Deadlock Detection in Distributed Systems

Ajay Kshemkalyani and Mukesh Singhal

Distributed Computing: Principles, Algorithms, and Systems

Chapter 10

Introduction

- Deadlocks is a fundamental problem in distributed systems.
- A process may request resources in any order, which may not be known a priori and a process can request resource while holding others.
- If the sequence of the allocations of resources to the processes is not controlled, deadlocks can occur.
- A deadlock is a state where a set of processes request resources that are held by other processes in the set.

System Model

- A distributed program is composed of a set of n asynchronous processes $p_1, p_2, \ldots, p_i, \ldots, p_n$ that communicates by message passing over the communication network.
- Without loss of generality we assume that each process is running on a different processor.
- The processors do not share a common global memory and communicate solely by passing messages over the communication network.

- There is no physical global clock in the system to which processes have instantaneous access.
- The communication medium may deliver messages out of order, messages may be lost garbled or duplicated due to timeout and retransmission, processors may fail and communication links may go down.
- We make the following assumptions:
  - The systems have only reusable resources.
  - Processes are allowed to make only exclusive access to resources.
  - There is only one copy of each resource.

- A process can be in two states: *running* or *blocked*.
- In the running state (also called *active* state), a process has all the needed resources and is either executing or is ready for execution.
- In the blocked state, a process is waiting to acquire some resource.

Wait-For-Graph (WFG)

- The state of the system can be modeled by directed graph, called a *wait for graph* (WFG).
- In a WFG , nodes are processes and there is a directed edge from node P_1 to mode 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 exists a directed cycle or knot in the WFG.

- Figure 1 shows a WFG, where process P_{11} of site 1 has an edge to process P_{21} of site 1 and P_{32} of site 2 is waiting for a resource which is currently held by process P_{21}.
- At the same time process P_{32} is waiting on process P_{33} to release a resource.
- If P_{21} is waiting on process P_{11}, then processes P_{11}, P_{32} and P_{21} form a cycle and all the four processes are involved in a deadlock depending upon the request model.

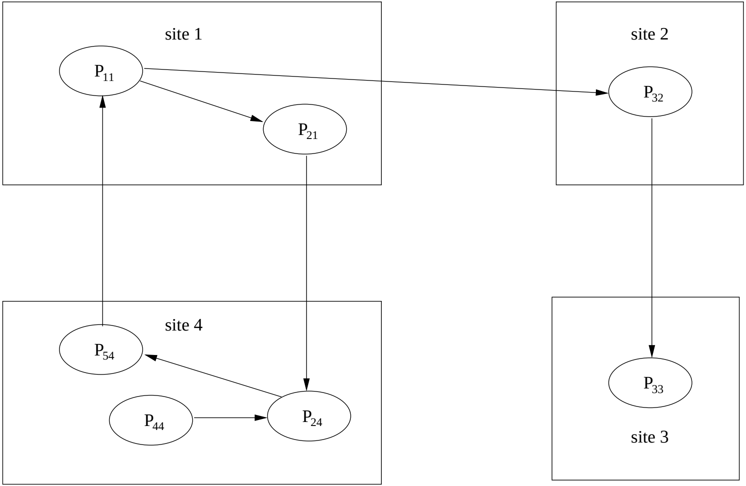

Figure 1: An Example of a WFG

Deadlock Handling Strategies

- There are three strategies for handling deadlocks, viz., deadlock prevention, deadlock avoidance, and deadlock detection.
- Handling of deadlock becomes highly complicated in distributed systems because no site has accurate knowledge of the current state of the system and because every inter-site communication involves a finite and unpredictable delay.
- Deadlock prevention is commonly achieved either by having a process acquire all the needed resources simultaneously before it begins executing or by preempting a process which holds the needed resource.
- This approach is highly inefficient and impractical in distributed systems.

- In deadlock avoidance approach to distributed systems, a resource is granted to a process if the resulting global system state is safe (note that a global state includes all the processes and resources of the distributed system).
- However, due to several problems, deadlock avoidance is impractical in distributed systems.
- Deadlock detection requires examination of the status of process-resource interactions for presence of cyclic wait.
- Deadlock detection in distributed systems seems to be the best approach to handle deadlocks in distributed systems.

Issues in Deadlock Detection

- Deadlock handling using the approach of deadlock detection entails addressing two basic issues: First, detection of existing deadlocks and second resolution of detected deadlocks.

- Detection of deadlocks involves addressing two issues: Maintenance of the WFG and searching of the WFG for the presence of cycles (or knots).

Correctness Criteria: A deadlock detection algorithm must satisfy the following two conditions:

(i) Progress (No undetected deadlocks):

- The algorithm must detect all existing deadlocks in finite time.
- In other words, after all wait-for dependencies for a deadlock have formed, the algorithm should not wait for any more events to occur to detect the deadlock.

(ii) Safety (No false deadlocks):

- The algorithm should not report deadlocks which do not exist (called *phantom or false* deadlocks).

Resolution of a Detected Deadlock

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

Distributed systems allow several kinds of resource requests.

The Single Resource Model

- In the single resource model, a process can have at most one outstanding request for only one unit of a resource.
- Since the maximum out-degree of a node in a WFG for the single resource model can be 1, the presence of a cycle in the WFG shall indicate that there is a deadlock.

The AND Model

- In the AND model, a process can request for more than one resource simultaneously and the request is satisfied only after all the requested resources are granted to the process.
- The out degree of a node in the WFG for AND model can be more than 1.
- The presence of a cycle in the WFG indicates a deadlock in the AND model.
- Since in the single-resource model, a process can have at most one outstanding request, the AND model is more general than the single-resource model.

- Consider the example WFG described in the Figure 1.
- P_{11} has two outstanding resource requests. In case of the AND model, P_{11}shall become active from idle state only after both the resources are granted.
- There is a cycle P_{11}->P_{21}->P_{24}->P_{54}->P_{11} which corresponds to a deadlock situation.
- That is, a process may not be a part of a cycle, it can still be deadlocked. Consider process P_{44} in Figure 1.
- It is not a part of any cycle but is still deadlocked as it is dependent on P_{24}which is deadlocked.

The OR Model

- In the OR model, a process can make a request for numerous resources simultaneously and the request is satisfied if any one of the requested resources is granted.
- Presence of a cycle in the WFG of an OR model does not imply a deadlock in the OR model.
- Consider example in Figure 1: If all nodes are OR nodes, then process P_{11} is not deadlocked because once process P_{33} releases its resources, P_{32} shall become active as one of its requests is satisfied.
- After P_{32} finishes execution and releases its resources, process P_{11} can continue with its processing.
- In the OR model, the presence of a knot indicates a deadlock.

The AND-OR Model

- A generalization of the previous two models (OR model and AND model) is the AND-OR model.
- In the AND-OR model, a request may specify any combination of *and* and *or* in the resource request.
- For example, in the AND-OR model, a request for multiple resources can be of the form x *and* (y *or* z).
- To detect the presence of deadlocks in such a model, there is no familiar construct of graph theory using WFG.
- Since a deadlock is a stable property, a deadlock in the AND-OR model can be detected by repeated application of the test for OR-model deadlock.

The $\binom{p}{q}$ Model

- The $\binom{p}{q}$ model (called the P-out-of-Q model) allows a request to obtain any k available resources from a pool of n resources.
- It has the same in expressive power as the AND-OR model.
- However, $\binom{p}{q}$ model lends itself to a much more compact formation of a request.
- Every request in the $\binom{p}{q}$ model can be expressed in the AND-OR model and vice-versa.
- Note that AND requests for p resources can be stated as $\binom{p}{p}$ and OR requests for p resources can be stated as $\binom{p}{1}$.

Unrestricted Model

- In the unrestricted model, no assumptions are made regarding the underlying structure of resource requests.
- Only one assumption that the deadlock is stable is made and hence it is the most general model.
- This model helps separate concerns: Concerns about properties of the problem (stability and deadlock) are separated from underlying distributed systems computations (e.g., message passing versus synchronous communication).