Page No.
Date

~~[illegible]~~ DS Active learning

Name- Rohit Saini
Panel- C
Batch- C2
Rollno- 41
erp- 1032200897

Batch-2

Q1. Explain Singhal's heuristic algorithm.

Ans. Singhal's heuristic algorithm is an approach to building a distributed system that ensures a total ordering of events in distributed system. The primary goal of this algorithm is to order events (such as message deliveries or local actions) consistently across all nodes in the distributed system, even in the presence of network delays node failures or message reordering. This algorithm is designed to be efficient and practical for real-world distributed systems.

The key idea behind Singhal's heuristic algorithm is to use timestamps to order events. Each event is associated with a timestamp, and these timestamps are used to determine the order in which events should be executed or delivered. The algorithm operates as follow:

1. Each node maintains a logical clock, which is incremented locally for each event it generates or receives.
2. when a node sends a message to another node, it includes its current timestamp with the message.
3. Upon a message (receiving), the receiving node compares the timestamp of the incoming message with its own timestamp. It uses the timestamp to order the messages. If the incoming message has a smaller timestamp, it is executed first; otherwise, it is delayed until the local time stamp. 'catch up'.
4. The algorithm also handles cases where timestamps are equal or nearly equal, by using additional heuristics, such as message priorities or message identifiers.

Page No.
Date

This algorithm is a heuristic because it may not always provide a precise total ordering of events in the presence of network delays and other factors. However, it is efficient and practical for many distributed systems and can provide a good approximation of event ordering.

Example: In this example, we have three nodes (A, B and C) in a distributed system, and events are occuring at these nodes.

1. Node A generates an event and assigns a logical timestamp of (1,A) to it, where (1,A) indicates that its first event at Node A.
2. Node A sends a message ~~and increments~~ with the event and its timestamp to Node B.
3. Node B receives the message and increments its own logical clock to (1,B). It compare the timestamp of the incoming message (1,A) with its local timestamp (1,B).
4. Since Node B's timestamp is equal to Node'A timestamp, Node B can execute the event from Node A without any delay.
5. After executing the event Node A, Node B generates its own event with a timestamp of (2,B).
6. Node B sends this event to Node C.
7. Node C receives the msg, increments its logical clock to (2,C) and compares it with the incoming message's timestamp (2,B).
8. Since Node 'C timestamp is greater than the incoming timestamp (2,B > 2,C) it processes the event from Node B and increments its own logical clock to (2,C)

Page No
Date

Q2. Give a suitable Example of Raymonds tree based algorithm.
Ans. Raymond's tree-based algorithm is distributed algorithm used to implement mutual exclusion (mutex) in a distributed system. Mutex is a fundamental synchronization mechanism that ensures that only one process or node can access a shared resources at a time to prevent concurrent conflicts. Raymonds algorithm is particularly suited for distributed systems where processes communicate via message passing.

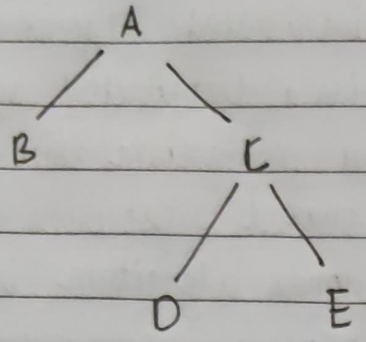
The algorithm uses a tree structure to manage the allocation of resources (critical section) among the nodes. Here's a simplified explanation of how it works:

1. Each node in distributed system is represented as process.
2. The processes are organized in a tree structure, typically rooted at a central node.
3. When a process wants to access the shared resource, it sends a request message. ~~to~~ the parent node in the tree.
4. The parent node forwards the request to its parent, and so on, until the request reaches the root of the tree.
5. The root grants access to the requesting process and sends a reply back down the tree.
6. The requesting process, upon receiving the reply, can access the shared resources.
7. Once the process is done with the resource, it sends a release message, which travels up the tree, allowing the other processes to ~~[illegible]~~ request access.

Raymond's algorithm ensures that only one process can access the resource at any given time while preventing deadlock. It also allows for fairness in resource allocation, as the tree structure ensures that processes eventually get a chance to access the resources.

Page No.
Date

Example



· In this example site A is the primary node that holds the token. The site below node A is the site B and C which are considered child nodes. These sites request the site A to enter the critical section. Site C is again divided into two subdivisions as node D & E.

· Like the above step, node C is the primary node with two sub-nodes as D and E. These nodes send the request to the parent node based on its requirement. As the request raised by nodes B and C is already present, the present request will be set in the queue.

· Based on the token received from the primary to secondary nodes, the sites enter the critical section.