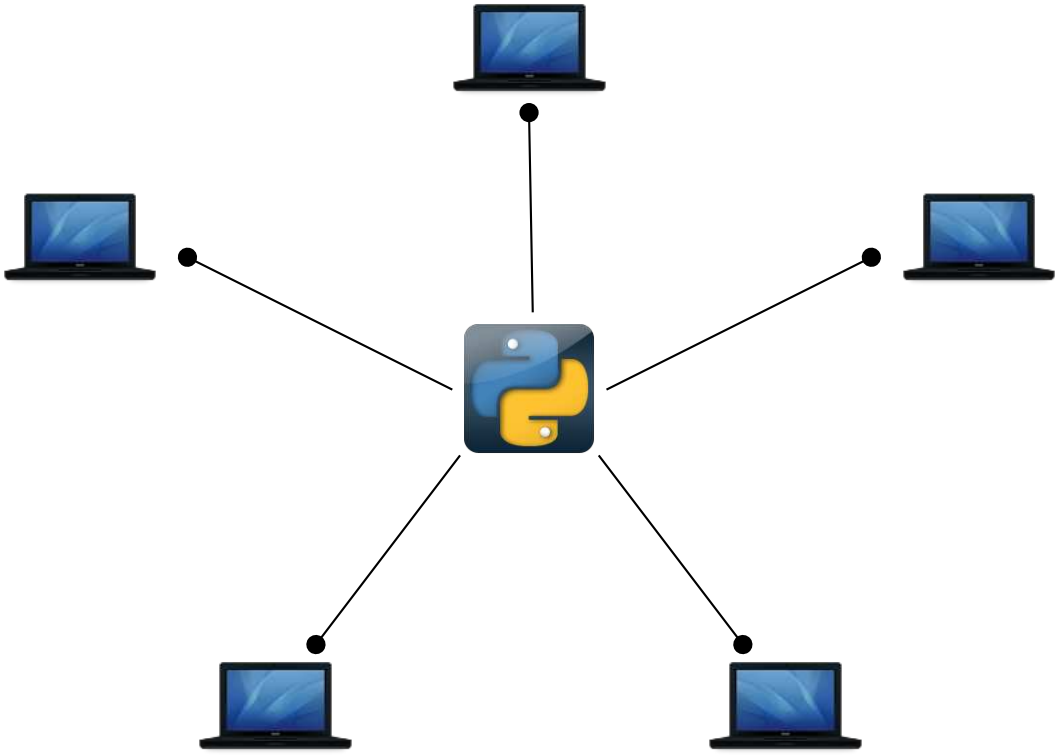
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία Client-Server Εφαρμογής για on-demand μεταφορά αρχείων με χρήση Multicast



ΙΕΡΩΝΥΜΟΣ ΑΛΕΞΑΝΔΡΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΧΑΡΧΑΛΑΚΗΣ ΣΤΕΦΑΝΟΣ

ΝΟΕΜΒΡΙΟΣ 2010

Δημιουργία Client-Server εφαρμογής για on-demand μεταφορά αρχείων με χρήση Multicast

Ιερώνυμος Αλέξανδρος

Νοέμβριος 2010

Επιβλέπων καθηγητής : Χαρχαλάκης Στέφανος

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας πτυχιακής εργασίας είναι η δημιουργία client – server εφαρμογής για μεταφορά αρχείων με χρήση multicast προς πολλούς παραλήπτες ταυτόχρονα. Για να γίνει δυνατή η αξιόπιστη αποστολή αρχείων με χρήση multicast στα πλαίσια της εφαρμογής δημιουργήθηκε ένα νέο πρωτόκολλο που υλοποιήθηκε στο επίπεδο εφαρμογών του μοντέλου αναφοράς TCP/IP. Το πρωτόκολλο υποστηρίζει αξιόπιστη μεταφορά αρχείων, έλεγχο συμφόρησης, συμπληρωματική λειτουργία και βασίζεται πάνω στο IP multicast για να μειώσει την περιττή κίνηση κατά την αποστολή.

Στην αρχή γίνεται μια εισαγωγή στο IP multicast και στους στόχους της πτυχιακής. Στην συνέχεια αναλύεται η βασική λειτουργία της εφαρμογής, οι αλγόριθμοι που υλοποιούνται, το πρωτόκολλο που δημιουργήθηκε και τα βασικά σημεία του κώδικα. Στο τέλος γίνονται κάποια σχόλια σχετικά με την υλοποίηση, παρουσιάζονται ορισμένα συμπεράσματα και το εγχειρίδιο της εφαρμογής.

Περιεχόμενα

1 ΕΙΣΑΓΩΓΗ

Σε τοπικά δίκτυα υπολογιστών (LANs) πολλές φορές υπάρχει η ανάγκη διαμοιρασμού μεγάλου όγκου πληροφορίας μεταξύ των υπολογιστών. Για να επιτευχθεί αυτό υπάρχει πληθώρα διαθέσιμων προγραμμάτων που ειδικεύονται στην μεταφορά αρχείων. Συνήθως τα δεδομένα που θέλουμε να μεταφέρουμε βρίσκονται σε έναν εξυπηρετητή (server) και πρέπει να μεταφερθούν σε πολλούς υπολογιστές (clients) ταυτόχρονα. Τα περισσότερα διαθέσιμα προγράμματα μεταφοράς αρχείων δεν κάνουν καλή διαχείριση των πόρων του δικτύου όταν χρειάζεται να γίνει παράλληλη μετάδοση σε πολλούς clients και αυτό κυρίως οφείλεται στα πρωτόκολλα στα οποία στηρίζονται. Στις παρακάτω ενότητες γίνεται αναφορά στα πρωτόκολλα που χρησιμοποιούν τέτοιου είδους εφαρμογές και αναλύονται τα μειονεκτήματα τους. Στην συνέχεια παρουσιάζονται οι εναλλακτικές επιλογές που υπάρχουν και πως μπορούν να χρησιμοποιηθούν για αξιόπιστη αποστολή δεδομένων.

1.1 Μεταφορά αρχείων με FTP

Τα περισσότερα προγράμματα μεταφοράς αρχείων στο επίπεδο εφαρμογών στηρίζονται στο πρωτόκολλο FTP[7] το οποίο στο επίπεδο μεταφοράς χρησιμοποιεί το πρωτόκολλο TCP[6]. Το TCP διαθέτει χαρακτηριστικά όπως αξιοπιστία, έλεγχο ροής και συμφόρησης που το κάνουν το ιδανικό πρωτόκολλο για χρήση σε εφαρμογές μεταφοράς αρχείων. Το TCP είναι ένα συνδεσμοστρεφές (connection-oriented) πρωτόκολλο, κάτι που σημαίνει ότι για να επικοινωνήσουν δυο σταθμοί θα πρέπει να δημιουργηθεί μια σύνδεση, ένα εικονικό κανάλι μεταξύ τους. Η δημιουργία σύνδεσης μεταξύ των δύο σταθμών σημαίνει ότι τα μηνύματα που ανταλλάσσονται είναι unicast, δηλαδή κάθε μήνυμα που στέλνεται έχει έναν μοναδικό παραλήπτη. Η παράλληλη μεταφορά δεδομένων από έναν server προς πολλούς clients με την χρήση του TCP, προϋποθέτει την δημιουργία ξεχωριστού καναλιού επικοινωνίας για κάθε client.

Το πρώτο πρόβλημα που παρουσιάζεται είναι ο ανταγωνισμός που δημιουργείται μεταξύ των παράλληλων συνδέσεων για το διαθέσιμο bandwidth του δικτύου. Αν διαθέτουμε μια γραμμή χωρητικότητας R και στέλνουμε δεδομένα παράλληλα σε N clients, στην ιδανικότερη περίπτωση κάθε client θα λαμβάνει με ταχύτητα $\approx R/\mathrm{N}$, ενώ αν λάβουμε υπόψιν την επιπλέον κίνηση που μπορεί να υπάρχει στο δίκτυο τότε υπάρχει μεγάλη πιθανότητα να γίνει άνιση κατανομή του bandwidth με αποτέλεσμα κάποιοι clients να επωφελούνται και κάποιοι άλλοι να αδικούνται όσον αφορά την ταχύτητα λήψης[4]. Ο δεύτερος περιορισμός αφορά την δημιουργία περιττής κίνησης στο δίκτυο. Πιο συγκεκριμένα επειδή τα μηνύματα είναι unicast, αν χρειάζεται να σταλεί ένα αρχείο μεγέθους S σε N clients, ένα αντίγραφο του αρχείου θα σταλεί σε κάθε έναν ξεχωριστά. Με αυτόν τον τρόπο ενώ το αρχείο έχει μέγεθος S, η κίνηση που δημιουργείται στο δίκτυο είναι μεγέθους $S*N$.

1.2 Μεταφορά αρχείων με Multicast

1.2.1 IP multicast

Η πολυδιανομή (multicast) ως έννοια στα δίκτυα υπολογιστών αναφέρεται στην ταυτόχρονη παράδοση ενός μηνύματος σε μια ομάδα υπολογιστών με μία μόνο μετάδοση από τον αποστολέα. IP multicast είναι η υλοποίηση του multicast στο επίπεδο διαδικτύου[1] με χρήση του πρωτοκόλλου IP. Στο multicast υπάρχει η έννοια της ομάδας (multicast group), όπου κάθε ομάδα είναι μια IP διεύθυνση από το εύρος 224.0.0.0-239.255.255.255. Ο αποστολέας δεν είναι υποχρεωμένος να γνωρίζει εξαρχής σε πόσους παραλήπτες απευθύνεται , απλά στέλνει τα δεδομένα στην IP διεύθυνση που αντιστοιχεί στο κατάλληλο multicast group. Από την μεριά τους οι ενδιαφερόμενοι clients πρέπει να δηλώσουν ότι θέλουν να λαμβάνουν μηνύματα για το συγκεκριμένο multicast group. Τα πλεονεκτήματα από την χρήση του IP multicast είναι :

- Μικρή επιβάρυνση του αποστολέα
- Αποδοτικότερη χρήση του διαθέσιμου bandwidth
- Μείωση της κίνησης στο δίκτυο

1.2.2 IP multicast και μεταφορά αρχείων

Το IP multicast βρίσκει συνήθως χρήση σε εφαρμογές πραγματικού χρόνου όπως μετάδοση φωνής και βίντεο όπου η αποστολή γίνεται με έναν σταθερό ρυθμό που ορίζει η εφαρμογή και τυχόν απώλειες κατά την μεταφορά είναι επιτρεπτές έως ένα σημείο. Ο λόγος που μπορεί να υπάρξουν απώλειες κατά την αποστολή multicast μηνυμάτων είναι ότι οι περισσότερες IP multicast εφαρμογές στο επίπεδο μεταφοράς χρησιμοποιούν το πρωτόκολλο UDP[5] καθώς το TCP είναι προσανατολισμένο στην αποστολή unicast μηνυμάτων και δεν είναι εφικτό να χρησιμοποιηθεί σε multicast εφαρμογές. Το UDP, σε αντίθεση με το TCP, δεν προσφέρει αξιοπιστία, έλεγχο ροής και συμφόρησης, προστασία από διπλότυπα πακέτα και παράδοση σε σειρά, για τον λόγο αυτόν μπορεί να υπάρξουν απώλειες κατά την μεταφορά. Στις εφαρμογές μεταφοράς αρχείων η απώλεια δεδομένων κατά την μεταφορά δεν είναι επιτρεπτή σε αντίθεση με τις εφαρμογές πραγματικού χρόνου που συνήθως χρησιμοποιούν το IP multicast. Αυτό σημαίνει ότι πολλά από τα χαρακτηριστικά που προσφέρει το TCP,στο UDP πρέπει να υλοποιηθούν προγραμματιστικά στο επίπεδο εφαρμογής.

[1] Μοντέλο αναφοράς TCP/IP

1.2.3 UDPcast

Το UDPcast είναι μια cross-platform εφαρμογή μεταφοράς αρχείων γραμμένη από τον Alain Knaff[2]. Δημιουργήθηκε για αποστολή αρχείων στο τοπικό δίκτυο με τη χρήση multicast ώστε να γίνεται γρήγορα η αποστολή ενός αρχείο σε πολλούς υπολογιστές ταυτόχρονα. Εξ' ορισμού η αποστολή δεδομένων γίνεται με multicast όμως υπάρχει και η επιλογή χρήσης broadcast για δίκτυα με υπολογιστές που δεν υποστηρίζουν multicast. Επίσης σε περίπτωση αποστολής σε έναν παραλήπτη η μετάδοση μετατρέπεται σε unicast. Η εφαρμογή χρησιμοποιεί δυο multicast κανάλια όπου στο ένα μεταφέρονται μηνύματα ελέγχου και στο άλλο τα δεδομένα. Υποστηρίζει λειτουργία χωρίς τη χρήση καναλιού επικοινωνίας όπου ο αποστολέας δεν ενημερώνεται κατά την αποστολή από τους παραλήπτες. Σε αυτήν την περίπτωση ο ρυθμός αποστολής ρυθμίζεται παραμετρικά και η αποστολή γίνεται με χρήση Forward Error Correction (FEC)[9]. Υπάρχουν παράμετροι που αυτοματοποιούν την έναρξη της αποστολής ώστε να ξεκινάει μόνο αν πληρούνται κάποια κριτήρια όπως ο αριθμός των συνδεδεμένων clients. Ο αποστολέας στέλνει το αρχείο ανά τμήματα και μετά την αποστολή κάθε τμήματος ζητά επιβεβαίωση ότι το τμήμα λήφθηκε από όλους ώστε σε διαφορετική περίπτωση να γίνει αναμετάδοση. Η εφαρμογή δίνει τη δυνατότητα στους παραλήπτες να συνδέονται και να ξεκινάνε τη λήψη ενώ η αποστολή είναι σε εξέλιξη αλλά να χάνουν τα δεδομένα που στάλθηκαν πριν τη σύνδεση τους.

1.3 Σκοπός πτυχιακής

Η εφαρμογή που σχεδιάστηκε στα πλαίσια της παρούσας πτυχιακής υλοποιεί ένα νέο πρωτόκολλο το οποίο δημιουργήθηκε για να επιτευχθεί αξιόπιστη και παράλληλη μεταφορά δεδομένων προς πολλούς παραλήπτες. Πιο συγκεκριμένα το πρωτόκολλο προσφέρει :

Αξιόπιστη μεταφορά αρχείων. Διασφαλίζεται ότι όλοι οι παραλήπτες θα λάβουν ολόκληρο το αρχείο.

Έλεγχο συμφόρησης. Ρύθμιση του ρυθμού αποστολής ώστε να επιτυγχάνεται βέλτιστο throughput.

Συμπληρωματική λειτουργία. Παραλήπτες μπορούν να συνδέονται και να λαμβάνουν δεδομένα ενώ η αποστολή είναι ήδη σε εξέλιξη.

Μείωση κίνησης στο δίκτυο με τη χρήση IP Multicast.

[2]http://udpcast.linux.lu/

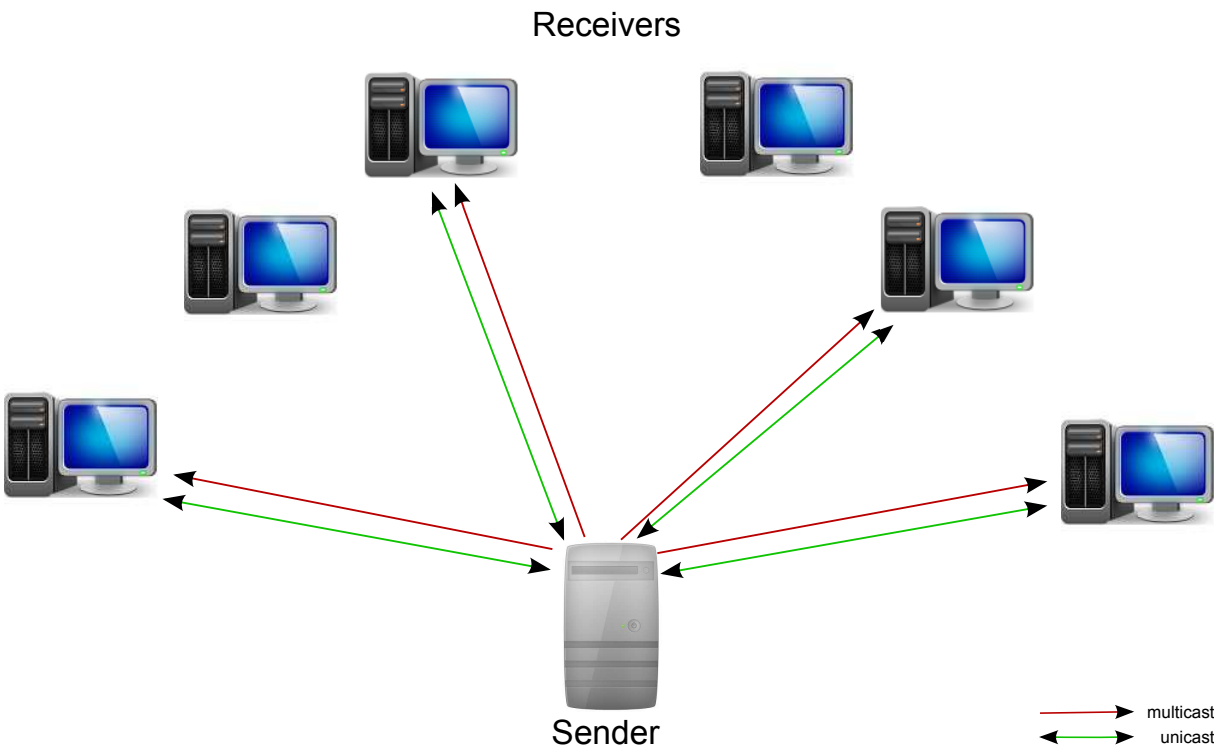


Figure 2.1: Μοντέλο αποστολής

2 ΛΕΙΤΟΥΡΓΙΑ

2.1 Σχεδίαση

Το κυρίως πρόγραμμα χωρίζεται σε δύο μέρη. Ο αποστολέας (server) εκτελεί το πρόγραμμα **sender** που είναι υπεύθυνο για την αποστολή δεδομένων ενώ στους παραλήπτες (clients) εκτελείται το πρόγραμμα **receiver** που αναλαμβάνει την λήψη των δεδομένων. Τα δύο αυτά μέρη υλοποιούν το πρωτόκολλο που σχεδιάστηκε στα πλαίσια της παρούσας πτυχιακής για να επιτευχθεί παράλληλη και αξιόπιστη μεταφορά δεδομένων. Το πρωτόκολλο υλοποιείται στο επίπεδο εφαρμογής και για επικοινωνία με το δίκτυο κάνει χρήση του μηχανισμού υποδοχών (sockets). Για την αμφίδρομη ανταλλαγή unicast μηνυμάτων μεταξύ sender και receivers χρησιμοποιούνται TCP sockets. Ο sender διατηρεί μια ανοιχτή TCP σύνδεση για κάθε receiver που λαμβάνει δεδομένα. Η χρήση του TCP διασφαλίζει την σωστή παράδοση των μηνυμάτων και μειώνει την πολυπλοκότητα του πρωτοκόλλου καθώς δεν χρειάζεται να γίνονται επιπλέον έλεγχοι ότι η παράδοση έγινε σωστά. Η αποστολή δεδομένων γίνεται με multicast. Για την αποστολή και παραλαβή multicast δεδομένων χρησιμοποιούνται UDP sockets. Η multicast κίνηση είναι μονόδρομη γιατί το multicast κανάλι χρησιμοποιείται μόνο για αποστολή δεδομένων από τον sender. Η unicast και η multicast κίνηση φαίνονται στην εικόνα 2.1

2.2 UDP sockets και multicast

Η αποστολή και λήψη multicast μηνυμάτων γίνεται δυνατή με τη βοήθεια UDP sockets. Για την αποστολή UDP datagrams χρειάζεται να δημιουργηθεί ένα UDP socket και να οριστεί η διεύθυνση και η θύρα (port number) στην οποία θα σταλούν τα δεδομένα ώστε η παράδοση στο άλλο άκρο να γίνει στη σωστή εφαρμογή. Το UDP είναι ασυνδεσμικό και δεν χρειάζεται να γίνει πρώτα σύνδεση με το άλλο άκρο για να σταλούν δεδομένα όπως στο TCP. Αυτό βέβαια σημαίνει ότι η εφαρμογή στην άλλη πλευρά δεν ξέρει πότε και τι δεδομένα θα σταλούν με αποτέλεσμα αν δεν περιμένει να λάβει κάτι , τα δεδομένα αυτά να χαθούν. Για να είναι multicast το μήνυμα που θα σταλεί, πρέπει να οριστεί ως διεύθυνση αποστολής μια multicast IP διεύθυνση από το εύρος 224.0.0.0 – 239.255.255.255. Το UDP socket δεν χρειάζεται να είναι μέλος του multicast group για να στείλει δεδομένα σε αυτό. Για τον έλεγχο της multicast αποστολής υπάρχουν οι παράμετροι (socket options):

- TTL
- Loopback
- Interface

Το TTL στο multicast, εκτός από την γνωστή σημασία που έχει στα IP πακέτα, σηματοδοτεί την εμβέλεια που θα έχει η multicast εκπομπή. Η loopback λειτουργία είναι ενεργοποιημένη από προεπιλογή και σημαίνει ότι αν ο αποστολέας είναι ο ίδιος μέλος του multicast group στο οποίο στέλνει , ένα αντίγραφο των δεδομένων θα γυρίσει πίσω σε αυτόν. Η επιλογή του interface υπάρχει για την περίπτωση που ο υπολογιστής είναι συνδεδεμένος σε περισσότερα από ένα δίκτυα και το interface στο οποίο θα σταλούν τα multicast δεδομένα πρέπει να οριστεί ρητά.

Συνήθως το λειτουργικό σύστημα είναι ρυθμισμένο να απορρίπτει multicast πακέτα που έρχονται από το δίκτυο. Για να λάβει ένα UDP socket multicast δεδομένα πρέπει να γίνει μέλος (join) ενός multicast group. Με αυτό τον τρόπο λέει στο λειτουργικό σύστημα να δέχεται και να προωθεί στην εφαρμογή τα multicast πακέτα που απευθύνονται στο συγκεκριμένο multicast group. Για να μπορέσει το UDP να κάνει αποπολύπλεξη των δεδομένων, πριν το join στο multicast group, το socket πρέπει να συσχετισθεί (bind) με ένα port number και ένα interface δικτύου.

2.3 Γλώσσα Προγραμματισμού

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής είναι η python[3] στην έκδοση 3. Η python είναι μια scripting γλώσσα προγραμματισμού υψηλού επιπέδου που δίνει έμφαση στη παραγωγικότητα και το καθαρό συντακτικό. Είναι μια γλώσσα γενικού σκοπού με μια πλούσια βιβλιοθήκη που την κάνει κατάλληλη

[3]http://www.python.org/

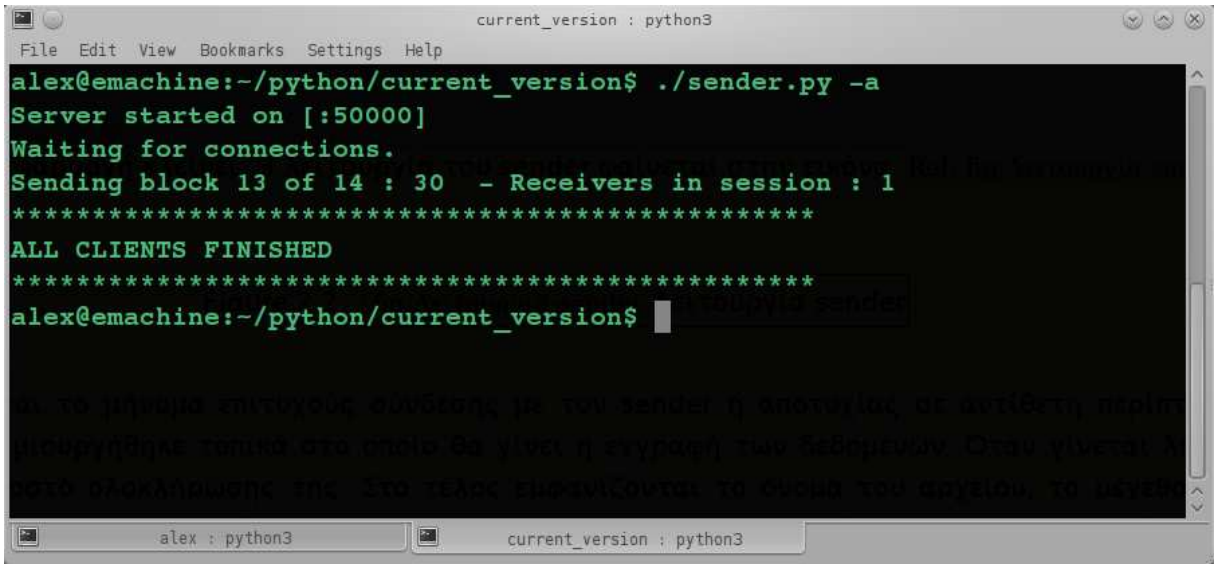


Figure 2.2: Λειτουργία sender

για ανάπτυξη πολλών ειδών εφαρμογών όπως web, databases, networking, GUI και άλλων. Έχει μικρή καμπύλη εκμάθησης και προσφέρει πολλές ευκολίες στον προγραμματιστή που αυξάνουν την παραγωγικότητα και κάνουν δυνατή την συγγραφή λειτουργικών προγραμμάτων σε μικρό χρονικό διάστημα. Ένας άλλος λόγος που προτιμήθηκε η python είναι τα πολλά έτοιμα εργαλεία που προσφέρει η βιβλιοθήκη της, τα οποία διευκόλυναν την λήψη πολλών σχεδιαστικών αποφάσεων κατά την ανάπτυξη της εφαρμογής. Ένας τομέας στον οποίον μπορεί να υστερεί σε σχέση με κάποιες άλλες γλώσσες προγραμματισμού είναι η ταχύτητα εκτέλεσης.

2.4 Λειτουργία εφαρμογής

Ο sender εκτελείται στο μηχάνημα που θα αναλάβει την αποστολή του αρχείου και ο receiver σε κάθε μηχάνημα που προορίζεται να λάβει το αρχείο. Τα δυο μέρη της εφαρμογής, sender και receiver, εκτελούνται από την γραμμή εντολών (command line) και η λειτουργία τους ρυθμίζεται από παραμέτρους που δίνονται κατά την έναρξη της εφαρμογής. Στην εξ' ορισμού λειτουργία ο sender ξεκινάει και περιμένει για συνδέσεις. Η αποστολή ξεκινάει με την σύνδεση των πρώτων receivers με τους υπόλοιπους να μπορούν να συνδέονται στην πορεία, ενώ σταματάει αυτόματα όταν λάβουν όλοι το αρχείο ή αποσυνδεθούν για άλλον λόγο. Κατά την λειτουργία της η εφαρμογή εκτυπώνει κάποια μηνύματα που δείχνουν σε πιο στάδιο βρίσκεται η αποστολή. Ο sender στην αρχή εκτυπώνει σε ποια διεύθυνση ακούει για συνδέσεις και κατά τη διάρκεια της αποστολής ποιο τμήμα του αρχείου στέλνει. Όταν η αποστολή ολοκληρωθεί εμφανίζεται το αντίστοιχο μήνυμα και η εφαρμογή κλείνει (εικόνα 2.2).

Στους receivers στην αρχή εμφανίζεται το μήνυμα επιτυχούς σύνδεσης με τον sender ή αποτυχίας σε αντίθετη περίπτωση. Στη συνέχεια εμφανίζεται μήνυμα σχετικά με το νέο

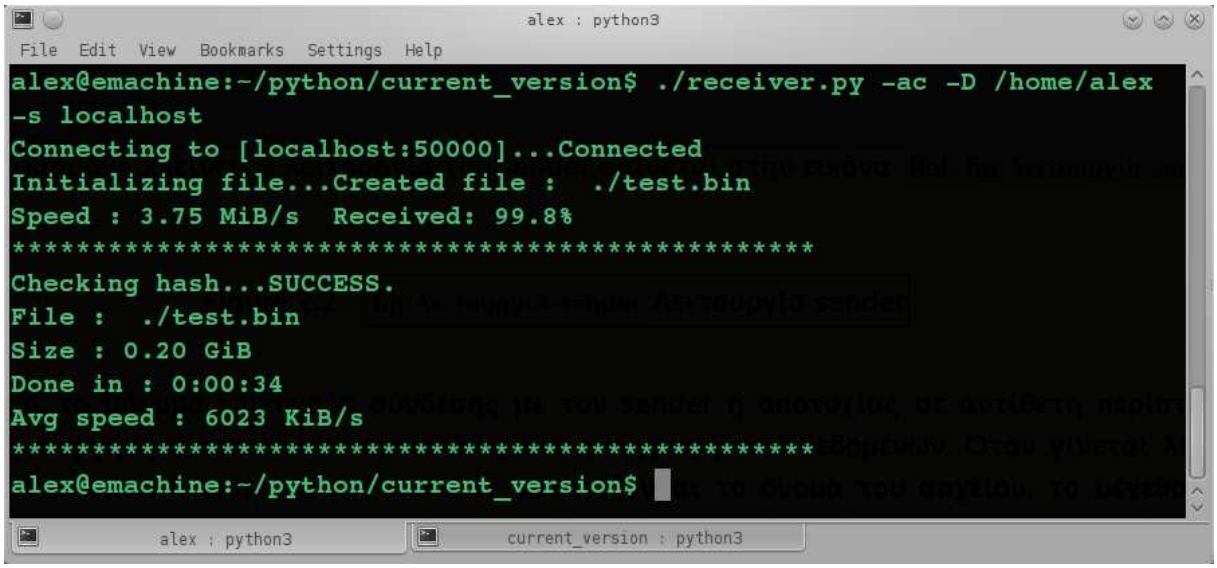


Figure 2.3: Λειτουργία receiver

αρχείο που δημιουργήθηκε τοπικά στο οποίο θα γίνει η εγγραφή των δεδομένων. Όταν γίνεται λήψη εμφανίζονται και αλλάζουν δυναμικά η ταχύτητα λήψης και το ποσοστό ολοκλήρωσης της. Στο τέλος εμφανίζονται το όνομα του αρχείου, το μέγεθος του, ο συνολικός χρόνος που διήρκεσε η λήψη του, ο μέσος ρυθμός λήψης και το αν λήφθηκε σωστά το αρχείο (εικόνα 2.3).

Εκτός από την αυτοματοποιημένη λειτουργία, ο sender και ο receiver υποστηρίζουν ένα πιο διαδραστικό τρόπο λειτουργίας με την χρήση **command line interface** (cli). Η cli λειτουργία προσφέρει στον χρήση έναν συγκεκριμένο αριθμό εντολών που δίνουν τη δυνατότητα η ρύθμιση κάποιων παραμέτρων να γίνεται πιο δυναμικά και λειτουργίες όπως σύνδεση του receiver στον sender και έναρξη της αποστολής να γίνονται χειροκίνητα. Η λειτουργία με cli περιγράφεται στην ενότητα 8.3.

2.5 Βασικός αλγόριθμος λειτουργίας

Πρώτα δημιουργείται μια νέα συνεδρία (session) στον sender όπου ορίζεται το αρχείο προς μεταφορά και το multicast group στο οποίο θα γίνει η αποστολή δεδομένων. Όταν ενεργοποιηθεί το session ο sender είναι έτοιμος να δεχτεί συνδέσεις. Όταν συνδέεται ένας receiver του αποστέλλεται ένα μήνυμα SESSION INFO[4] με πληροφορίες που αφορούν το ενεργό session. Το μήνυμα περιέχει λεπτομέρειες για το αρχείο και το multicast group στο οποίο θα γίνει η μετάδοση. Ο receiver εξετάζει το μήνυμα και στέλνει μήνυμα JOIN SESSION στον sender για να γίνει μέλος του session. Ο sender προσθέτει τον receiver στην λίστα με τους παραλήπτες και του στέλνει μήνυμα επιβεβαίωσης JOIN ACK. Ο receiver όταν λάβει το μήνυμα επιβεβαίωσης, αρχικοποιεί τις παραμέτρους για την λήψη του αρχείου και στέλνει το μήνυμα READY, το οποίο δηλώνει ότι είναι έτοιμος να λάβει δεδομένα.

[4]Η ανάλυση των μηνυμάτων του πρωτοκόλλου γίνεται στο κεφάλαιο 4

Όταν λάβει το πρώτο μήνυμα READY, ο sender ξεκινάει να στέλνει δεδομένα στη multicast διεύθυνση που ορίστηκε κατά την δημιουργία του session. Οι receivers που συνδέονται ενώ η αποστολή είναι σε εξέλιξη επαναλαμβάνουν την παραπάνω διαδικασία αρχικοποίησης και ξεκινάνε να λαμβάνουν το αρχείο από το σημείο στο οποίο βρίσκεται η αποστολή την ώρα που συνδέθηκαν.

Κατά την διάρκεια της αποστολής πάντα ένας από τους receivers ορίζεται ως πρωτεύων (prime). Ο prime receiver στέλνει πίσω στον sender ένα μήνυμα επιβεβαίωσης (acknowledgement, ACK) για κάθε κομμάτι του αρχείου που λαμβάνει. Έτσι βοηθάει τον sender να ρυθμίζει τον ρυθμό αποστολής και να ξέρει ποια κομμάτια του αρχείου έχει λάβει επιτυχώς ο prime receiver. Ο sender στέλνει το αρχείο με περάσματα (passes) όπου σε κάθε πέρασμα στέλνει τα κομμάτια του αρχείου που ο prime receiver δεν έχει λάβει κατά το προηγούμενο πέρασμα. Ενώ ο sender είναι επικεντρωμένος στον prime receiver οι υπόλοιποι receivers συνεχίζουν να λαμβάνουν παράλληλα και αν τελειώσουν πρώτοι ενημερώνουν τον sender ή περιμένουν να ερωτηθούν για τα κομμάτια του αρχείου που τους λείπουν ώστε να ολοκληρωθεί η λήψη. Όταν ένας receiver τελειώσει, ειδοποιεί τον sender με ένα μήνυμα DONE για να βγει από την λίστα με τους παραλήπτες. Όταν τελειώσουν όλοι και δεν υπάρχουν άλλοι ενεργοί receivers στο session, ο sender σταματάει την αποστολή και ολοκληρώνει το session.

2.6 Συνεδρία – Session

Το session δημιουργείται στον sender. Σκοπός του είναι να περιέχει τις απαραίτητες πληροφορίες που χρειάζονται για να γίνει σωστά η μετάδοση του αρχείου. Οι πληροφορίες που κρατούνται στο session είναι :

Session group Μια λίστα που περιέχει τους receivers που δήλωσαν ενδιαφέρον να λάβουν το αρχείο.

Αρχείο προς αποστολή Περιέχει τα στοιχεία του αρχείου που θα αποσταλεί.

Multicast group Είναι η multicast IP στην οποία θα σταλούν τα δεδομένα.

Κατάσταση του session Αν το session είναι ενεργό ή όχι.

Στόχος του session είναι όλοι οι ενδιαφερόμενοι receivers να μάθουν γρήγορα και με εύκολο τρόπο τις πληροφορίες που χρειάζονται για την λήψη του αρχείου. Η γενική ιδέα είναι ότι υπάρχει ένα περιβάλλον με πολλούς υπολογιστές όπου όλοι ή μια μεγάλη ομάδα αυτών, χρειάζεται να λάβει το ίδιο αρχείο. Σε αυτήν την περίπτωση δεν είναι αναγκαίο κάθε παραλήπτης να συνδέεται και να δηλώνει ρητά ποιο αρχείο χρειάζεται. Εφόσον το αρχείο προς αποστολή είναι γνωστό, είναι αποδοτικότερο να δημιουργηθεί μια συνεδρία στον sender και οι receivers να πάρουν αυτόματα τα δεδομένα που χρειάζονται από αυτόν. Για να ενεργοποιηθεί το session χρειάζεται να οριστεί το αρχείο και το multicast group

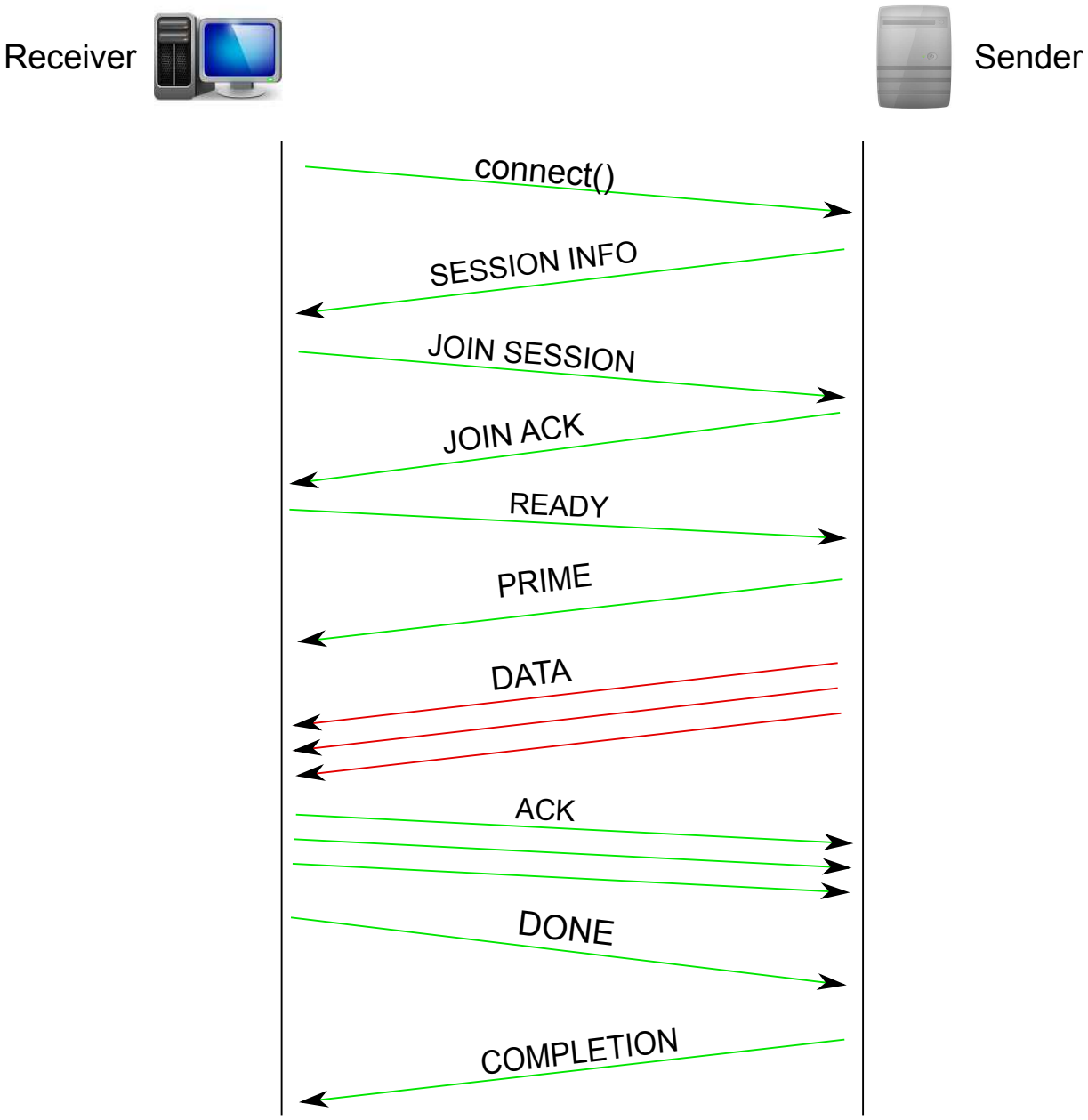


Figure 2.4: Ανταλλαγή μηνυμάτων μεταξύ sender - receiver

στο οποίο θα γίνει η μετάδοση. Αυτές είναι και οι πληροφορίες που χρειάζεται ένας receiver για να λάβει το αρχείο. Ο receiver πρώτα ελέγχει τα στοιχεία του αρχείου και αν είναι επιθυμητή η λήψη του στέλνει μήνυμα στον sender ότι θέλει να το λάβει και κάνει join στο multicast group. Στον sender υπάρχει μόνο ένα session ενεργό κάθε φορά και το session αυτό αφορά την μεταφορά μόνο ενός αρχείου. Για τον λόγο αυτόν τα μηνύματα του πρωτοκόλλου που έχουν σχέση με την διαχείριση του session δεν έχουν κάποιο αναγνωριστικό. Για παράδειγμά όταν ο sender δέχεται αίτημα συμμετοχής ενός receiver ξέρει ότι αυτό θα αφορά το μοναδικό session που μπορεί να είναι ενεργό εκείνη την στιγμή.

2.6.1 Session group

Με την βοήθεια του session group ο sender γνωρίζει πόσοι και ποιοι λαμβάνουν το αρχείο. Ο sender χρειάζεται να γνωρίζει πόσοι είναι συνδεδεμένοι ώστε να ορίζει πότε θα ξεκινά και πότε θα σταματά η αποστολή. Ο sender στέλνει δεδομένα μόνο όταν υπάρχει τουλάχιστον ένα μέλος στο session και η αποστολή σταματάει όταν το session group αδειάσει. Ξέροντας ποιοι λαμβάνουν το αρχείο βοηθά τον sender να σιγουρευτεί ότι όλοι θα λάβουν σωστά το αρχείο. Έχοντας τη λίστα με τους παραλήπτες ο sender μπορεί να ρωτά τον κάθε ένα ξεχωριστά και να μαθαίνει ποια κομμάτια τους λείπουν ώστε να εγγυηθεί την παράδοση τους. Μέλη του group είναι όσοι receivers θέλουν να λάβουν το αρχείο. Όσοι είναι συνδεδεμένοι σε μια δεδομένη χρονική στιγμή στον sender δεν είναι απαραίτητα μέλη του session. Ένας receiver μπορεί να συνδεθεί και να ανταλλάξει πληροφορίες για το session με τον sender χωρίς να είναι μέλος σε αυτό. Ο receiver γίνεται μέλος του session μόνο όταν ενημερώσει τον sender ότι ενδιαφέρεται να λάβει το αρχείο που διαφημίζει το session.

2.6.2 Multicast group

Είναι η multicast IP διεύθυνση στην οποία θα γίνει η αποστολή του αρχείου. Ο receiver μαθαίνει αυτήν την διεύθυνση όταν λάβει το μήνυμα SESSION INFO με τις πληροφορίες του session. Αν ενδιαφέρεται για το αρχείο και ενημερωθεί από τον sender ότι μπορεί να γίνει μέλος του session, κάνει join στο multicast group που θα γίνει η αποστολή.

2.6.3 Κατάσταση του session

Η κατάσταση του session ορίζει το αν ο sender διαφημίζει το session, επεξεργάζεται αιτήματα συμμετοχής σε αυτό και επίσης κάθε φορά που πρόκειται να στείλει δεδομένα συμβουλεύεται αν το session είναι ενεργό ή όχι.

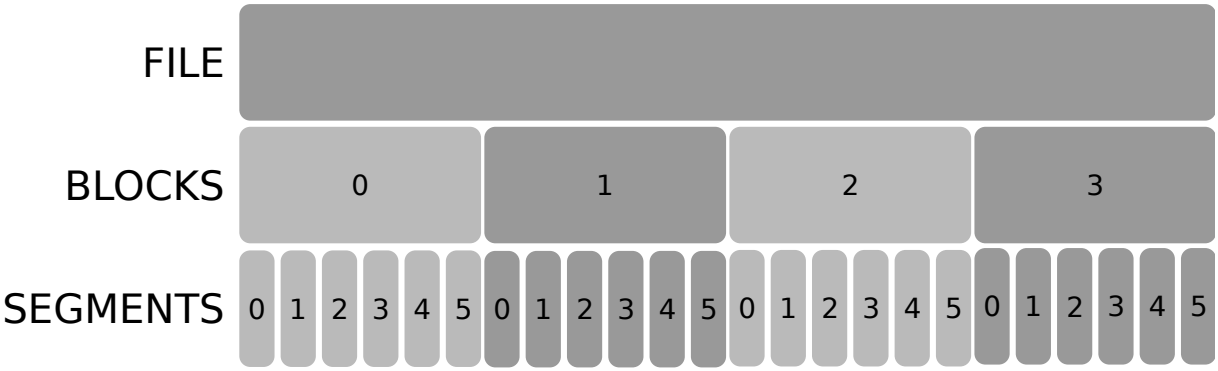


Figure 2.5: Χωρισμός αρχείου

2.6.4 Αρχείο προς αποστολή

Το αρχείο χωρίζεται σε τμήματα σταθερού μεγέθους που ονομάζονται segments. Μια ομάδα από segments σχηματίζει ένα block όπως φαίνεται στην εικόνα 2.5. Κάθε segment έχει μέγεθος 1456 bytes εκτός από το τελευταίο που μπορεί να έχει μικρότερο μέγεθος. Το μέγεθος του segment προκύπτει από την σχέση της εικόνας 2.6. Το DATA_HEADER είναι η κεφαλίδα (header) που μπαίνει στα μηνύματα δεδομένων και περιλαμβάνει το block και το segment number. Η αρίθμηση των blocks είναι αύξουσα και ξεκινάει από το 0 με το ίδιο να ισχύει για την αρίθμηση των segments μέσα σε κάθε block. Ο αριθμός των segments σε ένα block είναι σταθερός για όλα τα blocks εκτός από το τελευταίο που μπορεί να έχει λιγότερα segments. Η χωρητικότητα του block εξαρτάται από το πόσα segments μπορούν να περιγραφούν σε ένα μήνυμα αρνητικής επιβεβαίωσης (Negative Acknowledgement, NACK) που στέλνει ο receiver για να περιγράψει ποια segments δεν έλαβε από ένα συγκεκριμένο block. Η διάσπαση του αρχείου σε κομμάτια κάνει ευκολότερη την διαχείριση του. Οι πληροφορίες που περιλαμβάνει το session για το αρχείο είναι :

- Όνομα
- Μέγεθος
- Hash value
- Block capacity
- Segment size

Το μέγεθος σε συνδυασμό με το block capacity και το segment size χρειάζονται για να γίνει ο χωρισμός του αρχείου σε segments και blocks. Έχοντας αυτές τις τιμές sender

$$\underset{1500}{\text{ethernet MTU}} - \underset{20}{\text{IP header}} - \underset{8}{\text{UDP header}} - \underset{16}{\text{DATA HEADER}} = 1456 \text{ bytes}$$

Figure 2.6: Υπολογισμός segment size

και receivers μπορούν να αρχικοποιήσουν τις απαραίτητες δομές δεδομένων ώστε κατά την αποστολή να είναι γνωστό στους receivers πόσα έχουν να λάβουν και στον sender πόσα έχει να στείλει. Το μέγεθος του αρχείου επίσης βοηθάει τον receiver να ελέγξει αν υπάρχει διαθέσιμος χώρος στο μέσο όπου θα αποθηκευτεί το αρχείο. Το hash value είναι η έξοδος που παράγεται όταν το αρχείο περνά από την συνάρτηση κατακερματισμού MD5 και βοηθάει τον receiver να πιστοποιήσει ότι το έλαβε σωστά.

2.7 Μεταφορά αρχείου

Βασικός στόχος είναι το πρώτο έως και το τελευταίο κομμάτι του αρχείου να παραδοθούν σε όλους τους παραλήπτες που το έχουν ζητήσει. Η αποστολή ξεκινάει όταν κάνει join στο session ο πρώτος receiver. Κατά την αποστολή οι δύο βασικοί μηχανισμοί που μπαίνουν σε λειτουργία είναι η **αξιόπιστη παράδοση** και ο **έλεγχος συμφόρησης**. Η προϋπόθεση για να δουλέψουν αυτοί οι δυο μηχανισμοί είναι η ύπαρξη ανατροφοδότησης (feedback) από τους receivers. Κατά την διάρκεια της αποστολής ο sender δέχεται feedback με την μορφή :

- Θετικών επιβεβαιώσεων – Positive Acknowledgements (ACKs)
- Αρνητικών επιβεβαιώσεων – Negative Acknowledgements (NACKs)

Θετική επιβεβαίωση (ACK) είναι το μήνυμα που στέλνει ο receiver κάθε φορά που λαμβάνει επιτυχώς ένα segment. Αρνητική επιβεβαίωση (NACK) είναι το μήνυμα που στέλνει ο receiver κατόπιν ερώτησης του sender και περιγράφει ποια segments δεν έχει λάβει ο receiver για ένα συγκεκριμένο block. Το feedback από τα ACKs είναι συνεχές καθώς ACKs στέλνονται για κάθε segment που λαμβάνει ο receiver. Τα NACK σε αντίθεση δημιουργούν πολύ λιγότερη κίνηση καθώς σε ένα μήνυμα περιγράφεται μεγάλος αριθμός από segments και στέλνονται μόνο κατόπιν αίτησης από τον sender. Ο sender στέλνει αίτηση για NACK όταν γίνεται αλλαγή prime ώστε να μάθει ποια segments λείπουν από τον receiver που γίνεται ο νέος prime.

2.7.1 Congestion control

Στόχος του μηχανισμού ελέγχου συμφόρησης είναι να ρυθμίζει τον ρυθμό αποστολής ώστε ο αποστολέας να μην υπερβαίνει τον ρυθμό με τον οποίον μπορεί να λάβει ο παραλήπτης και να μην δημιουργεί συμφόρηση στο δίκτυο. Το TCP βασίζεται στον μηχανισμό κοινοποίησης του παραθύρου του παραλήπτη και στην ανίχνευση χαμένων πακέτων για να ρυθμίζει τον ρυθμό αποστολής. Το UDP από μόνο του δεν διαθέτει κανέναν μηχανισμό παρόμοιο με του TCP και τα ACKs στο επίπεδο εφαρμογής είναι ένας τρόπος για να αντισταθμιστεί αυτό. Χωρίς τον μηχανισμό κοινοποίησης παραθύρου ο μοναδικός τρόπος για να ενημερωθεί ο αποστολέας για την κατάσταση των παραληπτών και του δικτύου είναι μέσω της αναγνώρισης χαμένων πακέτων και timeouts. Το feedback από τα ACKs κάνει

δυνατή την εύρεση χαμένων πακέτων καθώς κάθε segment που στέλνεται έχει έναν αριθμό ακολουθίας και από τα ACKs που έρχονται ο αποστολέας μπορεί να εντοπίσει ποια segments έχουν χαθεί. Κάθε φορά που εντοπίζεται ένα χαμένο segment ή γίνεται timeout ο αποστολέας το εκλαμβάνει ως σύμπτωμα συμφόρησης στο δίκτυο και μειώνει τον ρυθμό με τον οποίον στέλνει.

2.7.2 Αξιόπιστη αποστολή

Κατά την παράλληλη μετάδοση σε πολλούς παραλήπτες για να θεωρηθεί η μεταφορά αξιόπιστη πρέπει όλοι να έχουν λάβει ολόκληρο το αρχείο. Εφόσον το UDP δεν είναι αξιόπιστο οι απαραίτητες ενέργειες για να λάβουν όλοι το αρχείο γίνονται στο επίπεδο εφαρμογής. Η βασική λειτουργία είναι ο sender να γνωρίζει τι έχει ληφθεί και τι όχι από τους receivers. Από το ξεκίνημα και κατά την διάρκεια της μετάδοσης ο sender γνωρίζει τι έχει στείλει και τι χρειάζεται να στείλει ξανά, ενώ οι receivers ξέρουν τι έχουν λάβει και ποια segments τους λείπουν για να ολοκληρωθεί η λήψη. Σε αυτό βοηθάει ο εντοπισμός χαμένων segments με την βοήθεια των ACKs και η δυνατότητα που έχουν οι receivers να πληροφορούν τον sender τι δεν έχουν λάβει με την χρήση NACK.

Η αποστολή του αρχείου γίνεται με περάσματα. Ο sender ξεκινάει και στέλνει διαδοχικά όλα τα segments του αρχείου και κατά την διάρκεια της αποστολής εντοπίζει και κρατά μια λίστα με χαμένα segments και μπορεί να πληροφορείται με NACK για την κατάσταση των receivers. Μετά την αποστολή του τελευταίου segment ελέγχει αν υπήρξαν απώλειες και ξεκινάει νέο πέρασμα στο οποίο στέλνει μόνο τα segments που είχαν χαθεί στο προηγούμενο πέρασμα. Ο τρόπος αποστολής με περάσματα και η δυνατότητα που έχουν οι receivers να ενημερώνουν τον sender για την κατάσταση τους με NACK δίνει την δυνατότητα στους receivers να συνδέονται ενώ η αποστολή είναι σε εξέλιξη, δηλαδή υποστηρίζεται το **late join**. Αυτοί οι receivers ενημερώνουν των sender ότι δεν έχουν λάβει τα segments που είχαν σταλεί πριν την σύνδεση τους ώστε ο sender να τα στείλει στο επόμενο πέρασμα.

2.7.3 PRIME Receiver

Στην παρούσα υλοποίηση ο sender αλληλεπιδρά με έναν μεγάλο αριθμό από receivers και εγγυάται την σωστή παράδοση του αρχείου σε όλους. Σε ένα τέτοιο σχήμα παίζει μεγάλο ρόλο στην σχεδίαση του πρωτοκόλλου ο τρόπος με τον οποίο θα ενημερώνεται ο sender από τους receivers και τι είδους feedback θα λαμβάνει ώστε να ικανοποιεί τις ανάγκες τους. Για παράδειγμά δεν είναι επιθυμητό να στέλνουν ACKs όλοι οι receivers γιατί αν μεγαλώσει πολύ ο αριθμός τους μπορεί να συμβεί feedback implosion, δηλαδή η κίνηση από τα ACKs θα επηρεάσει αρνητικά το δίκτυο και την αποστολή του αρχείου. Επιπλέον ο sender θα είναι υποχρεωμένος να διαθέσει πόρους για να επεξεργαστεί τα ACKs και να κρατάει πληροφορίες για την κατάσταση όλων των receivers στους οποίους στέλνει. Για την

αποφυγή του feedback implosion και ελάττωση του φόρτου στον sender, ACKs στέλνει μόνο ένας από τους receivers ο οποίος ονομάζεται prime. Prime ορίζεται ο πρώτος receiver που θα συνδεθεί στη αρχή του session. Αν ολοκληρώσει την λήψη ή αποσυνδεθεί για κάποιον άλλον λόγο πριν από τους άλλους, prime ορίζεται ο επόμενος receiver που συνδέθηκε μετά από αυτόν. Εφόσον ο prime είναι ο μόνος από τους receivers που στέλνει ACKs ο ρυθμός αποστολής και τα περάσματα που γίνονται για αποστολή χαμένων segments είναι με βάση τις δικές του ανάγκες. Η λύση του να επιλέγεται ένας receiver που στέλνει ACKs έχει καλύτερη απόδοση σε περιβάλλοντα όπου οι παραλήπτες έχουν παρόμοιες υπολογιστικές και δικτυακές δυνατότητες. Σε διαφορετική περίπτωση αν επιλεγεί ως prime ένας receiver μικρότερων δυνατοτήτων από τους υπόλοιπους θα καθυστερεί την αποστολή ενώ αντίθετα αν επιλεγεί ένας πολύ πιο ισχυρός, ο sender θα στέλνει με ρυθμό γρηγορότερο από όσο μπορούν να λάβουν οι υπόλοιποι.

2.7.4 Μεταφορά – sender

Όταν ενεργοποιηθεί το session ο sender περιμένει τους πρώτους receivers να κάνουν join. Σε κάθε receiver που συνδέεται, αποστέλλεται αυτόματα το μήνυμα SESSION INFO που περιέχει τις πληροφορίες του session. Ο receiver εξετάζει τις πληροφορίες που έλαβε και στέλνει το μήνυμα JOIN SESSION .Ο sender απαντά θετικά με το μήνυμα JOIN ACK και περιμένει να λάβει το μήνυμα READY για να ολοκληρωθεί η είσοδος του receiver στο group. Όταν κάνει join ο πρώτος receiver, ο sender τον ενημερώνει ότι είναι ο prime με το μήνυμα PRIME και ξεκινάει την αποστολή. Από εκείνο το σημείο ο sender επικεντρώνεται στον prime. Ο prime είναι αυτός που τροφοδοτεί τον sender με ACKs και ρυθμίζει τον ρυθμό αποστολής. Ο sender κρατάει πληροφορίες σχετικά με την πρόοδο λήψης του αρχείου μόνο για τον prime. Πιο συγκεκριμένα έχει μια λίστα με τα segments που έχει να στείλει και μια λίστα με segments που δεν έχουν επιβεβαιωθεί. Ο sender στέλνει διαδοχικά τα segments σε κάθε block όπως έχουν προκύψει από τον χωρισμό του αρχείου κατά την αρχικοποίηση του session. Όταν τελειώσει την αποστολή του τελευταίου block ελέγχει αν υπάρχουν segments που δεν έχει επιβεβαιωθεί η λήψη τους από τον prime. Αν υπάρχουν, ξεκινάει νέο πέρασμα και στέλνει μόνο τα segments που χάθηκαν κατά το προηγούμενο. Γίνονται όσα περάσματα χρειάζεται ώστε να λάβει ο prime ολόκληρο το αρχείο. Αν ο prime τελειώσει πριν τους υπόλοιπους στέλνει μήνυμα DONE και ο sender ορίζει ως prime τον επόμενο receiver που έχει σειρά. Ο sender στέλνει μήνυμα STATUS REQUEST στον νέο prime περιμένει να λάβει NACK για να συνεχίσει την αποστολή. Ο νέος prime ενημερώνει τον sender με την αποστολή NACKs σχετικά με τα segments που του λείπουν. Ο sender επεξεργάζεται τα NACKs και ακολουθεί την ίδια διαδικασία αποστολής με περάσματα. Εφόσον ο prime ρυθμίζει την αποστολή οι υπόλοιποι receivers που βρίσκονται στο παρασκήνιο λαμβάνουν δεδομένα με βάση τις δικές του ανάγκες. Αν κάποιος receiver τελειώσει πριν τον prime ειδοποιεί τον sender και βγαίνει από το group. Σε ένα δίκτυο όπου υπάρχει ομοιογένεια ανάμεσα στους receivers, η απώλεια που παρατηρείται σε αυτούς που λαμβάνουν στο

παρασκήνιο είναι παρόμοια με αυτή του prime, έτσι όταν έρχεται η σειρά τους να γίνουν prime ο αριθμός των segments που έχουν να λάβουν για να ολοκληρωθεί η λήψη είναι μικρός.

2.7.5 Μεταφορά - receiver

Όταν συνδεθεί στον sender, ο receiver λαμβάνει το μήνυμα SESSION INFO με τις πληροφορίες του session. Εξετάζει τις πληροφορίες που περιέχει το μήνυμα και πιο συγκεκριμένα ελέγχει αν έχει τον απαραίτητο διαθέσιμο χώρο για να το λάβει. Αν ο έλεγχος βγει θετικός στέλνει μήνυμα JOIN SESSION στον sender δηλώνοντας ότι θέλει να γίνει μέλος στο session. Όταν ο sender απαντήσει θετικά με JOIN ACK ο receiver αρχικοποιεί το αρχείο στο οποίο θα γράψει δεδομένα, κάνει join το multicast socket στο multicast group που έλαβε στο SESSION INFO και στέλνει το μήνυμα READY στον sender με το οποίο δηλώνει ότι είναι έτοιμος να λάβει το αρχείο. Αν είναι ο πρώτος που στέλνει μήνυμα READY στον sender η αποστολή ξεκινάει όταν λάβει ο sender το μήνυμα. Αν η αποστολή είναι ήδη σε εξέλιξη ο receiver ξεκινάει να λαμβάνει μόλις γίνει μέλος του multicast group και το μήνυμα READY δεν έχει κανέναν αντίκτυπο στον sender. Κατά την λήψη του αρχείου ο receiver ελέγχει αν το segment που έλαβε με το συγκεκριμένο block και segment number είναι στη λίστα με αυτά που έχει να λάβει. Αν είναι στη λίστα, τότε γράφει τα δεδομένα του segment στο αρχείο. Αν ο receiver έχει λάβει μήνυμα από τον sender ότι είναι ο prime τότε στέλνει ACK για τα segments που λαμβάνει. Στο ACK περιλαμβάνει το block και segment number του segment που έλαβε ώστε ο sender να καταλάβει σε ποιο segment αναφέρεται το ACK. Σε κάθε segment που λαμβάνει ο receiver ελέγχει αν έχει λάβει ολόκληρο το αρχείο. Αν έχουν ληφθεί όλα τα κομμάτια του αρχείου, τότε στέλνει το μήνυμα DONE στον sender το οποίο του λέει ότι ο receiver έχει ολοκληρώσει την λήψη του. Αν ο receiver έχει κάνει join στο session ενώ η αποστολή ήταν ήδη σε εξέλιξη τότε λαμβάνει κανονικά δεδομένα μέχρι να έρθει η σειρά του να γίνει prime. Τότε δέχεται μήνυμα STATUS REQUEST με το οποίο ο sender του ζητά να στείλει NACKs για κάθε block για το οποίο του λείπουν segments. Ο receiver στέλνει NACKs για τα blocks τα οποία είχαν σταλεί πριν κάνει αυτός join στο session.

2.7.6 Ολοκλήρωση μεταφοράς

Στην μεριά του sender η αποστολή ολοκληρώνεται όταν δεν υπάρχουν πια άλλα μέλη στο session. Το αν τα μέλη αποσυνδέθηκαν γιατί ολοκλήρωσαν την λήψη ή για κάποιον άλλον λόγο δεν έχει σημασία. Από την στιγμή που ξεκινήσει να στέλνει δεδομένα ο sender αν οποιαδήποτε στιγμή αδειάσει το session group τότε η αποστολή σταματά και το session απενεργοποιείται. Οι receivers όταν δουν ότι έλαβαν όλα τα κομμάτια του αρχείου στέλνουν μήνυμα DONE στον sender. Ο sender βγάζει τον receiver από το session group και απαντά με το μήνυμα COMPLETION. Όταν ο receiver λάβει το μήνυμα βγαίνει από

το multicast group και υπολογίζει το MD5 του αρχείου. Αν ταιριάζει με αυτό που έλαβε στο SESSION INFO τότε το αρχείο έχει ληφθεί σωστά.

3 ΑΛΓΟΡΙΘΜΟΣ

3.1 Λογικός χωρισμός αρχείου

Κατά την αρχικοποίηση του session όταν γίνει επιλογή του αρχείου που θα αποσταλεί, το αρχείο χωρίζεται σε blocks και segments. Με τον λογικό χωρισμό του αρχείου σε blocks και segments δεν χρειάζεται να φορτώνεται ολόκληρο το αρχείο στη μνήμη του υπολογιστή και να καταναλώνει πόρους. Ο χωρισμός του αρχείου βοηθά στην καλύτερη διαχείριση του και διευκολύνει τις διαδικασίες ανάγνωσης και εγγραφής κατά την αποστολή. Τα segments έχουν σταθερό μήκος 1456 bytes (segment size), το οποίο προκύπτει από την σχέση της εικόνας 2.6. Ο αριθμός των segments σε ένα block (block capacity) υπολογίζεται από το πόσα segments μπορούν να περιγραφούν σε ένα μήνυμα NACK. Τα μηνύματα NACK περιέχουν ένα bit mask όπου κάθε bit περιγράφει την κατάσταση ενός segment στο block, δηλαδή αν το segment έχει ληφθεί ή όχι. Πρώτα το μέγεθος του αρχείου διαιρείται με το segment size και υπολογίζεται ο συνολικός αριθμός των segments που θα αποτελούν το αρχείο. Στη συνέχεια ο συνολικός αριθμός των segments διαιρείται με το block capacity και προκύπτει ο συνολικός αριθμός των block. Οι τιμές του segment size και του block capacity είναι γνωστές στον sender κατά την εκκίνηση της εφαρμογής.

3.2 Αποστολή αρχείου

Πριν ξεκινήσει η αποστολή ο sender αρχικοποιεί δυο βασικές δομές δεδομένων τις οποίες χειρίζεται κατά την διάρκεια της αποστολής. Η πρώτη δομή περιγράφει τα segments τα οποία δεν έχουν ληφθεί από τον receiver που είναι prime. Είναι ένα dictionary που για κλειδί (key) έχει τον αριθμό ενός block και για value έχει ένα bit mask όπου τα bits χαρακτηρίζουν την κατάσταση των segments μέσα στο block. Αν η τιμή του bit είναι 1 (on) σημαίνει ότι ο prime δεν έχει λάβει ακόμα το segment στο οποίο αντιστοιχεί το bit, ενώ αν το bit έχει τιμή 0 (off) σημαίνει ότι έχει ληφθεί ACK για το συγκεκριμένο segment. Ο αριθμός των blocks και των segments που υπάρχουν στη δομή προέρχεται από τον λογικό χωρισμό του αρχείου με βάση το segment size και το block capacity. Παρακάτω αυτή η δομή θα αναφέρεται ως ***not_acked*** (segments για τα οποία δεν έχει ληφθεί ACK).

Η δεύτερη δομή είναι μια λίστα που περιέχει τους αριθμούς των segments που αντιστοιχούν στο block το οποίο βρίσκεται σε αποστολή. Αν η αποστολή βρίσκεται στο πρώτο πέρασμα τότε η λίστα θα περιέχει όλα τα segments του block ενώ αν έχουν προηγηθεί άλλα περάσματα τότε η λίστα θα έχει μέσα μόνο τους αριθμούς των segments που δεν ελήφθησαν κατά τα προηγούμενα περάσματα. Παρακάτω αυτή η δομή θα αναφέρεται ως

to_send (segments τα οποία έχουν δρομολογηθεί για αποστολή).

Όταν ξεκινάει η αποστολή η to_send περιέχει τους αριθμούς των segments από το πρώτο block. Σε κάθε αποστολή ο sender παίρνει τον αριθμό του segment που είναι να στείλει από την to_send, διαβάζει τα δεδομένα από το αρχείο που αντιστοιχούν στο συγκεκριμένο segment και τα στέλνει στο multicast socket. Το κομμάτι του αρχείου που πρέπει να διαβάσει ο sender ώστε να το στείλει υπολογίζεται με την βοήθεια του block και του segment number. Από τον αριθμό του block και το block capacity, εφόσον τα segments έχουν σταθερό μέγεθος, υπολογίζεται το byte από το οποίο ξεκινάει το block. Από τον αριθμό του segment υπολογίζεται από ποιο byte μέσα στο block πρέπει να διαβαστούν τα δεδομένα. Το read πριν την αποστολή γίνεται σε κομμάτια ίσα με το μέγεθος του segment, δηλαδή 1456 bytes. Μετά την αποστολή ο sender αφαιρεί αυτό το segment number από την to_send.

Όταν η to_send αδειάσει σημαίνει ότι έχουν σταλεί όλα τα segments για το τρέχον block. Ο sender κρατάει την τιμή από το τρέχον block σε μια μεταβλητή. Όταν έχει τελειώσει η αποστολή για ένα block ο sender ελέγχει αν αυτό ήταν το τελευταίο block του αρχείου ώστε να ξεκινήσει νέο πέρασμα ή περνά στην αποστολή segments από το επόμενο block. Κατά την αλλαγή του block το bit mask της not_acked που αντιστοιχεί στο επόμενο προς αποστολή block μετατρέπεται σε λίστα και αντιγράφεται στην to_send. Η to_send περιέχει μόνο εκείνα τα segments τα bits των οποίων ήταν on στο bit mask. Όταν ενημερωθεί η to_send επαναλαμβάνεται η ίδια διαδικασία. Όταν η αποστολή για όλα τα blocks τελειώσει, πριν ξεκινήσει νέο πέρασμα γίνεται έλεγχος αν υπάρχουν segments που δεν έχουν παραδοθεί κατά το πέρασμα που μόλις τελείωσε. Η δομή που ελέγχεται είναι η not_acked. Αν στη not_acked υπάρχουν bits με τιμή 1, σημαίνει ότι δεν έχουν παραδοθεί όλα τα segments και πρέπει να ξεκινήσει νέο πέρασμα. Αν δεν υπάρχουν segments που πρέπει να σταλούν ξανά τότε ο prime έχει λάβει όλα τα segments και πρέπει να ειδοποιήσει τον sender με ένα μήνυμα DONE. Αν μέχρι τον έλεγχο ο sender δεν έχει λάβει DONE από τον prime στέλνει μήνυμα STATUS REQUEST. Ένας receiver που έχει λάβει όλα τα segments και λαμβάνει μήνυμα STATUS REQUEST δεν έχει να στείλει NACKs και απαντάει με ένα μήνυμα DONE.

Όταν αλλάξει ο prime η not_acked έχει όλα τα bits 0. Για να ξεκινήσει νέο πέρασμα ο sender πρέπει να μάθει ποια segments λείπουν από τον νέο prime και του στέλνει STATUS REQUEST. Ο prime τότε απαντάει με ένα NACK για κάθε block από το οποίο του λείπουν segments. Τα NACK μηνύματα περιέχουν ένα bit mask σαν και αυτά που κρατούνται στη not_acked. Όταν ληφθεί ένα NACK το bit mask που υπάρχει στο NACK αντιγράφεται στη θέση του bit mask που υπάρχει στη not_acked για το αντίστοιχο block. Η λειτουργία της αποστολής με περάσματα φαίνεται παρακάτω στον ψευδοκώδικα 1.

Listing 1: Αποστολή με περάσματα

```
if send_from_current_block() == True:
```

```
        send_data()
else:
        if current_block == last_block:
                if has_more_to_send():
                        start_new_pass()
                else:
                        send_status_request()
        else:
                send_next_block()
```

3.3 Λήψη ACK

Το ACK είναι μια από τις δυο μορφές feedback που δέχεται ο sender. Το ACK συμβάλει στην επίτευξη αξιοπιστίας καθώς ο sender ενημερώνεται για το ποια segments ελήφθησαν σωστά και επίσης με την βοήθεια των ACKs λειτουργεί ο αλγόριθμος εντοπισμού χαμένων segments που με την σειρά του τροφοδοτεί τον αλγόριθμο που ρυθμίζει τον ρυθμό αποστολής. Όταν γίνεται λήψη ενός ACK πρώτα ενημερώνεται η δομή not_acked και το segment για το οποίο λήφθηκε το ACK σημειώνεται ως ληφθέν ώστε να μην σταλεί ξανά στο επόμενο πέρασμα αν υπάρχουν άλλα χαμένα segments. Μετά με την βοήθεια του block και segment number που υπήρχε στο ACK γίνεται έλεγχος αν βρέθηκαν χαμένα segments. Το αποτέλεσμα του ελέγχου για χαμένα segments επηρεάζει τις μεταβλητές που καθορίζουν την λειτουργία του αλγόριθμου ελέγχου συμφόρησης.

3.4 Εντοπισμός χαμένων segments

Για κάθε segment που λαμβάνει ο prime στέλνει και ένα ACK. Στο ACK περιλαμβάνεται το block και το segment number του segment που έλαβε ο prime. Ένα segment θεωρείται χαμένο όταν ληφθούν 3 ACKs που αφορούν segments που στάλθηκαν μετά από αυτό. Το ότι τα segments στάλθηκαν μετέπειτα φαίνεται από την σύγκριση του block και segment number στα ACKs. Ο sender στέλνει πάντα τα segments διαδοχικά, δηλαδή πρώτα στέλνονται τα segments με μικρότερο block και segment number. Αν για παράδειγμα ληφθούν ACKs για τα segments 6, 7, 8 του block 3 και δεν έχει ληφθεί ACK για το segment 5 του ίδιου block, τότε το segment 5 του block 3 θεωρείται χαμένο (εικόνα 3.1).

Στον sender ένα segment από την στιγμή που θα αποσταλεί χαρακτηρίζεται από τρεις καταστάσεις:

Lost – Segment που έχει χαρακτηριστεί σαν χαμένο

Acked – Segment για το οποίο έχει ληφθεί ACK

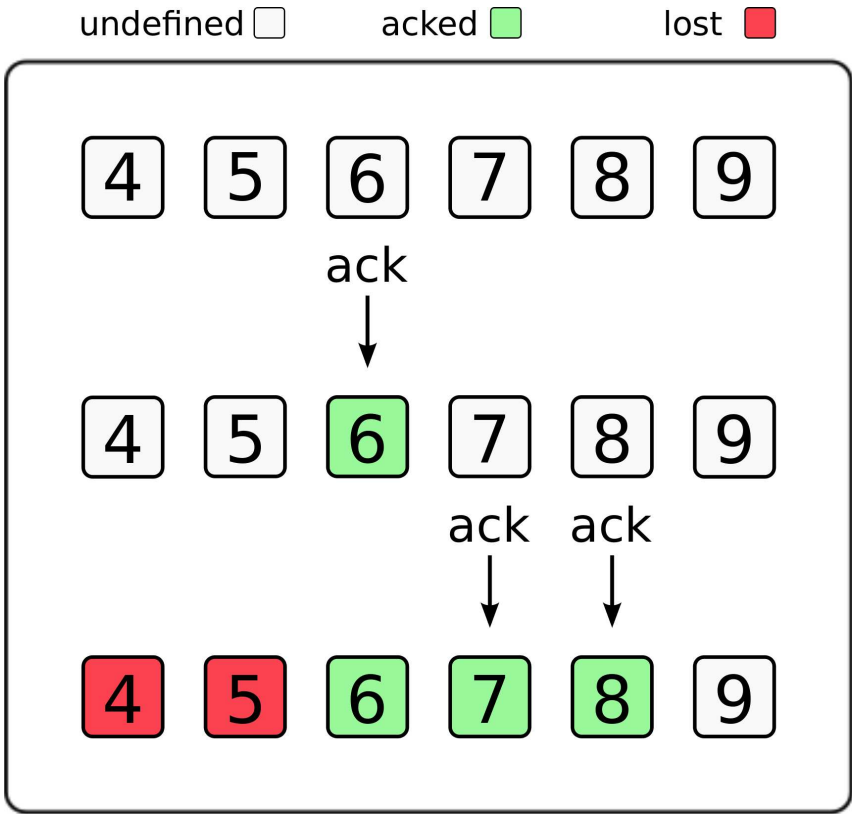


Figure 3.1: Εντοπισμός χαμένων segments

Undefined – Segment που έχει σταλεί αλλά δεν βρίσκεται σε καμία από τις προηγούμενες καταστάσεις

Undefined είναι το segment για το οποίο δεν έχει ληφθεί ACK και δεν έχει χαρακτηριστεί ως lost από ACKs που ελήφθησαν για άλλα segments. Τα undefined segments κρατούνται σε ένα dictionary με key το segment number και για value έναν μετρητή. Όταν ληφθεί ένα ACK, το segment στο οποίο απευθύνεται το ACK βγαίνει από το dictionary και αυξάνεται ο μετρητής σε όλα τα segments με μικρότερο segment number από αυτό που λήφθηκε. Όταν ο μετρητής ενός segment number γίνει 3, σημαίνει ότι έχουν ληφθεί τρία ACKs για segments με μεγαλύτερο segment number, επομένως το συγκεκριμένο segment χαρακτηρίζεται lost και βγαίνει από το dictionary (εικόνα 3.2). Για κάθε block υπάρχει ένα ξεχωριστό dictionary το οποίο κρατάει τα undefined segments. Όταν ξεκινάει η αποστολή για το επόμενο block πολλές φορές μένουν κάποια segments στο dictionary που αφορά το προηγούμενο block. Μετά τα πρώτα τρία ACKs που λαμβάνονται για ένα block, γίνεται έλεγχος του dictionary του προηγούμενου block και όσα segments υπάρχουν μέσα χαρακτηρίζονται ως lost.

3.5 Timeouts

Το timeout είναι ένας μηχανισμός για να ανιχνεύεται η καθυστέρηση στη λήψη ACKs που οφείλεται σε συμφόρηση του δικτύου ή μεγάλη υπερφόρτωση του receiver. Timeout συμβαίνει όταν ο χρόνος που περνά από την τελευταία αποστολή που έκανε ο sender

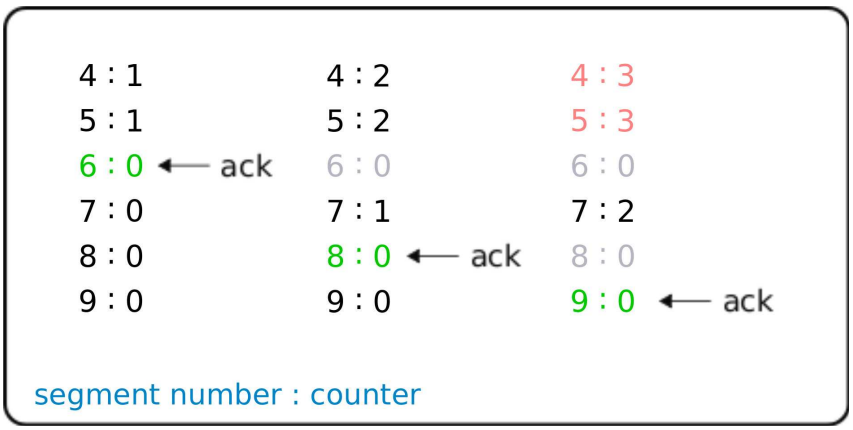


Figure 3.2: Εντοπισμός χαμένων segments στο dictionary

υπερβαίνει κάποιο συγκεκριμένο όριο. Όταν γίνεται αποστολή ενός segment ξεκινάει ένα χρονόμετρο και με κάθε νέα αποστολή το χρονόμετρο μηδενίζεται. Αποστολή segments γίνεται όταν το επιτρέπει ο μηχανισμός συρόμενου παραθύρου που ανανεώνεται από τα ACKs. Αν έχει περάσει πολύς χρόνος από την λήψη του τελευταίου ACK ο sender έχει στείλει τα segments που δικαιούται με βάση το παράθυρο και βρίσκεται σε κατάσταση αναμονής καθώς περιμένει να δεχτεί ACKs για να μπορέσει να στείλει περισσότερα segments. Το χρονικό όριο για το timeout ρυθμίζεται χειροκίνητα. Υπάρχει ένα όριο στον αριθμό των timeouts που μπορεί να προκαλέσει ένας prime. Ο prime μπορεί να αντιμετωπίζει προβλήματα στη λήψη και να προκαλεί timeouts μειώνοντας τον ρυθμό αποστολής. Ο sender για να προστατέψει τους άλλους receivers αλλάζει prime αν τα timeouts που προκαλεί ξεπεράσουν ένα όριο. Αυτή η λειτουργία φαίνεται στον αλγόριθμο 2.

Listing 2: Εντοπισμός Timeout

```
if current_time - last_send_time > TIMEOUT_THRESHOLD:
        timeout()
        timeout_count += 1
if timeout_count == MAX_TIMEOUTS:
        change_prime()
```

3.6 Έλεγχος συμφόρησης

Για τον έλεγχο συμφόρησης χρησιμοποιείται μηχανισμός συρόμενου παραθύρου. Ο sender κρατά ένα μεταβλητό παράθυρο το οποίο καθορίζει πόσα segments μπορεί να στείλει χωρίς να περιμένει να λάβει ACK. Το παράθυρο μεγαλώνει αθροιστικά κατά την περίοδο σωστής λειτουργίας και μειώνεται πολλαπλασιαστικά όταν ανιχνεύεται χαμένο segment. Ο αλγόριθμος ονομάζεται Additive Increase Multiplicative Decrease (**AIMD**)[4] και είναι αυτός που χρησιμοποιείται για την ρύθμιση του παραθύρου συμφόρησης (congestion window) του TCP. Επίσης ομοίως με το TCP ο αλγόριθμος περνάει από τις φάσεις **slow start** και **con-**

gestion avoidance[1]. Το αρχικό μέγεθος του παραθύρου είναι 1. Στη φάση του slow start το παράθυρο μεγαλώνει κατά 1 με κάθε ACK που έρχεται. Το slow start κρατάει μέχρι η τιμή του παραθύρου να ξεπεράσει το threshold. Το threshold είναι μια μεταβλητή τιμή που καθορίζει πότε σταματάει η φάση του slow start και ξεκινάει το congestion avoidance. Κατά την φάση του congestion avoidance το παράθυρο μεγαλώνει κατά 1 κάθε φορά που λαμβάνεται η τρέχουσα τιμή του παραθύρου σε ACKs. Αν το παράθυρο έχει μέγεθος **w**, τότε μετά από την λήψη **w** ACKs η τιμή θα γίνει **w + 1**. Το παράθυρο μεγαλώνει μέχρι να εντοπιστεί συμφόρηση. Τα δυο συμβάντα που ειδοποιούν τον sender ότι υπάρχει συμφόρηση και πρέπει να μειώσει το παράθυρο είναι τα χαμένα segments και τα timeouts. Όταν εντοπιστεί χαμένο segment το παράθυρο και το threshold γίνονται ίσα με το μισό της τιμής που είχε το παράθυρο πριν χαθεί το segment.

$$w' = threshold = w/2$$

Όταν συμβεί timeout το παράθυρο γίνεται 1 και το threshold γίνεται το μισό της παλιάς τιμής του παραθύρου.

$$threshold = w/2, w = 1$$

Listing 3: Αλγόριθμος ρύθμισης παραθύρου συμφόρησης

```
if timeout():
        threshold = w / 2
        w = 1

if lost > 0:
        w' = threshold = w / 2
        on_wire -= (lost + 1)
else:
        if w < threshold:
                #slow start
                w += 1
        else:
                #congestion avoidance
                w' += 1/w
```

Ο μηχανισμός του παραθύρου συμφόρησης υλοποιείται στον sender με την χρήση τριών βασικών μεταβλητών. Η πρώτη ρυθμίζει την τιμή του παραθύρου και ονομάζεται **window**, η δεύτερη είναι η **threshold** που ρυθμίζει την εναλλαγή ανάμεσα στο slow start και το congestion avoidance και η τρίτη είναι η **on_wire** η οποία κρατάει πόσα segments έχει στείλει ο sender χωρίς να λάβει ACK, που ουσιαστικά είναι τα undefined segments. Ο sender μπορεί να στείλει δεδομένα όσο το on_wire είναι μικρότερο του window. Με την

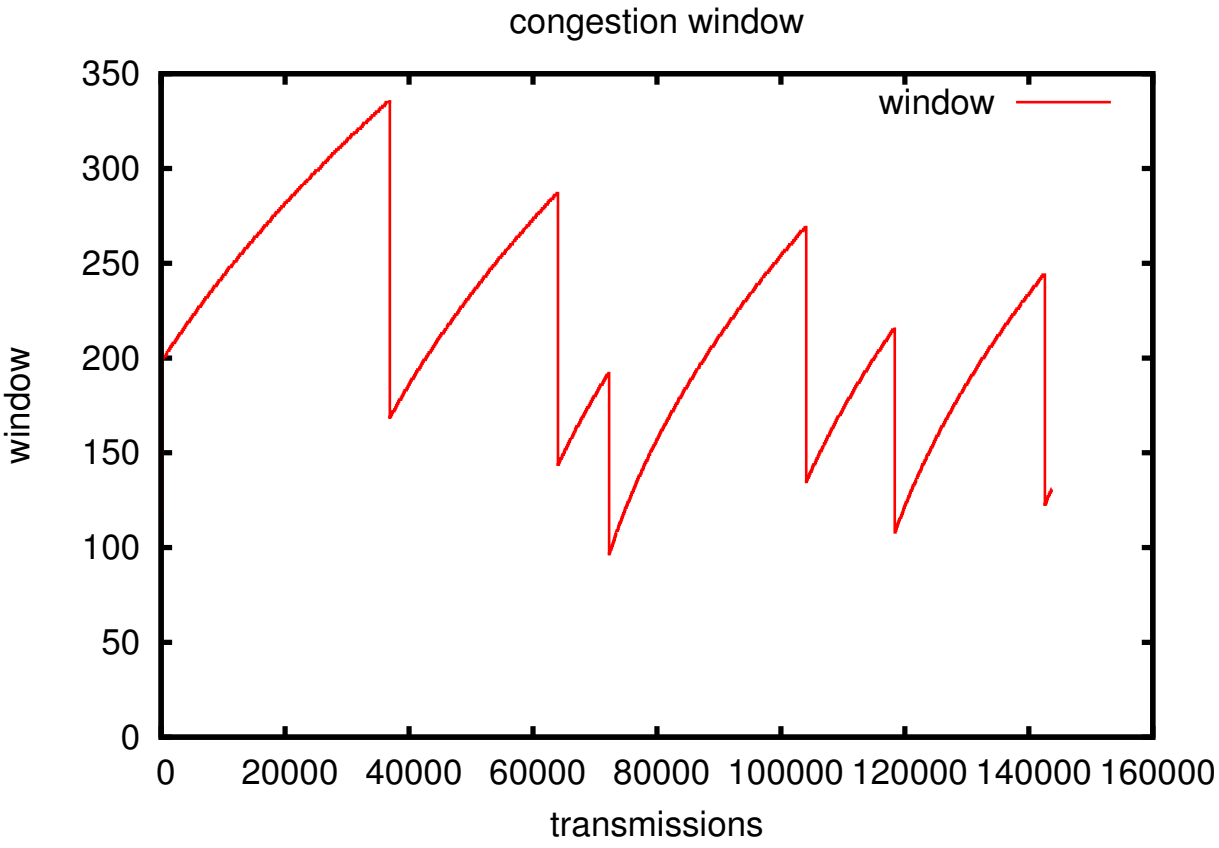


Figure 3.3: Παράθυρο συμφόρησης

λήψη ενός ACK πάντα ένα segment γίνεται acked και μπορεί κάποια να χαρακτηριστούν lost. Το on_wire με κάθε ACK μειώνεται κατά acked + lost όπου τα acked segments είναι πάντα 1 και τα lost μπορεί να είναι 0 ή περισσότερα. Το window και το threshold αλλάζουν σύμφωνα με τον αλγόριθμο AIMD που περιγράφτηκε παραπάνω. Η διακύμανση του παραθύρου κατά την διάρκεια ενός session φαίνεται στην εικόνα 3.3.

3.7 Λήψη αρχείου

Πρώτα ο receiver επεξεργάζεται τις πληροφορίες του SESSION INFO. Αρχικά με βάση το μέγεθος του αρχείου ελέγχει αν έχει τον διαθέσιμο χώρο για να το λάβει. Αν υπάρχει χώρος για το αρχείο, στέλνει μήνυμα JOIN SESSION στον sender. Ο sender απαντάει με μήνυμα JOIN ACK και ο receiver ετοιμάζεται να λάβει το αρχείο. Πρώτα αρχικοποιείται η δομή που κρατάει ο receiver για να παρακολουθεί ποια segments έχει λάβει και ποια του μένει να λάβει για να ολοκληρωθεί η λήψη. Αυτή η δομή είναι ίδια με τη δομή not_acked που έχει ο sender για να γνωρίζει ποια segments έχουν σταλεί σωστά, είναι δηλαδή ένα dictionary με key τον αριθμό του block και value ένα bit mask. Αν η τιμή του bit είναι 1 σημαίνει ότι το segment δεν έχει ληφθεί ακόμα, ενώ αν είναι 0 σημαίνει ότι ο receiver έχει λάβει το segment. Για να δημιουργήσει αυτή τη δομή ο receiver διαβάζει το μέγεθος του αρχείου, το segment size και το block capacity που έλαβε από το SESSION INFO. Παρακάτω

αυτή η δομή θα αναφέρεται ως ***to_receive*** (segments που έχει να λάβει ο receiver).

Μετά την to_receive δημιουργείται το αρχείο στο οποίο θα γίνει εγγραφή δεδομένων με όνομα αυτό που περιείχε το SESSION INFO. Τελευταία ρύθμιση είναι η είσοδος του UDP multicast socket στο multicast group στο οποίο θα σταλεί το αρχείο. Μόλις ο receiver ολοκληρώσει τις παραπάνω διεργασίες στέλνει μήνυμα READY για να ειδοποιήσει τον sender ότι είναι έτοιμος να ξεκινήσει την λήψη. Αν ο receiver είναι ο πρώτος που μπαίνει στο session, τότε η αποστολή ξεκινάει μόλις ο sender λάβει το μήνυμα READY. Αλλιώς αν η αποστολή είναι ήδη σε εξέλιξη ο receiver ξεκινάει να λαμβάνει μόλις γίνει μέλος του multicast group.

Όταν έρχεται ένα μήνυμα δεδομένων ο receiver ελέγχει το block και το segment number στη to_receive για να δει αν έχει ήδη λάβει το segment. Ο έλεγχος γίνεται γιατί κατά την αποστολή τα περάσματα γίνονται με βάση τις ανάγκες του prime receiver σε segments και οι receivers που λαμβάνουν στο παρασκήνιο δεν χρειάζεται να επεξεργάζονται χωρίς λόγο segments που έχουν λάβει ξανά. Στην αρχή όλα τα bits στην to_receive είναι 1, όταν καταφτάνει ένα segment για πρώτη φορά ο sender το επεξεργάζεται και αλλάζει την τιμή του αντίστοιχου bit σε 0. Μετά τον έλεγχο στη to_receive ο receiver προχωράει στην εγγραφή του segment στο αρχείο. Πρώτα υπολογίζεται το σωστό σημείο στο οποίο πρέπει να τοποθετηθούν τα δεδομένα μέσα στο αρχείο. Ο υπολογισμός γίνεται με τον ίδιο τρόπο που ο sender υπολογίζει από ποιο μέρος του αρχείου πρέπει να διαβάσει τα δεδομένα που θα στείλει. Ο receiver ελέγχει αν είναι prime και ανάλογα στέλνει ή όχι ACK για το segment που έλαβε. Σε κάθε segment που λαμβάνει ο receiver κάνει έλεγχο στη to_receive για να δει αν έλαβε όλα τα segments που αποτελούν το αρχείο. Αν έχει λάβει όλα τα segments στέλνει μήνυμα DONE και ο sender απαντάει με μήνυμα COMPLETION. Όταν ο receiver λάβει το μήνυμα COMPLETION ελέγχει το MD5 hash του αρχείου που έλαβε και το συγκρίνει με αυτό που υπήρχε στο SESSION INFO. Αν η τιμές ταιριάζουν σημαίνει ότι το αρχείο λήφθηκε σωστά. Ο έλεγχος του MD5 γίνεται για τοπική ενημέρωση του χρήστη ότι η μεταφορά έγινε σωστά. Όταν κατά την λήψη του αρχείου ο receiver δεχτεί το μήνυμα STATUS REQUEST απαντά με NACKs αν δεν έχει λάβει ακόμα όλα τα δεδομένα ή απαντά με DONE αν δεν έχει τίποτα άλλο να λάβει. Ο έλεγχος γίνεται στη to_receive. Αν το bit mask κάποιου block έχει bits με τιμή 1 τότε αυτό το bit mask μπαίνει σε ένα μήνυμα NACK μαζί με το block number και στέλνεται στον sender.

4 ΠΡΩΤΟΚΟΛΛΟ

4.1 Μορφή μηνυμάτων

Τα μηνύματα που ανταλλάσσονται μεταξύ sender και receivers χωρίζονται σε δύο κατηγορίες. Στη μια είναι τα μηνύματα του πρωτοκόλλου και στην άλλη είναι τα μηνύματα δεδομένων (εικόνα 4.1). Τα unicast μηνύματα ελέγχου που ανταλλάσσονται μέσω TCP

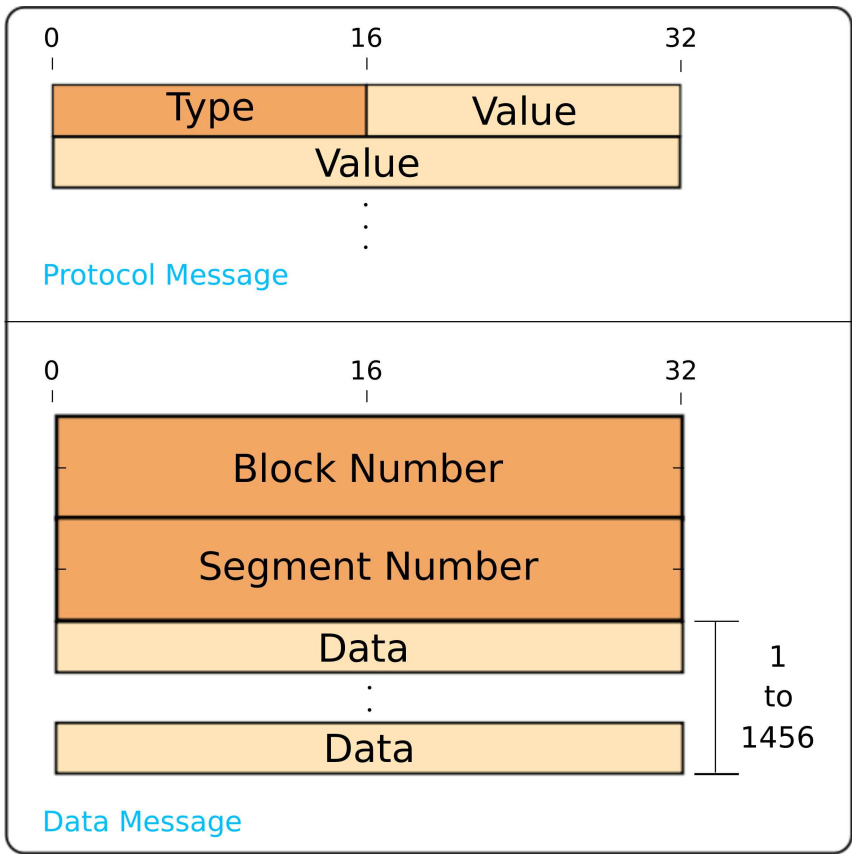


Figure 4.1: Μορφή Μηνυμάτων

είναι αυτά που ρυθμίζουν την λειτουργία του πρωτοκόλλου. Το header περιλαμβάνει το πεδίο TYPE και έχει μέγεθος 2 bytes. Το TYPE ορίζει τον τύπο του μηνύματος και είναι το πεδίο που εξετάζεται πρώτο όταν ληφθεί το μήνυμα και με βάση την τιμή του εκτελούνται οι ανάλογες ενέργειες. Το πεδίο VALUE περιέχει τις παραμέτρους του μηνύματος. Το μήκος του πεδίο VALUE είναι μεταβλητό καθώς κάθε μήνυμα έχει διαφορετικό αριθμό και τύπο παραμέτρων ενώ υπάρχουν και μηνύματα χωρίς παραμέτρους. Ο παραλήπτης κοιτώντας το πεδίο TYPE γνωρίζει αν το πεδίο VALUE έχει παραμέτρους και το εξετάζει ανάλογα. Τα μηνύματα δεδομένων αποτελούνται από τα πεδία DATA HEADER και DATA. Το DATA HEADER έχει μήκος 16 bytes και περιλαμβάνει τα πεδία BLOCK και SEGMENT που έχουν μήκος 8 bytes έκαστο. Το πεδίο DATA περιέχει τα δεδομένα του αρχείου και έχει μεταβλητό μήκος από 1 έως 1456 bytes. Στα μηνύματα δεδομένων δεν υπάρχει η ανάγκη για ένα πεδίο που να δηλώνει τον τύπο του μηνύματος γιατί είναι τα μόνα μηνύματα που λαμβάνει ο receiver στο multicast socket.

4.2 Μηνύματα

Το πρωτόκολλο αποτελείται από 16 μηνύματα τα οποία φαίνονται στον πίνακα 1 και αναλύονται παρακάτω.

Μήνυμα	Type
ACK	1
NACK	2
DONE	3
STATUS_REQ	4
COMPLETION	5
PRIME	6
NO PRIME	7
SESSION INFO	8
SESSION QUERY	9
JOIN SESSION	10
LEAVE SESSION	11
PLAIN MESSAGE	12
JOIN ACK	13
LEAVE ACK	14
NO SESSION	15
READY	16

Table 1: Μηνύματα Πρωτοκόλλου

4.2.1 Μήνυμα SESSION QUERY

Με αυτό το μήνυμα ο receiver ρωτά τον sender αν υπάρχει κάποιο ενεργό session και δηλώνει ότι επιθυμεί να λάβει πληροφορίες σχετικά με αυτό σε περίπτωση που υπάρχει. Το μήνυμα SESSION QUERY δεν έχει παραμέτρους.

4.2.2 Μήνυμα SESSION INFO

Το μήνυμα SESSION INFO στέλνεται από τον sender και περιέχει πληροφορίες για το ενεργό session. Οι παράμετροι που υπάρχουν στο μήνυμα είναι :

- Name – 250 bytes
- Size – 8 bytes
- Hash – 16 bytes
- Multicast group – 4 bytes
- Block capacity – 2 bytes
- Segment size – 2 bytes

Το Name είναι το όνομα του αρχείου. Ανεξάρτητα από το μήκος του ονόματος αυτό το πεδίο είναι πάντα 250 bytes. Το size είναι το μέγεθος του αρχείου σε bytes, ο receiver πριν κάνει αίτηση να μπει στο session ελέγχει αν έχει τον διαθέσιμο χώρο για να λάβει το αρχείο. Το Hash είναι η MD5 τιμή του αρχείου για να μπορεί να γίνει επιβεβαίωση ότι το

αρχείο λήφθηκε σωστά. Το multicast group είναι η multicast IP στην οποία στέλνονται τα δεδομένα. Το block capacity είναι ο αριθμός των segments σε ένα block. Το segment size είναι το σταθερό μέγεθος που έχουν τα segments στα οποία χωρίζεται το αρχείο πριν την αποστολή του.

4.2.3 Μήνυμα JOIN SESSION

Ο receiver στέλνει το μήνυμα JOIN SESSION όταν έχει επεξεργαστεί τα δεδομένα που υπάρχουν στο SESSION INFO και επιθυμεί να γίνει μέλος του session για να λάβει το αρχείο. Το μήνυμα JOIN SESSION δεν έχει παραμέτρους.

4.2.4 Μήνυμα LEAVE SESSION

Ο receiver στέλνει το μήνυμα LEAVE SESSION για να βγει από το session group και να σταματήσει να λαμβάνει το αρχείο. Το μήνυμα LEAVE SESSION δεν έχει παραμέτρους.

4.2.5 Μήνυμα JOIN ACK

Ο sender στέλνει το μήνυμα JOIN ACK ως απάντηση στο μήνυμα JOIN SESSION. Το μήνυμα JOIN ACK δεν έχει παραμέτρους.

4.2.6 Μήνυμα LEAVE ACK

Ο sender στέλνει το μήνυμα LEAVE ACK ως απάντηση στο μήνυμα LEAVE SESSION. Το μήνυμα LEAVE ACK δεν έχει παραμέτρους.

4.2.7 Μήνυμα PRIME

Ο sender στέλνει το μήνυμα PRIME στον receiver από τον οποίον θέλει να λαμβάνει ACKs. Όταν ένας receiver λάβει αυτό το μήνυμα ξεκινάει να στέλνει ACK για κάθε segment που λαμβάνει επιτυχώς. Το μήνυμα PRIME δεν έχει παραμέτρους.

4.2.8 Μήνυμα NO PRIME

Το μήνυμα NO PRIME στέλνεται από τον sender και σε αντίθεση με το PRIME ειδοποιεί τον receiver ότι πρέπει να σταματήσει να στέλνει ACKs. Το μήνυμα NO PRIME δεν έχει παραμέτρους.

4.2.9 Μήνυμα READY

Ο receiver στέλνει το μήνυμα READY όταν έχει αρχικοποιήσει όλες τις παραμέτρους που χρειάζονται για την λήψη του αρχείου και είναι έτοιμος να λάβει δεδομένα. Το μήνυμα READY δεν έχει παραμέτρους.

4.2.10 Μήνυμα ACK

Τα μηνύματα ACK στέλνονται από τον prime receiver για κάθε segment που λαμβάνει επιτυχώς από τον sender. Τα μηνύματα ACK βοηθούν τον sender να γνωρίζει τι έχει ληφθεί επιτυχώς, να εντοπίζει τι έχει χαθεί ώστε να το στείλει ξανά και να προσαρμόζει τον ρυθμό αποστολής του ανάλογα με τον φόρτο του δικτύου. Το μήνυμα ACK περιλαμβάνει τις παρακάτω παραμέτρους :

- Block number – 8 bytes
- Segment number – 8 bytes

Το block number είναι ο αριθμός του block και το segment number είναι ο αριθμός του segment μέσα στο block. Αυτές οι δυο παράμετροι χρειάζονται στο sender για να αναγνωρίσει σε ποιο segment από αυτά που έχει στείλει απευθύνεται το ACK.

4.2.11 Μήνυμα NACK

Τα μηνύματα NACK στέλνονται από τους receivers για να ενημερώσουν τον sender ποια segments δεν έχουν λάβει σε ένα συγκεκριμένο block. Οι παράμετροι που περιλαμβάνει είναι :

- Block number – 8 bytes
- NACK mask – μεταβλητό μέγεθος

Το block number είναι ο αριθμός του block που περιγράφει το NACK. Η NACK mask είναι ένα bit mask όπου τα bits αντιστοιχούν στα segments του block και η τιμή του bit (0 ή 1) δηλώνει αν το segment στο οποίο αντιστοιχεί το bit έχει ληφθεί ή όχι. Για παράδειγμά αν το 34ο bit έχει τη τιμή 1 σημαίνει ότι ο receiver δεν έχει λάβει ακόμα το segment με αριθμό 34 για το block με αριθμό Block number. Το μέγεθος της NACK mask αλλάζει ανάλογα με τον αριθμό των segments που περιγράφει κάθε φορά.

4.2.12 Μήνυμα STATUS REQUEST

Ο sender στέλνει το μήνυμα STATUS REQUEST όταν θέλει να μάθει για την κατάσταση ενός receiver και πιο συγκεκριμένα ποια segments δεν έχει λάβει. Ο receiver απαντά στέλνοντας NACK για κάθε block που δεν έχει λάβει ολόκληρο ή με DONE αν έχει λάβει όλο το αρχείο. Το μήνυμα STATUS REQUEST δεν έχει παραμέτρους.

4.2.13 Μήνυμα DONE

Μετά από κάθε επιτυχημένη λήψη ενός segment ο receiver ελέγχει αν έχει λάβει ολόκληρο το αρχείο. Αν κατά τον έλεγχο διαπιστώσει ότι δεν έχει να λάβει τίποτα άλλο, στέλνει το μήνυμα DONE για να ενημερώσει τον sender ότι έχει τελειώσει την λήψη του. Το μήνυμα DONE δεν έχει παραμέτρους.

4.2.14 Μήνυμα COMPLETION

Μόλις λάβει μήνυμα DONE από κάποιον receiver ο sender απαντάει με το μήνυμα COMPLETION. Το μήνυμα COMPLETION σηματοδοτεί για τον receiver το τέλος του session. Το μήνυμα COMPLETION δεν έχει παραμέτρους.

4.2.15 Μήνυμα PLAIN MESSAGE

Το μήνυμα PLAIN MESSAGE στέλνεται από τον sender και στόχος του είναι να εμφανιστεί ένα πληροφοριακό μήνυμα στην οθόνη του υπολογιστή που τρέχει το πρόγραμμα receiver στο άλλον άκρο. Το μήνυμα έχει μια παράμετρο :

- Message – μεταβλητό μήκος

Ο receiver μόλις λάβει το μήνυμα εκτυπώνει το περιεχόμενο του Message στην οθόνη.

4.2.16 Μήνυμα NO SESSION

Ο sender στέλνει το μήνυμα NO SESSION ως απάντηση στα μηνύματα JOIN SESSION, LEAVE SESSION και SESSION QUERY όταν δεν υπάρχει ενεργό session. Το μήνυμα NO SESSION δεν έχει παραμέτρους.

4.2.17 Μήνυμα DATA

Τα μηνύματα δεδομένων είναι αυτά που στέλνονται με multicast και περιέχουν τα κομμάτια του αρχείου που στέλνει ο sender. Τα μηνύματα δεδομένων περιλαμβάνουν τα πεδία :

- Block number – 8 bytes
- Segment number – 8 bytes
- Data – μεταβλητό

Το block number είναι ο αριθμός του block και το segment number είναι ο αριθμός του segment μέσα στο block. Αυτά τα δύο πεδία χαρακτηρίζουν τα περιεχόμενα του πεδίου Data που είναι δεδομένα του αρχείου. Το πεδίο Data έχει μεταβλητό μέγεθος 1 - 1456 bytes γιατί ενώ τα segments στα οποία χωρίζεται το αρχείο έχουν σταθερό μήκος, σχεδόν πάντα το τελευταίο segment έχει μικρότερο μήκος από τα υπόλοιπα. Το block και το segment number βοηθούν τους receivers να γράφουν τα δεδομένα στο σωστό σημείο μέσα στο αρχείο και να ενημερώνουν τον sender ποια κομμάτια έλαβαν σωστά με την αποστολή ACKs.

5 ΚΩΔΙΚΑΣ

5.1 Modules

Στην python η βασική μονάδα ομαδοποίησης κώδικα είναι το module. Τα modules είναι αρχεία κώδικα με την κατάληξη .py. Τα modules συμβάλουν στην επαναχρησιμοποίηση και την καλύτερη οργάνωση του κώδικα. Ένα module μπορεί να κάνει εισαγωγή (import) ένα άλλο module για να χρησιμοποιήσει των κώδικα που περιλαμβάνει. Ένα μεγάλο πρόγραμμα μπορεί να χωριστεί σε πολλά modules όπου το κάθε ένα αναλαμβάνει μια ξεχωριστή λειτουργία του προγράμματος. Ο κώδικας της εφαρμογής αποτελείται από έντεκα κλάσεις και είναι οργανωμένος σε έξι modules. Παρακάτω γίνεται μια αναφορά στα modules και την οργάνωση των κλάσεων και στη συνέχεια ακολουθεί μια πιο αναλυτική περιγραφή.

1. sender.py
    (a) Sender
    (b) SenderSession
2. receiver.py
    (a) Receiver
    (b) ReceiverSession
3. async.py
    (a) connection_handler
    (b) client_channel
    (c) multicast_channel
4. file.py
    (a) File
5. helper.py
    (a) Bitmask
    (b) Ack
    (c) Rtt
6. shared.py

5.2 Module shared.py

Το module shared.py περιέχει τις σταθερές που ρυθμίζουν την λειτουργία του πρωτοκόλλου και χρειάζονται στα υπόλοιπα modules για να λειτουργήσουν σωστά. Σε αυτό ορίζονται τα μήκη των πεδίων του πρωτοκόλλου ώστε οι παραλήπτες να μπορούν να διερμηνεύουν σωστά τα μηνύματα που λαμβάνουν. Οι σταθερές που ορίζονται είναι :

TYPE – 2 Bytes - Τύπος του μηνύματος

BLOCK_NUM – 8 Bytes - Αριθμός του block

SEG_NUM – 8 Bytes - Αριθμός του segment

TERMINATOR – 4 Bytes -Μήκος του συμβόλου που χρησιμοποιείται για τερματισμό των μηνυμάτων του πρωτοκόλλου

SEG_SIZE – 1456 Bytes - Μήκος ενός segment

MASK_SIZE – 1446 Bytes - Μήκος του bit mask

Ακολουθεί ο ορισμός των σταθερών που ρυθμίζουν την λειτουργία του congestion control. Οι τιμές που ορίζονται είναι :

WINDOW_MIN – 1 - Ελάχιστο μέγεθος του window

THRESHOLD – 200 - Αρχική τιμή του threshold

TIMEOUT – 0.5 sec - Το χρονικό όριο για την ανίχνευση timeout

MAX_TIMEOUTS – 10 - Μέγιστος αριθμός timeouts που μπορεί να προκαλέσει ένας receiver

Τέλος ορίζονται οι αριθμοί που αντιστοιχούν στους τύπους μηνυμάτων του πρωτοκόλλου. Sender και receiver χρησιμοποιούν αυτές τις αντιστοιχίσεις για να αναγνωρίζουν ποιο μήνυμα λήφθηκε από την τιμή του πεδίου TYPE. Τα μηνύματα και οι τιμές που αντιστοιχούν σε αυτά περιγράφονται στο κεφάλαιο 4.

5.3 Module helper.py

Το module helper.py αποτελείται από κλάσεις και μεθόδους οι οποίες εκτελούν γενικές λειτουργίες που χρειάζονται στον sender και στους receivers. Στο module υπάρχουν 3 κλάσεις και 8 μέθοδοι που περιγράφονται παρακάτω.

5.3.1 Κλάση Bitmask

Η κλάση `Bitmask` δημιουργήθηκε για την διαχείριση των bit masks που χρησιμοποιούν οι δομές not_acked και to_receive σε sender και receiver αντίστοιχα. Η διαχείρηση του bit mask μέσα από την κλάση κάνει πιο εύκολο τον χειρισμό των bit και παράγει πιο καθαρό και ευανάγνωστο κώδικα. Ένα bit mask είναι ένας ακέραιος αριθμός όπου τα bits που χρησιμοποιούνται για την αναπαράσταση του μπορούν να κωδικοποιηθούν ως flags, δηλαδή σαν διακόπτες όπου η κατάσταση ενός bit σε μια συγκεκριμένη θέση σημαίνει κάτι. Σε ένα bit mask οι βασικές ενέργειες που γίνονται σε ένα bit είναι ο ορισμός της τιμής του bit σε 1 (set), ο ορισμός της τιμής του bit σε 0 (clear) και ο έλεγχος της κατάστασης του bit (check). Οι παραπάνω ενέργειες γίνονται με τη χρήση δυαδικών τελεστών (bitwise operators) και bit shifting. Αν mask είναι ο ακέραιος και x είναι ο αριθμός του bit :

set mask $|= 1 << x$

clear mask $\&= \sim(1 << x)$

check mask $\&\ (1 << x)$

Οι βασικότερες μέθοδοι της κλάσης που υλοποιούν τις παραπάνω λειτουργίες είναι οι ομώνυμες `set`, `clear` και `check`. Μια ακόμα σημαντική μέθοδος είναι η `to_list` η οποία επιστρέφει μια λίστα που περιέχει τους αριθμούς που αντιστοιχούν στις θέσεις των bit που είναι set.

5.3.2 Κλάση Ack

Η κλάση `Ack` διαχειρίζεται τα ACKs που λαμβάνει ο sender και εντοπίζει χαμένα segments. Στον sender ένα στιγμιότυπο της κλάσης δημιουργείται για κάθε block του αρχείου. Μέσα στην κλάση διατηρείται ένα dictionary όπου για key έχει τον αριθμό του segment και για value έναν μετρητή. Κατά την αποστολή ενός segment καλείται η μέθοδος `add` με όρισμα τον αριθμό του segment ο οποίος μπαίνει στο dictionary και ο μετρητής του αρχικοποιείται σε μηδέν. Όταν ληφθεί το ACK για το segment καλείται η μέθοδος `received_ack` με όρισμα το segment number. Στη `received_ack` υλοποιείται ο αλγόριθμος που περιγράφτηκε στην ενότητα 3.4 για να εντοπιστούν τυχόν χαμένα segments. Τα χαμένα segments αφαιρούνται από το dictionary και η μέθοδος επιστρέφει τον αριθμό των χαμένων segments μαζί με τον αριθμό του τελευταίου χαμένου segment. Ο αριθμός του τελευταίου segment που χάθηκε χρειάζεται για τον αλγόριθμο που ρυθμίζει την λειτουργία του παραθύρου συμφόρησης στον sender.

5.3.3 Κλάση Rtt

Η κλάση `Rtt` χρησιμοποιείται για την μέτρηση του RTT των πακέτων δεδομένων που στέλνει ο sender. Η κλάση διαχειρίζεται ένα dictionary με key τον αριθμό ενός segment

και value τη χρονική στιγμή στην οποία στάλθηκε. Όταν στέλνεται ένα segment καλείται η μέθοδος `add` και ο αριθμός του μπαίνει στο dictionary με value τιν χρονική στιγμή που γίνεται η αποστολή. Όταν λαμβάνεται ένα ACK καλείται η μέθοδος `received_ack` με παράμετρο το segment number και υπολογίζει τον χρόνο που πέρασε από τη αποστολή του segment έως την λήψη του ACK. Η μέτρηση του RTT γίνεται για πληροφοριακούς λόγους και δεν ρυθμίζει τη λειτουργία του πρωτοκόλλου.

5.3.4 Μέθοδος calc_hash( )

Αυτή η μέθοδος παίρνει ως παράμετρο την διαδρομή ενός αρχείου και επιστρέφει την MD5 τιμή του.

5.3.5 Μέθοδοι για ανάλυση παραμέτρων

Είναι οι μέθοδοι `session_opt_parser`, `receiver_opt_parser`, `sender_opt_parser`. Σε αυτές τις μεθόδους γίνεται ο ορισμός των command line παραμέτρων που χρησιμοποιούν ο sender, ο receiver και η εντολή `session` στο cli του sender. Στην python η ανάλυση των command line options γίνεται με την βοήθεια του module `optparse`. Το `optparse` προσφέρει μια εύκολη διεπαφή για γρήγορη δημιουργία παραμέτρων. Πρώτα δημιουργείται ένα στιγμιότυπο της κλάσης `OptionParser` και πάνω σε αυτό μπορούν να προστεθούν παράμετροι με την βοήθεια της μεθόδου `add_option`.

5.3.6 Μέθοδος validate_ip()

Αυτή η μέθοδος ελέγχει αν το string που δέχεται ως παράμετρο είναι μια έγκυρη IP διεύθυνση ή ένα έγκυρο hostname.

5.3.7 Μέθοδος validate_multicast_addr()

Αυτή η μέδοδος ελέγχει αν η IP που δίνεται ως όρισμα είναι ένα έγκυρο multicast address.

5.3.8 Μέθοδος get_block_capacity()

Αυτή η μέθοδος υπολογίζει και επιστρέφει τον αριθμό των segments σε ένα block. Ο αριθμός προκύπτει με βάση τα bits που μπορούν να κωδικοποιηθούν σε ένα αντικείμενο της κλάσης `Bitmask` χωρίς το μέγεθος του να ξεπερνάει την σταθερά MASK_SIZE που ορίζεται στο module shared.py.

5.3.9 Μέθοδος separate_masks()

Αυτή η μέθοδος υπολογίζει και επιστρέφει το dictionary με τα bit masks που χρησιμοποιούν οι δομές `not_acked` και `to_send` σε sender και receiver.

5.4 Module file.py

Το module file.py περιλαμβάνει την κλάση `File` που σχεδιάστηκε για διαχείριση του file object που αντιστοιχεί στο αρχείο προς αποστολή. Σε αυτήν την κλάση εκτός από τις καθιερωμένες λειτουργίες ενός file object της python προστέθηκαν επιπλέον χαρακτηριστικά που χρειάζονται για να περιγραφεί το αρχείο σε ένα session. Οι μέθοδοι που προστέθηκαν είναι :

get_size – επιστρέφει το μέγεθος του αρχείου

get_blocks – επιστρέφει το dictionary με τα bit masks που αντιστοιχεί στο αρχείο με βάση τον χωρισμό του σε blocks και segments

get_block_count – επιστρέφει τον αριθμό των blocks που χωρίζεται το αρχείο

get_block_capacity – επιστρέφει την χωρητικότητα του block σε segments

get_hash – επιστρέφει το MD5 hash του αρχείου

5.5 Module async.py

Το module async.py περιέχει τις τρεις κλάσεις που αναλαμβάνουν την αλληλεπίδραση της εφαρμογής με το δίκτυο. Οι κλάσεις αυτές είναι :

connection_handler – Server που δέχεται συνδέσεις

client_channel – Κανάλι επικοινωνίας με τον receiver όπου ανταλλάσσονται τα μηνύματα του πρωτοκόλλου

multicast_channel – Κανάλι στο οποίο στέλνονται τα multicast δεδομένα

Οι κλάσεις αυτές χρησιμοποιούν τα modules `asyncore` και `asynchat` της python. Το module `asyncore` επιτρέπει την εύκολη συγγραφή εφαρμογών client – server που επικοινωνούν ασύγχρονα χωρίς την χρήση threads και την πολυπλοκότητα που εισάγεται με αυτά. Σε ένα client – server μοντέλο πολλές φορές η εφαρμογή πρέπει να χειριστεί πολλά sockets και να αποφασίζει πότε θα εκτελούνται οι λειτουργίες ανάγνωσης (read) και εγγραφής (write) σε κάθε socket. Ο τρόπος για να γίνει αυτό χωρίς την χρήση threads είναι με την κλήση συστήματος `select`. Η `select` δέχεται τρεις λίστες όπου η πρώτη έχει sockets που είναι υποψήφια για read, η δεύτερη έχει sockets που είναι υποψήφια για write και η τρίτη έχει sockets που θέλουμε να ελέγξουμε για σφάλματα. Η `select` επιστρέφει επίσης τρεις λίστες όπου στην πρώτη είναι τα sockets που έχουν έτοιμα δεδομένα για ανάγνωση, στην δεύτερη έχει sockets στα οποία μπορεί να γίνει write και η τρίτη sockets στα οποία παρουσιάστηκε σφάλμα. Με την χρήση της `select` μια εφαρμογή μπορεί να διανέμει σωστά τους πόρους του συστήματος και να διαχειριστεί αποδοτικά τα sockets. Το module

asyncore βασίζεται στη select αλλά απλοποιεί ακόμη περισσότερο τα πράγματα καθώς η κλάση asyncore.dispatcher προσφέρει έτοιμες τις απαραίτητες μεθόδους για χειρισμό των συμβάντων του socket. Η κλάση asyncore.dispatcher είναι μια wrapper κλάση για το αντικείμενο socket της python ή οποία εκτός από τις βασικές μεθόδους του socket ορίζει και επιπλέον μεθόδους για χειρισμό συμβάντων. Οι βασικότερες από αυτές είναι:

- handle_read - συμβάν ανάγνωσης από το socket
- handle_write - συμβάν εγγραφής στο socket
- handle_close - συμβάν κλεισίματος του socket
- handle_accept - συμβάν αποδοχής νέας σύνδεσης στον server

Ο απαραίτητος κώδικας που χρειάζεται να εκτελεστεί σε κάποιο από αυτά τα συμβάντα μπαίνει στην ανάλογη μέθοδο της κλάσης που θα κληρονομεί την asyncore.dispatcher. Η βασική ιδέα είναι ότι υπάρχουν τα κανάλια επικοινωνίας (channels) τα οποία είναι στιγμιότυπα κλάσεων που κληρονομούν την asyncore.dispatcher και υπερβαίνουν τις βασικές μεθόδους της. Η διαχείριση των καναλιών γίνεται από την μέθοδο asyncore.loop() στην οποία εκτελείται η select από τα αποτελέσματα της οποίας πυροδοτούνται τα ανάλογα συμβάντα για κάθε κανάλι. Η asyncore.loop() διατηρεί ένα map που περιέχει τα κανάλια που δημιουργούνται κατά την εκτέλεση της εφαρμογής και με βάση αυτό το map γίνονται οι έλεγχοι για read ή write συμβάντα.

Το module asynchat επεκτείνει την λειτουργία του asyncore με την κλάση asynchat.async_chat που είναι υποκλάση της dispatcher και προσφέρει νέες μεθόδους που διευκολύνουν την ανταλλαγή μηνυμάτων μεταξύ client και server. Οι δυο βασικές μέθοδοι που προσφέρει η async_chat είναι :

- collect_incoming_data
- found_terminator

Η async_chat επιτρέπει τον ορισμό ενός συμβόλου τερματισμού μηνυμάτων (terminator) ώστε η εφαρμογή να αναγνωρίζει πότε τελειώνει ένα μήνυμα και πότε ξεκινάει το άλλο μέσα σε ένα TCP stream. Η μέθοδος collect_incoming_data συσσωρεύει τα δεδομένα που λαμβάνει το κανάλι από το TCP stream. Όταν μέσα στα δεδομένα βρεθεί το σύμβολο που έχει οριστεί ως terminator σημαίνει ότι έχει ληφθεί ένα ολόκληρο μήνυμα και καλείται η μέθοδος found_terminator ώστε να γίνει επεξεργασία του μηνύματος.

5.5.1 Κλάση connection_handler

Η κλάση connection_handler είναι το server κομμάτι της εφαρμογής. Είναι υποκλάση της dispatcher και υπερβαίνει την μέθοδο handle_accept() που καλείται όταν συνδέεται ένας receiver.

5.5.2 Κλάση client_channel

Αυτή η κλάση είναι υποκλάση της `async_chat` και αναλαμβάνει την ανταλλαγή μηνυμάτων μεταξύ sender και receiver. Όταν ένας receiver συνδέεται στον sender δημιουργείται ένα αντικείμενο αυτής της κλάσης που αναλαμβάνει την επικοινωνία από εκείνο το σημείο. Η κλάση υπερβαίνει τις μεθόδους `collect_incoming_data` και `found_terminator` που λαμβάνουν και διαχωρίζουν τα μηνύματα που έρχονται από το δίκτυο. Η αποστολή μηνυμάτων γίνεται με την μέθοδο `push()` της `async_chat` η οποία δέχεται το μήνυμα προς αποστολή και το προωθεί στο δίκτυο όταν το επιτρέπει το κανάλι. Αυτή η κλάση χρησιμοποιείται αποκλειστικά για μεταφορά μηνυμάτων του πρωτοκόλλου μεταξύ sender και receiver.

5.5.3 Κλάση multicast_channel

Αυτή η κλάση κληρονομεί από την `dispatcher` και είναι υπεύθυνη για την αποστολή και παραλαβή multicast μηνυμάτων στο δίκτυο. Υλοποιεί τις βασικές μεθόδους `handle_read` και `handle_write` που αναλαμβάνουν την λήψη και την αποστολή δεδομένων όταν αυτό είναι δυνατόν. Σε αυτήν την κλάση σε αντίθεση με τις άλλες για να σταλούν multicast δεδομένα δημιουργείται ένα UDP socket. Το στιγμιότυπο της κλάσης που δημιουργείται σε sender και receiver είναι διαφορετικό. Στον sender που θέλει μόνο να στείλει δεδομένα, αυτό που χρειάζεται είναι να δημιουργηθεί ένα UDP socket και να οριστεί το TTL που επηρεάζει την εμβέλεια του multicast. Από εκείνο το σημείο μπορεί να γίνει αποστολή δεδομένων με την προϋπόθεση ότι η διεύθυνση που ορίζεται κατά την αποστολή των δεδομένων είναι μια έγκυρη multicast IP. Στον receiver το UDP socket πρέπει να συσχετιστεί με ένα port number και να κάνει join στο multicast group που θα σταλούν τα δεδομένα. Η κλάση υλοποιεί τις μεθόδους `join_group` και `leave_group` για να μπορεί το multicast socket να κάνει εύκολα join ή leave από το group. Η διαφοροποίηση για το πως θα αρχικοποιηθεί το αντικείμενο, δηλαδή αν θα χρησιμοποιηθεί για αποστολή ή για λήψη δεδομένων, γίνεται από το port number που περνιέται ως παράμετρος. Αν το port number είναι διάφορο του μηδενός σημαίνει ότι το socket θα αρχικοποιηθεί για να λάβει δεδομένα, ενώ αν είναι μηδέν συμβαίνει το αντίθετο.

Αν το multicast κανάλι θα μπει στο map του `asyncore.loop()` εξαρτάται από το τι επιστρέφουν οι μέθοδοι `readable()` και `writable()`. Αν για παράδειγμα η `readable()` επιστρέφει False τότε το κανάλι δεν θα ελεγχθεί για δεδομένα ενώ σε αντίθετη περίπτωση θα ελεγχθεί. Η τιμή που επιστρέφουν αυτές οι δυο μέθοδοι εξαρτάται από τις τιμές των αντίστοιχων μεταβλητών `is_readable` και `is_writable` η τιμή των οποίων μπορεί να αλλαχθεί για να ρυθμίζεται η λειτουργία του read και του write στο multicast socket.

Όταν δημιουργούνται στιγμιότυπα των παραπάνω κλάσεων στις κλάσεις `Sender` και `Receiver`, περνιέται ως παράμετρος μια αναφορά της αντίστοιχης βασικής κλάσης που

δημιουργεί το στιγμιότυπο. Η αναφορά στη βασική κλάση ονομάζεται master. Αυτή η αναφορά στη βασική κλάση χρειάζεται γιατί ο κυρίως κώδικας για τα συμβάντα read και write των sockets και για την αποδοχή νέων συνδέσεων βρίσκεται στις βασικές κλάσεις `Sender` και `Receiver`. Οι μέθοδοι που υπάρχουν στις κλάσεις που υλοποιούν τα channels απλά καλούν την αντίστοιχη μέθοδο στην βασική κλάση μέσω της αναφοράς master. Για παράδειγμα η μέθοδος `send_data()` της κλάσης `Sender` καλείται μέσω της μεθόδου `handle_write()` της κλάσης `multicast_channel`.

5.6 Module sender.py

Το module sender.py είναι το αρχείο που εκτελείται για να ξεκινήσει η εφαρμογή sender. Στο module περιλαμβάνονται οι κλάσεις `SenderSession` και `Sender`.

5.6.1 Κλάση SenderSession

Η κλάση `SenderSession` παρέχει στον sender τις πληροφορίες και τις μεθόδους για την διαχείριση του session. Η κλάση `SenderSession` διαχειρίζεται τις παρακάτω μεταβλητές :

- group – Λίστα με τα μέλη του session
- active – Κατάσταση του session. Αν είναι ενεργό ή όχι
- file – Αντικείμενο της κλάσης File που είναι το αρχείο προς αποστολή
- multicast_group – Η multicast διεύθυνση για αποστολή δεδομένων

Η διαχείριση του session γίνεται από τις παρακάτω μεθόδους:

- activate – Ενεργοποιεί το session
- deactivate – Απενεργοποιεί το session
- set_file – Επιλογή του αρχείου που θα αποσταλεί
- set_mcast_group – Επιλογή multicast διεύθυνσης
- join_group – Είσοδος ενός receiver στο group
- leave_group – Έξοδος ενός receiver από το group

5.6.2 Κλάση Sender

Η λειτουργία της κλάσης `Sender` βασίζεται στον χειρισμό συμβάντων που κάνει η `asyncore.loop()`. Στην αρχή δημιουργείται ένα στιγμιότυπο της κλάσης `connection_handler` το οποίο διαχειρίζεται τις νέες συνδέσεις. Για κάθε νέο receiver που συνδέεται δημιουργείται ένα αντικείμενο `client_channel` το οποίο μπαίνει αυτόματα στο map της `asyncore.loop()`. Για την αποστολή δεδομένων δημιουργείται ένα αντικείμενο της κλάσης `multicast_channel`. Η υπόλοιπη λειτουργία στηρίζεται στην διαχείριση συμβάντων (events) σε αυτά τα channels που γίνεται στην μέθοδο `run()` η οποία εκτελεί την `asyncore.loop()`. Για τον χειρισμό συμβάντων στα channels η κλάση ορίζει τις παρακάτω μεθόδους :

- handle_accept - Χειρισμός νέων συνδέσεων
- parse_msg – Ανάγνωση μηνυμάτων από το client channel
- handle_close – Διαχείριση κλεισίματος καναλιού
- send_data – Αποστολή δεδομένων
- handle_multicast_channel_error – χειρισμός σφάλματος στο multicast channel

Η `asyncore.loop()` ελέγχει συνέχεια τα ενεργά client channels για δεδομένα. Όταν ληφθεί ένα ολόκληρο μήνυμα καλείται η μέθοδος `found_terminator()` του ανάλογου channel στο οποίο ήρθε το μήνυμα. Μέσω της `found_terminator()` καλείται η `parse_msg()` η οποία αναλαμβάνει την ανάλυση του μηνύματος. Στην `parse_msg()` πρώτα ελέγχονται τα bytes στην αρχή του μηνύματος που αντιστοιχούν στο πεδίο TYPE. Με βάση την τιμή του πεδίου TYPE καλείται η μέθοδος που αντιστοιχεί στον αντίστοιχο τύπο μηνύματος. Για κάθε τύπο μηνύματος που μπορεί να λάβει ο sender υπάρχει μια μέθοδος `read_*()` στην οποία υπάρχει ο κώδικας που πρέπει να εκτελεστεί κατά την λήψη του μηνύματος. Σε κάθε `read_*()` μέθοδο περνιέται μια αναφορά στο channel από το οποίο λήφθηκε το μήνυμα ώστε ο sender να ξέρει σε πιο channel πρέπει να στείλει απάντηση αν αυτό είναι απαραίτητο.

Όταν το multicast channel είναι ελεύθερο καλείται η μέθοδος `send_data()` που αναλαμβάνει την αποστολή του αρχείου εφαρμόζοντας τον αλγόριθμο με τα περάσματα που περιγράφεται στην ενότητα 3.2. Όταν η αποστολή δεδομένων δεν είναι επιθυμητή η μεταβλητή `is_writable` του multicast channel γίνεται False ώστε κατά την κλήση της `asyncore.loop()` να μην ελεγχθεί αν το channel είναι διαθέσιμο για αποστολή. Αυτός ο μηχανισμός χρησιμοποιείται για τον έλεγχο του ρυθμού αποστολής. Η τιμή της μεταβλητής `is_writable` εξαρτάται από τις τιμές των μεταβλητών `window` και `on_wire` που ρυθμίζουν την λειτουργία του παραθύρου συμφόρησης. Όσο η `on_wire` είναι μικρότερη από την `window`, επιτρέπεται η αποστολή δεδομένων και η τιμή της `is_writable` είναι True ώστε αν το multicast channel είναι ελεύθερο να γίνει αποστολή. Όταν η `on_wire` είναι ίση ή μεγαλύτερη από την `window` τότε η τιμή της `is_writable` γίνεται False και η αποστολή

σταματάει. Στην `run()` πριν από κάθε εκτέλεση της `asyncore.loop()` ελέγχονται οι μεταβλητές `window` και `on_wire` για να αποφασιστεί η τιμή της `is_writable`. Οι μεταβλητές `on_wire` και `window` αλλάζουν σύμφωνα με τον αλγόριθμο AIMD (ενότητα 3.6) που υλοποιείται στην μέθοδο `read_ack()`.

Η `read_ack()` εκτελείται όταν ληφθεί ένα ACK. Πρώτα γίνεται ο έλεγχος για χαμένα segments και μετά ενημερώνονται με βάση τον αλγόριθμο οι μεταβλητές `window`, `on_wire` και `threshold`. Τα timeouts ελέγχονται με την μέθοδο `check_timeout()` που καλείται στην `run()`. Κάθε φορά που εκτελείται η `send_data()` αποθηκεύεται σε μια μεταβλητή το χρονικό σημείο που έγινε η τελευταία αποστολή. Στην `check_timeout()` υπολογίζεται πόσος χρόνος πέρασε από την τελευταία αποστολή και σε περίπτωση που ξεπερνάει το όριο σημαίνει ότι έχει συμβεί timeout. Όταν ανιχνευθεί timeout οι τιμές των `window` και `on_wire` επιστρέφουν στις αρχικές τους τιμές. Το χρονικό όριο για τα timeouts ορίζεται από την σταθερά TIMEOUT του module shared.py.

5.7 Module receiver.py

Το module receiver.py είναι το αρχείο που εκτελείται για να ξεκινήσει η εφαρμογή receiver. Στο module περιλαμβάνονται οι κλάσεις `ReceiverSession` και `Receiver`.

5.7.1 Κλάση ReceiverSession

Αυτή η κλάση χρησιμοποιείται για ομαδοποίηση των μεταβλητών που αφορούν το session στον receiver ώστε να γίνεται ευκολότερα η διαχείριση τους. Οι μεταβλητές που διαχειρίζεται η κλάση είναι :

filename – Όνομα αρχείου

filesize – Μέγεθος αρχείου

filehash – Το MD5 hash του αρχείου

m_addr – multicast διεύθυνση στην οποία θα ληφθεί το αρχείο

block_cap – Χωριτικότητα του block

seg_size – Το segment size

isset – Δηλώνει αν όλες οι μεταβλητές του session έχουν οριστεί.

5.7.2 Κλάση Receiver

Ο τρόπος λειτουργίας του κώδικα είναι ίδιος με αυτόν στην κλάση `Sender`. Υπάρχει η κεντρική μέθοδος `run()` που εκτελεί την `asyncore.loop()` η οποία παρακολουθεί τα

events των channels και εκτελεί τις ανάλογες μεθόδους για χειρισμό αυτών των events. Οι βασικές μέθοδοι για χειρισμό συμβάντων είναι :

- read_data - Ανάγνωση δεδομένων από το multicast κανάλι
- parse_msg – Ανάγνωση μηνυμάτων από το client channel
- handle_close – Διαχείριση κλεισίματος καναλιού
- handle_multicast_channel_error – χειρισμός σφάλματος στο multicast channel

Κατά την σύνδεση στον sender δημιουργείται ένα αντικείμενο `client_channel` που αναλαμβάνει την επικοινωνία με τον sender και ένα αντικείμενο `multicast_channel` που αναλαμβάνει την λήψη multicast δεδομένων. Όπως και στον sender τα νέα μηνύματα αναλύονται στην `parse_msg()` και καλείται η αντίστοιχη `read_*()` μέθοδος με βάση το TYPE. Όταν ο receiver θέλει να να κάνει join στο session δεν στέλνει κατευθείαν μήνυμα JOIN SESSION. Πρώτα στέλνει το μήνυμα SESSION QUERY και ενεργοποιεί τοπικά το flag `check_and_join`. Όταν ο sender απαντά με το μήνυμα SESSION INFO το επεξεργάζεται η μέθοδος `read_session_info()`. Σε αυτή την μέθοδο ελέγχεται αν είναι ενεργό το flag `check_and_join`. Αν είναι ενεργό τότε πρώτα γίνεται έλεγχος για τον διαθέσιμο χώρο στο αποθηκευτικό μέσο και μετά στέλνεται το μήνυμα JOIN SESSION. Αυτή η διαδικασία γίνεται ώστε ο receiver να έχει πάντα τις πιο πρόσφατες πληροφορίες για το session πριν στείλει αίτηση για συμμετοχή σε αυτό. Όταν ο sender απαντήσει με μήνυμα JOIN ACK, η μέθοδος `read_join_ack()` αρχικοποιεί τις απαραίτητες μεταβλητές για την λήψη του αρχείου. Όταν γίνει αρχικοποίηση όλων τον μεταβλητών στέλνεται το μήνυμα READY στον sender. Το μήνυμα READY προστέθηκε στο πρωτόκολλο ώστε ο sender να μην ξεκινάει να στέλνει πριν να είναι έτοιμος ο receiver γιατί σε περίπτωση που είναι ο prime θα προκαλέσει timeouts στον sender. Η μέθοδος `read_data()` εκτελείται αυτόματα όταν υπάρχουν διαθέσιμα δεδομένα στο multicast channel. Η μέθοδος `read_data()` επεξεργάζεται κάθε segment ξεχωριστά και αν είναι έγκυρο, γράφει τα δεδομένα στο αρχείο. Αν το flag `i_prime` είναι ενεργό σημαίνει ότι ο receiver είναι ο prime και πρέπει να στέλνει ACKS. Η μέθοδος `received_all()` ελέγχει την δομή `to_receive` για να διαπιστώσει αν έχουν ληφθεί όλα τα segments ώστε να σταλεί μήνυμα DONE στον sender.

5.8 Module cmd

Το module `cmd` προσφέρει την λειτουργικότητα του command line interface σε sender και receiver. Οι κλάσεις `Sender` και `Receiver` κληρονομούν την κλάση `Cmd` του module `cmd` η οποία προσφέρει τα εργαλεία για επεξεργασία command line εντολών. Σε αυτές τις κλάσεις εκτελείται ο βρόγχος `cmdloop()` ο οποίος εμφανίζει ένα prompt, δέχεται εντολές και τις ερμηνεύει. Για την δημιουργία μιας εντολής χρειάζεται να οριστεί μια μέθοδος με όνομα `do_<όνομα εντολής>` και μέσα να μπει ο κώδικας που θα εκτελείται όταν θα

δίνεται η εντολή. Επειδή ο βρόγχος `cmdloop()` μπλοκάρει την εκτέλεση του κώδικα, δηλαδή το πρόγραμμα μένει στο `cmdloop()` μέχρι αυτό να κλείσει, για να εκτελεστεί μαζί με την `asyncore.loop()` χρησιμοποιούνται διαφορετικά threads. Το `cmdloop()` τρέχει σε ξεχωριστό thread για να μην εμποδίζει την λειτουργία του `asyncore.loop()`. Η εκτέλεση του του `cmdloop()` γίνεται μόνο όταν δοθεί το option -i κατά την έναρξη της εφαρμογής. Η εξ' ορισμού λειτουργία είναι με ένα thread και χωρίς command line interface.

6 ΠΑΡΑΤΗΡΗΣΕΙΣ / ΣΧΟΛΙΑ

Παρακάτω σχολιάζονται κάποιες σχεδιαστικές αποφάσεις που πάρθηκαν στα πρώτα στάδια της υλοποίησης αλλά τελικά δεν υιοθετήθηκαν.

6.1 Αρχικό μοντέλο ανατροφοδότησης

Πριν η εφαρμογή πάρει την τελική της μορφή έγιναν αρκετές δοκιμές με διαφορετικά σχήματα αποστολής και μορφές ανατροφοδότησης. Αρχικά γινόταν αποστολή δεδομένων με λήψη ACKs από όλους τους receivers και υπήρχε ξεχωριστή δομή για κάθε receiver η οποία κρατούσε τα segments που πρέπει να σταλούν στον καθένα. Με αυτόν τον τρόπο δεν μπορούσε να γίνει σωστή ρύθμιση του ρυθμού αποστολής επειδή τα ACKs λαμβάνονταν από όλους τους receivers. Επίσης υπήρχε μεγαλύτερη επιβάρυνση του sender γιατί έπρεπε να διαχειριστεί μεγάλο αριθμό ACKs και κρατούσε ξεχωριστή δομή για κάθε receiver με αποτέλεσμα να μειώνεται η απόδοση όσο μεγάλωνε ο αριθμός των receivers. Στην συνέχεια προστέθηκε ένα αναγνωριστικό για τον prime receiver ώστε μόνο τα ACKs από αυτόν να λαμβάνονται υπόψιν για την ρύθμιση του παραθύρου συμφόρησης. Η χρήση του prime οδήγησε σε έναν πιο σωστό υπολογισμό του ρυθμού αποστολής αλλά πάλι υπήρχε επιβάρυνση όταν μεγάλωνε ο αριθμός των receivers.

Σε εκείνο το σημείο οι receivers δεν γνώριζαν εξαρχής τι έχουν να λάβουν και απλά έστελναν ACKs για κάθε segments που λάμβαναν και πολλές φορές επεξεργάζονταν segments που είχαν λάβει πριν αλλά ο sender τα έστελνε ξανά γιατί δεν τα είχε λάβει κάποιος άλλος receiver. Η ανεξέλεγκτη αποστολή ACKs για κάθε segment δημιουργούσε πρόβλημα και στον sender γιατί δεν μπορούσε να γίνει σωστός εντοπισμός χαμένων segments.

Ο sender κρατούσε ξεχωριστή δομή για κάθε receiver γιατί δεν είχε υλοποιηθεί ακόμα η αποστολή NACKs και οι receivers δεν μπορούσαν να τον πληροφορήσουν σχετικά με το ποια segments τους λείπουν. Ο μόνος τρόπος για να γνωρίζει ο sender την κατάσταση των receivers ήταν να λαμβάνει ACKs από όλους και να κρατάει ξεχωριστή δομή με χαμένα segments για τον κάθε ένα.

Σε επόμενη υλοποίηση προστέθηκε η δυνατότητα αποστολής NACKs και πλέον οι receivers είχαν μια δομή όπου κρατούσαν τα segments που έχουν να λάβουν. Στην αρχή αποστολή STATUS REQUEST για λήψη NACKs γινόταν στο τέλος ενός περάσματος και όχι

κατά την αλλαγή prime. Στη συνέχεια αφαιρέθηκαν οι δομές που κρατούσε ο sender για τους υπόλοιπους receivers καθώς η αποστολή γινόταν μόνο με βάση τα χαμένα segments του prime.

6.2 Αναφορά σε segments

Στις πρώτες εκδόσεις η διαχείριση του αρχείου γινόταν από μια ενιαία δομή χωρίς την ύπαρξη blocks. Η αναφορά σε κάθε segment γινόταν με την βοήθεια δυο μεταβλητών **from** και **to**. Το from ήταν το byte από το οποίο άρχιζε το segment και το to ήταν το byte στο οποίο τελείωνε το segment. Η μόνη ευκολία που υπήρχε από την χρήση του συνδυασμού from-to ήταν η άμεση αναφορά που μπορούσε να γίνει σε συγκεκριμένα segments του αρχείου χωρίς να χρειάζεται να γίνει δυναμικά ο υπολογισμός του byte από το οποίο ξεκινάει το segment για να γίνει ανάγνωση ή εγγραφή του. Η χρήση δυο μεταβλητών για αναφορά σε ένα segment έκανε δύσκολο τον εντοπισμό χαμένων τμημάτων και την διαχείριση της λίστας που τα περιείχε. Επίσης επειδή οι τιμές των from-to ήταν πολύ μεγάλες λόγο της άμεσης αναφοράς σε συγκεκριμένα bytes του αρχείου και η δομή ήταν ενιαία για όλο το αρχείο, αυτή η δομή δέσμευε περισσότερους πόρους από το σύστημα.

6.3 Αποστολή NACKs

Στην αρχή η δομή not_acked στον sender και η δομή to_receive στον receiver δεν χρησιμοποιούσαν bit masks αλλά λίστες για την διαχείριση των segments. Όταν έπρεπε να σταλεί NACK για ένα block η λίστα για το συγκεκριμένο block μετατρέπονταν σε ένα στιγμιότυπο της κλάσης Bitmask και στην συνέχεια γινόταν αποστολή του mask που υπήρχε σε αυτό το στιγμιότυπο. Στην μεριά του sender γινόταν η αντίθετη διαδικασία. Από το mask που υπήρχε στο NACK γίνονταν η δημιουργία ενός αντικειμένου Bitmask και ύστερα με την μέθοδο Bitmask.to_list() γινόταν ξανά μετατροπή σε λίστα.

Η παραπάνω διαδικασία είχε αρνητικό αντίκτυπο στην αποστολή όταν γινόταν μαζική αποστολή NACKs κατά την αλλαγή prime. Κατά την αποστολή ενός μεγάλου αρχείου όταν γίνεται prime ένας receiver που έχει κάνει late join έχει να στείλει έναν μεγάλο αριθμό από NACKs. Κατά την μαζική λήψη NACKs ο sender αφιέρωνε αρκετό χρόνο για την επεξεργασία τους με αποτέλεσμα να επηρεάζεται αρνητικά η αποστολή και να προκαλούνται timeouts.

Για να αντιμετωπιστεί αυτό το πρόβλημα οι λίστες αντικαταστάθηκαν από αντικείμενα Bitmask και τα NACKs πλέον περιλαμβάνουν ένα serialized αντικείμενο Bitmask. Στον sender γίνεται deserialization του αντικειμένου Bitmask το οποίο αντιγράφεται κατευθείαν στην δομή not_acked. Ο χρόνος που χρειάζεται για να γίνει serialization και deserialization είναι μηδαμινός σε σχέση με τις μετατροπές που έπρεπε να γίνουν πριν. Με την άμεση διαχείριση αντικειμένων Bitmask ο sender μπορεί να ανταποκριθεί στην επεξεργασία μεγάλου αριθμού NACKs χωρίς να επηρεάζεται η αποστολή.

7 ΕΠΙΛΟΓΟΣ / ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΒΕΛΤΙΩΣΗ

Η τελική εφαρμογή που υλοποιήθηκε στα πλαίσια της πτυχιακής πετυχαίνει τους βασικούς στόχους που τέθηκαν στην αρχή. Κατά την αποστολή προς πολλούς παραλήπτες η εφαρμογή εντοπίζει και καλύπτει τις ανάγκες των παραληπτών σε χαμένα segments ώστε να λάβουν όλοι με επιτυχία το αρχείο. Ο ρυθμός αποστολής ρυθμίζεται δυναμικά με βάση το feedback που λαμβάνει ο sender από τον prime receiver. Receivers μπορούν να συνδέονται ενώ η αποστολή βρίσκεται σε εξέλιξη και να λαμβάνουν στην πορεία τα δεδομένα που έχασαν στην αρχή. Το μεγαλύτερο μέρος του αρχείο στέλνεται μόνο μια φορά με χρήση multicast και στην συνέχεια γίνεται επαναποστολή μόνο των επιμέρους κομματιών που δεν παραδόθηκαν κατά την πρώτη μετάδοση.

Οι δύο βασικοί τομείς που δεν αναλύθηκαν είναι η ασφάλεια και η βελτιστοποίηση της ταχύτητας λειτουργίας της εφαρμογής. Από την μεριά της ασφάλειας σε μελλοντικές εκδόσεις μπορεί να προστεθεί υποστήριξη για κρυπτογράφηση των δεδομένων και των μηνυμάτων που στέλνονται σε μια συνεδρία. Επίσης μπορούν να προστεθούν ειδικά αναγνωριστικά στον sender και στους receivers ώστε να γίνεται ταυτοποίηση του αποστολέα πριν εξεταστεί ένα μήνυμα.

Σε δοκιμές που έγιναν παρατηρήθηκε ότι σε κάποιες περιπτώσεις δεν γίνεται βέλτιστη χρήση του throughput. Πιο συγκεκριμένα σε δίκτυο χωρητικότητας 1 Gbit/s παρατηρήθηκε ότι η ταχύτητα εκτέλεσης της εφαρμογής περιορίζει την ταχύτητα αποστολής με αποτέλεσμα να μην αξιοποιείται πλήρως το διαθέσιμο bandwidth. Αυτό οφείλεται εν μέρη στην γλώσσα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής και στους αλγορίθμους λειτουργίας του κώδικα. Η βελτιστοποίηση του κώδικα μπορεί να οδηγήσει σε καλύτερα αποτελέσματα με χρήση της ίδιας γλώσσας (python) αλλά για ακόμα καλύτερες επιδόσεις πρέπει να γίνει χρήση μιας πιο γρήγορης γλώσσας προγραμματισμού όπως η C με αντάλλαγμα μια πιο πολύπλοκη υλοποίηση από πλευράς κώδικα.

8 MANUAL

8.1 Sender

Για να λειτουργήσει ο sender πρέπει κατ ελάχιστο να οριστεί το αρχείο και η παράμετρος για ενεργοποίηση του session. Οι υπόλοιπες παράμετροι όπως για παράδειγμα η multicast IP και τα port numbers έχουν εξ' ορισμού τιμές οι οποίες μπορούν να αλλάξουν μόνο αν είναι απαραίτητο. Για να ξεκινήσει ο sender με υποστήριξη cli πρέπει να δοθεί μόνο η παράμετρος -i και όλες οι υπόλοιπες ρυθμίσεις μπορούν να γίνουν μέσω του cli.

Χρήση: ./sender.py -f FILE -a

Παράμετροι:

-h, –help Εκτύπωση λίστας με όλες τις παραμέτρους και τη χρήση τους.

-m MADDRESS, –maddress=MADDRESS Επιλογή multicast IP διεύθυνσης στην οποία θα γίνει η αποστολή των δεδομένων. Η εξ' ορισμού διεύθυνση είναι 227.0.0.1.

-f FILE, –file=FILE Ορισμός διαδρομής του αρχείου προς αποστολή. Η διαδρομή που δίνεται μπορεί να είναι απόλυτη ή σχετική. Αυτή η παράμετρος είναι υποχρεωτική όταν η εφαρμογή ξεκινάει χωρίς υποστήριξη command line interface.

-a, –activate Ενεργοποιεί αυτόματα το session κατά την εκκίνηση του sender. Αυτή η παράμετρος έχει ισχύ μόνο όταν δίνεται σε συνδυασμό με την -f επειδή για την ενεργοποίηση του session πρέπει πρώτα να έχει οριστεί το αρχείο προς αποστολή.

-t TTL, –ttl=TTL Ορισμός του TTL για τα multicast πακέτα. Καθορίζει την εμβέλεια των multicast πακέτων. Οι τιμές που καθορίζουν την εμβέλεια είναι :

0 – restricted to the same host

1 – restricted to the same subnet

32 – restricted to the same site

64 – restricted to the same region

128 – restricted to the same continent

225 – unrestricted

-i, –interactive Η εφαρμογή ξεκινάει με υποστήριξη command line interface.

-d, –debug Ενεργοποιεί την εμφάνιση μηνυμάτων αποσφαλμάτωσης κατά την λειτουργία της εφαρμογής.

-g, –plot Ενεργοποιείται η καταγραφή της διακύμανσης του window και του rtt σε αρχείο ώστε να μπορεί να γίνει αναπαράσταση με τη μορφή διαγράμματος.

-p MPORT, –mport=MPORT Ορίζει το multicast port το οποίο θα χρησιμοποιηθεί σε συνδυασμό με τη multicast IP για αποστολή των δεδομένων. Η προκαθορισμένη τιμή είναι 50005.

-P SERVERPORT, –serverport=SERVERPORT Ορισμός του port στο οποίο ο server θα δέχεται συνδέσεις. Η προκαθορισμένη τιμή είναι 50000.

8.2 Receiver

Για να λειτουργήσει ο receiver πρέπει κατ ελάχιστο να δοθεί η διεύθυνση του sender και οι παράμετροι -a, -c για να πραγματοποιηθεί αυτόματη σύνδεση και είσοδος σε ενεργό session στον sender. Γι να ξεκινήσει ο receiver με υποστήριξη cli πρέπει να δοθεί μόνο η παράμετρος -i και όλες οι υπόλοιπες ρυθμίσεις μπορούν να γίνουν μέσω του cli.

Χρήση: ./receiver.py -ac -s SERVERADDR

Παράμετροι:

-h, –help Εκτύπωση λίστας με όλες τις παραμέτρους και τη χρήση τους.

-d, –debug Ενεργοποιεί την εμφάνιση μηνυμάτων αποσφαλμάτωσης κατά την λειτουργία της εφαρμογής.

-g, –plot Ενεργοποιείται η καταγραφή της διακύμανσης του window και του rtt σε αρχείο ώστε να μπορεί να γίνει αναπαράσταση με τη μορφή διαγράμματος.

-s SERVERADDR, –serveraddr=SERVERADDR Ορισμός της IP διεύθυνσης του sender. Αυτή η παράμετρος είναι υποχρεωτική όταν η εφαρμογή ξεκινάει χωρίς υποστήριξη command line interface.

-p SERVERPORT, –serverport=SERVERPORT Ορισμός του port στο οποίο ο sender δέχεται συνδέσεις. Η εξ' ορισμού τιμή είναι 50000.

-m MPORT, –mport=MPORT Ορισμός του multicast port στο οποίο ο receiver περιμένει για δεδομένα. Η εξ' ορισμού τιμή είναι 50005.

-c, –connect Ο receiver συνδέεται αυτόματα στον sender κατά την εκκίνηση. Αυτή η επιλογή είναι υποχρεωτική όταν η εφαρμογή ξεκινάει χωρίς υποστήριξη command line interface και πρέπει δίνεται σε συνδυασμό με την -s.

-a, –autojoin Κάνει τον receiver να στείλει αυτόματα μήνυμα JOIN SESSION κατά την σύνδεση του στον sender. Αν υπάρχει ενεργό session στον sender αυτή η επιλογή αυτοματοποιεί την συμμετοχή του receiver στο session. Αυτή η επιλογή είναι υποχρεωτική όταν η εφαρμογή ξεκινάει χωρίς υποστήριξη command line interface και δίνεται σε συνδυασμό με τις -c, -s.

-i, –interactive Η εφαρμογή ξεκινάει με υποστήριξη command line interface.

-D, –directory=DIRECTORY Η διαδρομή στην οποία θα αποθηκευτεί το αρχείο.

8.3 Λειτουργία με cli

Όταν η εφαρμογή εκτελείται με υποστήριξη command line interface (cli) ο χρήστης έχει στη διάθεση του μια σειρά εντολών που του επιτρέπουν να ρυθμίζει πιο ευέλικτα τις παραμέτρους λειτουργίας της εφαρμογής. Η εφαρμογή εμφανίζει ένα prompt στο οποίο ο χρήστης μπορεί να εισάγει εντολές όπως στην εικόνα 8.1. Η εντολή help εμφανίζει μια λίστα με τις διαθέσιμες εντολές και μπορεί να πάρει ως παράμετρο το όνομα μιας εντολής για να εμφανίσει περισσότερες πληροφορίες για αυτήν.

8.3.1 Sender

Παρακάτω παρατίθενται οι εντολές που υπάρχουν διαθέσιμες στον sender και περιγράφεται η λειτουργία τους.

listchannels Εμφανίζει τη λίστα τους receivers που είναι συνδεδεμένοι στον sender. Οι receivers σε αυτή τη λίστα δεν είναι απαραίτητα μέλη του session.

exit Κλείνει την εφαρμογή sender.

session [options] Είναι η εντολή που χρησιμοποιείται για την διαχείριση του session στον sender. Η εντολή session παίρνει επιπλέον παραμέτρους που ρυθμίζουν τις διάφορες μεταβλητές του session.

- **-f FILE, –file=FILE** Ορισμός διαδρομής του αρχείου προς αποστολή. Η διαδρομή που δίνεται μπορεί να είναι απόλυτη ή σχετική.
- **-a, –activate** Ενεργοποιεί το session.
- **-d, –deactivate** Απενεργοποιεί το session.
- **-i, –info** Εκτυπώνει τις πληροφορίες του session.
- **-r, –reset** Μηδενίζει όλες τις παραμέτρους του session.
- **-m MULTICAST, –multicast=MULTICAST** Επιλογή multicast IP διεύθυνσης στην οποία θα γίνει η αποστολή των δεδομένων.
- **-l, –listconnected** Εμφανίζει μια λίστα με τους receivers που είναι μέλη του session και λαμβάνουν το αρχείο.

8.3.2 Receiver

Παρακάτω παρατίθενται οι εντολές που υπάρχουν διαθέσιμες στον receiver και περιγράφεται η λειτουργία τους. Στον receiver η μορφή του prompt αλλάζει ανάλογα με το αν είναι συνδεδεμένος ή όχι στον sender (εικόνα 8.2). Κάποιες εντολές δεν μπορούν να εκτελεστούν πριν τη σύνδεση στον sender και η εναλλαγή του prompt βοηθάει τον χρήστη να γνωρίζει αν ο receiver είναι συνδεδεμένος ώστε να εκτελεί μόνο συγκεκριμένες εντολές.

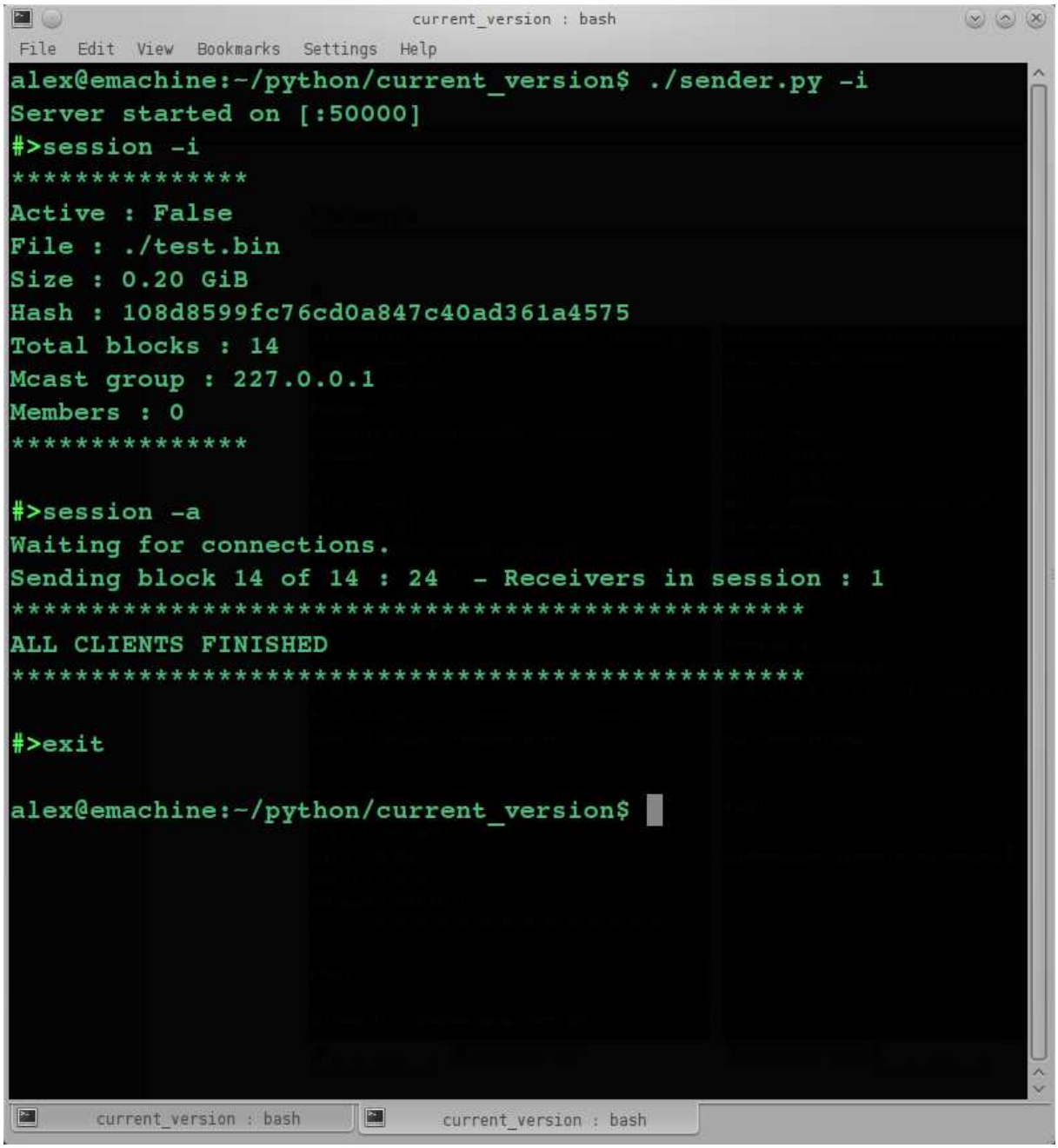


Figure 8.1: Λειτουργία cli στον sender

connect Πραγματοποιεί σύνδεση στον sender.

disconnect Πραγματοποιεί αποσύνδεση από τον sender.

exit Τερματίζει την εφαρμογή receiver.

server [ip | **hostname]** Ρυθμίζει την IP διεύθυνση του sender στην οποία θα προσπαθήσει να συνδεθεί ο receiver κατά το connect.

setdir [path] Ορίζει την διαδρομή στην οποία θα γίνει η αποθήκευση του αρχείου. Η διαδρομή που θα δοθεί μπορεί να είναι απόλυτη ή σχετική.

stat Εμφανίζει την διαδρομή στην οποία θα αποθηκευτεί το αρχείο και τον διαθέσιμο χώρο που υπάρχει.

session Στέλνει ένα μήνυμα SESSION QUERY στον sender και αν υπάρχει ενεργό session εμφανίζει τα στοιχεία του.

join Ο receiver ζητά να γίνει μέλος του ενεργού session στον sender.

leave Ο receiver ζητά να βγει από το session στο οποίο είναι μέλος.

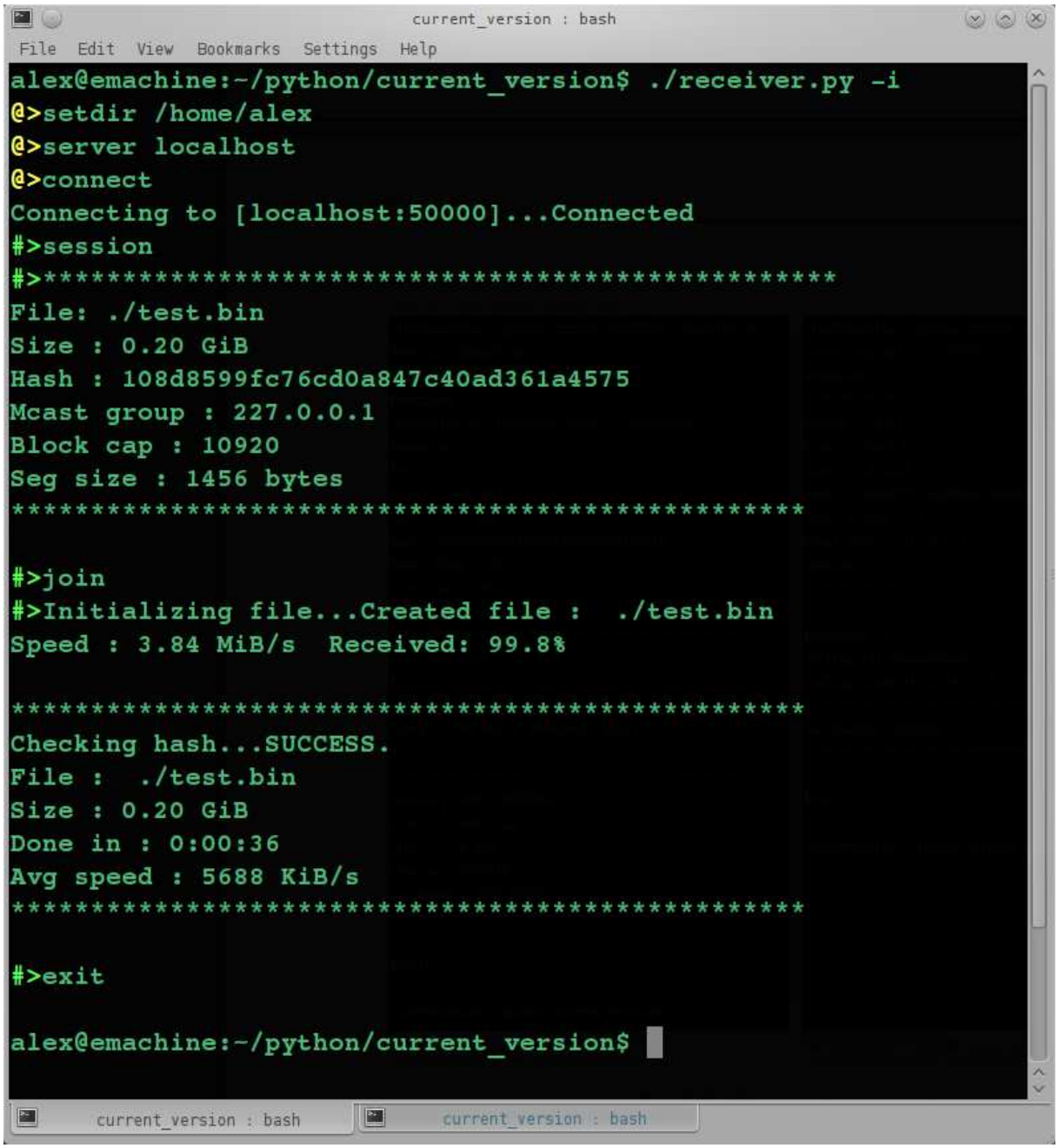


Figure 8.2: Λειτουργία cli στον receiver

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999. Obsoleted by RFC 5681, updated by RFC 3390.

[2] Douglas Comer, editor. *Internetworking with TCP/IP - Principles, Protocols, and Architectures, Fourth Edition*. Prentice-Hall, 2000.

[3] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, and M. Luby. The Reliable Multicast Design Space for Bulk Data Transfer. RFC 2887 (Informational), August 2000.

[4] James F. Kurose and Keith W. Ross. Computer networking: A top-down approach featuring the internet. *Addison Wesley*, page 688, 2001.

[5] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[6] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[7] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959 (Standard), October 1985. Updated by RFCs 2228, 2640, 2773, 3659, 5797.

[8] B. Quinn and K. Almeroth. IP Multicast Applications: Challenges and Solutions. RFC 3170 (Informational), September 2001.

[9] M. Watson, M. Luby, and L. Vicisano. Forward Error Correction (FEC) Building Block. RFC 5052 (Proposed Standard), August 2007.

[10] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M. Luby. Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer. RFC 3048 (Informational), January 2001.