James Roesemann
CSCi 379
Project 1
Discussion questions.

- How would TCP or UDP affect the application? Why did I pick TCP over UDP?

TCP is more reliable than UDP. TCP provides error checking. If a packet drops during transmission or is transmitted erroneously it can be resent. In addition, TCP packets are assembled in their proper order. UDP on the other hand does not provide the same reliability. It does not have the similar error checking, packets are used as the arrive, and if a packet drops it is not retransmitted. UDP is faster than TCP but it sacrifices reliability. If I had gone with UDP, it is likely that some of the data would have never made it from client, to server, back to client, and that data could be out of order. For a fast-local connection UDP probably would have been fine, but for me the fact that both applications need all the transmitted data in the correct order in order to function made TCP the natural choice.

- Is my protocol statefull or stateless? How would the other have affected my application?

A state full protocol is a protocol is one that retains information about its connection during its life. A stateless protocol is one that does not retain this information. The client protocol knows the server Ip address and port number of the protocol it transmitted data to. The data it sent to the server is unique and the information it receives would likely be unique as well. For this reason, it can only accept traffic from the specific server it transmitted data to when using that port. The server must keep track of the port number and ip address of the clients it communicates with. It does this so that it does not mix up data from multiple clients. It is able to return specific communications to specific clients. If the server program provided a simple service such as merely accepting some transmitted data then it could be considered stateless in such a case the server would not need to remember who sent it the data, it already has it and does not need to retransmit. Our server accepts data and must return a manipulation of that data from a specific client and no one else. it would have to have a state full connection to achieve this. I don't believe you could do the assignment is asking with a stateless protocol.

- Is the server program able to communicate with multiple clients at the same time?

The server program was able to communicate with multiple client programs on multiple computers while running. I did not perform these communications simultaneously but I see no reason why they couldn't have. The client programs communicate with the same port numbers but with different ip addresses, so even if the server received packets simultaneously it should have been able to differentiate between them and communicate without any mixed signals.

The only time I could not get multiple clients to communicate at once was if I tried to run more than one on the same machine at the same time. Only on process could send or receive on a specific port at once. The process would have to be terminated before a new one could use that port.