Operating System I - 91.515

Assignment-3

Hetali Shah Gala

Hetali\_ShahGala@student.uml.edu

I felt I have achieved 80% success from the solution which I prepared for this assignment. In my solution, I have used Ricart and Agrawal’s algorithm with messages of type: REQUEST and REPLY. I have used distributed system involving 4 fully connected nodes and the buffer manager will run on a separate system. The description about the design of my solution and the problems faced during implementation are explained below.

**Design** **Summary**

I have created two files *nc.c* and *ringbuffer.c* which contains the code for node-controller and ring-buffer respectively. Node controllers 1, 2, 3 and 4 are started in CS91515-1, CS91515-2, CS91515-3 and CS91515-4 respectively. Ring buffer is started in CS91515-5. All systems are started in port number 9000.

**Work Flow of Node Controller**

1. Every node controller is started by passing its number/id to the main function, number varying from 1 to 4.
2. Depending upon the node controller’s id no of server or client is created are started. Node-controller-1 has 3 accepts, node-controller-2 has 2 accept and one connect, node-controller-3 has 1 accept 2 connect and node-controller-4 makes connect to the rest of the node-controller.
3. Node-controllers are started one after the other in the order of their ID, NC 1 is started first, followed by 2, 3 and 4 respectively.
4. Once all the node-controllers are connected they will connect to ring-buffer.
5. When a connection is established between node controllers, a thread is created to do the read operation on that channel. In every node-controller totally of 4 thread is created to just to handle read operation, one for every node controller, and one for buffer.
6. After the connection is made with all systems producer and consumer are initiated.
7. Since I am using Ricart and Agrawal’s algorithm, each node controller have 8 queues.
8. Producer and Consumer use different queues, and they use one queue for one flavors and since producer produces 4 flavors and consumer consumes 4 flavors, they end up using 8 queues.
9. In each node controller is a single process which contains 30 threads of producer and 50 threads of consumer.
10. All queues are handled by the Queue manager function.
11. Every new request from external node controller or from internal client is sent to queue manager, i.e. either producer or consumer makes request or socket thread listening the channel established between node controllers, will send the struct received or created to the queue manager.
12. Struct has following information in it
    1. Time Stamp
    2. Node Controller ID
    3. Message Type: reply or request or delete
    4. Client ID: to know who exactly created the request
    5. Flavor
    6. Client type: Producer or Consumer
    7. Reply count: to notify the client when every node-controller replied
13. When Queue manager receives a struct, it checks for the message type
    1. If the message type is request and if it is created by the client then apart from adding the message to the queue, it also sends message to the other node controllers. But if is created by other node controller then it is just added to the queue.
    2. If the message type is reply then it checks to which message was it replied and then it increases the reply count of that message and if every node controller has replied then it signals the client who is waiting to do CS.
    3. If the message type is delete, this is sent by the ring buffer upon the completion of the task, then queue manager deletes the message from the node and also sends the replies to the other node controllers and replies the message of other node controller till it sees a message from its own client.
14. Producer and Consumer are waiting for the signal from the queue manager to proceed to do the critical section and once they get the signal they will a write message to the ring buffer.
15. Every producer and consumer takes a lock on a flavor before sending a request to Queue manager.
16. Consumer consumes only at flavor at a time, i.e. to collect 1 dozen it has to send message to queue manager for 12 times, which in turn send 12 message to each node-controller.

**Work Flow of Ring Manager**

1. When Ring buffer is started it starts a server.
2. Server accepts four connection from node controllers and creates thread and to that thread it pass the file descriptor of client which connected it.
3. Thread is listening on the file descriptor which it received from the server.
4. When a thread receives a message/struct from the node controller it check the client type and based on it creates a thread for producer or consumer.
5. At any given time there could be max of 4 consumer and 4 producer threads in the ring buffer.
6. Producer doesn’t need to take a lock to stop other producers from adding same flavor donut to the ring buffer because that part was taken by Ricart and Agrawal’s theorem.
7. Producer does need to take lock on flavor to so that it can stop consumer from accessing the same flavor buffer. Once the producer is finished accessing the buffer he signals the consumer.
8. Consumer is similar to producer but he signals producer.
9. Once when the thread is done accessing he sends the message to Node controller who created the message, changing the message type as deleted.

**Challenges Faced**

I was unable to get the all system to interact, though it worked when only 2 nodes connected but failed when system increased more than 2.

I am confident of the logic developed by me but I was unable to achieve 100% success in given time.