St. Francis Institute of Technology
Department of Computer Engineering

Academic Year: 2021-2022 **Semester: VIII**
Subject: Distributed Computing **Class / Branch / Division: BE/CMPN/A**
Name: Rebecca Dias Roll Number:18

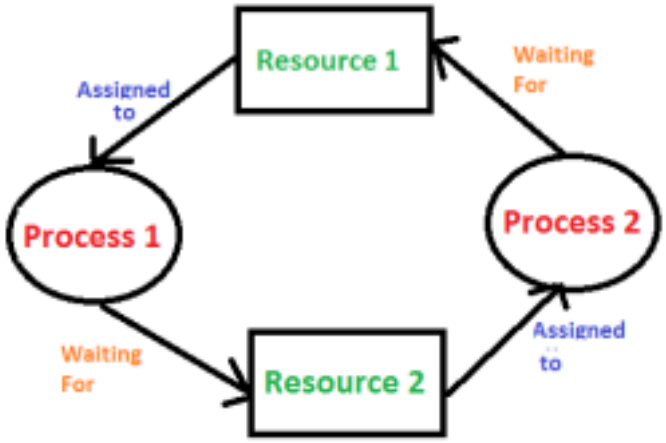
Experiment No: 09

Aim: Implement Deadlock management in Distributed systems

Theory:

1. Deadlock:

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



Three commonly used strategies to handle deadlocks are as follows:

- **Avoidance:** Resources are carefully allocated to avoid deadlocks.
- **Prevention:** Constraints are imposed on the ways in which processes request resources in order to prevent deadlocks.
- **Detection and recovery:** Deadlocks are allowed to occur and a detection algorithm is used to detect them. After a deadlock is detected, it is resolved by certain means.

Necessary conditions for Deadlocks

- **Mutual Exclusion**

  A resource can only be shared in a mutually exclusive manner. It implies, if two processes cannot use the same resource at the same time.

- **Hold and Wait**

  A process waits for some resources while holding another resource at the same time.

- **No preemption**

The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

- **Circular Wait**

  All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

2. Chandy-Misra-Haas's deadlock detection algorithm

Chandy-Misra-Haas's distributed deadlock detection algorithm is an edge chasing algorithm to detect deadlock in distributed systems.

In edge chasing algorithms, a special message called probe is used in deadlock detection. A probe is a triplet (i, j, k) which denotes that process P_i has initiated the deadlock detection and the message is being sent by the home site of process P_j to the home site of process P_k.

The probe message circulates along the edges of WFG to detect a cycle. When a blocked process receives the probe message, it forwards the probe message along its outgoing edges in WFG. A process P_i declares the deadlock if probe messages initiated by process P_i returns to itself.

Algorithm:

Process of sending probe:

1. If process P_i is locally dependent on itself then declare a deadlock.
2. Else for all P_j and P_k check following condition:
   (a) Process P_i is locally dependent on process P_j
   (b) Process P_j is waiting on process P_k
   (c) Process P_j and process P_k are on different sites.

If all of the above conditions are true, send a probe (i, j, k) to the home site of process P_k.

On the receipt of probe (i, j, k) at home site of process P_k:

1. Process P_k checks the following conditions:
   (a) Process $P_k$is blocked.
   (b) $dependent_k[i]$ is *false*.
   (c) Process P_k has not replied to all requests of process P_j
   If all of the above conditions are found to be true then:
   1. Set $dependent_k[i]$ to true.
   2. Now, If k == i then, declare the P_i is deadlocked.
   3. Else for all P_m and P_n check following conditions:
   (a) Process $P_k$is locally dependent on process P_m and
   (b) Process P_m is waiting upon process P_n and
   (c) Process P_m and process P_n are on different sites.
   Send probe (i, m, n) to the home site of process $P_n$if above conditions satisfied.

Thus, the probe message travels along the edges of the transaction wait-for (TWF) graph and when the probe message returns to its initiating process then it is said that deadlock has been detected.

Advantages:

- There is no need for special data structure. A probe message, which is very small and involves only 3 integers and a two dimensional boolean array dependent, is used in the deadlock detection process.
- At each site, only a little computation is required and overhead is also low
- Unlike other deadlock detection algorithms, there is no need to construct any graph or pass nor to

pass graph information to other sites in this algorithm.

- Algorithm does not report any false deadlock (also called phantom deadlock).

Disadvantages:

- The main disadvantage of a distributed detection algorithm is that all sites may not be aware of the processes involved in the deadlock; this makes resolution difficult. Also, proof of correction of the algorithm is difficult.

Implementation:

Code:

```
def DeadLock(initiator):
    print(f"Initiating Chandy Misra Haas Algorithm from Process {initiator} >>>> ")
    flag = False
    prob_msg_list = []
    dead_proc_init = ""
    dead_proc_recv = ""
    print("Probe Message Creation >>>>")
    for i in range(procs):
        for j in range(procs):
            if(resource_array[i][j] == 1):
                prob_msg = (initiator, i, j)
                print(f"Probe Message : {prob_msg} with Initiating
Process:{initiator} and Receiving Process:{j}")
                prob_msg_list.append(prob_msg)
    for i in range(procs):
        for j in range(procs):
            if(prob_msg_list[i][0] == prob_msg_list[j][2]):
                flag = True
                dead_proc_init = prob_msg_list[i][0]
                dead_proc_recv = prob_msg_list[j][2]
    if(flag == False):
        print(f"\nFrom the probe message list we can see that,\nNo Deadlock was not
detected in the system when initiator is process {initiator}")
    else:
        print(f"\nFrom the probe message list we can see that,\nThe Deadlock was
detected in the system when initiator is process {initiator} when the initiating
process is \n{prob_msg_list[i][0]} and Receiving Process is also
{prob_msg_list[i][0]}\n")


procs = int(input("Enter the number of processes >>>> "))
print("Enter the resources of each process >>>> ")
resource_array = [(list(map(int, input().split()))) for i in range(procs)]

for i in range(procs):
    DeadLock(i)
```

Output:

```
PS C:\Users\PC 3\Documents\clare> python exp8.py
Enter the number of processes >>>> 4
Enter the resources of each process >>>>
0 1 0 0
0 0 1 0
0 0 0 1
0 1 0 0
Initiating Chandy Misra Haas Algorithm from Process 0 >>>>
Probe Message Creation >>>>
Probe Message : (0, 0, 1) with Initiating Process:0 and Receiving Process:1
Probe Message : (0, 1, 2) with Initiating Process:0 and Receiving Process:2
Probe Message : (0, 2, 3) with Initiating Process:0 and Receiving Process:3
Probe Message : (0, 3, 1) with Initiating Process:0 and Receiving Process:1

From the probe message list we can see that,
No Deadlock was not detected in the system when initiator is process 0
Initiating Chandy Misra Haas Algorithm from Process 1 >>>>
Probe Message Creation >>>>
Probe Message : (1, 0, 1) with Initiating Process:1 and Receiving Process:1
Probe Message : (1, 1, 2) with Initiating Process:1 and Receiving Process:2
Probe Message : (1, 2, 3) with Initiating Process:1 and Receiving Process:3
Probe Message : (1, 3, 1) with Initiating Process:1 and Receiving Process:1

From the probe message list we can see that,
The Deadlock was detected in the system when initiator is process 1 when the initiating process is
1 and Receiving Process is also 1
```

```
Initiating Chandy Misra Haas Algorithm from Process 2 >>>>
Probe Message Creation >>>>
Probe Message : (2, 0, 1) with Initiating Process:2 and Receiving Process:1
Probe Message : (2, 1, 2) with Initiating Process:2 and Receiving Process:2
Probe Message : (2, 2, 3) with Initiating Process:2 and Receiving Process:3
Probe Message : (2, 3, 1) with Initiating Process:2 and Receiving Process:1

From the probe message list we can see that,
The Deadlock was detected in the system when initiator is process 2 when the initiating process is
2 and Receiving Process is also 2

Initiating Chandy Misra Haas Algorithm from Process 3 >>>>
Probe Message Creation >>>>
Probe Message : (3, 0, 1) with Initiating Process:3 and Receiving Process:1
Probe Message : (3, 1, 2) with Initiating Process:3 and Receiving Process:2
Probe Message : (3, 2, 3) with Initiating Process:3 and Receiving Process:3
Probe Message : (3, 3, 1) with Initiating Process:3 and Receiving Process:1

From the probe message list we can see that,
The Deadlock was detected in the system when initiator is process 3 when the initiating process is
3 and Receiving Process is also 3
```

Conclusion:

In this experiment we studied Deadlock management in Distributed systems. Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. And we also implemented Chandy-Misra-Haas's deadlock detection algorithm in python.