ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ

Project_Phase2

ΑΝΑΣΤΑΣΙΟΣ ΜΠΕΝΟΣ 3130141
ΕΥΘΥΜΙΟΣ ΤΣΟΠΕΛΑΣ 3130210

(Οδηγίες για Compile στο τέλος της αναφοράς.)

Αναφορά Παραδοτέου

Περιβάλλον δοκιμών:

Στα πλαίσια της εργασίας υλοποιήσαμε ένα εικονικό περιβάλλον που συντονίζει τον διαμοιρασμό ηλεκτρικής ενέργειας μεταξύ χρηστών. Οι χρήστες μπορούν τόσο να ζητούν όσο και να προσφέρουν μονάδες ενέργειας από και προς τους υπόλοιπους χρήστες της κοινότητας αντίστοιχα και όπως περιγράφεται παρακάτω. Στόχος της κοινότητας είναι να υπάρχει αδιάλειπτη προσφορά μονάδων ενέργειας σε όλους τους χρήστες που έχουν ανάγκη και η ανταλλαγή των μονάδων ενέργειας να πραγματοποιείται με δίκαιο τρόπο, βάση του κανόνα: *max(s – b, 0) * t που εξηγείται στην εκφώνηση*. Ως προς την υλοποίηση, αναπτύξαμε το λογισμικό του Server (SocketServer) και των energy users/χρηστών (SocketClient01) της υπηρεσίας. Κατά την εκκίνηση του προγράμματος,ο Server είναι σε θέση να ανταποκρίνεται σε αιτήματα των energy users για την αποστολή μονάδων ηλεκτρικής ενέργειας. Για τον σκοπό αυτό ο Server γνωρίζει την ποσότητα ενέργειας που διαθέτει ο κάθε energy user του δικτύου. Στην πραγματικότητα μέσω της μεθόδου SignIn λαμβάνει αιτήματα σύνδεσης από τον εκάστοτε user με τα στοιχεία του.

Από την άλλη πλευρά ο client μόλις δημιουργηθεί τρέχει την μέθοδο register στην οποία καλείται ο χρήστης να συμπληρώσει τα στοιχεία του και στη συνέχεια με την introduce_myself στέλνει για πρώτη φορά τα προσωπικά του στοιχεία στον server.Από την στιγμή αυτή και έπειτα η κεντρική μέθοδος που χρησιμοποιείται είναι η circle of life μέσα στην οποία ο client “ζεί” και πραγματοποιεί ενέργειες όπως να καταναλώνει μονάδες ενέργειας,να ελέγχει το απόθεμα του,να εκκινεί διαδικασίες ανατροφοδότησης και εν ολίγοις πραγματοποιεί επικοινωνίες είτε με άλλους clients είτε με τον server κ.α.

Θέλοντας να εξηγήσουμε λεπτομερώς αυτή τη θεμελιώδη μέθοδο του client αρκεί να περιγράψουμε το γενικότερο σενάριο του κύκλου ζωής ενός energy user (βλέπε ΣΕΝΑΡΙΟ).

Κατα την υλοποιηση του δεύτερου παραδοτέου αξιοποιήσαμε την δημιουργία τριών βασικών concurrent hashmaps S_List, B_List, T_List οι οποίες αποθηκεύουν το σύνολο των ενεργειακών μονάδων που ο εκάστοτε χρήστης προμήθευσε προμηθεύτηκε αντίστοιχα. Αρχικοποιουμε με 0 τα values κατα την διάρκεια του signin και στη συνέχεια τις ενημερώνουμε όποτε βρίσκεται κάποιος Supplier. Αντίστοιχες μέθοδοι S,B_List_Update πραγματοποιούν αυτή την ενημέρωση και επηρεάζουν την T_List η οποία αποθηκεύει το χρονικό διάστημα που μεσολαβεί από την τελευταία επιλογή ενός χρήστης ως προμηθευτή. Καθοριστικό ρόλο παίζει η SBT_List στην οποία αποθηκεύουμε το όνομα του user ως key και το SBT score του, το οποίο υπολογίζεται βάση του πιθανο-θεωρητικού κανόνα που δόθηκε στην εκφώνηση (μέσω της μεθόδου SBT_List_Use_of_Formula). Στη συνέχεια, με την κλήση της μεθόδου Sort αποθηκεύουμε ταξινομημένα κατα φθίνουσα σειρά (ως προς SBT Score) τα username που έχουμε στη SBT_List.

<u>ΣΕΝΑΡΙΟ</u>

Όταν κάποιος client , μέσω της κατανάλωσης ενέργειας που επιτελεί, χάσει το 20% ή το 50% ή το 90% του αρχικού του αποθέματος, κάθε φορά στέλνει στον master server ένα αίτημα με το ποσό ενέργειας που ζητά (10% του αρχικού αποθέματος). Ο server με την σειρά του εκκινεί μια διαδικασία αναζήτησης του client που φέρει το μεγαλύτερο απόθεμα (μέσω της myMap concurrent hash λίστας που έχει δημιουργήσει κατα το Registration). Άν το αίτημα του client είναι μικρότερο από το 10% του αποθέματος του υποψήφιου supplier τότε το αίτημα γίνεται δεκτό (1. Single Supplier, βλ. Υπόδειγμα 1). Άν ο server δεν μπορεί να βρει Single Supplier τότε απαντάει στον αιτών client ~~> fail . Σε αυτό το σημείο ο client αποφασίζει να ακολουθήσουν το Plan A, κατα το οποίο ο server καλείται να ψάξει για πλήθος απο suppliers, το οποίο θα μπορεί να ικανοποιήσει αθροιστικά το αρχικό αίτημα του client. Η λογική που ακολουθείται είναι η εξής:

Έστω Z η αιτούμενη ποσότητα ενέργειας του client .

ΑΡΧΗ
Για τον 1ο στην Καταταξη του SORT.txt:
Δίνει το 10% του αποθέματος του, ώστε να καλύψει όσο περισσότερο μπορει.
Αφαιρώ απο το Z, τις μονάδες που εξασφαλίστηκαν.
Suppliers_List -> + { Το username του , Το ποσό που έδωσε}
Αν το αίτημα Z Ικανοποιήθηκε σταματώ , αλλιώς:

Για τον 2ο στην Καταταξη του SORT.txt:
Επαναλαμβάνω.

…

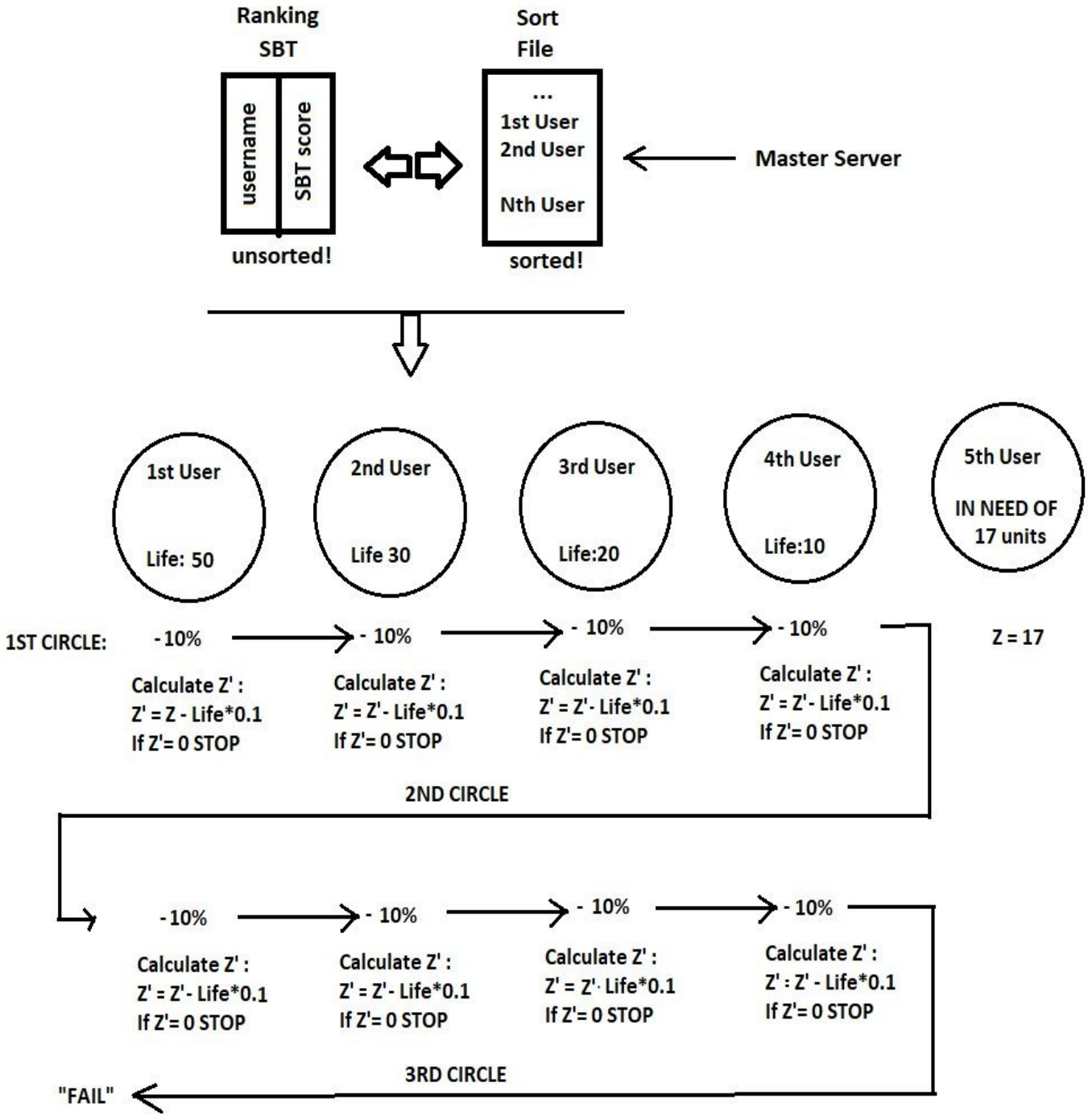
Για τον τελευταίο στην Καταταξη του SORT.txt:
Δίνει το 10% του αποθέματος του, ώστε να καλύψει όσο περισσότερο μπορει.
Αφαιρώ απο το Z, τις μονάδες που εξασφαλίστηκαν.
Suppliers_List -> + { Το username του , Το ποσό που έδωσε}
Αν το αίτημα Z Ικανοποιήθηκε σταματώ ,

Αλλιώς:* Ολοκληρώθηκε ένας Κύκλος. *Επαναλαμβάνω από το σημείο ΑΡΧΗ και κάθε φορά κατασχετω το 20% του αποθέματος του εκάστοτε client.

ΑΝ ΟΔΗΓΗΘΟΎΜΕ ΣΤΗΝ ΑΝΆΓΚΗ ΓΙΑ ΕΚΙΝΙΣΗ ΤΡΙΤΟΥ ΚΥΚΛΟΥ, ΤΟ ΑΡΧΙΚΟ ΑΙΤΗΜΑ ΔΕΝ ΘΑ ΙΚΑΝΟΠΟΙΗΘΕΊ.

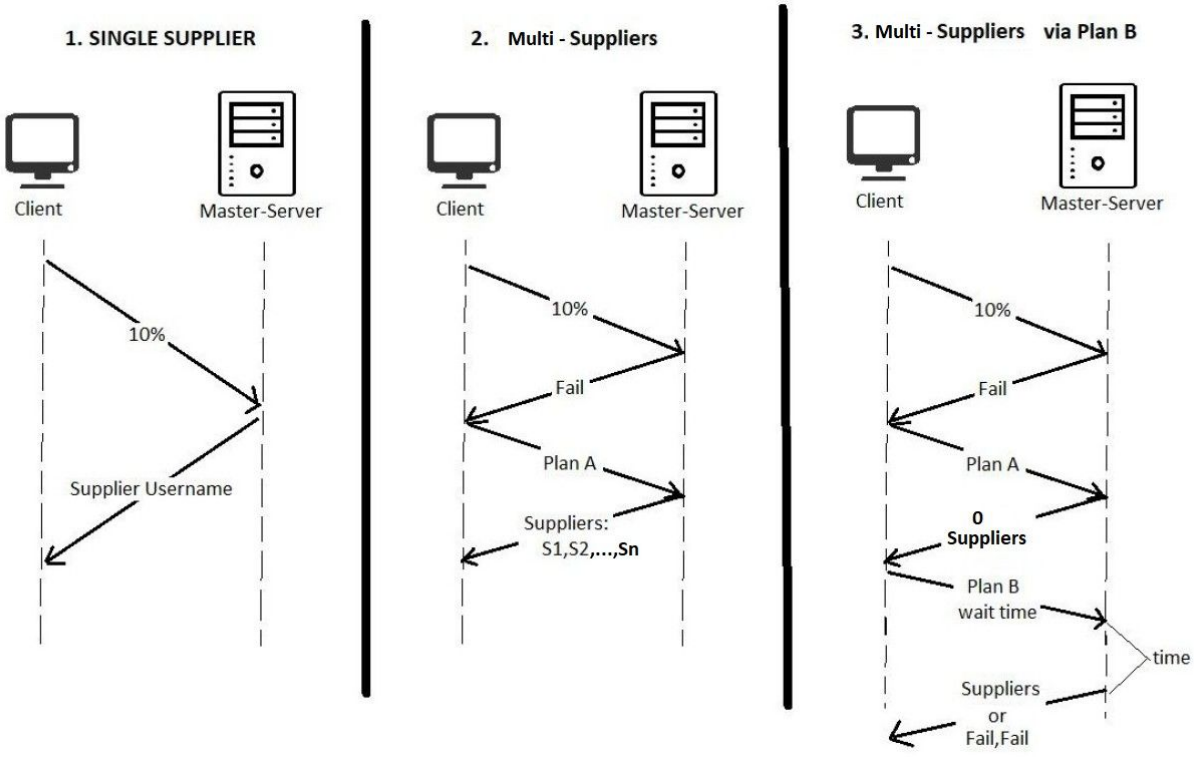
***** Ο Client που αιτείται, ασφαλώς, δεν δύναται να επιλέξει τον εαυτό του ώς Προμηθευτη , συνεπώς προσπερνά, το όνομα του, στο SORT.txt ,ανεξαρτήτως κατάταξης.***
***** Αν το υπολειπόμενο ποσό του αιτήματος (Z') είναι μικρότερο από το εκάστοτε 10% που οφείλει να παρέχει ο κάθε client, τότε ασφαλώς θα κατασχεθεί ένα μέρος του (όσο χρειάζεται) και το υπόλοιπο θα επιστραφεί στον client.***

Σχηματικά έχουμε:

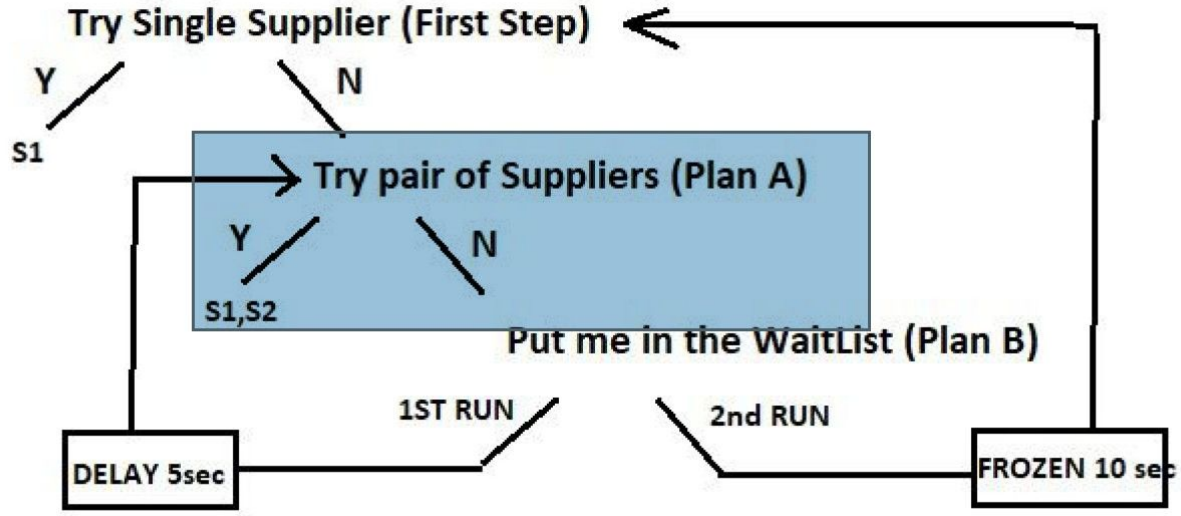


Αν ο server βρεί αρκετούς suppliers ώστε να ικανοποιήσει το εν λόγω αίτημα τότε αποστέλλει το πλήθος τους.
Αν ο server δεν μπορεί να ικανοποιήσει το αίτημα τότε αποστέλλει στον client "0".
Αν ο client λάβει "0" τότε καταλήγει στο plan B.
Πρόκειται για μία ιδιαίτερη περίπτωση κατα την οποία ο Client δηλώνει στον server ότι επιθυμεί το αίτημα ανατροφοδότησης του, να αποθηκευτεί και να επανεξεταστεί έπειτα από κάποιο συγκεκριμένο χρονικό διάστημα . Με το πέρας αυτού του διαστήματος ο Server εφαρμόζει για άλλη μία φορα το Plan A . Εν τέλει, είτε θα βρεί πλήθος διαθέσιμων προμηθευτών είτε θα αποστείλει τελική απόρριψη στον Client.
Αν ο Client λάβει μια τέτοια απόρριψη , χαρακτηρίζεται "σε κρίση" και σταματά την κατανάλωση ενέργειας για δέκα δευτερόλεπτα. Έπειτα είναι σε θέση να επαναλάβει εξ αρχής την όλη διαδικασία

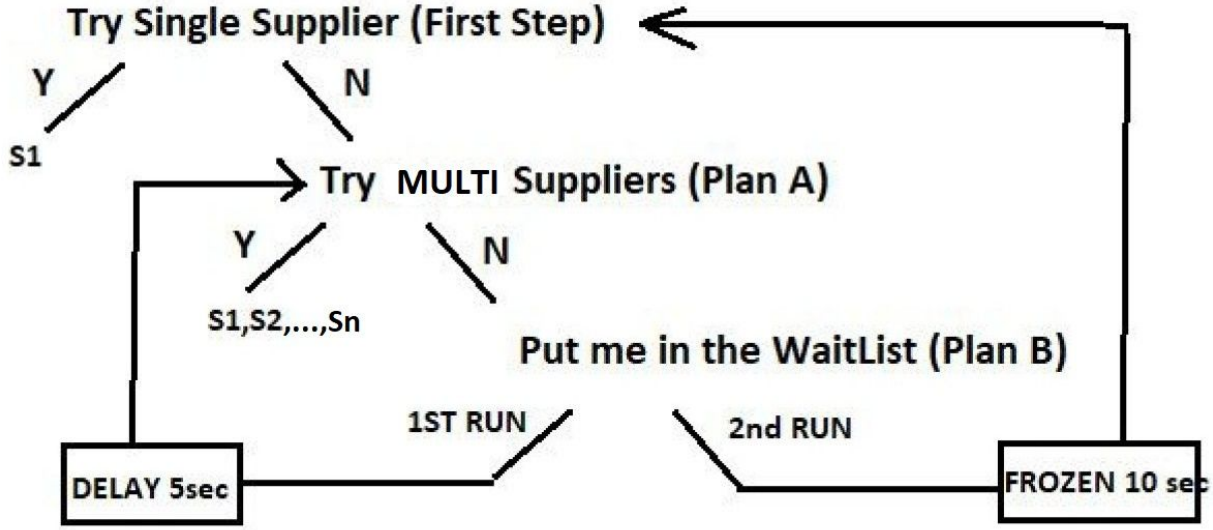
<u>Υπόδειγμα 1</u>



ΠΡΙΝ:

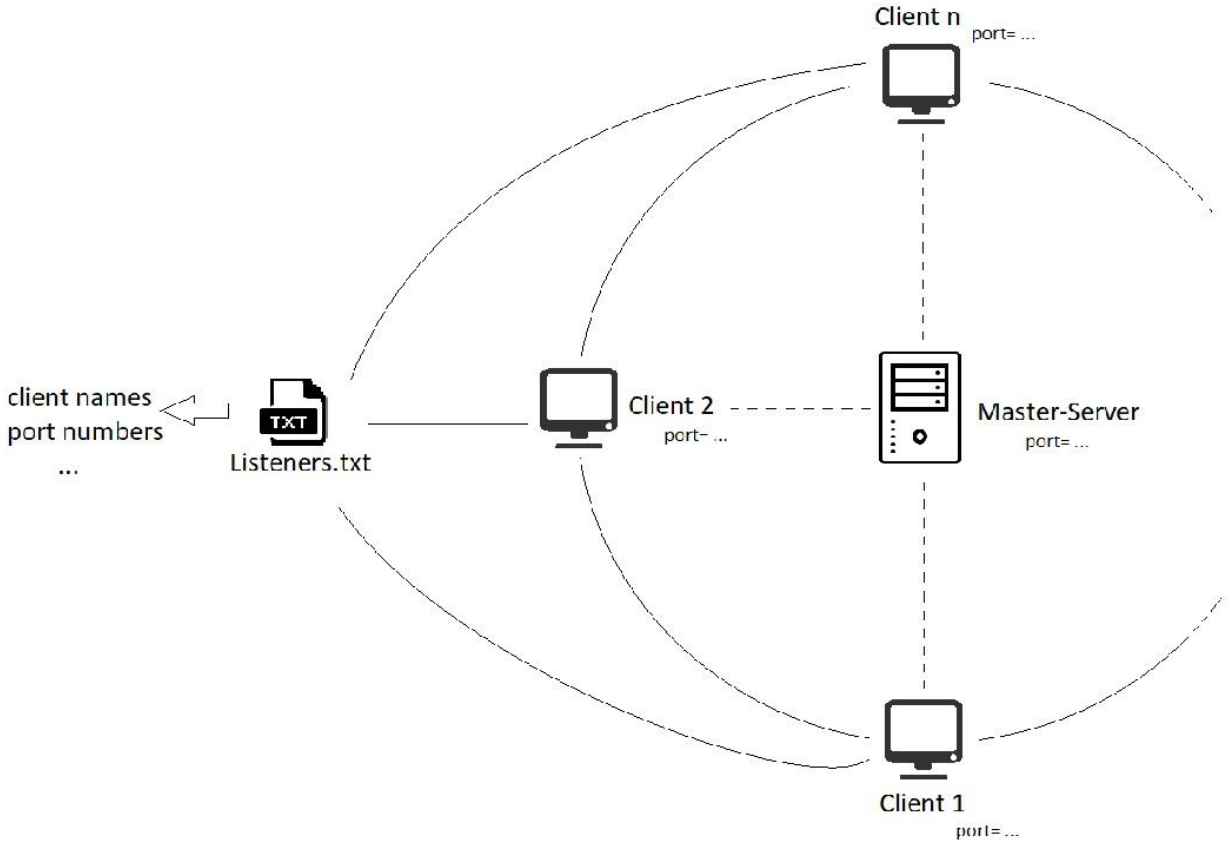

Try Single Supplier (First Step)
Y
N
S1
Try pair of Suppliers (Plan A)
Y
N
S1,S2
Put me in the WaitList (Plan B)
1ST RUN
2nd RUN
DELAY 5sec
FROZEN 10 sec

ΜΕΤΑ:


Try Single Supplier (First Step)
Y
N
S1
Try MULTI Suppliers (Plan A)
Y
N
S1,S2,...,Sn
Put me in the WaitList (Plan B)
1ST RUN
2nd RUN
DELAY 5sec
FROZEN 10 sec

Όταν ο εκάστοτε client έχει στην κατοχή του τα Username των/του Supplier τότε πραγματοποιεί αναζήτηση στο αρχείο Listeners.txt , με σκοπό να ενημερωθεί για την port στην οποία ακούει/ακούν ο supplier/suppliers. Να σημειωθεί ότι όταν ένας client εκκινείται καταγράφει σε αυτό το κοινόχρηστο αρχείο το Username και το port στο οποίο θα ακούει για πιθανά αιτήματα τροφοδοσίας απο άλλους clients. Εφόσον ενημερωθεί, προχωρά στην επίτευξη επικοινωνίας με τον/τους προμηθευτή/προμηθευτές του και εν συνεχεία στην λήψη των μονάδων. Αυτό μας κάνει σαφές ότι πρόκειται για μια peer-to-peer εφαρμογή κατα την οποία οι energy users δεν επικοινωνούν μόνο με τον master-server αλλά και μεταξύ τους (βλ. Υπόδειγμα 2).

Υπόδειγμα 2



Οδηγίες Compile:

Αλλαγή της IP στις γραμμες:
SocketClient01: 23, 162, 201, 230

- SocketClient01 ~~>Allow Parallel Run
- Run: SocketServer
- Run: SocketClient (1st Time: for 1st Energy User)
- Run: SocketClient (2nd Time: for 2nd Energy User)
  ...
- Run: SocketClient (n Time: for n Energy User)