Contents

Introduction

This documentation file is a comprehensive description of the whole usage and implementation of the DAS (distributed averaging system). The program is made in java and it uses UDP (User Datagram Protocol), which adheres to the project requirements.

Program rundown

The program can operate in two modes: “Master mode” and “Slave mode”.

Master mode listens for incoming UDP messages in a loop which are numbers, then processes them and does certain things based on the numbers:

-If the value is different from 0 or -1, it is printed to the console along with the currently stored numbers and stored in the program’s memory

-if the value equals 0 the process computes an integer average value of all non-zero numbers received since its start, including the value of <number> which is the initial umber specified when running the application. The value is rounded down to the nearest integer. Then the program prints this average to the console, lastly using the socket it works on it sends a broadcast message including the computed average value to all the machines working in its local network to the port of the number equal to <port>.

-If the value equals to -1, the process prints this value to the console. Using the socket it works on; it sends a broadcast message including the value -1 to all the machines working in its local network to the port of the number equal to <port> (so the same on which it works). Lastly it closes the used socket and terminates them.

Slave mode creates a UDP socket (datagram socket) opening a UDP port with a random number as this port (relying on the operating system for the port generation) and then, using this socket, it sends (to a process working on the same machine on a port with a number <port>) a message including the value of the parameter <number>. After a message is sent, the process terminates.

Implemented features

UDP Communication: The program uses UDP for all network communication, such as the project requirements said

Master and Slave Modes: Both modes of operation are implemented, the mode is automatically chosen based on the port's availability.

Number Processing: The Master processes incoming numbers (which can be doubles) according to the project requirements.

Broadcasting Messages: The Master broadcasts computed averages and termination signals to the local network.

Error Handling: The program handles invalid input and exceptions.

Correct double handling: The program takes care of occasional errors that result in using double type variables

Not implemented features, which would be nice to have.

Since the program uses UDP it would be nice to confirm that the numbers have been received. However, this is impossible since the program uses UDP. I personally would prefer to use TCP in this project since the numbers are quite important for us.

Difficulties and Errors encountered

One of the most important things for me to grasp was UDP and datagram packets. I personally dislike the UDP protocol, I'd much rather work with TCP, because we care for the numbers sent.

Regardless, I had to get accustomed to UDP, that was the most difficult part of the task for me.

Another difficult thing was storing the numbers, first I tried to use arrays, but then shifted to using an array list

One of the technical difficulties I encountered was whenever the slave mode sends 0 the master broadcasts the average value, the master mode then was putting this value in the numbers list, I needed to add a few lines that check whether the broadcast is from itself (equal port numbers), this way the program behaves correctly.

Later I noticed that it would be nice if the program could handle numbers as doubles, so I needed to change the number processing, then I recalled that whenever I use doubles there is sometimes an error with the decimal point, for example sometimes 5 is stored as 5.0000000001, if this happens then the comparison will work in a wrong way, so I introduced a variable called EPSILON that should take care of this potential error.