**Distributed Systems Job Scheduler Using Largest Available Server**

Luke Glover

Jarrod Adair

Frederick Jack

**Introduction**

Introduction (½ page): What this project (focusing on Stage 1) is about, including the goal of the project and Stage 1.

In a distributed system job allocation is very important for the efficient use of resources. Scheduling is a decision-making process that typically uses an optimizing algorithm for dispatching of jobs. This project is implementing a client-side job scheduler working with a distributed systems server simulator. The task of Stage 1 of the project is to schedule and dispatch every job to the largest server available in each simulation. The client is connected through a socket, so it is language independent, but this implementation is written in Java. The goal of this stage of the project is to successfully run and scheduled all available jobs without failures.

**System Overview**

System overview (½ page): high-level description of the system (both client-side simulator and server-side simulator with the focus being your client-side simulator), preferably, with a figure (your own, not one in ds-sim User Guide) showing the workflow/working of the system.

At a high level the process is started by the Client initiating a connection with the Server through a socket. The socket is the middleware that connects the Client and Server and allows them to send messages to each other. A Java socket uses TCP (Transfer Control Protocol) to create a connection. After the Client initiates a connection the Server, the Server then reads the specified configuration file, it writes a ds-system.xml file which contains the servers that are available for the Client to use. The Client will read the ds-system.xml file to find out that information. The Client’s core function is essentially a large loop that functions as a state machine. The steps are that the Server sends a single job with details to the client for scheduling. The Client then sends a command to the Server for it to process the job with the largest server it has available. Once the Server has processed a job it sends a completion message, and this process repeats until there are no new jobs and the Server tells the client it has no more jobs. The Client then sends a quit command to the Server and closes the connection. The output the scheduling is stored in a ds-jobs.xml file and it contains information about each job process.

![Diagram

Description automatically generated]()

As showed in the figure above the distributed system has six components.

socket – is the process that connects the client and server through a local host on port 50000.

server – also known as ds-server, simulates a cloud server and is written in C. The socket allows the client-server connection to exist without having the same language.

configuration file - contain the information about jobs such as the submit time, job ID, estimated run time and the resource requirements needed to process.

ds-system.xml file - contains the information about the servers available: how many of a particular server the bootup time the cost and the server resource specs.

client – program written to receive messages from server and to act as a job scheduler.

ds-jobs.xml file – contains the output of processed job after the connection is closed.

**Design**

Design (1 page): design philosophy, considerations and constraints, functionalities

of each simulator component focusing on the client-side simulator.

**Implementation**

Implementation (2 pages): brief description of any implementation specific information including technologies, techniques, software libraries and data.

structures used. How each of components/functions of your simulator is implemented including who oversees which function(s) and how they have led the design and development.

References including project git repository/wiki, e.g., GitHub and Bitbucket.

\* The numbers of pages for each section are also a suggestion.