Tyler Friesen - 104823930

**Communication Model**

Architecture: Client-Server

Communication Model: Interactive (Push-and-Pull), client needs to be able to push that they’re available to the server, pull the task from the server, push finished work back to the server

Communication Scope: One-to-One is used when the server and client communicate with each other, while multicasting is used as the server can communicate with multiple clients at once.

**Design Goals**

Design Goals:

* Simplicity: Messages will be small and easy to understand
* Efficiency: Small message size will allow for more efficiency and throughput
* Scalability and Extensibility: Message types allow for bother servers and clients to be added, as well as adding different job types

Reliable Message Exchange: Yes, need to know that the correct work is done, and the correct result is returned

Is Communication Security Important?: No, not worried about security for a small project like this

Persistent Connections?: No, once the client has the job, they don’t need to keep the connection open. They do need to be able to return the result of the job to the server

Bandwidth or Latency Restrictions or Requirements: No

Error and Failure Handling: Clients will send acknowledgement after receiving a job and servers will send acknowledgement after receiving the result of a job. If acknowledgement is not received, the last message will be sent again after a timeout.

**Message Design and Communication Rules**

The messages will be binary messages sent over the connection.

Message types:

* *Connection message*: The clients will send a message to the server indicating that they are available to take a job
* *Job message*: The server will send a message to the client containing the job
* *Acknowledgement message*: A message confirming the last message was received
* *Job return* message: The client will send a message to the server with the result of the completed job

Structure:

All messages will have a header with the type of message and size of the payload defined.

Below are the payload type definitions;

Connection message: Client sends message saying it is available and gives an overview of its available resources

Job message: Server sends the job with an id number

Acknowledgment message: Client sends back acknowledgement that the job has been received with the id number of the job

Job return message: Client sends the result of the job to the server along with the id number of the job

Application Flow:

Server:

* Server will remain in the idle state until a client initiates a connection.
* Server will send client a job and re-enter idle state
* Server will receive connections and send jobs while client completes the job
* Server will receive job result from client, send acknowledgement and re-enter idle state

Client:

* Client will connect to the server and wait for a job
* When a job is received the client will send acknowledgement
* Client will work on the job
* When the job is completed the client will send the result to the server and wait for acknowledgement
* If the acknowledgement is not received before the timeout, the client will resend the job return message and wait, otherwise it will try to connect to another server for a job