COMP-3670 Assignment 2

Sean Janssens, Gabe Oprescu, Vlad Roata

1. Our network application protocol will be a client and server architecture, where the clients will be made up of all the job-seekers, and the server(s) will be the job-creators. It will be an interactive (push and pull) communication model because the job-seeker(client) will have to receive the job from the job-creator, and in turn will also have to return the completed job to the job-creator(server). For the job-creator(server), it will send the job to the job-seeker and will receive the completed job from the job-seeker after they have completed the job. The communication scope will be both one-to-one, and multicast. It will be one-to-one because the input and output from a job-seeker will only be shared with the specific job creator that created the job. But it can also be multicast because a job-creator can share the same job between multiple different job seekers.
   Our network application protocol will be a simple, efficient two-way communication link between the server and the client, as we need to ensure that the job seeker/creator will receive the correct data quickly, but not as quick as possible. It needs to be reliable upon receiving and sending messages to and from the client/server. We need to ensure this as if somewhere in the process of the handshake between the client and server, we could lose information that would cause the job to become incomplete. We should have a secure connection such that this data will not be able to be read by invaders that wish to cause harm to the server in such a way that it could crash from too many requests in a DOS attack. Due to the nature of our task, we do not require the client to be connected to the server persistently. They are only going to connect to the server when they wish to complete a job. The server in a realistic environment would be kept running 24/7 so any job seekers at any time of the day would be able to access it. If an error were to occur during runtime, then we must close the current operation that occurred during that error, as this could incur effects to the server that could ruin the integrity of the data.

Message format will consist of simple strings. These strings will contain the type of job that is being offered to the seeker (I.e: “add 2+3”). The structure of the message is as follows: Job Type, Description, Required Number of Seekers, Accept/Decline. A full message might appear like this: "Job Type: Math | Description: Add the sum of two numbers | Seekers required : 1 | Accept? (Y|N)". This type of message transmits all the information required to complete the job.

Communication rules are as follows: seekers establish a connection to a creator (for simplicity, they already know the creator’s IP) and await a job. The job description string is sent from the creator to the seeker. The job type is randomly selected but some information can be controlled by the user (deciding the numbers that will be used for an addition, for example). The seeker has the option to type Y or N to accept or decline the job, respectively. If the job is declined, then the connection is closed. The same job will be offered to the next seeker. If it is accepted, the seeker reads the message and performs the appropriate task. A seeker that is currently working on a task is identified as “busy”, so another task may not be attributed to the same seeker until its task is complete, although the same task may also be assigned to another seeker if the job is large or complex enough to call for it. If the seeker receives a ping from a creator, it will return its status (busy or not busy). Once the seeker completes the job, it is no longer identified as “busy” and returns the appropriate message or result to the creator.

2.

Creating a custom application layer protocol will be essential in needing being successful in creating our application. It will allow us to customize our protocol to fit the boundaries that may arise in our program, in which an already existing protocol may not be as helpful. Our protocol will help us by with these custom attributes:

- The protocol should have a bare minimum level of security. Just basic encryption/obfuscation of back-and-forth messages as to deter exploitation of seeker resources where one could queue an infinite number of jobs if they knew what source message to send to seekers.
- The protocol should maximize throughput and not let seekers idle so long as there are potential jobs to be assigned.
- Messages should contain all possible information needed for completing a job.
- The protocol should be aware of job constraints such as requiring multiple seekers.