ASSIGNMENT NO. 5

Problem Statement :

Implement token ring based mutual exclusion algorithm.

Tools/Environment :

Ubuntu , JDK

Theory :

Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process cannot enter its critical section while another concurrent process is currently present or executing in its critical section i.e only one process is allowed to execute the critical section at any given instance of time.

In Distributed systems, we neither have shared memory nor a common physical clock and there for we cannot solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used.

Requirements of Mutual exclusion Algorithm:

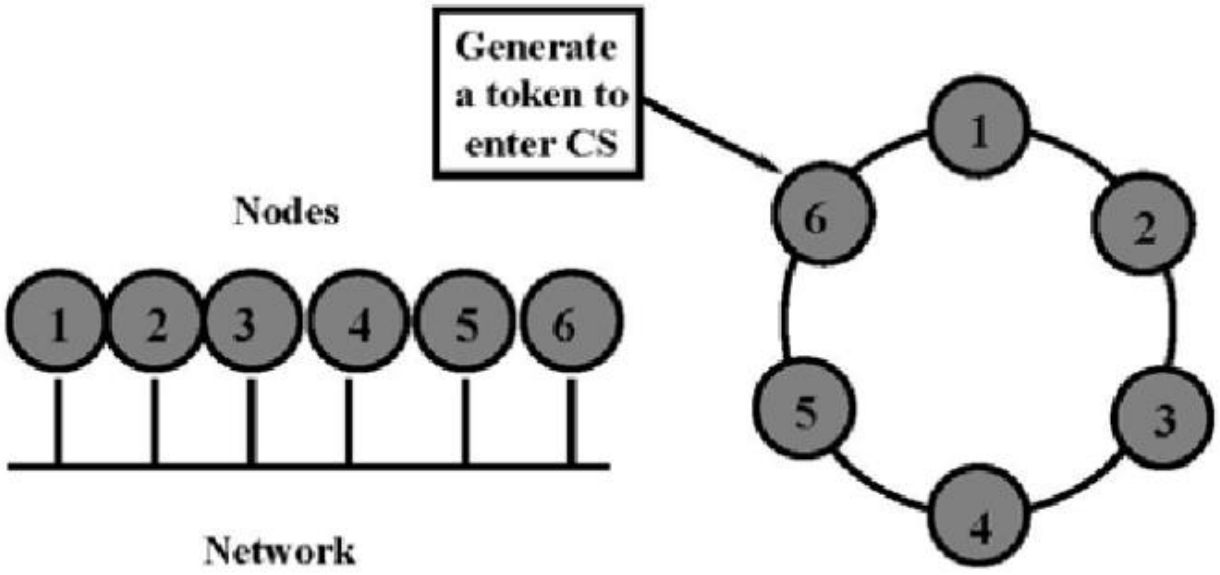
- **No Deadlock:**
  Two or more site should not endlessly wait for any message that will never arrive.
- **No Starvation:**
  Every site who wants to execute critical section should get an opportunity to execute it in finite time. Any site should not wait indefinitely to execute critical section while other site are repeatedly executing critical section
- **Fairness:**
  Each site should get a fair chance to execute critical section. Any request to execute critical section must be executed in the order they are made i.e Critical section execution requests should be executed in the order of their arrival in the system.
- **Fault Tolerance:**
  In case of failure, it should be able to recognize it by itself in order to continue functioning without any disruption.

Token Based Algorithm:

Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus network of processes. A logical ring is constructed with these processes and each process is assigned a position in the ring. Each process knows who is next in line after itself.

Algorithm :

- In this algorithm it is assumed that all the processes in the system are organized in a logical ring. The figure blow describes the structure.
- The ring positions may be allocated in numerical order of network addresses and is unidirectional in the sense that all messages are passed only in clockwise or anti-clockwise direction.
- When a process sends a request message to current coordinator and does not receive a reply within a fixed timeout, it assumes the coordinator has crashed. It then initializes the ring and process Pi is given a token.
- The token circulates around the ring. It is passed from process k to k+1 in point to point messages. When a process acquires the token from its neighbor it checks to see if it is attempting to enter a critical region. If so the process enters the region does all the execution and leaves the region. After it has exited it passes the token along the ring. It is not permitted to enter a second critical region using the same token.
- If a process is handed the token by its neighbor and is not interested in entering a critical region it just passes along. When no processes want to enter any critical regions the token just circulates at high speed around the ring.
- Only one process has the token at any instant so only one process can actually be in a critical region. Since the token circulates among the process in a well-defined order, starvation cannot occur.
- Once a process decides it wants to enter a critical region, at worst it will have to wait for every other process to enter and leave one critical region.
- The disadvantage is that if the token is lost it must be regenerated. But the detection of lost token is difficult. If the token is not received for a long time it might not be lost but is in use.



Conclusion:

Thus we have studied token ring based mutual exclusion algorithm.