Assignment-3

1. Consider the following simple method to collect a global snapshot (it may not always collect a consistent global snapshot): an initiator process takes its snapshot and broadcasts a request to take the snapshot. When some other process receives this request, it takes a snapshot. Channels are not FIFO. Prove that such a collected distributed snapshot will be consistent iff the following holds (assume there are n processes in the system and Vti denotes the vector timestamp of the snapshot taken process pi):

   (Vt1[1], Vt2[2],....., Vtn[n] = max (Vt1, Vt2,....., Vtn)

   Don't worry about channel states.

2. What modifications should be done to the Chandy–Lamport snapshot algorithm so that it records a strongly consistent snapshot (i.e., all channel states are recorded empty).

3. Consider a system of n processes executing Maekawa's mutual exclusion algorithm. Suppose a process Pi makes a request to enter CS. How many times Pi may have to yield (relinquish) before it can enter the CS? Let the size of the quorum (subset) of a process be q = √n. Please give your answer in terms of n, q.

4. Susuki-Kasami mutual exclusion algorithm maintains a queue in the token. If such a queue is not maintained, then how will it affect the algorithm?

5. Does Ricart-Agarwala mutual exclusion algorithm require the channels to be FIFO? If so, please explain with an example how the example will be incorrect when the channels are non-FIFO? Otherwise, give justification as to why it will work correctly in spite of channels being non-FIFO.

6. In a distributed system, the following figure under the communication deadlock model. Find out if node 1 will detect a communication deadlock. Briefly trace the Fig:

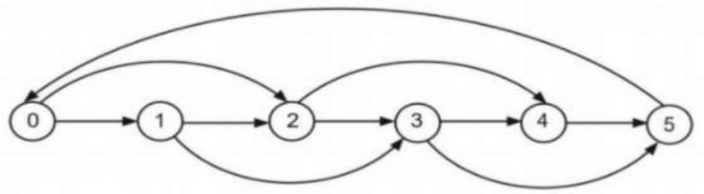

Fig: A WFG using the OR model

7. Design an algorithm to count the total number of processes in a unidirectional ring of unknown size. Note that any process in the ring can initiate this

computation and more than one process can concurrently run the algorithm.

8. Consider Peterson's leader election algorithm which works for unidirectional rings. As per this algorithm, a node i stays red after a round if alias(N) > max(alias,alias(NN)). Suppose we change it as follows: if alias(N) > max(alias, alias(N, N)) then the process N(i) stays red while processes i, NN(i) become black. What is the effect of this change on the algorithm?

****