Sean Maguire
Project 4
Experiment Analysis

Design

Bandwidth

a. To find the bandwidth used by the server and one client, I first determined the size of the UDP header and IP4 header. According to Microsoft.com, an entire UDP header is 20 bytes [1]. I then added this constant to the size of the server and client messages individually.
b. Because I was calculating the bandwidth, rather than observing it with a packet sniffer, I only ran the simulation once to verify the length of the server messages.
c. I recorded the length of server messages various amounts of clients in Excel.
d. The system conditions were those of the WPI CCC3 and CCC4 machines. I connected to them through the VPN with Putty on a Windows machine. The CCC machines are running Linux with 64 bit processors and a 64 bit Java interpreter.
e. I made sure I was using the IP4 header by adding `-Djava.net.preferIPv4Stack=true` to the java command in the makefile. The server and client are on the same, 100 millisecond sleep timer. Microsoft suggests that the maximum size of a UDP packet could be 65507 bytes, with a 20 byte header and a 8 byte UDP header. However, other sources such as stackoverflow.com suggest that a stable UDP packet should not exceed 512 bytes of data. In any case, my server messages should be far less than 512 bytes and can be sent in a single packet. The server messages are dynamically generated and have a variable length based on the number of horizontal lines, vertical lines, players and bullets. My client's update messages are static in size, sending only 7 bytes of data with each key press. The client's login message is slightly longer, sending 8 bytes plus the IP address which is a minimum of 8 bytes and a maximum of 15 bytes.

Processing

a. To find how long the server took to create an update, I used Java's `System.nanoTime()` as soon as the server's main loop started and just before the loop reached the sleep statement. This gave a good measurement of how long the entire update process took.
b. Because the server sends 10 messages per second, I let the server run for a 10 seconds with different numbers of clients. I fired some bullets and then averaged the processing time for clients simply running and clients running with bullets.
c. My server printed its processing time to the console, so I copied the data from the console and pasted it into Excel.
d. The system conditions were those of the WPI CCC3 and CCC4 machines. I connected to them through the VPN with Putty on a Windows machine. The CCC machines are running Linux with 64 bit processors and a 64 bit Java interpreter.
e. I am considering the time it takes the server to print to the console to be negligible.

Results

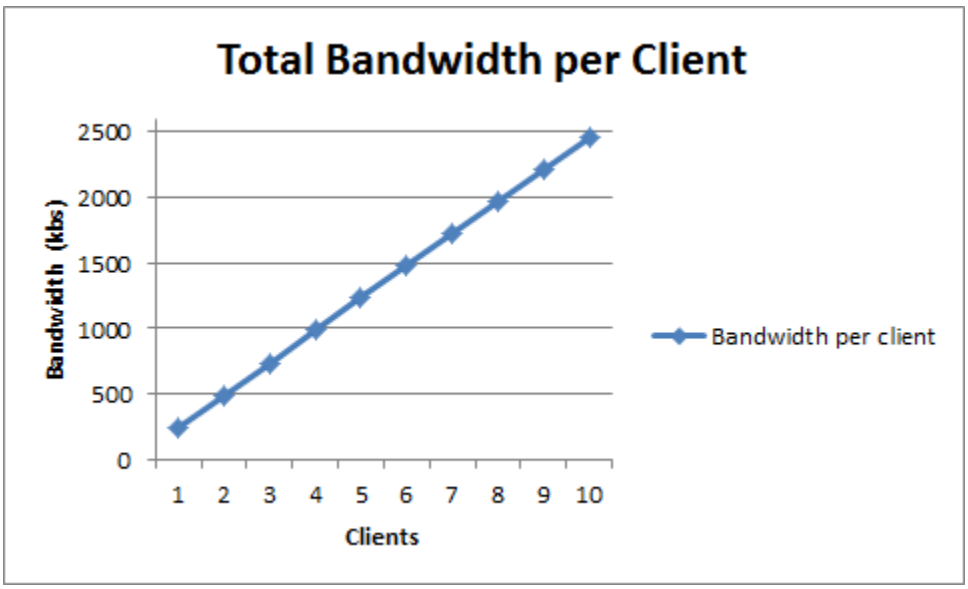
Bandwidth

C->S login: 21 byte login + 20 byte UDP header + 8 byte IP4 header = 49 bytes
S->C map: 26 bytes + 20 byte UDP header + 8 byte IP4 header = 54 bytes
S->C update: 18 bytes + 20 byte UDP header + 8 byte IP4 header = 46 bytes
Total to join a game: 149 bytes

For a client to respond:
C->S update: 9 bytes + 20 byte UDP header + 8 byte IP4 header = 37 bytes

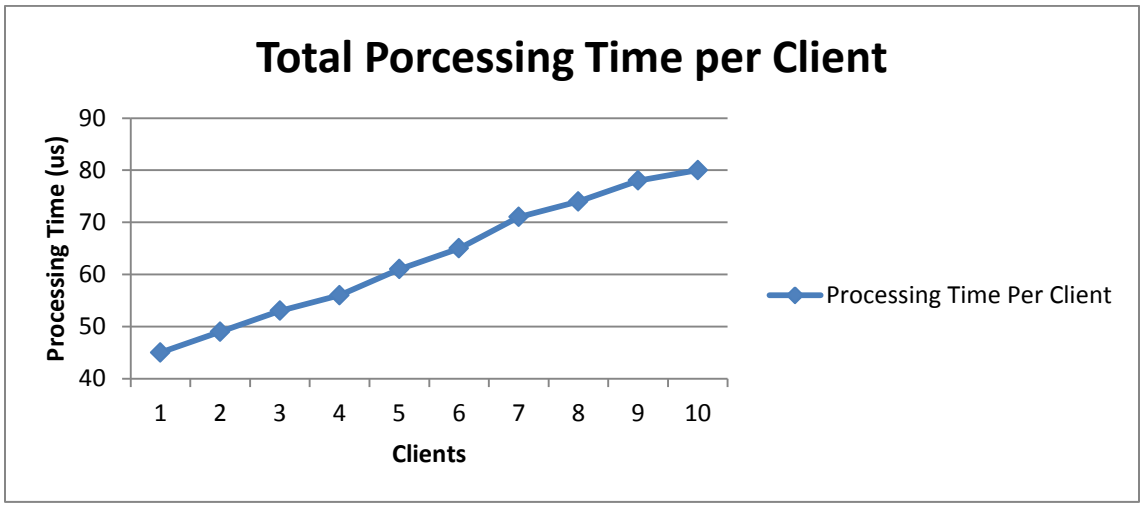
Each additional client needs to login and receive a map. Once a new client has joined there is an additional ~9 bytes (depending on position) for the server and each additional bullet is ~6 bytes (depending on position) for the server.

N = number of clients
So to find the bandwidth per client I am using: (149+37+54+6)*N



This graph shows bandwidth of 1-10 clients interacting with the server. It does not take into account that with more people playing, it is likely that there will be more than a constant number of bullets fired.

Processing



Analysis

Bandwidth

The graph that I created is linear. I believe that if I were to use a packet sniffer with multiple people playing the game, I would see an exponential increase in bandwidth. This is because as more people join the server, players need to dodge more bullets and have more targets to fire at. It was hard for me to simulate a game with 10 people as a single person on a single computer. I think the bottleneck on the game would be bandwidth, because in the next section, you will see that I believe the server could handle about 1000 clients.

Processing

The processing time is increasing as each client joins. My processing time measurements were taken in microseconds. To extrapolate these results, I want the processing time to reach around 100 milliseconds. If the processing time reached 100 milliseconds, it would mean that the server is maxed out for each loop and would respond late to the clients, slowing down play. Using this train of thought, I would estimate that my server could handle about 1000 clients.

References

[1] Microsoft. “Getsockopt() Function Returns a Different Maximum UDP Message Size Than You Expect in Windows 2000 SP3.” http://support.microsoft.com/kb/822061/. 2001. 14 Dec 2011.