**BTC4201 / ICS4104: Distributed Systems**

**Assignment: Inter-process Communications in Distributed Environment**

110703 Ryan Jatelo

110542 Andrew Karanja

**KnockKnock Server**

The ServerSocket provides a system-independent implementation of the server side of the client/server socket connection. Typically, the ServerSocket class sits on top of a platform-dependent implementation hiding the details of any particular system. The constructor throws an exception in the case it cannot listen to the port (most likely the port is in use).

If the connection is successful, it uses accept() to accept the connection from a client listening on the same port.

The next lines open an input stream on the socket. An output stream is also opened on the same socket. These are local strings used to read from and write to the socket.

From here we create a constructor for the knockknock protocol which keeps track of the current question and the state that the question is in. outputLine = kkp.processInput(null); gets the first line of text that the server says to the client from the KKProtocol. The next two lines write to the output stream connected to the "client socket" and then flush the output stream. This sequence of code initiates the conversation between the client and the server.

The next section of code is a loop that reads from and writes to the socket thereby sending messages back and forth between the client and the server while they still have something to say to each other.

The last few lines help in closing the connection.

**KnockKnock Client**

This class implements the client program that speaks to the Server. For this to work, the server should be running already.

The first thing that the client program does is open a socket on the port that the server is listening to on the machine that the server is running on. It uses a hostname to represent a local network.

The client then opens an input and output stream on the socket and sets up some local variables, followed by the loop that implements the communication between the client and the server. The server speaks first, so the client must listen first which it does by reading from the input stream attached to the socket. In the case the client says “no” on the last question the server responds with “bye” and closes the connection.

**KnockKnock Protocol**

The KKProtocol class implements the protocol that the client and server use to communicate. This class keeps track of where the client and the server are in their conversation and serves up the server's response to the client's statements.

**References**

(*Writing the Server Side of a Socket*, n.d.)