IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.
www.ijiset.com
ISSN 2348 – 7968

A Survey on Distributed Deadlock Detection Algorithm and its performance Evolution

Sailen Dutta Kalita[1], Minakshi Kalita[2],Sonia Sarmah[3]
[1,2,3] Department of Computer Science and Engineering and Information Technology,
School of Technology, Assam Don Bosco University, Azara,Guwahati,Assam,India
E-mail:[1]sailen.dk@gmail.com.[2]minakshi0991@gmail.[3]sonia.sarmah@gmail.ac.in

Abstract- Deadlock is one of the most serious problems in distributed systems environment and detection of deadlock has undergone extensive study. A deadlock is a condition in a system where a process cannot proceed because it needs to obtain a resource held by another process which itself is holding a resource that the other process needs .In literature, various techniques have been discussed so far which are used to prevent, detect and resolve the deadlocks. In this paper we survey the various detection algorithms and their performance in distributed environment. The paper introduces a uniform framework for the discussion of these algorithms.

1. INTRODUCTION

Deadlock is an important issue in distributed systems. Two type of deadlock have been discussed in the literature: Resource deadlock and communication deadlock. In resource deadlocks, processes can simultaneously wait for several resources and cannot proceed until they have acquired all those resources. In communication deadlocks, processes wait to communicate with other processes among a set of processes and no process in the set ever initiates any further communication until it receives the communication for which it is waiting. There are three strategies to handle deadlocks: Deadlock prevention, Deadlock avoidance and Deadlock detection.

i) **Prevention:** Guaranteeing that deadlocks can never occur. It is inefficient as it decreases the system concurrency and in many systems, future requirements are unpredictable.

ii) **Avoidance:** Detecting potential deadlocks in advance and taking action to insure that deadlock will not occur. It is impractical because of huge storage requirements and expensive communication costs.

iii) **Detection:** Allowing deadlocks to form and then finding and breaking the deadlocks. It does not have a negative effect on system throughput. The detection of deadlock s involves two issues: Maintenance of the WFG and search of the WFG for the presence of cycles (or knot). A correct deadlock detection algorithm must satisfy the following two conditions:-

Progress: No undetected deadlocks. The algorithm must detect all existing deadlocks in finite time.

Safety: The algorithm should not report deadlocks which are non-existent (called phantom deadlocks). It is very difficult to design a correct deadlock detection algorithm in distributed system because sites may obtain out of date and inconsistent WFGs of the system as a result sites my detect a cycle that does not exist, but whose different segment were existing in the system in different times. This is the primary reason why many deadlock detection algorithms reported in the literature are incorrect.

2. Survey of Distributed Deadlock Detection Algorithm:

Numerous algorithms in distributed deadlock detection have been proposed in the literature**,** Depending upon the manner in which WFG information is maintained and the search for a cycles is carried out deadlock detection algorithm are fall into three categories: Centralized control, Distributed control, and Hierarchical control

Centralized control: In Centralized deadlock detection algorithms, a designated site (control site) has the responsibility of constructing the global WFG and searching it for cycles. The control site may maintain the global WFG constantly or may build it whenever deadlock detection is to be carried out by soliciting the local WFG from every site.

Deadlock resolution is simple –the control site has complete information about the deadlock cycle and it thus optimally resolves the deadlock. However it have a single point of failure and communication links near the control site are likely to be congested because the control site receives WFG information from all the sites.

Distributed control: In Distributed deadlock detection algorithms, the responsibility for detecting a global deadlock is shared equally among all sites. These algorithm are difficult to design due to the lack of globally shared memory-sites may collectively report the existence of a global cycle after seeing its segments at different instants, In addition, deadlock resolution is difficult in Distributed deadlock detection algorithms, as several sites can detect the same deadlock and not be aware of the other sites or processes involved in the deadlock.

Hierarchical control: In Hierarchical deadlock detection algorithms, sites are arranged in a hierarchical manner and site detect deadlock involving only its descendant sites. They are the best of both Centralized control and distributed control in that here no single point failure and a site is not bogged down by deadlock detection activities with which it is not concerned. However, Hierarchical deadlock detection algorithms require special care while arranging the sites in a hierarchy.

In this section, a few algorithms for distributed deadlock detection are surveyed.

Centralized Versus Distributed Deadlock Detection:

There are a number of reasons why distributed deadlock detection seems more attractive than a centralized scheme, that is, one in which a single agent is responsible for deadlock detection. First, a centralized deadlock detection algorithm is vulnerable to failures of the central detector. Hence special provisions for this kind of faults have to be made, resulting in long delays until a new central agent is determined and supplied with up-to-date wait-for information. Distributed algorithms deal with these kinds of problems in a much more natural way. Furthermore, because of the heavy traffic to and from the central agent, this agent can constitute a performance bottleneck, limiting the overall performance of the DBS. More evidence for the superiority of distributed schemes is supplied by the observation that for typical applications most WFG cycles are very short. Bernstein give theoretical reasons for the pre- dominance of short paths in WFGs. In particular, for most applications over 90% of WFG cycles can be expected to be of length 2. The same figure also appears in an empirical study .The observation that deadlock cycles are short makes centralized deadlock detection an even less attractive choice. With a global algorithm there may be a significant time and message overhead in assembling all the local WFGs at the global detector. Thus, a distributed deadlock might go undetected for quite a while. Since most deadlocks involve only two sites, they can detect the deadlock more efficiently by communicating directly. Mitchell and Merritt [1984] present a fully distributed deadlock detection algorithm that has a very simple and appealing correctness proof and that, according to the authors, had been implemented in a DBS in less than an hour.

2.1 Centralized Algorithms:

Ho-Ramamurthy 2-phase Algorithm: It is a centralized algorithm. Each site maintains a status table for all processes initiated at that site, i.e., it records all the blocked resources and all the resources being waited on. After that the Controller sends requests for the status table to each of the sites[12]. Then the Controller creates Wait-for-graph using the tables from all sites to check whether there exists cycle or not. If no cycle exists, that means no deadlock. If cycle exists, the Controller again requests for status table to create WFG based only on the common transactions in the 2 tables. If the same cycle is detected again, then the system is deadlocked. But this algorithm detects false deadlock.

Ho-Ramamurthy 1-phase Algorithm: It is a centralized algorithm. Each site maintains 2 status table-one for Resource status, another for Process Status[12]. The Controller collects all the tables from each of the sites and creates WFG to check whether there exists cycle or not. If cycle do not exists, that means no deadlock; if exists, it is deadlock.

2.2 Distributed Algorithm:

Chandy-Misra-Haas Algorithm: Each of the processes requests for multiple resources, as a result process may waiting for two or more than two resources at the same time[10]. At this time, the waiting process generates a probe message containing-the process being blocked, the process sending the message and the process receiving the message, and sent it to the process holding the resource. After receiving of the probe message, the recipient checks to see whether itself is waiting for any process. If so, the probe message is updated by changing the last two data and keeping the first data same, and sent it to another process holding the resources. If this message goes around all the way and comes back to the

original sender, which initiates the probe message, that means there exists a cycle, i.e., there is a deadlock.

Obermarck's Algorithm: It is a path-pushing distributed deadlock algorithm. Each of the sites maintains local Wait-for-graph[14]. In this case, processes are called Transactions-t1, t2, t3 …. etc. Transactions consist of a number of sub transactions and at each point in time only one sub transaction is active. Transactions are totally ordered. There is special virtual node called "*Ex*", which abstracts remote sites in the WFG. Upon receiving deadlock detection information, a site constructs received info with its local WFG detected and resolves all local deadlocks (which do not contain node Ex). For all cycles (Ex, t1, t2, …, tk, Ex) it sends this path to all other sites where a sub transaction of t1, t2, .., tk may be waiting to receive a message from a local sub transaction (and remote sub transaction at that site has higher priority than local sub transaction). In this algorithm, Phantom Deadlock is detected.

Bracha's Algorithm: In this paper two algorithms are discussed, First one for dynamically changing systems and the other for a static system where transmission delays are ignored for simplification[4]. The algorithms use a model similar to the AND/OR in which transactions can request any M available from a pool of N resources. An OR request corresponds to M = 1 while an AND request corresponds to M = N.

The static algorithm consists of two phases: A notification phase, in which transactions are notified that a deadlock detection algorithm has started, and a granting phase, in which active transactions simulate granting of requests. Deadlocked nodes are the nodes that are not made active by the second phase. For static systems with messages in transit, the authors have used colored wait-for graph to take into account messages in transit in the communication channels, and to represent a static snapshot of the ongoing activities in the system.

Mitchell's Algorithm: This is an edge chasing algorithm in which each transaction uses a public and private label for deadlock detection[3]. Initially these labels for all transactions have the same value. The scheme is explained in the following four steps:

i) Block Step: When a transaction becomes blocked waiting for a second transaction, both of its labels are incremented to a value greater than that of the blocking transaction.

2) Active Step: A transaction becomes active when it gets a resource, times out or fails, or when the owner of a resource changes.

3) Transmit Step: When a blocked transaction discovers that its public label is smaller than that of the blocking transaction, it changes its label to a value equal to that of the blocking transaction.

4) Detect Step: When a transaction receives its own public label back, a deadlock is detected. When a transaction begins to wait for a resource held by another transaction, it executes the block step. When a transaction becomes active, it executes the active step. Periodically,

the blocked transactions read the public label of the blocking transaction. Based on the ordering between this label and its own public label, the transmit step explained above is executed. Due to the transmit step, the largest public label migrates in the opposite direction along the edges of the wait-for graph.

This algorithm is easily implemented and does not detect false deadlocks in the absence of process failures. It also can detect all existing deadlocks.

Kawazu's Algorithm : The authors divide this algorithm into two phases. In the first phase local deadlocks are detected, and in the second phase global deadlocks are detected in the absence of local deadlocks[15]. The local deadlock detection step is initiated when a transaction begins to wait for a resource. If no local deadlock is detected, the detection of possible global deadlocks begins. To detect global deadlock, the local wait-for graphs are gathered to construct a pseudo-wait-for graph. This graph does not necessarily represent the true status of transactions. This scheme suffers from phantom deadlocks, because each local wait-for graph is not collected at the same time due to communication delays. Also, in case a transaction simultaneously waits for more than one resource, some global deadlocks may go undetected since the global deadlock detection is initiated only if no local deadlock is detected.

Brian M. Johnston Algorithm: In this paper Brian M. Johnston proposed an algorithm that uses an update message The function of update message is two fold: first to modify the Wait-for variables and second to check the occurrence of deadlock[13]. As compared to many recent algorithms in this field, the proposed algorithm can detect the most frequent deadlocks with minimum message passing.

With message complexity defined as the number of messages transmitted between initiating a update message and detecting the cycle. In the worst case, in a system of n transactions, the overall message complexity of the proposed algorithm is O(n).

B. M. Alom Algorithm: B. M. Alom has introduced a deadlock detection and resolution technique using the concept of priorities. In this technique a priority based table is used[11]. This algorithm maintains a list of all the transactions, and whenever a deadlock cycle is detected, the priorities of the transactions constituting the deadlock is checked. The transaction with the least priority is aborted so that the resources held by it can be set free and can be granted over to the waiting transactions. But it has been found that if the order of the priorities is changed this algorithm fails to detect deadlocks. B. M. Alom has introduced a deadlock detection and resolution technique using the concept of priorities. In this technique a priority based table is used. This algorithm maintains a list of all the transactions, and whenever a deadlock cycle is detected, the priorities of the transactions constituting the deadlock is checked. The transaction with the least priority is aborted so that the resources held by it can be set free and can be granted over to the waiting transactions. But it has been found that if the order of the priorities is changed this algorithm fails to detect deadlocks.

Chim-fu yeung Algorithm: He presented an improvement over the algorithm proposed by Chandy & Mishra in which Deadlock is found by passing special messages called probe messages along the edges of a wait-for graph[8]. This process has several advantages as compared to Chandy's algorithm, such as:

i) This algorithm is error free.

ii) It suffers very little performance degradation as compared to the original one.

iii) Even for large values of multi-programming level, probe-based algorithm can outperform time–out.

iv) The rate of probe initiation is a dominant factor in determining system's performance.

In this algorithm he focus on clearing the dependency table regularly. If the dependency tables are cleared periodically then the number of probe messages will be great as many probes needed to be propagated. As the controller initiates the probes periodically, the number of probe messages could be used to estimate the total blocking time period of a process.

3. Performance Evolution:

We have seen a lot of algorithms to detect Dead lock But some of the algorithm have some drawbacks like in case of Chandy and Mishra's algorithm, it may detect false deadlock. Even in this algorithm a lot of performance degradation is seen whereas if we compare it to the algorithm proposed by Chim –fu-Yeung, we seen that algorithm is error free. It suffers very little performance degradation as compared to the original one. The rate of probe initiation is a dominant factor in determining system's performance.

Similarly, when we see Kawaju's algorithm , it is realized that it has problem of detecting phantom deadlocks. The algorithm proposed by B.M.Alom has made use of the priority based technique and we have analyzed that If any other order of priorities of the transactions is taken instead of the one he has taken in his paper then deadlock cannot be detected or it detects false deadlocks.

The algorithm proposed by Brian M Jhonston make use of update message but in this there are no criteria of deciding that which transaction needs to be aborted. They are simply aborting the transaction for which the intersection values of Wait_for variable and Request_Q variables are not NIL. But we think it is not a fair decision t abort any transaction like this, without even realizing how old that transaction is and to what extent is the system dependent on it and thus we think that some other concept has to be used here so that we abort the youngest transaction.

Table:1.Performance evolution

Algorithm	No. of messages	Detection Delay	Message Length
Ho-Ramamoorthy Algorithm	N/A	O(n)	N/A
Obermarck's Algorithm	n(n-1)/2	*O*(n)	O(n)
Chandy-Misra-Haas's Algorithm	m(n-1)/2	*O*(n)	3-word

Brian M. Johnston Algorithm	N	*O(n)*	N/A

4. Conclusion:

A large numbers of algorithms were published addressing the problem of distributed deadlock detection .We believe that recent developments in the area of distributed deadlock detection algorithms has rendered much of the older work obsolete. In this paper we survey the various detection algorithms and their performance in distributed environment .Here we focused on a small number of concepts and does not claim completeness in the sense that all known approaches have been covered.

REFERENCES

[1] Christian Lambertz, ,Mila Majster-Cederbaum,, "Efficient deadlock analysis of component – basedsoftwarearchitectures,"DepartmentofComputerScience,UniversityofMannheim,68131 Mannheim, Germany.

[2] Andrews G.R. and Levin G.M., "On-The-Fly Deadlock Prevention". ACMSIGACT-SIGOPS Symposium on Principles of Distributed Computing 1982. 165-172.

[3] Badal D.Z. and Gehl M.T., "On Deadlock Detection in Distributed Computing Systems". IEEE INFOCOM 1983.

[4] Haas L.M., "Two Approaches to Deadlock in Distributed Systems", Ph.D.Thesis.Department of Computer Science. The University of Texas at Austin. August 1981.

[5]Goldman B., "Deadlock Detection in Computer Networks", MIT/LCS/TR-185. Sep 1985

[6] HIMANSHI GROVER, SURESH KUMAR, "ANALYSIS OF DEADLOCK DETECTION AND RESOLUTION TECHNIQUES IN DISTRIBUTED DATABASE ENVIRONMENT" at IJCES. Sep 2012

[7] Brian M. Johnston, Ramesh Dutt Javagal: Ajoy Kumar Datta, Sukumar ghosh, "A Distributed Algorithm for Resource Deadlock Detection", Department of Computer Science , IEEE , pp-252 -256 ,1991

[8] Chim fu Yeung , "A new distributed deadlock detection algorithm for distributed databases systems", Department of Computer science, pp-506 -510, 1983

[9] R. Obermarck, "Distributed Deadlock Detection Algorithm," ACM Transaction on Database Systems, vol. 7:2, pp. 187-208, 1982

[10] Chandy X. M. and Mishra J., "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems " in ACM, 1982

[11] B.M. Monjurul Alom, Frans Henskens, Michael Hannaford "Deadlock Detection Views of Distributed Database", IEEE Sixth International Conference on Information Technology: New Generations, Page(s):730–737, 2009

[12] G. S. Ho and C. V. Ramamoorthy, "Protocols for Deadlock Detection in Distributed Database Systems" IEEE Transaction on Software Engineering, vol. 8:6, pp. 554-557, 1982.

[13] Brian M. Johnston, Ramesh Dutt Javagal: Ajoy Kumar Datta, Sukumar ghosh, "A Distributed Algorithm for Resource Deadlock Detection", Department of Computer Science , IEEE , pp-252 -256,1991

[14] R. Obermarck, "Distributed Deadlock Detection Algorithm," ACM Transaction on Database Systems, vol. 7:2, pp. 187-208, 1982

[15] S. Kawazu, S. Minami, K. Itoh, and K. Teranaka, "Two-Phase Deadlock Detection Algorithm in Distributed Databases" in IEEE, 1979

Biographies of Author

First Author: SAILEN DUTTA KALITA

Qualification:

1. M.Tech Pursuing ' Department of Computer Science and Engineering and Information Technology,

School of Technology, Assam Don Bosco University, Azara,Guwahati,Assam,India

2. B.Tech,year-2011,North-Eastern Regional Institute of Science andTechnology

Publication:

1. **Attacks and countermeasures in Mobile AD HOC Network-An analysis,** International Journal On Advanced Computer Theory And Engineering (IJACTE), ISSN (Print): 2319-2526, Volume -4, Issue -3, 2015

Current research interests:

1. Computer network and information security

Second Author: MINASKHI KALITA

1. M.Tech pursuing , ' Department of Computer Science and Engineering and Information Technology,

ISSN 2348 – 7968

School of Technology, Assam Don Bosco University, Azara,Guwahati,Assam,India
2. BE,year-2013.IST,Guwahati Univesity

Third Author: SONIA SARMAH

1. Designation: Assistant Professor,′ Department of Computer Science and Engineering and Information Technology, School of Technology, Assam Don Bosco University, Azara,Guwahati,Assam,India