**Operating Systems**

**Final Project Part 1**

The final project consists of modifications to the producer-consumer server. For part 1, you will add 2 new features to your clients.

Modify the 2 client programs, so that they do not send clients all at once, but send them little by little, so that the producers program will create a stream of producer clients that arrive over time (same for the consumers).

This requires a new command-line argument, arrival rate, which is understood to be in seconds. So a rate of 1, means 1 per second, a rate of 0.5 means 2 per second, etc. Realistic arrivals usually follow a Poisson process. For example, if an average 120 people go to a shop per day, they don’t come exactly 12 per hour or one every 5 minutes, they come in clusters.

Here is some code that produces such distribution of clients. Every time this function is called, it will generate a time to wait before sending the next client. It is random around the desired rate.

/\*

\*\* Poisson interarrival times. Adapted from various sources

\*\* r = desired arrival rate

\*/

double poissonRandomInterarrivalDelay( double r )

{

return (log((double) 1.0 -

((double) rand())/((double) RAND\_MAX)))/-r;

}

Since the interarrival time may be a fraction of a second, you cannot use the sleep function (seconds), you must use the function usleep (microseconds).

The second feature to add is that some percentage of your clients should wait SLOW\_CLIENT seconds before sending their identity (PRODUCE or CONSUME); they are kind of misbehaving clients. Those misbehaving clients should be randomly distributed among the good ones, not all at one time. You may implement this as you like. It also requires a command-line argument. The value SLOW\_CLIENT will be defined in the header and represent the number of seconds to wait before sending the initial message (set it to 3).

So update the command-line signature of your clients to:

producers [host] port num rate bad

consumers [host] port num rate bad

* Since the clients should be getting served, allow a maximum of 2000 clients (num) to be specified on the command line.
* rate should be a floating point number greater than 0.
* bad should be an integer between 0 and 100, which specifies the percentage of badly behaved clients who wait SLOW\_CLIENT seconds before sending their identity. If bad is 50, then approximately half the clients send their message immediately and half wait before sending. If bad is 0, all clients behave normally.