Distributed Deadlock Detection

- A distributed system is a network of sites that exchange information with each other by message passing.
- A process can request and release resources in any order, which may not be known a priori and a process can request some resources while holding others.
- If sequence of the allocation of resources to processes is not controlled in such environment, deadlocks can occur.

Preliminaries –

(1) The system model →

The problem of deadlocks has been generally studied in distributed systems under the following model –

- The systems have only reusable resources
- Processes are allowed only exclusive access to resources.
- There is only one copy of each resource.

→ A process can be in two states: running or blocked.

→ In running state (active state), a process has all the needed resources and is either executing or is ready for execution.

→ In the block state, a process is waiting to acquire some resources.

(ii) • Resource Vs Communication Deadlocks -

• There are two types of deadlock. Resource Deadlock and Communication Deadlock.

• In resource deadlocks, processes can simultaneously wait for several resources and can not proceed until they have acquired all those resources.

• A set of processes is resource - deadlocked if each process in the set requests resources held by another process in the set and it must receive all of the requested resources before it can become unblocked.

• In communication deadlocks, processes wait to communicate with other processes among a set of processes.

• A set of processes is communication - deadlocked if each process in the set is waiting to communicate with other process in the set ~~and no process in~~

• And no process in the set ever initiates any further communication until it receives the communication for which it is waiting.

→ 'Wait to communicate' can be viewed as a 'wait to acquire a resource.

system ... ...ch ... other ... resource ... No ... leadi...

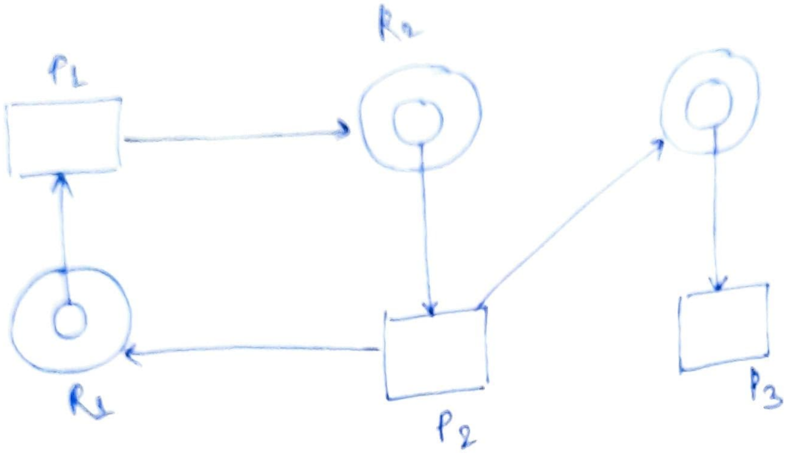
(iii) A graph-theoretic model -

- The state of process-resource interaction can be modeled by a bi-partite directed graph called a resource allocation graph.
- The nodes of this graph are processes and resources of a system.
- And edge of the graph depict assignments or pending requests.
- A pending request is represented by a request edge directed from the node of a requesting process to the node of the requested resource.
- A resource assignment is represented by an assignment edge directed from the node of an assigned resource to the node of the assigned process.
- A system is deadlocked if its resource allocation graph contains a directed cycle or a knot.

Knot → A knot K in a graph is a nonempty set of nodes such that for every node x in K, all nodes in K and only the nodes in K are reachable from x.

$((\forall x \forall y \in K : x \to y)$ AND $(\forall x \in K \; \exists z :: x \to z \Rightarrow z \in K))$

- No node in a knot is a sink or has a path leading to a sink.
- An active process corresponds to sink node in an expedient general resource graph.
- The absence of a knot in an expedient general resource graph does not imply freedom from dead lock.



Deadlock with no knot.

Wait-For Graphs →

- In WFG, nodes are processes and there is a directed edge from node P_1 to node P_2 if P_1 is blocked and is waiting for P_2 to release some resource.
- A system is deadlocked if and only if there is a directed cycle or knot in the WFG.

Deadlock Handling Strategies

- Deadlock handling is complicated because no one site has accurate knowledge of the current state of the system.
- And because every intersite communication involves a finite and unpredictable delay.

There are three deadlock handling strategies –

(i) Deadlock Prevention
(ii) Deadlock Avoidance
(iii) Deadlock Detection

(i) Deadlock Prevention →

- Deadlock prevention is commonly achieved by either having a process acquire all the needed resources simultaneously before it begins execution
- or by preempting a process that holds the needed resources.
- A process requests or releases a remote resource by sending a request message or release message to the site where the resource is located.
- This method has a number of drawback and tends to deadlock.
- For eg. If process P_1 of site S_1 and process P_2 of site S_2 demands resource R_3 and R_4 simultaneously from site S_3 and S_4 respectively.

If site S_3 grants R_3 to P_1 and site S_4 grants R_4 to P_2 then there will be deadlock.

- To prevent deadlock, force processes to acquire needed resources one by one but this method is highly inefficient and impractical.

(ii) <u>Deadlock Avoidance</u> →

- In this, a resource is granted to a process if the resulting global system state is safe.
(Global state includes all the processes and resources of the system)

- Because of the following reason deadlock avoidance can be impractical -

(a) Every site has to maintain information on the global state of the system which requires huge storage requirements.

(b) The process of checking for a safe global state must be mutually exclusive.
Because if several sites concurrently perform checks for a safe global state, they may all find the state safe but the net global state may not be safe.

(c) Due to the large number of processes and resources it will be computationally expensive.

(iii) Dead lock Detection →

safer / an

- Deadlock detection requires an examination of the sta... of process-resource interactions for the presence of cyclical wait.

- Dead lock detection has two favorable conditions :

(1) Once a cycle is formed in the WFG, it persists until it is detected and broken and.

(2) cycle detection can proceed concurrently with the normal activities of a system and hence does not have a negative effect on the system throughput.

Issues In Dead Lock Detection And Resolution

It is categorized in two fields —

(i) Detection of existing deadlocks

(ii) Resolution of detected deadlocks

(I) Detection -

- It involves two issues :

— maintenance of the WFG

— search of the WFG for cycles (or knots)

A correct deadlock detection algorithm must satisfy the following two conditions :-

(a) Progress - No (Undetected deadlocks) - Algorithm must detect all existing deadlocks in finite time

Safety - No false deadlocks →

- The algorithm should not report deadlocks which are non-existent (called phantom deadlocks).
- It is difficult to design a correct dealock detection algorithm because sites may obtain out of date and inconsistent WFGs of the system.
- As a result sites may ~~obtain~~ ~~out~~ detect a cycle that does not exist, but whose different segments were existing in the system at different time.

Resolution →

- Deadlock resolution involves breaking existing wait-for dependencies in the system to resolve the deadlock.
- It involves rolling back one or more processes that are deadlocked and assigning their resources to blocked processes in the deadlock so that they can resume execution.

Centralized Deadlock-Detection Algorithms

- In simplest completely centralized algorithm a site called the control site, maintains the WFG of the entire system.
- Checks it for the existence of deadlock cycles.