PRODUCER – CONSUMER PROBLEM IN MULTIPLE HOSTS USING LAMPORT MUTUAL EXCLUSION ALGORITHM

Problem definition:

The problem is a producer-consumer problem requiring an arbitrary number of producers, an arbitrary number of consumers and a buffer manager. In addition, a node controller process is required on each host which could ever have any producer or consumer processes running on it. (Producers and consumers may actually be implemented as threads or processes as you choose, but the buffer manager must be a process which creates either child processes or threads to handle each request which arrives from a producer or a consumer.)

Files Used:

1. buffermanager.c : The buffermanager.c file consists of code for creating a shared memory segment (4 ring buffers) and semaphore arrays for producer and consumer. It also listens continuously for any connection from producers and consumers, accepts them and execute the process of producing or consuming the donuts based on the request.

2. nodecontroller.c : The nodecontroller.c consists of code for implementing Lamport's logical clock algorithm for communication between the nodes. The node controller receives messages through message queue from producers and consumers requesting for producing or consuming donuts. Based on request the node controller requests all other nodes for processing the request and when it receives reply from other node controllers, it sends a message back to producer or consumer for continuing the production or consumption of donuts through buffer manager.

3. netproducer.c : This file has code for making a request to node controller for producing the donuts. Once it gets a reply message from its node controller, it makes a request to buffer manager for a socket connection. After receiving a connection, sends the request for producing a donut of a type randomly. Once it is done, sends a message to node controller saying that it has done its job.

4. netconsumer.c : This file has code for making a request to node controller for consuming the donuts. Once it gets a reply message from its node controller, it makes a request to buffer manager for a socket connection. After receiving a connection, sends the request for consuming a donut of a type randomly. Once it is done, sends a message to node controller saying that it has done its job.

5. msg_util.c & utilities.c : These files consists of functions for generating a message, reading a message, signal and wait functions.

6. Other files: buf_mngr.h, ddonuts.h – header files and Makefile

7. Output files: output.txt for consumer output

Success rate:

The success rate for the assignment is 90. As I am able to connect the node controllers and produce and consume donuts.

Execution Process:

1. open three different terminals for running the node controllers in each host
2. Go to cs91515-1, cs91515-2 & cs91515-4 and execute the following command at each host.
   Cs91515-1$ ./nodec 0
   Cs91515-2$ ./nodec 1

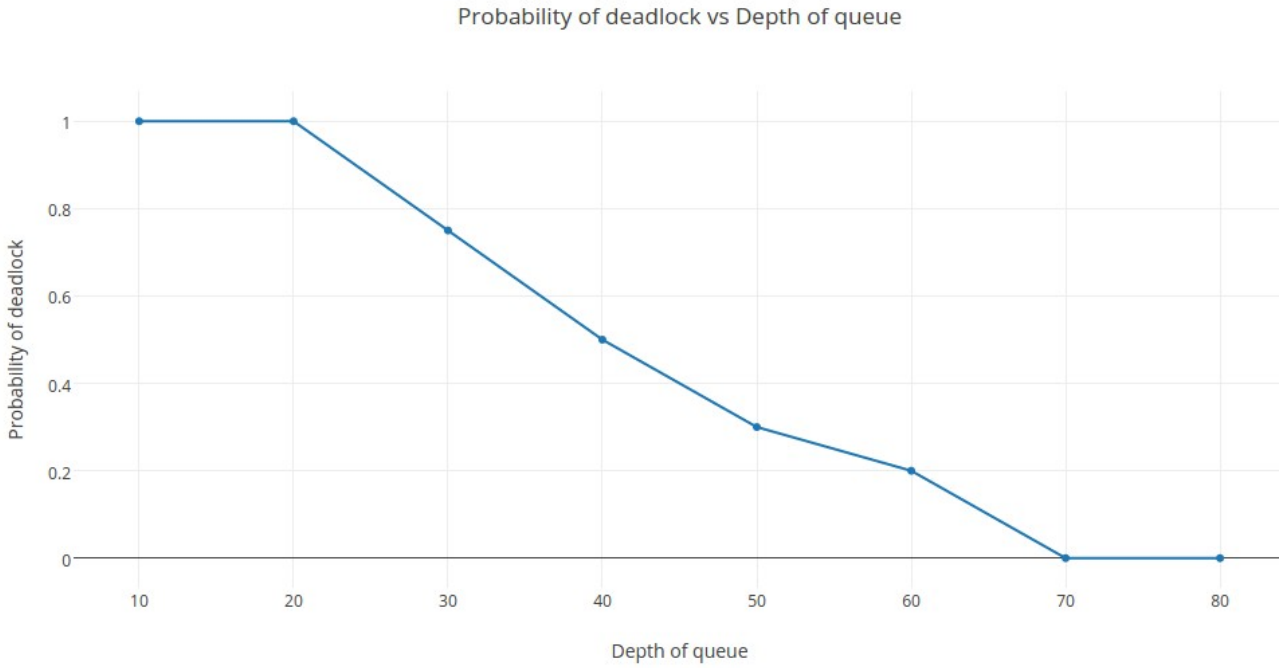
Cs91515-4$ ./nodec 2

3. Open another termial and run buffer manager executable file at one host(I have taken cs91515) as
   cs91515-6$ ./bm
4. Run all the producers first followed by consumers in the three hosts where the node controllers are being executed to exectute the process.
   Cs91515-1$ ./np cs91515-6 1 0 - for producer
   Cs91515-1$ ./nc cs91515-6 1 0 - for consumer

Experiment 1:

Produce a graph for the probability of deadlock vs the depth of the queue for 5 producer and 15 consumers.

Expected Result:



Conclusion: As the depth of the queue decreases the probability of the deadlock increases.