

CASE STUDY NO : 06

Date	1/10/20
Page No.	31

Case Study On :- Study of Dead locking in distributed system.

Deadlocks in Distributed System -

In a distributed system deadlock can neither be prevented nor avoided as the system is so vast that it is impossible to do so. Therefore, only deadlock detection can be implemented. The techniques of deadlock detection in the distributed system require the following.

Some people make a distinction between two kinds of distributed deadlocks: communication deadlocks and resource deadlocks. A communication deadlock occurs, for example, when process A is trying to send a message to process B, which in turn is trying to send one to process C, which is trying to send one to A. There are various scenarios in which this situation leads to deadlock, such as buffers being available. A resource deadlock occurs when processes are fighting over exclusive access to I/O devices, files, locks or other resources.

1. The osflock algorithm (ignore the problem).

2. Detection (let deadlocks occur, detect them, and try recover).
3. Prevention (statically make deadlocks structurally impossible).
4. Avoidance (avoid deadlocks by allocating resources carefully).

Progress -

The method should be able to detect all the deadlocks in the system.

Safety -

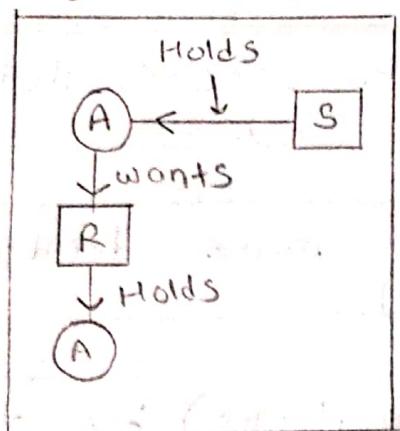
The method should not detect false or phantom deadlocks.

These are three approaches to detect deadlocks in distributed systems. They are as follows:

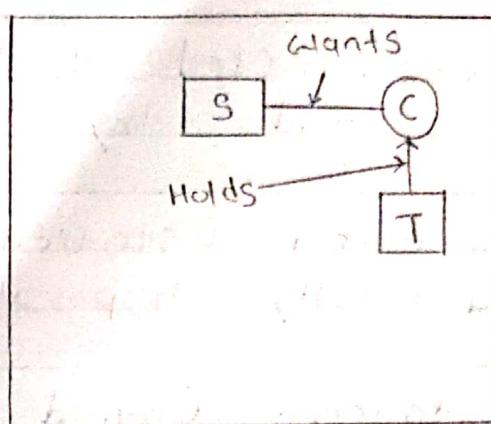
1. Centralized Deadlock Detection -

In the centralized approach, there is only one responsible resource to detect deadlock. The advantage of this approach is that it is simple and easy to implement, while the drawbacks include excessive workload at one node, single-point failure (that is the

Machine 0



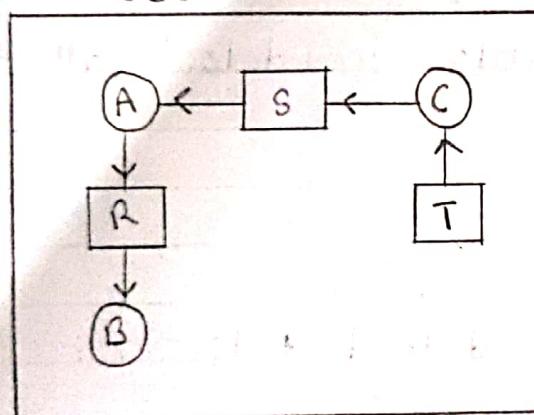
Machine 1



a initial state

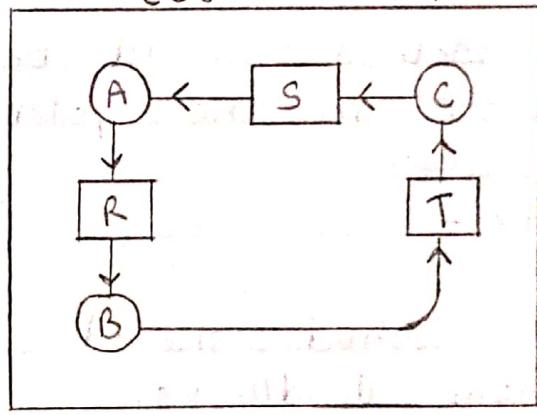
b initial state

Coordinator



c

Coordinator



d

- Initial resource graph for machine 0.
- Initial resource graph for machine 1.
- The coordinator's view of the world.
- The situation after the delayed message.

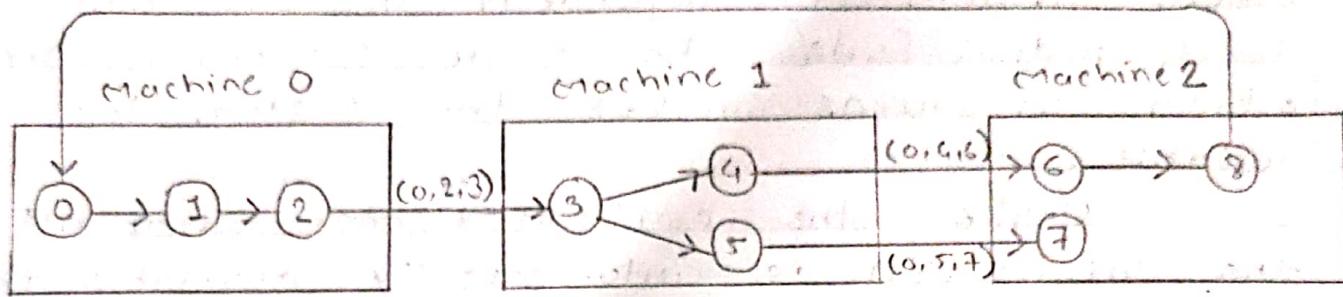
Date	1/20
Page No.	33

whole system is dependent on one node if that node fails the whole system crashes) which in turns makes the system less reliable.

Unlike the centralized case, where all the information is automatically available in the right place, in a distributed system it has to be sent there explicitly. Each machine maintains the graph for its own processes and resources. Several possibilities exist for getting it there. First, whenever an arc is added or deleted from the resource graph, a message can be send to the coordinator providing the update. Second periodically, every process can send a list of arcs added or deleted since the previous update. This method requires fewer messages than the first one. Third, the coordinator can ask for information when it needs it.

2. Distributed Deadlock Detection.

In the distributed approach different nodes work together to detect deadlocks. A single point failure (that is the whole system is dependent on one node if that node fails the whole system crashes) as the workload is equally divided among all nodes. The speed of deadlock detection also increases.



The Chandy - Misra - Haas distributed deadlock detection algorithm

Chandy - Misra - Haas distributed deadlock detection algorithm is a distributed deadlock detection algorithm. It is based on the concept of a global state space. The algorithm uses a distributed consensus mechanism to reach a global agreement on the state of the system. The algorithm consists of two main phases: a pre-computation phase and a detection phase. In the pre-computation phase, each node maintains a local state space and a local history of events. The local state space contains information about the current allocation of resources and the pending requests. The local history contains information about the sequence of events that have occurred since the last update. In the detection phase, each node sends its local state space and local history to all other nodes. Each node then performs a local consistency check to determine if there is a deadlock. If a deadlock is detected, the node sends a message to all other nodes to inform them of the deadlock. The other nodes then perform their own local consistency checks and send messages back to the deadlock node. This process continues until a solution is found or a timeout occurs.

Date	1/120
Page No.	34

Many distributed deadlock detection algorithms have been published. Surveys of the subject are given in Knupp (1987) and Singhal (1989). Let us examine a typical one here, the Chandy - Misra - Haas algorithm (Chandy et al., 1983). In this algorithm, processes are allowed to request multiple resources (e.g., locks) at once, instead of one at a time. By allowing multiple request simultaneously, the growing phase of transaction can be speeded up considerably. The consequence of this change to the model is that a process may now wait on two or more resources simultaneously.

3. Hierarchical Deadlock Detection

This approach is the most advantageous. It is the combination of both centralized and distributed approaches of deadlock detection in a distributed system. In this approach, some selected nodes or clusters of nodes are responsible for deadlock detection and these selected nodes are controlled by a single node.

