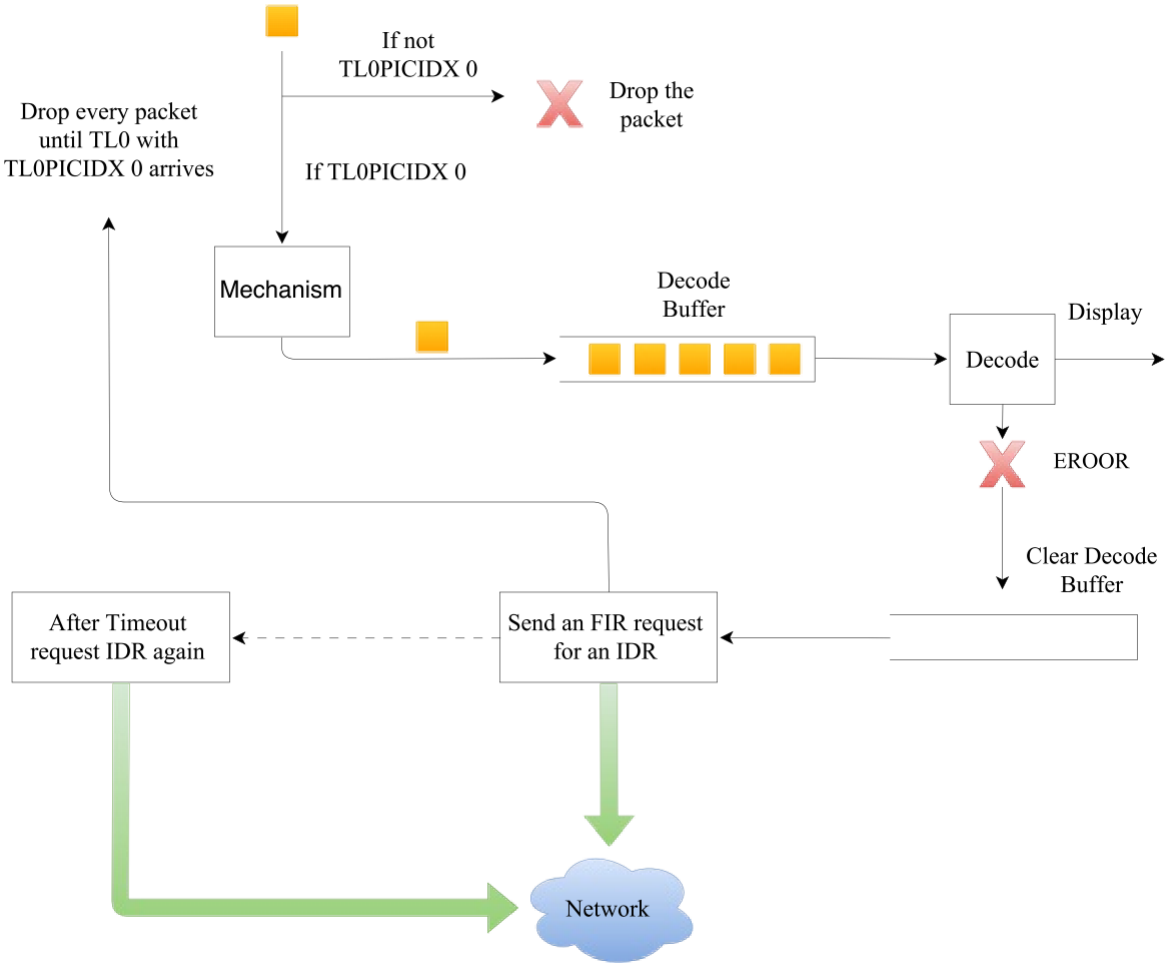
Συνεπώς, από τη λίστα με τα TL0_info εντοπίζουμε το πιο παλαιό TL0_info του οποίου η μεταβλητή tl0_completed είναι false. Για αυτό το TL0_info, για εκείνα τα στοιχεία του πίνακα seq_index που είναι false αντιστοιχίζουμε τους RTP αριθμούς ακολουθίας που τα αντιπροσωπεύουν και τους στέλνουμε μέσω RTCP NACK πακέτων στον πομπό. Έπειτα, εντοπίζουμε το επόμενο TL0_info που δεν είναι ολοκληρωμένο και επαναλαμβάνουμε την ίδια διαδικασία κ.ό.κ.. Η διαδικασία εύρεσης και αίτησης των χαμένων TL0 πακέτων φαίνεται στην Εικόνα 43.



Εικόνα 43: Διαδικασία εύρεσης χαμένων TL0 πακέτων και αίτηση αναμετάδοσης

Στο επόμενο βήμα, αφού εντοπίσαμε και ζητήσαμε όλα τα χαμένα πακέτα, πρέπει να τοποθετήσουμε το νέο αυτό πακέτο στον jitter buffer. Από τον έλεγχο που κάναμε πριν σχετικά με το αν το πακέτο το έχουμε πάρει ξανά ή όχι, είμαστε σίγουροι πως δε θα υπάρχει μέσα στον buffer, οπότε και το τοποθετούμε στη σωστή σειρά. Στη συνέχεια παίρνουμε το πρώτο πακέτο από τον jitter buffer και εκτελούμε τον αλγόριθμο ελέγχου τις ακεραιότητας των frames, προκειμένου να αποφασίσουμε αν το πακέτο πρέπει να προωθηθεί προς τον αποκωδικοποιητή ή όχι. Σε περίπτωση που υπάρχει κάποιο πρόβλημα, τότε ενημερώνουμε τις κατάλληλες μεταβλητές ώστε όλα τα επόμενα πακέτα που σχετίζονται με τα προβληματικά frames να μην προωθηθούν προς τον αποκωδικοποιητή. Με τον τρόπο αυτό, εξασφαλίζουμε ότι τα frames που θα χτίζει ο αποκωδικοποιητής θα είναι πάντα έγκυρα και χωρίς σφάλματα.

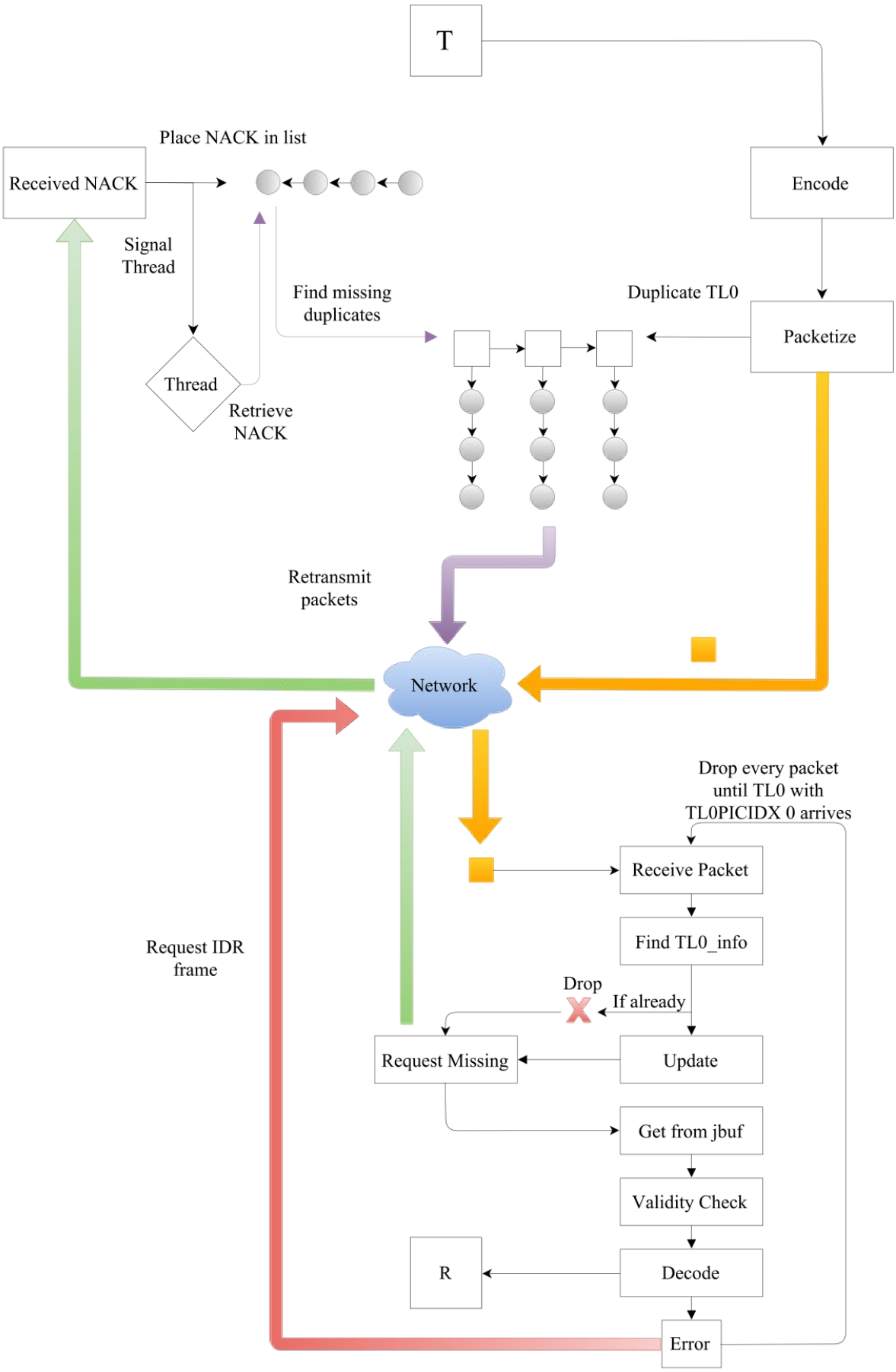
Σε περίπτωση, όμως που προκύψει κάποιο σφάλμα στην αποκωδικοποίηση, τότε είμαστε σίγουροι πως το πρόβλημα προκύπτει κατά την αποκωδικοποίηση ενός TL0 AU, γεγονός που μπορεί να συμβεί μόνο σε περιπτώσεις μεγάλων απωλειών. Τότε, επειδή είναι αδύνατο να επανακτήσουμε την ακολουθία των TL0 σε σύντομο χρονικό διάστημα, στέλνουμε ένα αίτημα ανανέωσης του frame στον αποστολέα, όπου αναγκάζουμε τον κωδικοποιητή να παράξει μια IDR εικόνα, ώστε να αρχίσουμε το χτίσιμο της αλυσίδας των TL0 από την αρχή. Παράλληλα, φροντίζουμε να καθαρίσουμε τη μνήμη του αποκωδικοποιητή από υπολείμματα προηγούμενων AUs και ενεργοποιούμε ένα μηχανισμό, ο οποίος κατά την άφιξη των πακέτων σε περίπτωση που έχουμε ζητήσει IDR φροντίζει να τα “πετάει” μέχρι να δεχτεί ένα TL0 με TL0PICIDX ίσο με το 0. Επιπλέον, επειδή τα IDR πακέτα μπορεί να χαθούν, φροντίζουμε ύστερα από κάποιο χρονικό διάστημα που περιμένουμε χωρίς να μας έρθουν τα απαιτούμενα πακέτα να ζητήσουμε ξανά από τον κωδικοποιητή να παράξει μια IDR εικόνα. Η διαδικασία ανάκαμψης σε περίπτωση σφάλματος παρουσιάζεται στην Εικόνα 44.



Εικόνα 44: Διαδικασία ανάκαμψης σε περίπτωση σφάλματος

Τέλος, γνωρίζοντας πλέον την πλήρη διαδικασία που ακολουθεί ο αλγόριθμός μας κατά τη λήψη ενός νέου πακέτου που αφορά κάποιο TL0 επίπεδο ή ενός πακέτου που αφορά κάποιο ενισχυμένο επίπεδο, επιστρέφουμε να μελετήσουμε την περίπτωση που όπου λαμβάνουμε ένα πακέτο το οποίο αφορά κάποιο TL0 επίπεδο, αλλά το έχουμε

πάρει ξανά. Το πακέτο αυτό είναι ένα αναμεταδιδόμενο πακέτο και το ότι το έχουμε πάρει ξανά οφείλεται στο γεγονός ότι είναι πιθανό να χάθηκε, όπως συμβαίνει με τα κανονικά μεταδιδόμενα πακέτα, και να το ζητήσαμε παραπάνω από μία φορές. Όταν λοιπόν εντοπίσουμε ότι ένα πακέτο το έχουμε λάβει ξανά, δεν πρέπει να ενημερώσουμε κάτι στη δομή TL0_info που του αντιστοιχεί αλλά ούτε να επιχειρήσουμε να το τοποθετήσουμε στον jitter buffer. Αυτό που χρειάζεται να κάνουμε είναι να εντοπίσουμε και να ζητήσουμε όλα τα χαμένα TL0 πακέτα και στη συνέχεια να πάρουμε ένα στοιχείο από τον jitter buffer και να αποφασίσουμε αν θα το δώσουμε για αποκωδικοποίηση. Η τελική μορφή που λαμβάνει το σύστημα ανίχνευσης και αναμετάδοσης χαμένων TL0 πακέτων παρουσιάζεται στην Εικόνα 45.



Εικόνα 45: Η τελική μορφή του συστήματος ανίχνευσης και αναμετάδοσης TL0 πακέτων

Συνοψίζοντας, δημιουργήσαμε ένα μηχανισμό υψηλής ανθεκτικότητας ανάμεσα στον πομπό και το δέκτη και προσδώσαμε τη δυνατότητα στο BareSIP να φροντίζει για την ακεραιότητα των AU που επρόκειτο να τροφοδοτήσει στον αποκωδικοποιητή, ώστε η προσπάθεια που καταβάλει για την άμεση αποκατάσταση των χαμένων TL0 πακέτων με τη χρήση αναμετάδοσης, να είναι ικανή για την διατήρηση της ακολουθίας τους. Με τον τρόπο αυτό, όπως θα παρουσιάσουμε και στο κεφάλαιο των αποτελεσμάτων, πετυχαίνουμε η αναπαραγωγή βίντεο στην έξοδο να είναι εύρωστη και χωρίς μεγάλες διακοπές ή παραμορφώσεις, καθώς το πλήθος των frames που καταφέρνουμε να αναπαράξουμε διατηρείται υψηλό παρά τις απώλειες που προκύπτουν.

6. ΜΕΤΡΗΣΕΙΣ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

Στην ενότητα αυτή θα περιγράψουμε τις μετρήσεις που πραγματοποιούμε, τόσο για το μηχανισμό συνδιάσκεψης όσο και για το μηχανισμό αναμετάδοσης TL0 πακέτων, καθώς και τα αποτελέσματα που προκύπτουν από τις μετρήσεις αυτές.

6.1 Μηχανισμός συνδιάσκεψης πολλαπλών χρηστών στο BareSIP

Στα πειράματα που πραγματοποιούμε, μελετάμε την απόδοση ως προς την καθυστέρηση αποστολής και λήψης της αρχικής έκδοσης του BareSIP και της έκδοσης του BareSIP όταν αυτή χρησιμοποιεί το μηχανισμό συνδιάσκεψης για δύο μέχρι πέντε συμμετέχοντες. Στην περίπτωση των δύο συμμετεχόντων, συγκρίνουμε την απόδοση των δύο αυτών διαφορετικών συστημάτων, καθώς επιτελούν την ίδια λειτουργία, ενώ για τρεις έως πέντε χρήστες θα μελετήσουμε απλά την απόδοση του BareSIP όταν αυτή χρησιμοποιεί το μηχανισμό συνδιάσκεψης. Επιπλέον, θα εκτιμήσουμε την καθυστέρηση εξυπηρέτησης του κάθε πακέτου στον εξυπηρετητή του συστήματος.

Ως προς τη μετάδοση, ο συνολικός χρόνος μετάδοσης ενός πακέτου ήχου ή βίντεο σε κάθε τερματικό κόμβο είναι το άθροισμα των επιμέρους καθυστερήσεων για τη σύλληψη των δεδομένων (μικρόφωνο, κάμερα), την κωδικοποίηση και την αποστολή. Σύμφωνα με τον τρόπο που λειτουργεί το BareSIP, στην περίπτωση του ήχου, τα δείγματα που συλλαμβάνονται από το μικρόφωνο έχουν διάρκεια όση ορίζει η διάρκεια του πακέτου (packet time – ptime) και συνεπώς τα πακέτα που συλλαμβάνονται, που κωδικοποιούνται και που αποστέλλονται, περιέχουν δεδομένα διάρκειας packet time. Στην περίπτωση του βίντεο, η κάμερα συλλαμβάνει ένα ολόκληρο frame και ο κωδικοποιητής τοποθετεί τα κωδικοποιημένα δεδομένα σε ένα ή περισσότερα RTP πακέτα μεγέθους pktsize. Στις μετρήσεις που πραγματοποιούμε, για τον υπολογισμό της καθυστέρησης μετάδοσης ενός πακέτου δε λαμβάνουμε υπόψη μας την καθυστέρηση για τη σύλληψη των δεδομένων, αλλά μελετάμε από τη στιγμή που τα δείγματα τροφοδοτηθούν στο BareSIP μέχρι τη στιγμή που αποστέλλονται στο δίκτυο.

Στο σύστημα MusiNet, στο οποίο ενσωματώθηκε ο μηχανισμός συνδιάσκεψης, εστιάζουμε στη μείωση της καθυστέρησης για τη μετάδοση και λήψη πακέτων ήχου, καθώς είναι ο βασικότερος παράγοντας για τη βελτίωση της εμπειρίας. Μια παράμετρος, λοιπόν, που επηρεάζει τη συνολική καθυστέρηση του συστήματος είναι η διάρκεια των πακέτων (ptime). Η χρήση μεγαλύτερου buffer σημαίνει ότι η πηγή ήχου συλλαμβάνει μεγαλύτερα χρονικά διαστήματα, γεγονός που αυξάνει την καθυστέρηση από άκρο σε άκρο, καθώς πρέπει να περιμένουμε περισσότερο για να γεμίσει ο buffer. Όπως αναφέραμε προηγουμένως, για τις μετρήσεις μας αγνοούμε το χρόνο που απαιτείται για τη σύλληψη δειγμάτων από την πηγή. Ωστόσο, η αύξηση της διάρκειας των πακέτων σημαίνει ότι πρέπει να διαχειριστούμε μεγαλύτερα πακέτα, γεγονός που οδηγεί σε αύξηση στην καθυστέρηση επεξεργασίας των πακέτων, μολονότι μειώνεται το πλήθος των πακέτων που χρειάζεται να μεταδώσουμε.

Τα πειράματα διεξήχθησαν σε LAN, όπου πραγματοποιήσαμε μια σειρά κλήσεων με σχεδόν με σχεδόν πανομοιότυπο περιεχόμενο ήχου/βίντεο διάρκεια περίπου 60 sec – όλα τα αποτελέσματα αναφέρονται σε μέσες τιμές. Οι μετρήσεις πραγματοποιήθηκαν σε ένα Apple Mac mini, το οποίο τρέχει ένα στιγμιότυπο του BareSIP, με τους υπόλοιπους συμμετέχοντες να εκτελούνται σε ένα αντίστοιχο μηχάνημα το οποίο εκτελεί ένα στιγμιότυπου του BareSIP για κάθε συμμετέχοντα. Κάθε τερματικός κόμβος χρησιμοποιεί επεξεργαστή Intel Core i5 στα 2.5 GHz, με λογισμικό MacOS X version 10.9.5. Για την κωδικοποίηση του ήχου χρησιμοποιούμε τον κωδικοποιητή Opus ενώ

για την κωδικοποίηση βίντεο τον H.263. Για τη λήψη και αναπαραγωγή του ήχου χρησιμοποιούμε το CoreAudio, για τη λήψη βίντεο το AVCapture API και για την αναπαραγωγή βίντεο το OpenGL. Ο εξυπηρετητής είναι εγκατεστημένος σε ένα MacBook Pro με επεξεργαστή Intel Core i7 στα 2.7 GHz, με λογισμικό MacOS X version 10.9.5.

Στον Πίνακας 1 παρουσιάζουμε την καθυστέρηση μετάδοσης ανά πακέτο σε microseconds (μsec) για ήχο και βίντεο. Με “P2P” συμβολίζουμε την αρχική έκδοση του BareSIP, χωρίς ενδιάμεσο εξυπηρετητή. Με “VC” συμβολίζουμε την έκδοση του BareSIP όταν λειτουργεί σε κατάσταση συνδιάσκεψης πολλαπλών χρηστών, και ο αριθμός που ακολουθεί συμβολίζει το πλήθος των συμμετεχόντων. Στην περίπτωση αυτή, όλες οι ροές δεδομένων διέρχονται μέσων του εξυπηρετητή.

Καθυστέρηση μετάδοσης ήχου ανά πακέτο (*μ*sec)					
ptime	P2P	VC2	VC3	VC4	VC5
20	166.54	170.08	173.11	174.6	173.24
10	115.39	120.31	120.92	120.92	120.9
5	72.6	76.36	76.19	76.4	79.74
Καθυστέρηση μετάδοσης βίντεο ανά πακέτο (*μ*sec)					
pktsize	P2P	VC2	VC3	VC4	VC5
1024	1663.98	1778.97	1804.79	1797.44	1821.24

Πίνακας 1: Καθυστέρηση μετάδοσης ανά πακέτο ήχου/βίντεο σε μsec

Τόσο για τον ήχο όσο και για το βίντεο, παρατηρούμε ότι καθώς αυξάνεται ο αριθμός των συμμετεχόντων αλλά και το μέγεθος του πακέτου, η καθυστέρηση μετάδοσης γενικά αυξάνεται. Στην περίπτωση του ήχου, μειώνοντας τη διάρκεια του πακέτου (ptime), τόσο σε κατάσταση P2P όσο και σε VC παρατηρούμε μια σημαντική μείωση στην καθυστέρηση μέχρι και 57%. Τόσο στην περίπτωση του ήχου όσο και του βίντεο, με την αύξηση του αριθμού των συμμετεχόντων και διατηρώντας το μέγεθος του πακέτου σταθερό, παρατηρούμε μια αύξηση στην καθυστέρηση μέχρι και 10% αλλά της τάξης των μsec, γεγονός που την καθιστά αμελητέα.

Ως προς τη λήψη, ο συνολικός χρόνος λήψης δεδομένων είναι το άθροισμα του χρόνου για τη λήψη των RTP πακέτων, του χρόνου αναμονής στους jitter buffers, του χρόνου αποκωδικοποίησης και του χρόνου αναπαραγωγής. Σύμφωνα με τον τρόπο που λειτουργεί το BareSIP, μετά τη διαδικασία της αποκωδικοποίησης, καλούνται συγκεκριμένοι χειριστές (handlers) οι οποίοι είτε αποθηκεύουν τα αποκωδικοποιημένα δείγματα ήχου στον buffer αναπαραγωγής είτε απεικονίζουν το βίντεο frame. Στην περίπτωση του βίντεο, ένα πακέτο μπορεί να περιέχει ένα ολόκληρο ή ένα τμήμα ενός frame. Σε κάθε περίπτωση, μελετάμε το χρόνο που χρειάζεται για τη λήψη ενός πακέτου, χωρίς να υπολογίζουμε το τμήμα που αφορά την αναπαραγωγή ήχου και βίντεο. Εκτός από το περιεχόμενο και τη διάρκεια των πακέτων, μια άλλη παράμετρος η οποία επηρεάζει σημαντικά τη διαδικασία λήψης είναι το μέγεθος του jitter buffer, το οποίο ορίζεται ως πλαίσια/πακέτο. Στον Πίνακας 2 παρουσιάζουμε την καθυστέρηση λήψης ανά πακέτο ήχου/βίντεο σε milliseconds (msec), συμπεριλαμβανομένου και το μέγεθος του jitter buffer σαν επιπλέον παράμετρο.

Για τη λήψη ήχου, το οποίο είναι το πιο κρίσιμο ζήτημα στο MusiNet, διατηρώντας τη διάρκεια του πακέτου και το μέγεθος του jitter buffer σταθερά, η εκτιμώμενη καθυστέρηση είναι σχεδόν πανομοιότυπη τόσο σε κατάσταση P2P όσο και σε κατάσταση VC. Συγκεκριμένα, η εκτιμώμενη καθυστέρηση είναι περίπου ίση με τη διάρκεια του πακέτου πολλαπλασιασμένη με το μέγεθος του jitter buffer. Για τη λήψη βίντεο, διατηρώντας το μέγεθος του jitter buffer σταθερό, παρατηρούμε ότι καθώς ο αριθμός των συμμετεχόντων αυξάνεται, η καθυστέρηση επεξεργασίας αυξάνεται μέχρι και 38%. Επιπλέον, τόσο σε κατάσταση P2P όσο και σε VC, μειώνοντας το μέγεθος του jitter buffer, παρατηρούμε ότι οι εκτιμώμενες καθυστερήσεις μειώνονται σχεδόν κατά το ίδιο ποσοστό.

Καθυστέρηση λήψης ήχου ανά πακέτο (msec)						
jbuf	ptime	P2P	VC2	VC3	VC4	VC5
10	20	199.658	200.087	200.069	199.914	200.083
10	10	99.834	100.065	100.064	100.034	100.025
10	5	49.989	50.045	50.025	50.028	50.014
5	20	99.773	100.099	100.099	100.078	100.048
5	10	49.959	50.059	49.956	50.049	50.031
5	5	25.011	25.047	25.046	25.044	25.037
1	20	20.003	20.097	20.122	20.098	20.056
1	10	10.029	10.073	10.057	10.065	10.061
1	5	5.031	5.0565	5.062	5.059	5.060
Καθυστέρηση λήψης βίντεο ανά πακέτο (msec)						
jbuf	ptime	P2P	VC2	VC3	VC4	VC5
10	1024	206.363	271.162	285.208	275.401	283.088
5	1024	105.546	129.341	140.409	137.017	139.058
1	1024	20.652	21.865	23.623	23.001	23.778

Πίνακας 2: Καθυστέρηση λήψης ανά πακέτο ήχου/βίντεο σε msec

Όσον αφορά την επίδοση του εξυπηρετητή, παρουσιάζουμε την καθυστέρηση μεταβίβασης (relay) ανά πακέτο ήχου στον Πίνακας 3. Παρατηρούμε ότι καθώς αυξάνεται ο αριθμός των συμμετεχόντων και το packet time, το σύστημα παρουσιάζει μια αύξηση στην καθυστέρηση επεξεργασίας μέχρι και 260%. Σημειώνεται, ωστόσο, ότι οι μετρήσεις αυτές είναι της τάξης των μsec.

Καθυστέρηση μεταβίβασης ανά πακέτο ήχου στον εξυπηρετητή (*μ*sec)				
ptime	VC2	VC3	VC4	VC5
20	96.75	147.22	249.12	217.79
10	87.94	114.93	152.49	134.31
5	68.54	91.68	114/79	139.98

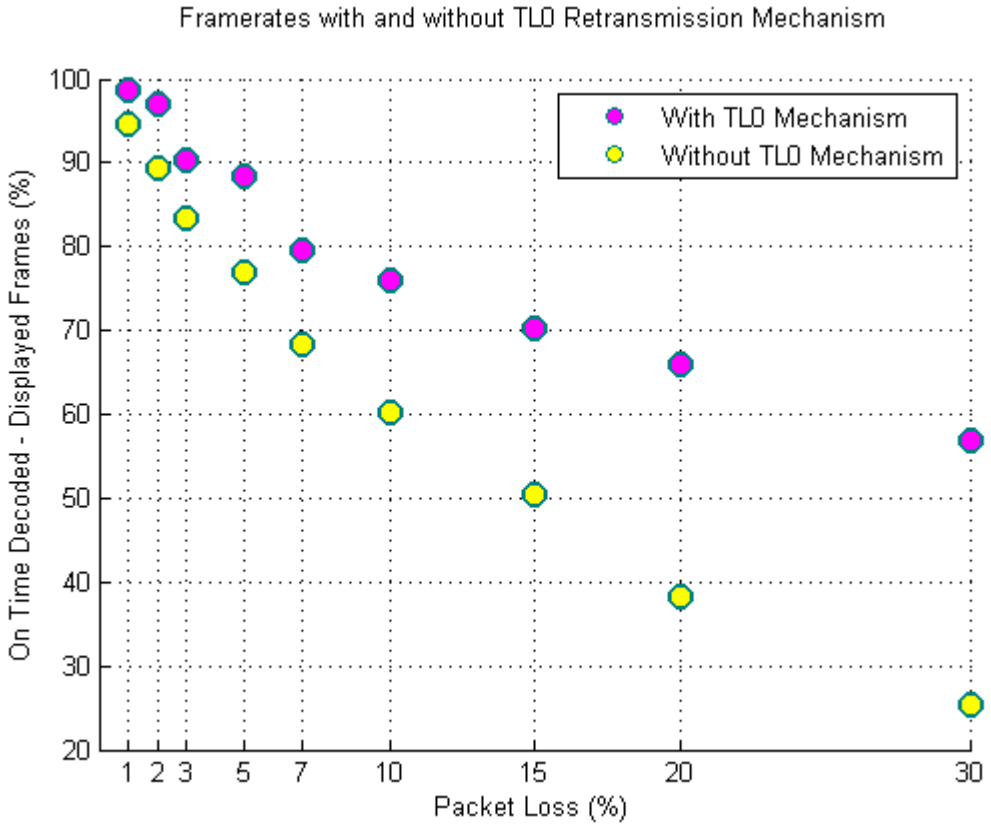
Πίνακας 3: Καθυστέρηση μεταβίβασης ανά πακέτο ήχου στον εξυπηρετητή

6.2 Σύστημα αναμετάδοσης TL0 πακέτων βίντεο στο BareSIP

Δεδομένου ότι ο μηχανισμός αναμετάδοσης TL0 πακέτων βίντεο δημιουργήθηκε με προοπτική την ενσωμάτωσή του στο ερευνητικό πρόγραμμα SeNSE, οφείλουμε να μελετήσουμε κατά πόσο η απόδοσή του μπορεί να χρησιμοποιηθεί σε ένα πραγματικό σύστημα τηλεοπτικής μετάδοσης. Συνεπώς, για την αξιολόγηση της απόδοσης τους συστήματος πραγματοποιήθηκαν πειράματα που είχα ως στόχο να προσδιορίσουν κατά πόσο η επιλογή της επέκτασης SVC και των δυνατοτήτων που προσφέρει, σε συνδυασμό με το μηχανισμό αναμετάδοσης TL0 πακέτων βίντεο, μπορεί να χρησιμοποιηθεί σε ένα πραγματικό σύστημα τηλεοπτικής μετάδοσης. Στις μετρήσεις που πραγματοποιήσαμε, λοιπόν, επιλέξαμε να αξιολογήσουμε την ποιότητα του λαμβανόμενου βίντεο αλλά και την αποτελεσματικότητα του μηχανισμού αναμετάδοσης, όταν η εφαρμογή λειτουργεί σε ένα πραγματικό δίκτυο. Για τη μέτρηση της ποιότητας του λαμβανόμενου βίντεο, ως μέτρο ποιότητας χρησιμοποιήθηκε ο αριθμός των εικόνων (frames) που αποκωδικοποιήθηκαν σωστά και εντός του χρόνου που οφείλει μια εικόνα να προβληθεί (ο χρόνος αυτός καθορίζεται από τον αριθμό των εικόνων ανά δευτερόλεπτο – Frames per Second – fps), σε σχέση με το ποσοστό των απωλειών που εισάγει το δίκτυο. Για τη μέτρηση της βελτίωσης που εισάγει ο μηχανισμός, ως μέτρο χρησιμοποιήθηκε ο λόγος των ωφέλιμων bytes, δηλαδή των bytes που αποκωδικοποιήθηκαν, προς το συνολικό αριθμό των bytes που έστειλε ο κωδικοποιητής, σε σχέση με το ποσοστό των απωλειών που εισάγει το δίκτυο.

Ως προς τις συνθήκες υπό τις οποίες εκπονήθηκαν τα πειράματα, η μετάδοση πραγματοποιήθηκε σε τοπικό δίκτυο, όπου δύο υπολογιστές ήταν συνδεδεμένοι ενσύρματα σε ένα δρομολογητή. Και στους δύο υπολογιστές εκτελούταν το λογισμικό BareSIP, όπως αυτό τροποποιήθηκε, αλλά το ένα τερματικό επιτελούσε τη λειτουργία της σύλληψης, της κωδικοποίησης και της αποστολής βίντεο ενώ το άλλο επιτελούσε τη λειτουργία της λήψης, της αποκωδικοποίησης και της αναπαραγωγής. Και τα δύο αυτά μηχανήματα είναι NB Toshiba Satellite N50-A-10V, με επεξεργαστή Intel Core i7-4710HQ, CPU 2.50GHz x 8 και με λειτουργικό σύστημα Ubuntu 14.04 LTS. Επιπλέον, προκειμένου να υπάρχει συνέπεια στη σύγκριση των αποτελεσμάτων ανάμεσα στις διαφορετικές μετρήσεις, τόσο στην περίπτωση της έκδοσης με το μηχανισμό όσο και στην περίπτωση της έκδοσης χωρίς το μηχανισμό χρησιμοποιήθηκε ένα raw αρχείο βίντεο, μορφής YUV, αποτελούμενο από 300 frames και ρυθμό προβολής 30 fps, ενώ σε τεχνικό επίπεδο ο κωδικοποιητής είχε ρυθμιστεί να παράγει 1 IDR frame κάθε 120 frames (GOP 120). Επίσης, η προσομοίωση του πραγματικού δικτύου για τις απώλειες πακέτων βίντεο πραγματοποιήθηκε από μια διεργασία διακριτού χρόνου, της οποίας η συνάρτηση πυκνότητας πιθανότητας σε κάθε βήμα ήταν ομοιόμορφα και ανεξάρτητα κατανεμημένη. Με άλλα λόγια, κάθε πακέτο είχε μια συγκεκριμένη πιθανότητα να χαθεί ανεξάρτητα από τα άλλα πακέτα. Στην περίπτωση που μελετάμε, ένα RTP πακέτο αντιστοιχεί σε ένα NALU, με μέγιστο μέγεθος τα 1.5KB, και συνεπώς για κάθε ένα πακέτο που χάνεται, πρακτικά χάνεται ένα NALU. Τέλος, να αναφέρουμε ότι τα αποτελέσματα των πειραμάτων που προέκυψαν αποτελούν ένα συνδυασμό εκτελέσεων με τη χρήση του μηχανισμού αναμετάδοσης των TL0 NALUs και χωρίς τη χρήση του μηχανισμού. Πιο συγκεκριμένα, επειδή η διάρκεια του βίντεο είναι ιδιαίτερα μικρή (11sec) αλλά και λόγω της φύσης της διεργασίας προσομοίωσης των απωλειών πακέτων βίντεο, υπήρξαν περιπτώσεις όπου οι απώλειες πακέτων που προέκυπταν υπερτερούσαν ή υστερούσαν του ποσοστού που είχαμε θέσει, με αποτέλεσμα να παίρνουμε ακραίες μετρήσεις ή μετρήσεις με πολύ μεγάλη απόκλιση από το γενικότερο μέσο όρο. Για το λόγο αυτό, πραγματοποιήσαμε πολλές επαναλήψεις και κρατήσαμε ως τελική μέτρηση το μέσο όρο των παρατηρήσεων, ενώ απορρίψαμε τις μετρήσεις εκείνες, οι οποίες είχαν πολύ μεγάλη απόκλιση από το επιθυμητό ποσοστό απωλειών πακέτων.

Ξεκινώντας με την πρώτη μετρική που αφορά την αξιολόγηση της ποιότητας του λαμβανόμενου βίντεο, μελετάμε το λόγο $\frac{Effective\_Frames}{Total\_Transmitted\_Frames}$, όπου ως *Effective_Frames* ορίζουμε τον αριθμό των frames που αποκωδικοποιήθηκαν σωστά και ως *Total_Transmitted_Frames* τον αριθμό των frames που μετέδωσε ο αποστολέας, στην περίπτωσή μας 300, συναρτήσει του ποσοστού απωλειών του δικτύου. Οι τιμές που προέκυψαν κατά την εκτέλεση των πειραμάτων τόσο με τη χρήση του μηχανισμού αναμετάδοσης όσο και χωρίς αυτόν, παρουσιάζονται στην Εικόνα 46. Όπως παρατηρούμε, για μικρό ποσοστό απωλειών της τάξης του 1-2%, τόσο στην περίπτωση όπου χρησιμοποιείται ο μηχανισμός αναμετάδοσης όσο και στην περίπτωση που δε χρησιμοποιείται, επιτυγχάνεται ένας πολύ καλός ρυθμός σωστής αποκωδικοποίησης και αναπαραγωγής των frames, γεγονός που σημαίνει ότι για μικρές απώλειες ο θεατής έχει αρκετά



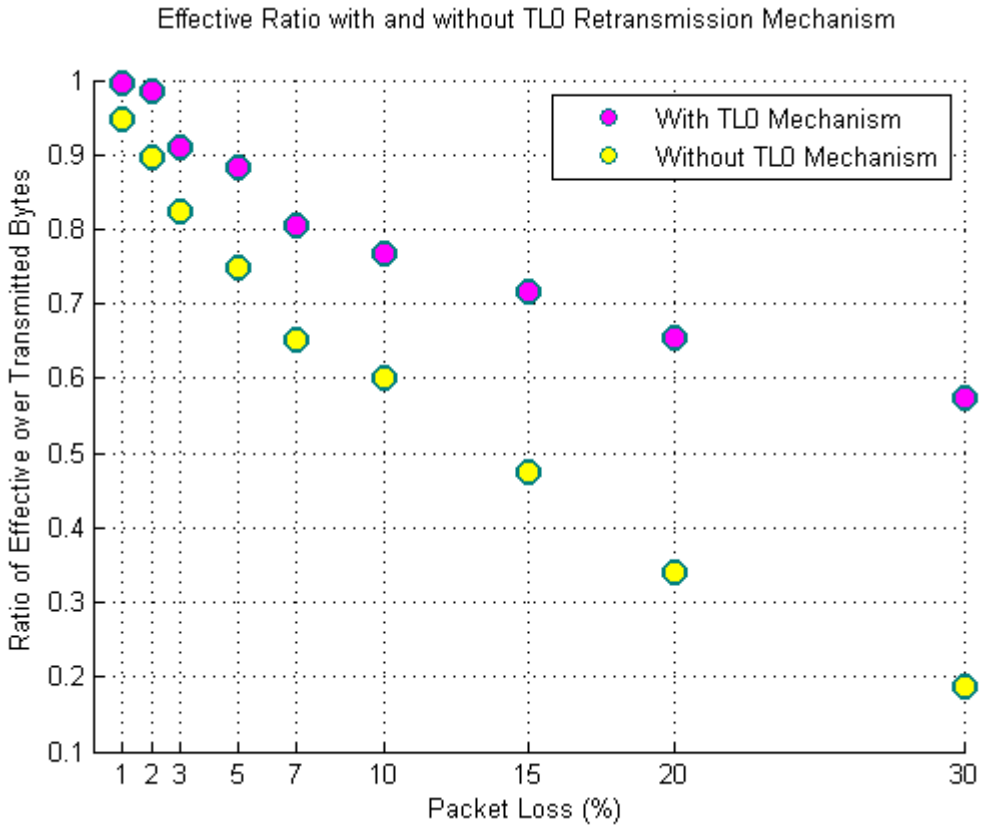
Εικόνα 46: Ο αριθμός των ωφέλιμων frames συναρτήσει των απωλειών

ικανοποιητική εμπειρία βίντεο, σε βαθμό που δεν μπορεί να αντιληφθεί τις τυχόν απώλειες. Καθώς όμως το ποσοστό των απωλειών αυξάνει, παρατηρούμε ότι οι διαφορές ανάμεσα στην περίπτωση μετάδοσης με τη χρήση του μηχανισμού αναμετάδοσης και στην περίπτωση μετάδοσης χωρίς το μηχανισμό είναι αρκετές και άμεσα αντιληπτές από το χρήστη. Φαίνεται πως όταν ο μηχανισμός δεν είναι ενεργοποιημένος, η ποιότητα στην έξοδο μειώνεται σχεδόν εκθετικά, σε αντίθεση με την περίπτωση που ο μηχανισμός ενεργοποιημένος, όπου φαίνεται μια σημαντική προσπάθεια αναχαίτισης των απωλειών των frames, διατηρώντας την ποιότητα του βίντεο σε αρκετά υψηλό επίπεδο. Η διαφορά είναι ιδιαίτερα αισθητή στην περίπτωση

που έχουμε μεγάλες απώλειες πακέτων της τάξης του 30%, όπου περίπου το 55% των frames καταφέρνουμε να το αποκωδικοποιήσουμε και να το αναπαράγουμε, σε αντίθεση με το 15% που επιτυγχάνουμε χωρίς τη χρήση του μηχανισμό. Επιτυγχάνουμε δηλαδή περίπου 265% καλύτερη επίδοση.

Συνεπώς, από τα αποτελέσματα που προκύπτουν από τη μετρική που παρουσιάσαμε στην Εικόνα 46, διαπιστώνουμε ότι η ποιότητα του λαμβανόμενου βίντεο είναι ιδιαίτερα υψηλή, παρά την αύξηση των απωλειών του δικτύου, γεγονός που την καθιστά ικανή για να χρησιμοποιηθεί σε τηλεοπτική μετάδοση, δεδομένου ότι οι συνθήκες του δικτύου θα είναι παρόμοιες, όπως και οι ανοχές που έχουμε θέσεις για τις απώλειες.

Περνώντας στη δεύτερη μετρική που αφορά την αξιολόγηση της βελτίωσης που εισάγει ο μηχανισμός αναμετάδοσης TL0 πακέτων, μελετάμε το λόγο $\frac{Effective\_Bytes}{Total\_Transsmited\_Bytes}$, όπου ως *Effective_Bytes* ορίζουμε τον αριθμό των bytes που χρησιμοποιήθηκαν για αποκωδικοποίηση, ενώ ως *Total_Transmitted_Bytes*, στην περίπτωση που δε χρησιμοποιούμε το μηχανισμό,



Εικόνα 47: Ο ρυθμός των ωφέλιμων bytes συναρτήσει των απωλειών

ορίζουμε τα bytes που παράγει ο κωδικοποιητής και, στην περίπτωση που χρησιμοποιούμε το μηχανισμό, ορίζουμε τα bytes που παράγει ο κωδικοποιητής + τα bytes των αναμεταδόσεων. Η μετρική αυτή παρουσιάζεται στην Εικόνα 47 και ουσιαστικά μας δείχνει κατά πόσο τα bits που παρήγαγε ο κωδικοποιητής χρησιμοποιήθηκαν στην αποκωδικοποίηση, δηλαδή την αποτελεσματικότητα του να διατηρηθούν οι χρήσιμες ιδιότητες της κωδικοποίησης, όπως προβλέπεται από το

πρότυπο H.264, χωρίς να εξαναγκάζεται ο κωδικοποιητής να παράγει πολλά IDR frames, συναρτήσει του ποσοστού των απωλειών πακέτων. Η μετρική αυτή μπορεί να μας δώσει μια έμμεση πληροφορία σχετικά με την ποιότητα του βίντεο, αντιστοιχίζοντας στην τιμή 1 την ιδανική περίπτωση όπου όλα τα bits που δημιούργησε ο κωδικοποιητής χρησιμοποιήθηκαν για αποκωδικοποίηση και σε μικρότερες τιμές τις υπόλοιπες περιπτώσεις. Σημειώνουμε ότι δεν υπάρχει μία προς μία αντιστοιχία με την προηγούμενη μετρική για το frame rate, καθώς, για τον ίδιο αριθμό των frames που αποκωδικοποιούνται, ο λόγος αυτός μπορεί να έχει διαφορετικές τιμές. Στην Εικόνα 47 παρατηρούμε πως για μικρό ποσοστό απωλειών της τάξης του 1-2%, ο λόγος αυτός σε κάθε περίπτωση είναι αρκετά κοντά στο 1 και συνεπώς υπάρχει ένα ιδιαίτερα καλό ποσοστό αξιοποίησης των bits που παράγει ο κωδικοποιητής. Ωστόσο, καθώς το ποσοστό των απωλειών αυξάνει, στην περίπτωση που ο μηχανισμός είναι ενεργοποιημένος παρατηρούμε ότι η αξιοποίηση των bits που παράγει ο κωδικοποιητής είναι σαφώς ανώτερη από την περίπτωση που ο μηχανισμός είναι απενεργοποιημένος.

Συνοψίζοντας, οι δύο μετρικές που μελετήσαμε μας οδηγούν στο συμπέρασμα πως με τη χρήση του μηχανισμού αναμετάδοσης TL0 πακέτων βίντεο μπορούμε να πετύχουμε μεγαλύτερη ποιότητα στην έξοδο του δέκτη αλλά και να αξιοποιήσουμε αρκετά πιο αποτελεσματικά τα bits που παράγει ο κωδικοποιητής. Ο συνδυασμός των μετρικών αυτών μας επιβεβαιώνει πως με τη διατήρηση των TL0 AU στην έξοδο, στις περισσότερες περιπτώσεις το λαμβανόμενο βίντεο θα είναι υψηλής ποιότητας και εύρωστο.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Δεδομένου ότι το MusiNet είναι ένα σύστημα με ιδιαίτερα υψηλές απαιτήσεις ως προς την καθυστέρηση του ήχου, δημιουργήσαμε έναν πολυνηματικό μηχανισμό ο οποίος μπορεί να διατηρήσει τη συνολική καθυστέρηση του ήχου σε πολύ χαμηλά επίπεδα, σε σημείο που να μην είναι αντιληπτή από το ανθρώπινο αυτί. Τα αποτελέσματα αυτά ενισχύονται ακόμη περισσότερο με το γεγονός ότι η καθυστέρηση αυτή παραμένει σχεδόν σταθερή, ανεξάρτητα από την αύξηση των συμμετεχόντων στη συνδιάσκεψη. Το γεγονός αυτό μας δίνει τη δυνατότητα να εστιάσουμε και να μελετήσουμε παραμέτρους όπως η καθυστέρηση του δικτύου, οι απώλειες πακέτων, οι συνδυασμοί των ptime και jitter buffer size κ.ά., οι οποίες θα ανταποκρίνονται στις συνθήκες ενός πραγματικού δικτύου και να μελετήσουμε τεχνικές οι οποίες θα μας επιτρέψουν να έχουμε αντίστοιχα καλές επιδόσεις σε μια πραγματική μουσική επίδοση. Παράλληλα, αξίζει να μελετήσουμε την πολυπλοκότητα που προσδίδει στο σύστημα ο πολυνηματικός μηχανισμός και να αναπτύξουμε τεχνικές ώστε να γίνει πιο ελαφρύς και με μεγαλύτερη αποδοτικότητα. Τέλος, μπορούμε να εστιάσουμε στη βελτίωση της απόδοσης του βίντεο, όπου μπορούμε να αξιοποιήσουμε τα χαρακτηριστικά του SVC για να πετύχουμε μια εξίσου καλή απόδοση με στόχο τον τέλειο συγχρονισμό του με τον ήχο.

Όσον αφορά το μηχανισμό αναμετάδοσης TL0 πακέτων βίντεο, είδαμε ότι η συνεισφορά του σε ένα πραγματικό σύστημα τηλεοπτικής μετάδοσης είναι ιδιαίτερα μεγάλη, καθώς επιτυγχάνει πολύ υψηλές επιδόσεις παρά την αύξηση των απωλειών του δικτύου. Σε επόμενο στάδιο μπορούμε να διαμορφώσουμε τον κωδικοποιητή να αντικαθιστά με “ελαττωματική” πληροφορία τα frames τα οποία χάνονται ή δεν είναι ολοκληρωμένα, προκειμένου να μελετήσουμε την απόδοση του συστήματος σε επίπεδο Y-PSNR. Επιπλέον, η μορφή του αλγορίθμου στο δέκτη είναι τέτοια που μπορεί να αποτελέσει αντικείμενο έρευνας, προκειμένου να δημιουργηθεί ένας πολυπαραμετρικός αλγόριθμος, ο οποίος θα μπορεί να προσαρμόζεται ανάλογα με τη φύση της εφαρμογής (τηλεοπτική μετάδοση, μετάδοση σε πραγματικό χρόνο μέσω διαδικτύου, μουσική επίδοση μέσω διαδικτύου κ.ά..) και να μπορεί να αποφασίζει τον τρόπο με τον οποίο πρέπει να διαχειριστεί τα χαμένα TL0 πακέτα. Τέλος, ενδιαφέρον παρουσιάζει η περίπτωση όπου μεταδίδουμε και βελτιωτικά χωρικά επίπεδα, ιδιαίτερα σε εφαρμογές όπου το μέγεθος της εικόνας αλλάζει διαρκώς, προκειμένου να δημιουργηθεί ένας αλγόριθμος ο οποίος θα φροντίζει τόσο για τα TL0 πακέτα αλλά και για τα πακέτα του πιο σημαντικού χωρικού επιπέδου, εκείνα δηλαδή με SL0 ίσο με το 0.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Base Layer	Επίπεδο βάσης
Client	Πελάτης
Contributing Source	Συμβαλλόμενη πηγή
Data Stream	Ροή δεδομένων
Decode	Αποκωδικοποίηση
Demultiplexing	Αποπολυπλεξία
Dependency	Εξάρτηση
Effective frame	Ωφέλιμη εικόνα
Encode	Κωδικοποίηση
Enhancement Layer	Βελτιωτικό/ενισχυμένο επίπεδο
Fidelity	Πιστότητα
File descriptor	Περιγραφέας αρχείου
Frame	Εικόνα, Πλαίσιο
Handler	Χειριστής
Instance	Στιγμιότυπο
Internet	Διαδίκτυο
Intra-layer	Ενδο-επίπεδο
Layer	Επίπεδο
Media	Μέσο(-α)
Media Stream	Ροή μέσου
Media Type	Τύπος μέσου
Motion Compensation	Αντιστάθμιση της κίνησης
Multicast	Πολυεκπομπή
Multi-Conference	Συνδιάσκεψη πολλαπλών χρηστών
Multiplexing	Πολυπλεξία
Multi-threading	Πολυνηματικός
Open Source	Ανοιχτού κώδικα
Payload	Ωφέλιμο φορτίο
Payload Formats	Μορφές ωφέλιμου φορτίου
Payload Type	Τύπος ωφέλιμου φορτίου
Picture parameter set	Σύνολο παραμέτρων εικόνας
Port	Θύρα
Real-time	Πραγματικού χρόνου
Retransmission	Αναμετάδοση
Robust	Εύρωστος
Scalable	Κλιμακωτός
Sequence Number	Αριθμός ακολουθίας
Sequence parameter set	Σύνολο παραμέτρων ακολουθίας
Server	Εξυπηρετητής
Session	Σύνοδος
Stream	Ροή/ρεύμα
Synchronization Source	Πηγή συγχρονισμού
Temporal Layer	Χρονικό επίπεδο
Thread	Νήμα
Timestamp	Χρονοσφραγίδα
User-agent	Πράκτορας χρήστη

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ACK	Acknowledgement
AU	Access Unit
AVC	Advanced Video Coding
BLP	Bitmask of following Lost Packets
CNAME	Canonical Name
DID	Dependency ID
DON	Decoding Order Number
FCI	Feedback Control Information
FMT	Feedback Message Type
FSEQ	First Sequence number
FSN	First Sequence Number
GOP	Group of Pictures
HDTV	High Definition Television
IDR	Instantaneous Decoding Refresh
IDRPICID	Instantaneous Decoding Refresh Picture ID
IP	Internet Protocol
LSEQ	Last Sequence Number
MANE	Media Aware Network Element
MCU	Multipoint Control Unit
MPEG	Moving Picture Experts Group
MSG	Medium-grain Scalability
MST	Multi-Session Transmission
NACK	Negative Acknowledgement
NAL	Network Abstract Layer
NALUC	Network Abstract Layer Unit Counter
NAT	Network Address Translation
NMP	Networked Music Performance
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PACSI	Payload Content Scalability Information
PCMU	Pulse Code Modulation mu-law

PID	Packet ID
PT	Payload Type
QID	Quality ID
RFC	Request for Comments
RTCP	Real-time Transport Control Protocol
RTCP APP	RTCP Application
RTCP FB	RTCP Feedback
RTCP RR	RTCP Receiver Report
RTCP SDES	RTCP Source Description
RTCP SR	RTCP Sender Report
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SEI	Supplemental Enhancement Information
SFU	Selective Forward Unit
SIP	Session Initiation Protocol
SL0	Spatial Layer Zero (0)
SNR	Signal to Noise Ration
SSRC	Synchronization Source
STAP	Single Time Aggregation Packet
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
TID	Temporal ID
TL0	Temporal Layer Zero (0)
tl0_dep_rep_idx	Temporal Level Zero Dependency Representation Index
TL0D	Temporal Layer Zero Descriptor
TL0PICIDX	Temporal Layer Zero Picture Index
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VC	Video Conference
VCL	Video Coding Layer
WebRTC	Web Real Time Communication

ΑΝΑΦΟΡΕΣ

[1] *MusiNet: Comprehensive design and implementation of a networked music performance system*, National Strategic Reference Framework (NSRF) – Research Funding Program: THALIS – University of Crete – MUSINET.

[2] J. Lazzaro, J. Wawrzynek, "A case for network musical performance". *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*. ACM Press New York, NY, USA. pp. 157–166, 2001.

[3] J. Postel, "User Datagram Protocol", RFC768, 1980.

[4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC3550, 2003.

[5] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p54.

[6] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p55.

[7] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p61

[8] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p69

[9] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p64

[10] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p58

[11] Colin Perkins, *RTP: Audio and Video for the Internet*, Addison Wesley, June 2003, p76

[12] J. Ott, S. Wenger, N. Sato, C. Burmeister, J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC4585, 2006

[13] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC3261, 2002.

[14] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", RFC4566, 2006.

[15] Creytiv software. [Online]. Available: http://creytiv.com

[16] M. Westerlund, B. Burman, C. Perkins, H. Alvestrand, "Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams", Internet-Draft, 2014.

[17] M. Westerlund, S. Wenger, "RTP Topologies", RFC5117, 2008.

[18] M. Westerlund, S. Wenger, "RTP Topologies", Internet-Draft, 2015.

[19] W. Jiang, H. Schulzrinne, "*Modeling of Packet Loss and Delay and Their Effect on Real-Time Multimedia Service Quality*", 2000.

[20] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 22: Feb. 2014

[21] Y. –K. Wang, R. Even, T. Kristensen, R. Jesup, "RTP Payload Format for H.264 Video", RFC6184, 2011

[22] *Advanced Video Coding for Generic Audiovisual Services – Annex G: Scalable Video Coding*, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 22: Feb. 2014

[23] S. Wenger, Y. –K. Wang, T. Schierl, A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC6190, 2011.

[24] Y. - K. Wang, M. M. Hannuksela, S. Pateux, A. Eleftheriadis, S. Wenger, "System and Transport Interface of SVC", *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9,* 2007.

[25] A. Eleftheriadis, "The Future of Telepresence", *IMTC,* 2010.