Project Part 2

Δίκτυα Υπολογιστών

Εαρινό Εξάμηνο 2018

Γ. Δ. Σταμούλης

Ομάδα 9

Κατωπόδης Αντώνιος 3140076

Κουτσοπούλου Αθανασία Μαρία 3140092

Χασακής Διονύσιος 3140219

Ανάλυση Υλοποίησης

Σκοπός του project είναι η δημιουργία ενός προγράμματος που χρησιμοποιεί UDP τεχνολογίες αλλά μεταφέρει αξιόπιστα τα αρχεία, ως παραλλαγή του TCP.

Για αυτόν τον σκοπό χρησιμοποιήσαμε τις DatagramSocket και DatagramPacket της Java, προκειμένου να υλοποιήσουμε εμείς οι ίδιοι τους διάφορους ελέγχους στην επικοινωνία Server – Client.

Όταν ανοίγουμε τον Server δίνουμε ως είσοδο την IP και το port του. Σε αυτό το port δημιουργείται το default socket στο οποίο ακούει ο Server, και όπου στέλνει κάθε client την πρώτη φορά. Δηλαδή κάθε client στέλνει το πρώτο πακέτο το «SYN» για να ξεκινήσει το 3-Way-Handshake σε αυτό το port.
Όταν ο Server «ακούσει» από το default socket, ελέγχει αρχικά εάν το flag του datagramPacket έχει την τιμή Packet.N0_DATA_PACKET που υποδηλώνει ότι το datagramPacket δεν μεταφέρει κάποιο data, αλλά ο σκοπός του είναι να πραγματοποιηθεί το Handshake. Εάν το flag δεν έχει αυτήν την τιμή, ο server αγνοεί το datagramPacket, αφού πρόκειται για πακέτο data που επιχείρησε να στείλει κάποιος client χωρίς να έχει εγκαθιδρύσει σύνδεση με τον server μέσω ενός επιτυχημένου Handshake. Εάν από την άλλη είναι το flag είναι πράγματι Packet.NO_DATA_PACKET τότε ξεκινά ένα νέο thread που θα είναι υπεύθυνο για την εξυπηρέτηση της προκείμενης συνεδρίας με τον Client αυτό.

Κάθε φορά που δημιουργεί ο Server νέο thread για την συγκεκριμένη μεταφορά από τον συγκεκριμένο Client, αναθέτει και νέο DatagramSocket μέσω του οποίου θα επιτυγχάνεται η επικοινωνία. Με αυτόν τον τρόπο ο Server συνεχίζει να ακούει για άλλους Client στο default port. Ο Server-thread στέλνει το SYN/ACK απάντηση πίσω στον Client, όπου περιέχει το νέο port που χρησιμοποιείται από το socket που δημιούργησε ο Server για τον Client, και το οποίο περιμένει να επιβεβαιώσει ο Client ότι το έλαβε. Έχουμε θέσει timeout, όπου αν λήξη ξαναστέλνει το SYN/ACK μέχρι 3 φορές. Εάν δεν λάβει απάντηση από τον Client τότε ενημερώνει για τον τερματισμό του, και κλείνει το thread και το DatagramSocket. Για να αποδεχτεί το πακέτο που λάβει ως την απάντηση του Client θα πρέπει το datagramPacket εκτός από το σωστό SEQ να έχει και flag ίσο με NO_DATA_PACKET. Όταν λοιπόν λάβει την απάντηση ACK από τον Client, τότε «εγκαθιδρύεται» η επικοινωνία μέσω του νέου DatagramSocket ενημερώνουν τόσο ο Client όσο και ο Server ότι Handshake έγινε και ο Server αναμένει την μεταφορά του αρχείου.

Σε αυτό το σημείο ας μιλήσουμε λίγο για τον Client. Όταν ξεκινά ο Client του δίνουμε την IP και το port του Server καθώς επίσης το αρχείο (ονομα.κατάληξη) που επιθυμούμε να μεταφέρουμε στον Server και το path του directory στο οποίο βρίσκεται, και τέλος το μέγεθος του payload. Αρχικά ο client θα διαβάσει όλο το αρχείο σε ένα array από byte και στην συνέχεια θα φτιάξει ένα Hashmap με όλα τα διαφορετικά payload των πακέτων που θα χρειαστεί να στείλει για να μεταφερθεί το αρχείο. Τα keys του hashmap είναι ένας counter ουσιαστικά που κρατά την σειρά των payload με την σειρά που αν ενωθούν δίνουν το αρχικό data byte buffer του αρχείου. Εξαίρεση είναι το πρώτο στοιχείο του HashMap με key -1 όπου περιέχει το όνομα του αρχείου (με την κατάληξη) αποθηκευμένο σε byte array.

Έτσι λοιπόν όταν επιτύχει το Handshake, αφού περιμένει λίγο ο Client (χρήση thread sleep) ξεκινά να στέλνει έναν ένα τα values του Hashmap με τα payloads, αφού τα μετατρέψει πρώτα σε datagramPacket. Το datagramPacket έχει την εξής μορφή:

Header 3 bytes
{
όπου Header: 1 byte flags → όπου στα 3 τελευταία bits έχουμε με την σειρά το SEQ NUMBER, FLAG NO DATA PACKET, FLAG NOT LAST PACKET
2 bytes payload size
}
Payload x bytes
CheckSum 8 bytes

Αφού το πρωτόκολλο που υλοποιήσαμε είναι το Stop and Wait, τόσο για το SEQ number των πακέτων όσο και για το ACK number αρκεί ένα bit 0 ή 1.
Όμοια το flag No Data Packet είναι μηδέν αν είναι πακέτο που μεταφέρει file data, διαφορετικά (αν χρησιμοποιείται για handshake) είναι 1.
Τέλος, το flag Not Last Packet, είναι 1 αν είναι τελευταίο πακέτο, αλλιώς είναι μηδέν.
Σημείωση: αν ένα πακέτο είναι No Data Packet τότε πάντα το Not Last Packet είναι μηδέν, διότι πολύ απλά δεν θα του δώσει ποτέ ο Client τιμή όταν στέλνει No Data Packet. Έτσι, για να σταλθεί στον Server ένα Termination "signal", αντί να προσθέσουμε νέο flag που θα γινόταν 1 μόνο στην περίπτωση αυτή, αποφασίσαμε να χρησιμοποιήσουμε ως «flag Termination» και τα τρία flags μαζί. Έτσι, όταν θέλουμε να στείλουμε Termination signal στέλνουμε τα τρία bits 111.

πχ. Αν ένα header έχει flags 0000 00101 τότε συμπεραίνουμε:
SEQ NUMBER = 1
FLAG NO DATA PACKET = 0
FLAG NO LAST PACKET = 1

Προκυμμένου να διευκολυνθεί η διαχείριση των datagram packet, δημιουργήσαμε νέα κλάση για το μοντέλο των πακέτων που στέλνουμε. Έτσι ο Client κάθε φορά έχοντας υπόψιν αυτή, δημιουργεί για το εκάστοτε payload το αντίστοιχο byte array προσθέτοντας μπροστά από το payload το header και μετά από το payload το CheckSum. Μέσω αυτού του array δημιουργεί το datagram packet και το στέλνει στον Sever μέσω του νέου port που έχει οριστεί για την επικοινωνία Client-Server αυτή.
Όταν ο server λάβει το datagram packet θα μετατρέψει το data του datagram σε ένα πακέτο από την κλάση Packet του μοντέλου μας και μετά με βάση αυτό θα κάνει διάφορους ελέγχους για να αποφανθεί για την συνέχεια.
Εάν το πακέτο είναι corrupt το αγνοεί.
Εάν το πακέτο είναι valid αλλά δεν έχει τον αναμενόμενο seq number τότε το κάνει drop γιατί είναι duplicate και κάνει retransmit το τελευταίο ACK mesage.
Εάν είναι valid και σωστό seq number, την πρώτη φορά γνωρίζουμε ότι το πακέτο αφορά τον τίτλο του αρχείου (μαζί με την κατάληξη), επομένως δημιουργεί το file με το όνομα που βρίσκεται στο payload.

Όλες τις υπόλοιπες φορές, το πακέτο μεταφέρει μέρος του file, οπότε το γράφει το payload ως byte στο file που άνοιξε.

Σημείωση, ο Client κάθε φορά έχει ένα timeout μέχρι να λάβει το ACK με το σωστό ACK number. Αν το timeout λήξει κάνει retransmit.

Τέλος, γίνεται έλεγχος τόσο στα πακέτα που στέλνει ο Client στον Server, όσο και στα ACK που στέλνει ο Server.
Για τα πακέτα κάναμε έλεγχο μέσω CheckSum. Το οποίο υπολογίζει ο Client το βάζει στο τέλος του πακέτου μετά το payload και μετά ο Server υπολογίζει πάλι το CheckSum με βάση το πακέτο που έλαβε (αφαιρώντας το CheckSum) και αν προκύψει το ίδιο με αυτό που είναι στο τέλος του πακέτου τότε θεωρείται σωστό – valid πακέτο.
Για τα ACK, καθώς στέλνουμε μονάχα 1 bit κάτι τέτοιο δεν είχε νόημα. Επομένως για αυτά, αποφασίσαμε, ότι αντί να στέλνουμε 1 bit, θα στέλνουμε 3 με τον ίδιο αριθμό. Έτσι ο Client αν δει ότι το ACK δεν αποτελείται από τρία ίδια bits θεωρεί ότι έχει γίνει κάποιο λάθος, και το αγνοεί.

Περιγραφή και Τεκμηρίωση

Αρχικά έχουμε τον BaseServer η οποία είναι μία abstract class από την οποία κληρονομούν ο Client και ο Server. Ο Server δέχεται ένα IP και ένα port στο οποίο θα ακούει για connections. Ο Client παίρνει ως είσοδο το IP και το port του Server και ξεκινάει να κάνει το 3-way handshake με τον Server. Ο Server παίρνει το πρώτο μήνυμα στο main thread και έπειτα ξεκινά ένα νέο thread και ανοίγει ένα νέο socket με το οποίο ο client θα επικοινωνεί μαζί του. Έτσι το συγκεκριμένο thread θα ακούει στο ανοιγμένο socket, από και μέσω του οποίου θα επικοινωνεί με τον client. Με το που ανοίγει ένα νέο socket, ο client ενημερώνεται και πλέον δεν επικοινωνεί στο main port. Ο λόγος είναι για να πετύχουμε παράλληλα connections στον Server και όσο το δυνατόν παραλληλλοποίηση. Οπότε το main socket μένει ανοιχτό και διαχειρίζεται μόνο νέα connections, όλα τα υπόλοιπα τα αγνοεί, καθώς δεν είναι αρμόδιο να τα διαχειριστεί. Κάθε φορά και ο client και ο server κατασκευάζουν ένα αντικείμενο Packet το οποίο εμπεριέχει όλες τις πληροφορίες οι οποίες θα αποθηκευτούν στον buffer για πιο εύκολη διαχείριση. Τέλος, η GUIClient τρέχει το Java Swing GUI μέσω του οποίου ο χρήστης μπορεί να δώσει IP και Port στον Server, να τον εκκινήσει και να τον διακόψει, να ανοίξει ένα νέο client, να ορίσει το IP και το Port του Server, το payload size αλλά και να επιλέξει το αρχείο που θέλει να μεταφερθεί. Το πρόγραμμα έχει τη δυνατότητα να τρέξει και από command line ξεκινώντας τον ServerSpawner και ClientSpawner αντίστοιχα και είτε δίνοντας τιμές στα args είτε γράφοντας στο command line.

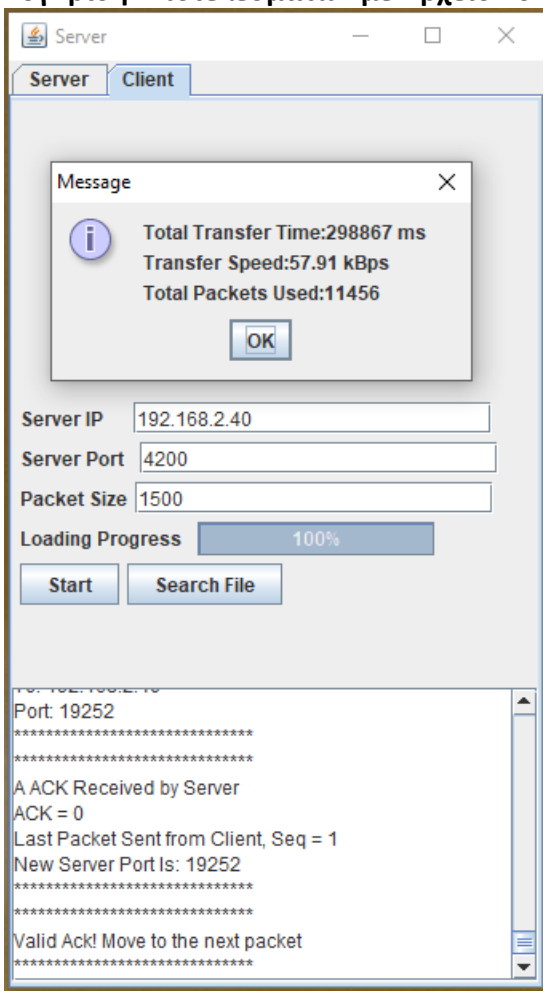
Για τη μεταγλώττιση του προγράμματος θα πρέπει να ανοίξουμε το project με το IntelliJ IDEA και να τρέξουμε τις αντίστοιχες κλάσεις που επιθυμούμε, είτε να το κάνουμε import σε eclipse/netbeans. Υπάρχει και η δυνατότητα να τρέξουμε τις κλάσεις που επιθυμούμε και από command line, αλλά λόγω διαφορετικών πακέτων αυτή η διαδικασία είναι πιο περίπλοκή.

Γενικότερα Σχόλια

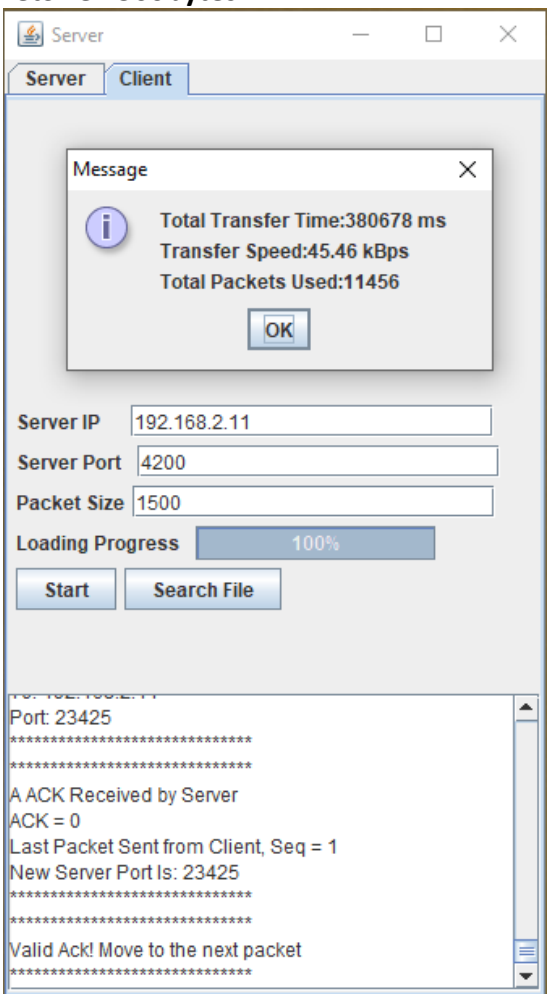
Διαχείριση των connection των Client – Server και το πως να κάνουμε τον Server να ανοίγει ένα Thread που επικοινωνεί μέσω ενός ξεχωριστού Socket με τον κάθε Client, με αποτέλεσμα πολλοί Client να μπορούν να επικοινωνούν με τον Server παράλληλα.

Μία εναλλακτική που ξεκινήσαμε στην υλοποίηση της εργασίας μας ήταν αρχικά χρησιμοποιούσε 1 ξεχωριστό byte για κάθε ένα από τα τρία flags του header, και χρησιμοποιούσε 4 bytes για το payload size. Όμως στην συνέχεια αντιληφθήκαμε ότι ήταν άσκοπη «σπατάλη» αφού η ίδια λειτουργία μπορούσε να επιτευχθεί με λιγότερα bytes ανά datagram packet. Επιπλέον, εναλλακτική ήταν ο τρόπος χειρισμού του handshake, όπου δεν είχαμε βάλλει τον server να κλείνει το connection αν δεν λάβει το ACK στο SYN/ACK, αλλά τελικά καταλήξαμε ότι ήταν πιο ορθή η νέα μας υλοποίηση για να έχει γίνει ολοκληρωμένα το connection establish.

Σύγκριση Αποτελεσμάτων με Αρχείο 16Mb, PacketSize 1500bytes

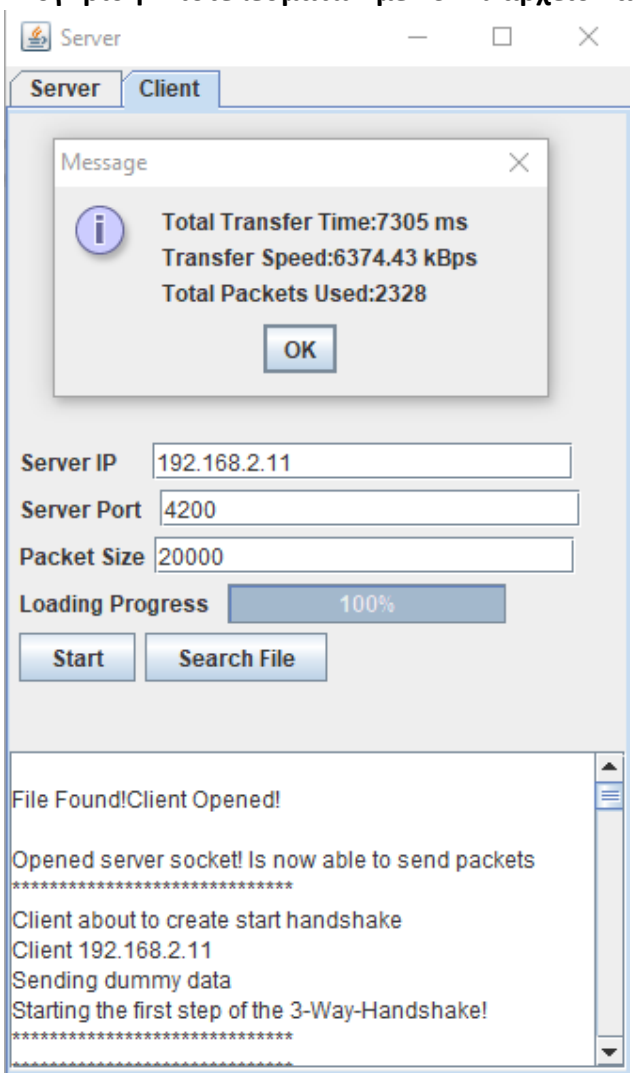


Σε Διαφορετικά PC

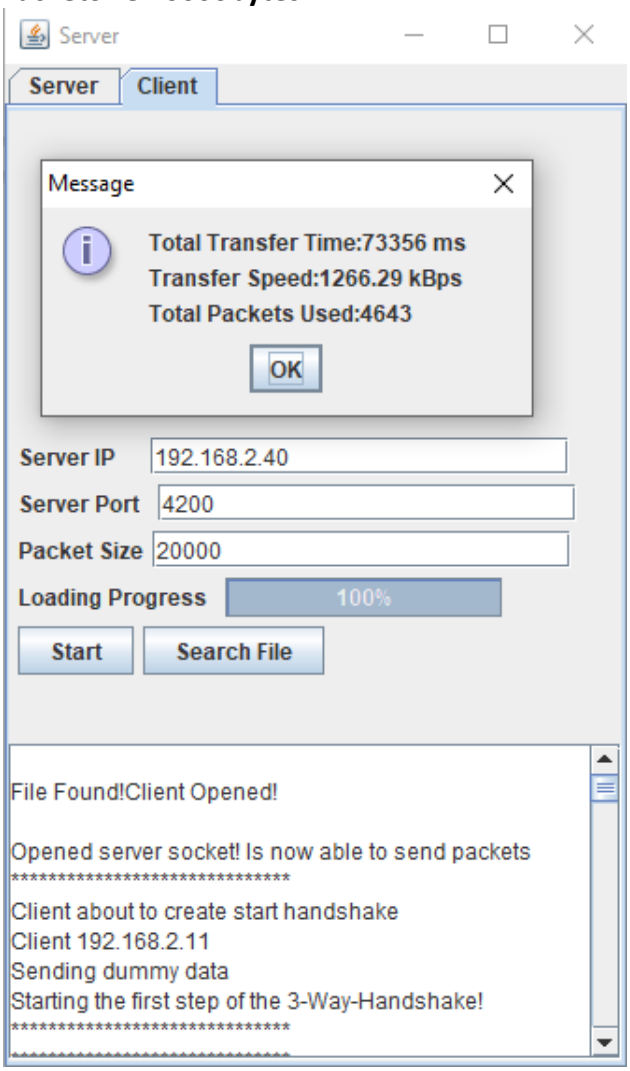


Στο ίδιο PC

Σύγκριση Αποτελεσμάτων με 43Mb αρχείο και PacketSize 20000bytes



Στο ίδιο PC



Σε Διαφορετικά PC

Παρατηρούμε πως στη περίπτωση που έχουμε μικρό μέγεθος πακέτων ο χρόνος έχει απόκλιση γιατί τα πακέτα είναι μικρά και πολλά και υπάρχει κάποιο delay(lock i/o) στον ίδιο δίσκο με αποτέλεσμα τη μεγαλύτερη καθυστέρηση, ενώ στην δεύτερη περίπτωση είναι πιο εμφανής η διαφορά καθώς πολύ περισσότερα πακέτα θα γίνουν drop.

Παρατήρηση: Για την έναρξη του σέρβερ θα πρέπει να αποφεύγεται η χρήση localhost όταν πρόκειται να σταλθεί πακέτο από τον ίδιο τον υπολογιστή, αλλά η χρήση της διεύθυνσης NAT.