My client.cpp was created using a model of request threads, Producer/consumer buffer, and an event handler. After starting the client and data server, the producer/consumer buffer is created using a standard semaphore implementation. Next, the request threads are created, one for each of the specified number of patients in the problem. In this case there were 3 patients that required data requests. Next, the event handler is created. This is where the various “working threads” will be contained. When the file descriptor detects something going on, the event handler will select that thread to complete the required tasks. 3 statistic threads were also created to keep track of the number of requests per each patient. All of these threads were created using the pthread\_create() function, by passing in the required data for the helper function that was linked to the corresponding thread type. Sometime in the event handler helper function, there is a segmentation fault that causes the program to seize. This causes me to not be able to accrue data on the various inputs of the function. Although, I can make an educated guess at the results of the function depending on the inputs. Clearly, increasing the number of request threads will increase the computation time of the program. I think that increasing the size of the buffer and the number of working threads would decrease the computation time until a certain point, where the time would then increase again. I believe this program would be faster than MP3, because there is less switching between threads, which causes some time increase. Using an event handler is a better option then working with several working threads.