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. (103)

- 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) Deadlock Detection →

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

- Deadlock 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 Deadlock 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.

- All sites request and release resources by sending request resource and release resource message to the control site respectively.
- It updates its WFG, when receives resource request and release request.
- Checks the WFG for deadlocks whenever a request edge is added to the WFG.
- Simple but is highly inefficient because all resource acquisition and release requests must go through the control site, even when the resource is local.
- This results in large delays in responding to user requests, large communication overhead, and the congestion of communication links near the control site.
- Reliability is poor because if control site fails, entire system comes to a halt.
- These problems can be mitigated by having each site maintain its resource status (WFG) locally and by having each site send its resource status to a control site periodically for WFG and deadlock detection.
- However due to communication delay and lack of perfectly synchronized clocks, the control site may get inconsistent view and detect false deadlocks.
- For e.g. two resources R_1 and R_2 are stored at sites S_1 and S_2 respectively.

(107)

Suppose the following two transactions T_1 and T_2 are started almost simultaneously at sites S_3 and S_4 respectively:

T_1	T_2
lock R_1	lock R_1
unlock R_1	unlock R_1
lock R_2	lock R_2
unlock R_2	unlock R_2

- Suppose that the lock (R_1) request of T_1 arrives at S_1 and locks R_1 followed by the lock (R_1) request of T_2, which waits at S_1.
- At this point S_1 reports its status $T_2 \Rightarrow T_1$ to a designated site.
- Thereafter, T_1 unlocks R_1, T_2 locks R_1, T_1 makes a lock (R_2) request to S_2.
- T_2 unlocks R_1 and makes a lock (R_2) request to S_2.
- Now suppose that the lock (R_2) request of T_2 arrives at S_2 and lock R_2 followed by the lock (R_2) request of T_1 which wait at S_2.
- At this point S_2 reports its status $T_1 \rightarrow T_2$ to the designated site, which after constructing the global WFG reports a false deadlock $T_1 \rightarrow T_2 \rightarrow T_1$.

The Ho-Ramamoorthy Algorithm

To solve the false detection of deadlock Ho and Ramamoorthy gave two centralized deadlock detection algorithm.

(i) The Two-phase Algorithm -

- In this algo., every site maintains a status table that contains the status of all the processes initiated at that site.
- Status of a process includes all resources locked and all resources being waited upon.
- According to this information, designated site constructs a WFG and searches for cycles.
- If there is no cycle then system is deadlock free.
- Otherwise the designated site again requests status tables from all the sites and again construct a WFG using only those transactions which are common to both reports.
- If the same cycle is detected again the system is declared deadlocked.
- By selecting only the common transactions found in two consecutive reports, they claimed that algo. gets a consistent view of the system.

And if a deadlock exists, it was argued the same wait-for condition must exist in both reports.

- However this claim proved to be incorrect but reduces the probability of getting an inconsistent view.

(ii) The One-Phase Algorithm →

- The one-phase algo. requires only one status report from each site
- However, each site maintains two status tables: a resource status table and a process status table.
- The resource status table at a site keeps track of the transactions that have locked or are waiting for resources stored at that site.
- The process status table at a site keeps track of the resources locked by or waited for by all the transactions at that site.
- Periodically, a designated site requests both the tables from every site, constructs a WFG using only those transactions for which the entry in the resource table matches the corresponding entry in the process table and searches the WFG for cycles.

- If no cycle is found then system is deadlock free otherwise deadlock is detected.

- This algo does not detect false deadlock because it eliminates the inconsistency in state information by using only the information that is common to both tables.

- This eliminates inconsistencies introduced by unpredictable message delays.

- For e.g. If the resource table at site S_1 indicates that resource R_1 is waited upon by a process P_2 (i.e. $R_1 \leftarrow P_2$)

- And the process table at site S_2 indicates that process P_2 is waiting for resource R_1 (i.e. $P_2 \rightarrow R_1$) then edge $P_2 \rightarrow R_1$ in the WFG reflects the correct system state.

- The one phase algo is faster and requires fewer messages as compared to two phase algo.