Name: Dawood Sarfraz

Rollno: 20p-0153

Section: BSCS-6B

Distributed Mutual Exclusion

Distributed mutual exclusion is a technique used in distributed computing systems to manage access to shared resources among multiple processes running on different nodes in the network. The goal of distributed mutual exclusion is to ensure that no process is left waiting indefinitely to access the resource.

There are several algorithms are used to implement distributed mutual exclusion including Maekawa & Suzuki-Kasami algos.

Maekawa Algorithm:

The Maekawa algorithm is distributed mutual exclusion algorithm used to ensure that only one process can access a shared resource at a time in distributed computing system.

The Maekawa algorithm is based on the used of token, which is passed from process to process in a circular fashion. When a process wants to access the shared resource, it requests the token from process that currently holds it. The process that holds the token grants access to the shared resource to the requesting process & then releases the token, which is passed on the next process in the circular sequence.

To Ensure that the token is passed in a timely & efficient manner, the Makawae algorithm uses a set of "keepers". Keepers are processes that monitor the token & ensure that it is passed in a timely & correct fashion. If a keeper

defects that the token is lost or has been held for too long by a process, it takes corrective actions to recover the token & ensure it continues to be passed in the correct order.

The Maekawa algo is fault-tolerant, it can handle failures of individual process or keepers. If a process or keepers is fail it recovers.

Suzuki Kasami:

It is based on logical clocks, which are used to order events in the system. Each process in the system maintains a logical clock, which is a non-decreasing counter that represents the order in which events occure in the system.

When a process wants to access the shared resource, it sends a request message to all other processes in the system, along with its current logical clock value. upon receiving a request message, a processes logical clock value, the receiving process grants access to the share resource to the requesting process. otherwise, the receiving process defers its decision until a later time.

In addition, each process maintains a set of outstanding requests which contains the logical clock values of all the processes that have requested access to the shared resource but have not yet been granted access. when a process grants access to the shared resources to a requesting process, it removes the requesting processes logical clock value from its outstanding requests set.

To handle multiple requests simultaneously, the Suzuki Kasami algorithm uses a queue to store requests that cannot be granted immediately. It adds to the queue & waits until it can grant access to the requesting process.