International Journal of Computer Applications (0975 – 8887)
Innovations in Computing and Information Technology (Cognition 2015)

A Study of Distributed Deadlock Handling Techniques

Garima Rana
Amity University, Haryana

Parveen Kumar
Amity University, Haryana

Kanchan Choudhary
Amity University, Haryana

Dilshana Khurshid
Amity University, Haryana

ABSTRACT
A deadlock is a situation where a process or a set of processes is blocked, waiting on an event that will never occur. In this case of a deadlock, the intervention of a process outside of those involved in the deadlock is required to recover from the deadlock. The formation and existence of deadlocks in a system lowers system efficiency. Therefore, avoiding performance degradation due to deadlocks requires that a system be deadlock free or that deadlocks be quickly detected and eliminated. In this paper, we study deadlock handling strategies in distributed system. Several deadlock techniques based on various control organisations are described. Pros and cons of these techniques are discussed and their performance is compared.

1. INTRODUCTION
A distributed system is a set of autonomous processes that communicate with each other to perform some task. It also includes single machine with multiple communicating processes. In computer systems, many transactions may compete for a finite number of resources at the same time. While the request for a particular resource is ongoing, a transaction may enter a wait state if the request is not granted due to non-availability of the resource. Some time a situation may arise wherein waiting processes may not ever get a chance to change their states. This condition arises when the requested resources are held by other waiting processors. This situation is termed as deadlock [4, 10].

Deadlocks can be handled in three ways:

1.1 Deadlock prevention
In this scheme, all the resources that a transaction requires are pre-declared. This strategy defines that, a request is granted to the transaction, only if, all the resources, it requires are available and the system in turn guarantees that none of these resources would be required by any ongoing transaction. In this approach, all the resources required are reserved in advance. However, no priority must be set for different processes. Deadlock prevention has two obvious disadvantages: First, concurrency is reduced due to pre allocation of resources. Second, evaluation of the safety of the request results in additional overhead. Prevention is the only feasible scheme for handling deadlocks in systems that have no provision for restoring states [5, 10].

1.2 Deadlock Avoidance
The deadlock avoidance says that before starting any transaction, it is not necessary to determine the resources they require. If the requested resources are unavailable for any transaction, still the transaction can proceed. The transactions are allowed to wait for a particular time interval if the requested resource is been occupied by other transaction. In this conflict either the requested

transaction or the victim selection criteria for the abortion of a transaction vary depending on the avoidance scheme used [5, 10]. In distributed systems, the deadlock avoidance scheme is considered as more attractive than prevention scheme because such systems already have ability to abort transactions.

1.3 Deadlock Detection
When conflicts between transactions occur then the requesting transactions are handled by allowing the requesting transactions to wait freely. The outcome of this may be deadlock and hence it must be detected and later resolved. One of the most important tasks performed by the detection algorithm is to find cycles among transactions each waiting for a resource held by the other. To find the deadlock cycles among the transactions, we use directed graph. In the graph, the vertices are marked as transactions and resources, whereas, the edges represent the requests and allocations [9, 10].

2. CONTROL ORGANISATION FOR DISTRIBUTED DEADLOCK DETECTION
Deadlock detection algorithms can be categorized as distributed, centralized or hierarchical. The main demerit of deadlock detection is the additional overhead incurred due to detection of cycles in the graph and abortion and restart of transaction upon detection of deadlocks [9, 10].

2.1 Centralized Control
In the centralized deadlock detection algorithm, a control site (designated site) holds the responsibility of developing the global WFG and finding it for cycles. This control site may maintain the global WFG constantly or it may build it whenever deadlock detection is to be carried out. As compared to other detection algorithms, the centralized deadlock detection algorithms are conceptually simple and easy to implement. However, at a point the centralized deadlock detection algorithm fails. When the sites receive WFG information from all other sites then the communication between the sites get clogged [2, 10].

2.2 Distributed Control
In the distributed deadlock detection algorithm, the selection process of global deadlock is shared among all sites. A distributed deadlock algorithm is different from centralized detection algorithm in the way that this is not more prone to a single point of failure and no site is swamped with deadlock detection activity. In addition to this, a deadlock detection is initiated only a point when a deadlock cycle has a waiting

process. The algorithms of distributed deadlock detection scheme are difficult to design due to non presence of globally shared memory- sites altogether report the presence of a global cycle after observing its segments at different instants. Also the difference between the centralized and distributed also lies in the fact that a number of sites may initiate deadlock detection for the same deadlock in the distributed deadlock control [7, 10].

2.3 Hierarchical Control

In hierarchical deadlock detection algorithms, sites are sequentially arranged in a hierarchical manner and a site detects deadlock containing only its descendant sites. The hierarchical is considered as the best deadlock control scheme as it get the best of both centralized as well as distributed and there is no single point of failure. However, this requires some special care as the sites should be arranged in a hierarchy [10].

3. REVIEW OF SOME LEADING DISTRIBUTED DEADLOCK DETECTION ALGORITHMS

As the name suggests, this algorithm altogether cooperate with all the sites to detect a cycle in the state graph that is used to distribute over many sites of the system. The distributed deadlock detection algorithm can be initiated either by the local site of the process or by the site where the process waits.

There are four main algorithms for distributed deadlock detection scheme:

1. Path pushing algorithm
2. Edge chasing algorithm
3. Diffusion computation algorithm
4. Global state detection algorithm

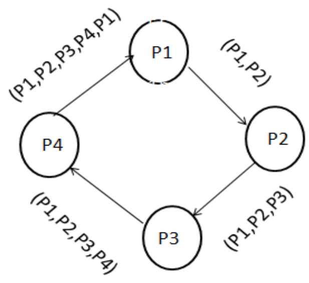
3.1 Path Pushing Algorithm [8]

A path pushing deadlock detection algorithm track a path for the information about the wait for dependencies graph. Obermarck’s algorithm defines a path pushing algorithm. This was designed for distributed database system: hence, transactions are termed as processes denoted by, T1, T2, T3…….Tn.

However a transaction may also include a number of sub transactions that normally execute at different sites.

The Algorithm

Deadlock detection at a site follows the following iterative processes,
Initially the sites wait for the information about the deadlock from all other sites.
The site collects the information and combines it with its local TWF graph and constructs a new updated TWF graph. It then finds all the cycles and breaks only those cycles which do not contain the node ‘Ex’.
The present site transmit the transaction nodes into string form ’Ex->T1->T2->ex’ to all other sites, where a sub transaction of T2 is waiting to receive a message from the sub transaction of T2 at this site.
The path pushing algorithm reduces the traffic of message passing by lexically ordering transactions and sending the string ‘Ex, T1, T2, T3, Ex’ to different sites only if the priority of T1 is higher than T3 in lexical ordering. Obermarck gave an informal correctness proof of the algorithm. It is found that this algorithm is incorrect becauseitdetectsphantom Deadlocks



.

Fig 1: An example of Obermarck’s path-pushing algorithm.

3.2 EDGE CHASING ALGORITHM [3]

Chandy-Misra-Hass’s gave another distributed deadlock detection algorithm which was named as Edge chasing algorithm. This algorithm uses a special message called a probe. If a process requires any resource, it sends a request for it. Unfortunately, resource fails or times out then the process generate a probe message and send it to all the processes holding one or more of its requested resource.

Each probe message include the following information:

- the id of the process that is blocked,
- the id of the process is sending this particular version of the probe message; and,
- the id of the process that should receive this probe message.

The starting process Pj is dependent on the terminal process in a sequence of processes Pj, Pi1, Pi2,….Pim, Pk such that each process is blocked and each process except Pj is waiting for the resources which is being held by other processes. Process Pj is locally dependent upon Process Pk if Pj is dependent upon Pk and both the processes are at the same site.

The Algorithm

The system executes the following algorithm to determine if a blocked process is deadlocked:

```
If Pi is locally dependant on itself
  then declare deadlock.

else for Pk and Pj
```

- Pj is waiting for Pk and,
- Pj and Pk are situated on different sites, transfer probe (i. j, k) to Pk(home site).

Thus along the edges of the global TWF graph the probe is successfully propagated and a deadlock is detected as soon as the probe message returns to its initiating process.

The advantages of probe message are,

1. The message has a fixed length.
2. No false state is found in this algorithm.
3. Computations are very little. Hence, it is very easy to implement.
4. No need for special data structure.

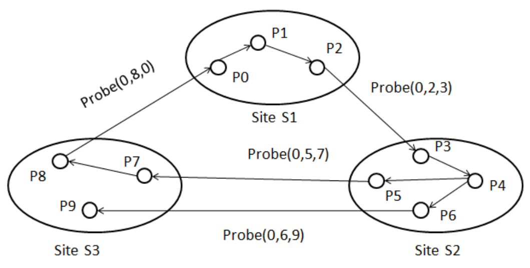


Fig 2: An example of Chandy et al. edge-chasing algorithm.

3.3 A DIFFUSION COMPUTATION BASED ALGORITHM [6]

In diffusion computation based distributed deadlock detection algorithms; deadlock detection computation is diffused through the WFG of the system. Chandy et al.'s distributed deadlock detection algorithm illustrates the technique of diffusion computation based algorithm.

A diffusion computation is initialized by the process to determine if it is deadlocked or not. The message used in diffusion computation is in the form of a query(i,j,k) and a reply(i,j,k), denoting that they belong to diffusion computation initiator by the process Pi and then get transfer from process Pj to process Pk. A process can be in two states: active state or blocked state. In active state, a process is in the executing state whereas in blocked state, the process is in waiting state to acquire a resource. A blocked state broadcasts a message to all the states from whom it is waiting to receive the message. The message is discarded if it is received by the active process whereas if the message is received by a blocked process then the following action takes place:

If this is the first query message received by Pk for the deadlock detection initiated by Pi(called the engaging query), then it propagates the query to all the processes in its dependent set and sets a local variable numk(i) to the number of query messages sent.

If this is not an engaging query, then Pk returns a reply message to it immediately, provided Pk has been continuously blocked since it received the corresponding engaging query. Otherwise, it discards the query.

A local Boolean variable waitk(i) at process Pk denotes the fact that it has been continuously blocked since it received the last engaging query from process Pi.

The Algorithm

Here we describe the Chandy et al.'s diffusion computation based deadlock detection algorithm.

```
**Initiate a diffusion computation for a blocked process Pi:**
  Send query (i, i, j) to all processes Pj in the dependent set
DSi of Pi;
numi(i):= |DSi|; waiti(i):= true;

**When a blocked process Pk receives a query (i, j, k):**
 if this is the engaging query for process Pk
 then send query (i, k, m) to all Pm in its dependent set DSk;

if this the engaging query for process Pk;
 else if waitk(i) then send a reply(i, k, j) to Pj.
**When a process Pk receives a reply (i, j, k).**
If waitk(i)
  then begin
    numk(i):= numk(i) -1;
    if numk(I) = 0
then if i=k then **declare a deadlock**
else send reply (i,k,m) to Pm.
```

3.4 A GLOBAL STATE DETECTION BASED ALGORITHM [1]

Using the global state detection approach, there exist three deadlock detection algorithms to detect the distributed deadlocks. First approach is algorithm by Bracha and Toueg consists of two phases. In the initial phase, the algorithm records a description of distributed WFG and in the second phase, the algorithm regulates the granting of requests to check for various deadlocks. The first phase terminates after the second phase has been terminated because the first phase is nested within the second phase. Now the, the algorithm by Wang et al. also includes two phases. In the first phase a place description of distributed WFG is recorded whereas in the second phase, the static WFG recorded in the initial phase is reduced to detect any deadlocks. This algorithm has one major feature that is, both the phases occur simultaneously.

Now, the algorithm by Wang et al. also includes two phases. In the first phase a description of distributed WFG is recorded whereas in the second phase, the static WFG recorded in the initial phase is reduced to detect any deadlocks. This algorithm has one major feature that is, both the phases occur simultaneously.
The Kshemkalyani- Singhal algorithm consists of only one single phase, which consists of a fan-out sweep of messages outwards from an initiator process and a fan in sweep of messages inwards to the initiator process [5].

4. CONCLUSION

The detection of deadlocks requires performing two tasks: first, maintaining a WFG; second, searching the WFG for cycles. Depending upon the way the WFG is maintained and the way a control to carry out the search for cycles is structured, deadlock detection algorithms are classified into three categories: centralized, distributed, and hierarchical. Distributed deadlock detection algorithms can be divided into four classes; path-pushing, edge-chasing, diffusion computation, and global state detection. In path-pushing algorithms, wait-for dependency information of the global WFG is disseminated in the form of paths. In edge chasing algorithms, special messages called probes are circulated along the edges of the WFG to detect a cycle. When a blocked process receives a probe, it propagates the probe along its outgoing edges in WFG. A process declares a deadlock when it receives a probe initiated by it. Diffusion computation type algorithms make use of echo algorithms to detect deadlock. Deadlock detection messages are successively propagated through the edges of the WFG. Global State Detection based algorithms detect deadlocks by taking a snapshot of the system and by examining it for the condition of a deadlock.

5. REFERENCES

[1] Bracha,G., and S. Toueg, "Distributed Deadlock Detection," Distributed Computing, vol. 2, 1987.

[2] Chandy, K.M., and L.Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems,"ACM Trans. On Computer Systems, Feb.1985.

[3] Chandy, K.M.,J. Mishra, and L.M.Haas."Distributed Deadlock Detection," ACM Trans. On Computer Systems, May 1983.

[4] Herman,T., and K.M. Chandy." A Distributed Procedure to Detect AND/OR Deadlocks." Tech. Report TR LCS-8301, Dept. of Computer Sciences, University of Texas at Austin, 1983.

[5] Kshemkalyani, A.D., and M. Singhal, "Efficient Detection and Resolution of Generalized Distributed Deadlocks," IEEE Transactions on Software Engineering, vol.20, no.1, Jan. 1994.

[6] Menasce, D.E., and R.R. Muntz, "Locking and Deadlock Detection in Distribute Databases," IEEE Trans. On Software Engineering, May 1979.

[7] Mitchell, D.P., and M.J. Merritt, " A Distributed Algorithm for Deadlock Detection and Resolution,"Proc. Of the ACM Conference on Principles of Distributed Computing, Aug.1984.

[8] Obermarck,R."Distributed Deadlock Detection Algorithm,"ACM Trans. On Database Systems,June 1982.

[9] Singhal,M.,"Deadlock Detection in Distributed Systems," IEEE Computer, Nov..1989.

[10] Singhal., Shivaratri. "Advanced Concepts in Operating Systems" McGrawHill, 1994.