**ΕΡΓΑΣΙΑ ΔΙΚΤΥΑΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ**

**ΦΟΙΤΗΤΗΣ: ΣΩΤΗΡΙΟΣ ΦΤΙΑΚΑΣ**

**ΑΕΜ: 3076**

**ΠΕΡΙΕΧΟΜΕΝΑ**

Εισαγωγη 2

ΚΛΑΣΕΙΣ 2

Email 2

Account 2

Server 3

Client 7

ClientHandler 8

Testing 12

ΠΡΩΤΟΚΟΛΛΟ 12

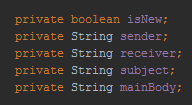
**ΕΙΣΑΓΩΓΗ**

Στόχος της εργασίας ήταν η εξοικείωση μας στην χρήση των **Sockets**, των **Input/Output Streams** και του **Multi-Theading**. Μας ζητήθηκε λοιπόν να δημιουργήσουμε ένα απλοποιημένο κατανεμημένο σύστημα ηλεκτρονικής αλληλογραφίας, το οποίο θα αποτελείται από έναν **Server-Εξυπηρετητή** το οποίο θα μπορεί να εξυπηρετεί ταυτόχρονα δύο και περισσότερους **Clients-Πελάτες**, οι οποίοι με της σειρά τους θα στέλνουν διάφορα αιτήματα στον Server.

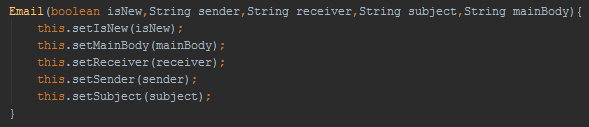
**ΚΛΑΣΕΙΣ**

**1) Email**

Η κλάση αυτή υλοποιεί τα Email που στέλνονται από χρήστη σε χρήστη. Κάθε Email έχει τα εξής χαρακτηριστικά:



Τα οποία αρχικοποιούνται στον constructor της κλάσης:

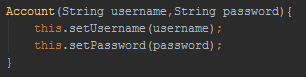


**2) Account**

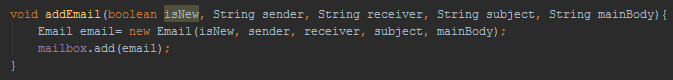
Η κλάση αυτή υλοποιεί τα Accounts των χρηστών που γίνονται registered στον Server. Κάθε λογαριασμός έχει τα εξής χαρακτηριστικά:



Τα οποία αρχικοποιούνται στον constructor της κλάσης:



Επίσης, πρόσθεσα την συνάρτηση **AddEmail** η οποία προσθέτει ένα νέο Email στην λίστα του χρήστη:

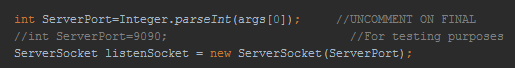


**3)Server**

Ορίζω αρχικά δύο σταθερές με το πλήθος των αρχικών χρηστών και το πλήθος των αρχικών μηνυμάτων τους.



Έπειτα πραγματοποιώ την σύνδεση του server στη θύρα που δίνει ο χρήστης ως 1ο όρισμα του προγράμματος:



Έφτιαξα τα μηνύματα που θα επαναλαμβάνονται συνέχεια ως String μεταβλητές:



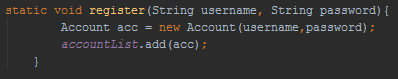
Καθώς επίσης και μία λίστα όπου αποθηκεύω όλα τα Accounts που έχουν δημιουργηθεί:



Οι συναρτήσεις που μας ζητήθηκαν υλοποιήθηκαν ως εξής:

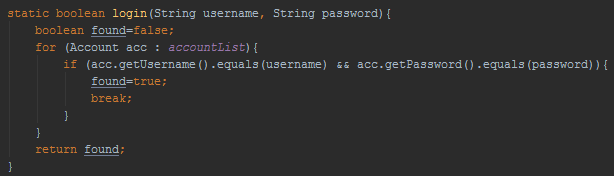
* **register**

Παίρνει ως ορίσματα το όνομα χρήστη και το συνθηματικό και αποθηκεύει το Account στην λίστα με τα Accounts



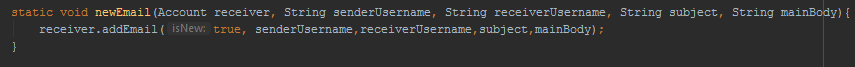
* **login**

Παίρνει ως ορίσματα το όνομα χρήστη και το συνθηματικό του και προσπαθεί να κάνει ταυτοποίηση. Επιστρέφει μια boolean τιμή true αν το Account υπάρχει, διαφορετικά επιστρέφει false.

****

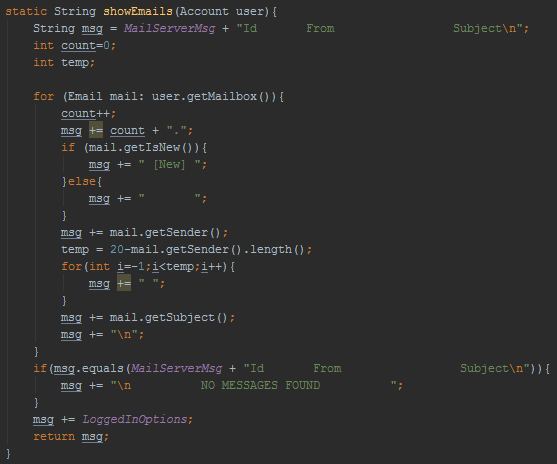
* **newEmail**

Δημιουργεί ένα νέο Email στο Account του παραλήπτη με τη χρήση της συνάρτησης addEmail.

****

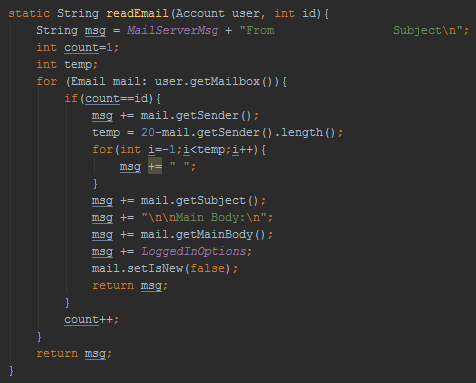
* **showEmails**

Εμφανίζει όλα τα Email του χρήστη αριθμημένα, ενώ αν δεν υπάρχουν μηνύματα εμφανίζει **NO MESSAGES FOUND.**

****

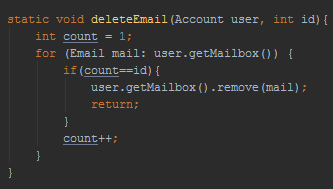
* **readEmail**

Παίρνει ως ορίσματα το Account και τον αριθμό του Email που ζητήθηκε και το εμφανίζει.

****

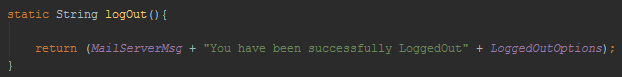
* **deleteEmail**

Παίρνει ως ορίσματα το Account και τον αριθμό του Email που ζητήθηκε και το διαγράφει.



* **logout**

Εμφανίζει το μήνυμα αποσύνδεσης του χρήστη.

****

* **exit**

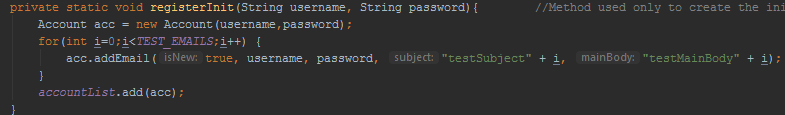
Εμφανίζει το μήνυμα εξόδου από τον διακομιστή και τερματίζει την σύνδεση απελευθερώνονται τους πόρους.



**Εκτός από τις συναρτήσεις που μας ζητήθηκαν, δημιούργησα και κάποιες βοηθητικές συναρτήσεις:**

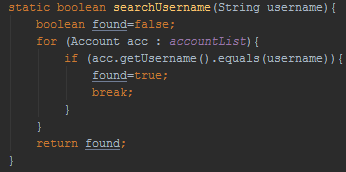
* **registerInit**

Αυτή η συνάρτηση δημιουργεί τους αρχικούς test-users με τα sample emails τους κατά την εκκίνηση του Server.



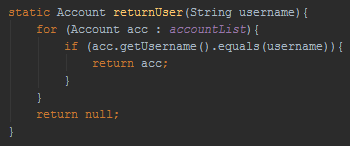
* **searchUsername**

Αυτή η συνάρτηση ψάχνει στη λίστα με τα Account για να βρει αν υπάρχει ένας συγκεκριμένος χρήστης. Αν υπάρχει επιστρέφει true, διαφορετικά επιστρέφει false.

****

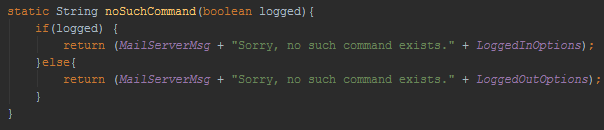
* **returnUser**

Αυτή η συνάρτηση επιστρέφει το αντικείμενο Account που ανήκει στον χρήστη με όνομα το όρισμα username.

****

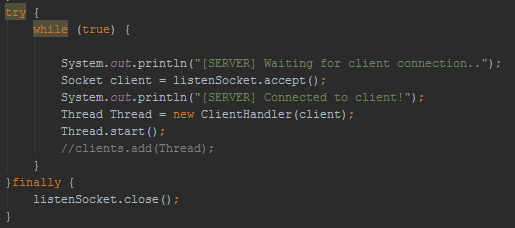
* **noSuchCommand**

Αυτή η συνάρτηση επιστρέφει ένα κατάλληλο μήνυμα σφάλματος όταν ο χρήστης δώσει λανθασμένη εντολή ή εντολή που δεν υπάρχει καν.

****

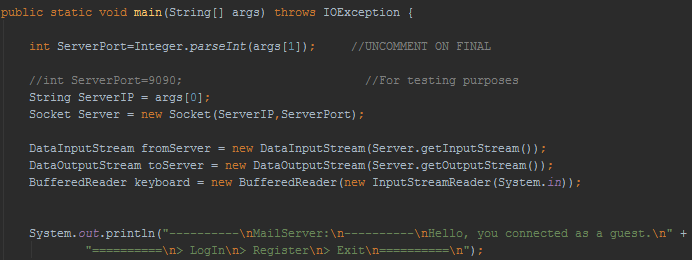
**Ο τρόπος που λειτουργεί ο Server είναι ο εξής:**

Σε έναν ατέρμονα βρόγχο ελέγχουμε συνεχώς σε ένα καινούριο Socket με όνομα client για αιτήματα από καινούριους πελάτες και τα αποδεχόμαστε, εμφανίζοντας το κατάλληλο μήνυμα σύνδεσης στο panel του server. Στη συνέχεια, δημιουργούμε ένα Thread, στο οποίο τρέχουμε ένα αντικείμενο της κλάσης **ClientHandler (θα εξηγηθεί στη συνέχεια, με λίγα λόγια διαχειρίζεται τους clients)** με όρισμα τον client που μόλις συνδέθηκε και ξεκινάμε με την εντολή start() το Thread. Αυτό γίνεται έτσι ώστε να τρέχουμε ταυτόχρονα πολλά Threads για την ταυτόχρονη εξυπηρέτηση πολλών χρηστών. Στο τέλος, με το κλείσιμο του Server απελευθερώνουμε τους πόρους.

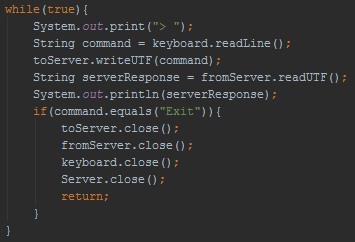
****

**4) Client**

Στην κλάση του πελάτη, ορίζουμε ένα Socket με την Localhost IP **127.0.0.1 (1ο όρισμα)** και με τη **θύρα** **(2ο όρισμα)** στην οποία ακούει ο Server έτσι ώστε να έχουμε στην οθόνη μας επικοινωνία μεταξύ client-server. Στη συνέχεια ορίζουμε **ενα Output Stream** με όνομα **toServer** το οποίο εξυπηρετεί στο να στέλνει ο πελάτης δεδομένα στον server, και **δύο Input Streams**, το ένα με όνομα **fromServer** το οποίο θα δέχεται τα δεδομένα που στέλνει ο server, και ένα με όνομα **keyboard** που θα δέχεται τα δεδομένα που δίνει ο χρήστης μέσω του πληκτρολογίου του.



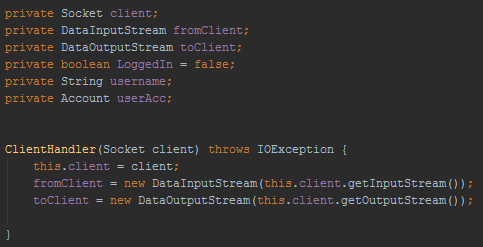
Τέλος, δημιουργούμε έναν ατέρμονα βρόγχο στον οποίο ο χρήστης δίνει δεδομένα μέσω του πλητρολογίου, τα στέλνει στον server, o server τα επεξεργάζεται και απαντάει κατάλληλα στο αίτημα. Όταν γίνει exit, οι πόροι (Sockets, Streams) απελευθερώνονται και η επικοινωνία με τον server τερματίζεται.



**5) ClientHandler**

Η κλάση αυτή είναι υπεύθυνη για την διαχείριση των Thread που εξυπηρετούν τους πελάτες, λαμβάνοντας αιτήματα από αυτούς και απαντώντας τους με κατάλληλα μηνύματα και ενδείξεις.

Παρακάτω δίνονται τα πεδία και ο κατασκευαστής της κλάσης:



Χρησιμοποιούμε όπως φαίνεται κι εδώ ένα Input και ένα Output Stream για την λήψη αιτημάτων από τους πελάτες και την μεταφορά δεδομένων προς αυτούς.

Στην συνάρτηση **run** των Threads, έχουμε έναν ατέρμονα βρόγχο στον οποίο περιμένουμε τις εντολές του κάθε client και τρέχουμε μια μεγάλη **switch case** η οποία έχει όλες τις διαφορετικές εντολές-αιτήματα που μπορεί να δώσει ένας πελάτης.

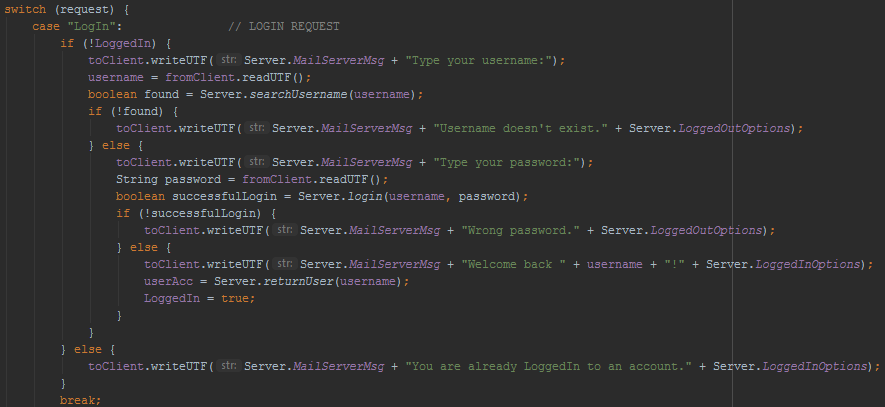
Οι περιπτώσεις των **requests** είναι οι εξής:

* **LogIn**

Ο χρήστης πληκτρολογεί το **όνομα χρήστη** του. (Αν δεν υπάρχει του εμφανίζει κατάλληλο μήνυμα και τον επιστρέφει στις αρχικές επιλογές)

Ο χρήστης πληκτρολογεί το **συνθηματικό** του. (Αν δεν γίνει επιτυχημένη ταυτοποίηση του εμφανίζει κατάλληλο μήνυμα και τον επιστρέφει στις αρχικές επιλογές)

Αν έγινε επιτυχημένη ταυτοποίηση, του εμφανίζει μήνυμα Welcome Back με τις καινούριες εντολές.



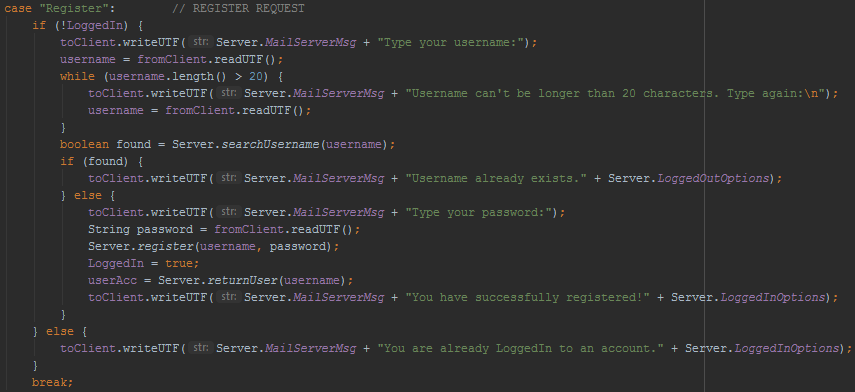
* **Register**

Ο χρήστης πληκτρολογεί το **όνομα χρήστη** του. (Αν υπάρχει ήδη του εμφανίζει κατάλληλο μήνυμα και τον επιστρέφει στις αρχικές επιλογές)

Αν το όνομα χρήστη είναι **>20 χαρακτήρες**, ζητάει καινούριο όνομα χρήστη.

Ο χρήστης πληκτρολογεί το **συνθηματικό** του. (Αν δεν γίνει επιτυχημένη ταυτοποίηση του εμφανίζει κατάλληλο μήνυμα και τον επιστρέφει στις αρχικές επιλογές)

Μόλις δώσει τον κωδικό (θεωρώ δεδομένο ότι τον βάζει σωστά οπότε δεν υλοποίησα επιβεβαίωση κωδικού 2η φορά), του έρχεται **μήνυμα επιτυχούς εγγραφής** και **συνδέεται αυτόματα** στον καινούριο του λογαριασμό.



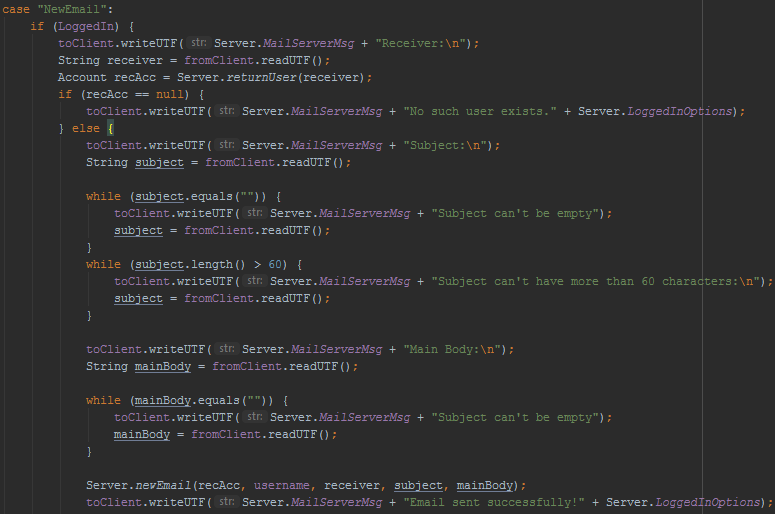
* **NewEmail**

Ο χρήστης δίνει το **όνομα του παραλήπτη**. (Αν δεν υπάρχει τέτοιος χρήστης του εμφανίζει κατάλληλο μήνυμα και τον επιστρέφει στις αρχικές επιλογές)

Έπειτα δίνει το **θέμα-subject** του Email, επιβεβαιώνοντας ότι **δεν είναι κενό** και ότι **δεν περιλαμβάνει περισσότερους από 60 χαρακτήρες.**

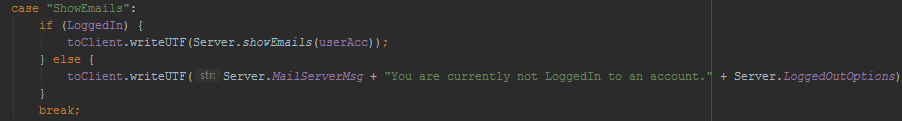
Τέλος, δίνει το **περιεχόμενο** του Email, επιβεβαιώνοντας ότι **δεν είναι κενό.**

Μόλις δώσει και το περιεχόμενο, του επιστρέφεται μήνυμα επιτυχούς αποστολής του μηνύματος.



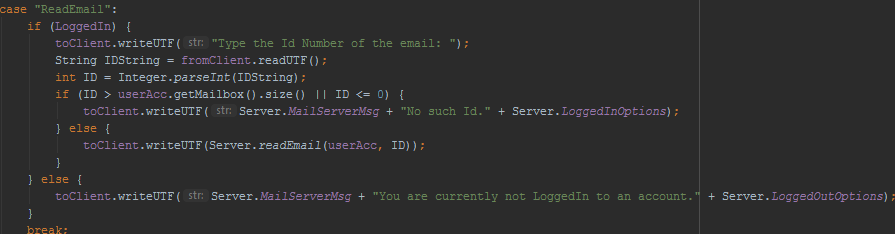
* **showEmails**

Με την εντολή αυτή, ο server επιστρέφει στον χρήστη **αριθμημένα όλα τα μηνύματα** που υπάρχουν αποθηκευμένα στην λίστα με τα Email του λογαριασμού του.

****

* **ReadEmail**

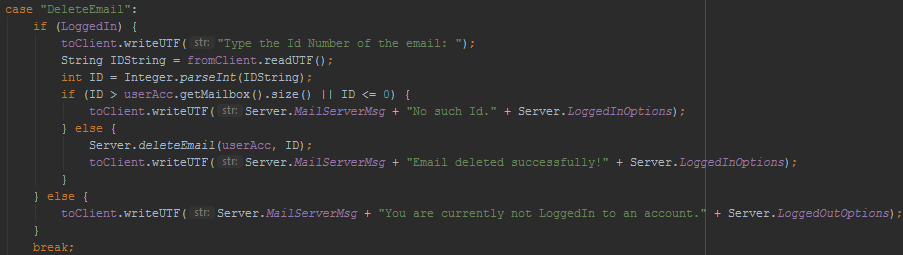
Ο χρήστης δίνει τον **αριθμό ID του Email** που θέλει να διαβάσει και ο Server του επιστρέφει το περιεχόμενο αυτού του μηνύματος, μαζί με το θέμα και τον αποστολέα.

****

* **DeleteEmail**

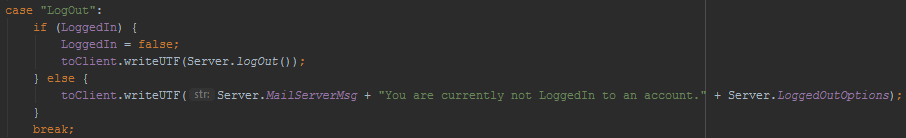
Ο χρήστης δίνει **τον αριθμό ID του Email** που θέλει να διαγράψει. (**γίνεται έλεγχος ώστε να υπάρχει αυτός ο αριθμός**)

Έπειτα του έρχεται μήνυμα επιτυχημένης διαγραφής του μηνύματος.



* **LogOut**

Η εντολή Logout **αποσυνδέει τον χρήστη από το Account του** και τον επιστρέφει στο πάνελ με τις αρχικές επιλογές (LogIn, Register, Exit).

****

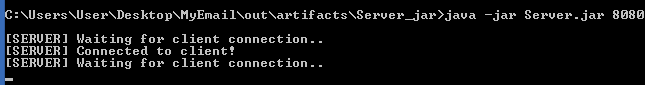
* **Exit**

Η εντολή Exit **τερματίζει τη συγκεκριμένη σύνδεση** client-server.

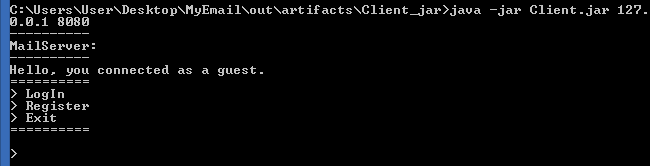


**TESTING**

**Σύνδεση του Server (και μήνυμα επιτυχούς σύνδεσης με Client):**

****

**Σύνδεση Client - Server:**

****

**ΠΡΩΤΟΚΟΛΛΟ (TCP)**

Η επικοινωνία του Server με τους Client γίνεται μέσω του πρωτοκόλλου TCP. Αυτό λειτουργεί ως εξής:

H υπηρεσία TCP υλοποιείται με τη δημιουργία από τον αποστολέα και τον παραλήπτη τερματικών σημείων (end points) που ονομάζονται υποδοχές(sockets).

Δημιουργείται μια διαδικασία στον **host 1(server)** και προσκολλάται σε μια **port p** οπότε και μπλοκάρεται περιμένοντας μια σύνδεση**.** Μια άλλη διαδικασία στον **host 2(client)** από **άλλο port** ζητάει να συνδεθεί στην **port p** του **host 1.** Χρησιμοποιείται μια **three-way handshake** ανάμεσα στον client που ζητά τη σύνδεση και τον server.Η σύνδεση θεωρείται **επιτυχής** και ακολουθείται μια διαδικασία ανταλλαγής πακέτων. Η διαδικασία του server είναι συνεχώς **σε κατάσταση block** και περιμένει τα δεδομένα που θα στείλει ο client. Στο τέλος, ο client στέλνει αίτημα αποσύνδεσης, οπότε και **η σύνδεση τερματίζεται** ενώ **οι πόροι αποδεσμεύονται**.

Όσον αφορά τα πακέτα, το TCP δέχεται τα data streams, **τα τεμαχίζει σε κομμάτια** και στέλνει κάθε κομμάτι ως **ένα ξεχωριστό IP datagram**. Στον αποδέκτη, το TCP **επαναδιατάσσει τα datagrams** που φθάνουν εκτός σειράς.

Μια σύνδεση TCP είναι ένα **full duplex end-to-end byte stream**. Κάθε byte σε μια σύνδεση TCP έχει το δικό του **32-bit ακολουθιακό αριθμό**. Τα δεδομένα ανταλλάσσονται σε **segments** αποτελούμενα από τον **header** μήκους **20-bytes** ακολουθούμενου από τα δεδομένα. Χρησιμοποιείται ένα **πρωτόκολλο ολισθαίνοντος παραθύρου** σύμφωνα με το οποίο ο μεταδότης εκκινεί ένα **χρονόμετρο αναμετάδοσης** μόλις μεταδίδει ένα segment. O αποδέκτης στέλνει πίσω ένα segment που περιέχει ένα **acknowledgement number**. Aν πριν φθάσει το acknowledgement το χρονόμετρο αναμετάδοσης του μεταδότη κάνει **time out,** το segment **επαναμεταδίδεται**.