Project 2 part 1: Evaluation of TCP behavior

Colin Miller, Nick Adman

Method: We wrote client and server code. The server acted as the server in our proxy; it changed state, without telling our client code. Both were written in Java. We ran the client and the server on separate machines. We then put the server into different states. To do this we would accept one connection and print out the message (to make sure there was a normal connection established) then we would do a .close() on the connection, disconnect the internet, or whatever was required for the given test. After this the server had a empty while(true) loop to makes sure it was still running, and the connection had not been closed as the JVM shutdown. Below is the table of the results we got, with the server state in the leftmost column and the client actions on the top row.

Client Action

Server State

	Minding Business	Read	Write	Close	Shutdown Input	Shutdown Output
Closed	Appears normal	Reads null from the server	The server responds null first time, the second time gives a Socket Exception: Broken pipe	Normal	Normal	Normal
Shutdown (Write)	Appears normal	Server responds to client with null	From the client side, the connection appears normal	Normal	Normal	Normal
Shutdown (Read)	Appears normal	Server never responds to client, client blocks on read	From the client side, the connection appears normal	Normal	Normal	Normal

Process Stop / Server Exception Thrown	Appears normal	Reads null from the server	Server responds null first time, second time gives a Socket Exception: Broken Pipe	Normal	Normal	Normal
Network Failure	Appears normal	Server never responds to client, client blocks on read	From the client side, the connection appears normal	Normal	Normal	Normal
Client loses Internet connection	Appears normal	Read times out, throws Socket Exception: Operation timed out	Appears normal	Normal	Normal	Normal
Server not started	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*
Server is normal	Appears Normal	Appears Normal	Appears Normal	Appears Normal	Appears Normal	Appears Normal

* If the server is not started, then the client will throw an error when trying to connect: We tried to connect a client to the server before starting the server code. The client immediately threw a java.net.ConnectException: Connection refused.