A Review of Distributed Deadlock Detection Techniques based on Diffusion Computation Approach

Sonia Singh
Department of Computer Science and Engineering
Manav Rachna International University
Faridabad, India

S. S. Tyagi, PhD.
Department of Computer Science and Engineering
Manav Rachna International University
Faridabad, India

ABSTRACT
A deadlock is a system state in which every process in some group requests resources from other processes in the group, and then waits indefinitely for these requests to be satisfied. Deadlocks have a very adverse effect on the efficient working of operating system therefore they should be either prevented, avoided or if exist should be detected and resolved. Because distributed systems are more vulnerable to deadlocks, the problems of deadlock detection and resolution have long been considered important problems in such systems. This paper provides a comprehensive review of the some of the existing techniques for deadlock detection in distributed environment.

General Terms
Distributed systems, Deadlocks, Cycles and Knots, Algorithms

Keywords
Deadlock Detection, Distributed environment, Diffusion computation

1. INTRODUCTION
A distributed system can be visualized as a set of sites, each site consisting of a number of independent processes. No process knows the global state of the whole system. The processes communicate through messages. The communication is asynchronous and a message may take an arbitrary but finite time. A distributed operating system is one that looks to its user like an ordinary centralized operating system but runs on multiple, independent central processing units (CPUs). The key concept here is transparency. In other word, the use of multiple processors should be invisible to the user. A distributed operating system (OS) is an operating system is a generalization of a traditional operating system. A distributed OS provides the essential services and functionality required of an OS, adding attributes and particular configurations to allow it to support additional requirements such as increased scale and availability. To a user, a distributed OS works in a manner similar to a single-node, monolithic operating system. That is, although it consists of multiple nodes, it appears to users and applications as a single-node.

In a distributed system if a process needs a resource on another site for its computation, it sends a message to the manager of that site through a communication network to request access to the resource. If the resource is available, it will be granted to the requesting process; otherwise, the requesting process may need to wait until the resource is acquired. In this environment, a deadlock may occur in which processes involved in the deadlock are waiting indefinitely in a circular fashion until a special action is taken.

A deadlock is a system state in which every process in some group requests resources from other processes in the group, and then waits indefinitely for these requests to be satisfied. Because distributed systems are vulnerable to deadlocks, the problems of deadlock detection and resolution have long been considered important problems in such systems. To detect deadlocks in a distributed system, the global state of the system is commonly modeled by a logical structure called the *wait-for graph* (WFG). *A WFG* is a directed graph, in which a vertex represents a process, and an edge ***(i, j)*** indicates that process ***i*** has requested a resource from process j, and j has not granted the request. Detecting generalized deadlocks in distributed systems is a difficult problem, because it requires detection of a complex topology in the global WFG.

2. MODELS OF DEADLOCKS
Depending on the applications, system allows a number of different kinds of resource request. Based on the underlying resource-request models [1,7], the deadlock detection algorithms are classified into following categories-

2.1 Single Request 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.

2.2 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.

2.3 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. In the OR model, the presence of a knot indicates a deadlock.

2.4 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.

2.5 P-out-of-Q Model

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, P-out-of-Q model l lends itself to a much more compact formation of a request. Every request in the P-out-of-Q model can be expressed in the AND-OR model and vice-versa. Note that AND requests for p resources can be stated as P-out-of-P model and OR requests for p resources can be stated as P-out-of-1. This model is also known as *Generalized Model*. Hence, a process resource request is expressed as a predicate involving the required resources and the logical AND and OR operators. Since AND and OR model are the special case of P out-of Q model, it is also referred as the Generalized Model. The generalized request model is quite common in many domains including resource management in distributed operating systems, communicating sequential processes and quorum consensus algorithms in distributed databases [11,12,16]. A deadlock in the generalized model is referred as the generalized deadlock. Since the existence of cycle or knot in the WFG are insufficient to determine a deadlock in the generalized model, it is very difficult to detect as well as resolve compared to the AND and OR deadlock. Hence, very few generalized deadlock detection and resolution algorithms [4,5,7,8,10,12,15,16] have been proposed in the literature.

3. TYPES OF DEADLOCK DETECTION ALGORITHMS

3.1 Path Pushing

The first distributed algorithms for the deadlock problem maintained the notion of an explicit global WFG, which had worked so well in the centralized case. In path-pushing algorithms, distributed deadlocks are detected by maintaining an explicit global WFG. The basic idea is to build a global WFG for each site of the distributed system. In this class of algorithms, at each site whenever deadlock computation is performed, it sends its local WFG to all the neighboring sites. After the local data structure of each site is updated, this updated WFG is then passed along to other sites, and the procedure is repeated until some site has a sufficiently complete picture of the global state to announce deadlock or to establish that no deadlocks are present. This feature of sending around the paths of global WFG has led to the term path-pushing algorithms. Many of these algorithms were found to be incorrect as they were not able to detect true deadlocks or detected phantom deadlocks.

3.2 Edge Chasing Algorithms

In an edge-chasing algorithm, the presence of a cycle in a distributed graph structure is be verified by propagating special messages called probes, along the edges of the graph. These probe messages are different than the request and reply messages. The formation of cycle can be detected by a site if it itself receives the matching probe sent by it previously. Whenever a process that is executing receives a probe message, it discards this message and continues. Only blocked processes propagate probe messages along their outgoing edges. Main advantage of edge-chasing algorithms is that probes are fixed size messages which are normally very short.

3.3 Global State Detection Based Algorithms

A consistent snapshot of a distributed system can be obtained without freezing the underlying computation and If a stable property holds in the system before the snapshot collection is initiated, this property will still hold in the snapshot. Therefore, distributed deadlocks can be detected by taking a snapshot of the system and examining it for the condition of a deadlock. This type of algorithms fall into the category of Global state detection.

3.4 Diffusion Computation Based Algorithms

In diffusion computation based distributed deadlock detection algorithms; deadlock detection computation is diffused through the WFG of the system. These algorithms make use of echo messages to detect deadlocks. This computation is superimposed on the underlying distributed computation. If this computation terminates, the initiator declares a deadlock. To detect a deadlock, a process sends out messages along all the outgoing edges in the WFG. These messages are successively propagated (i.e., diffused) through the edges of the WFG.

The general idea behind diffusion computation approach is this: A process determines if it is deadlocked by initiating a diffusion computation. The messages used in diffusion computation take the form of a query (i, j, k) and a reply (i, j, k), denoting that they belong to a diffusion computation initiated by a process Pi and are being sent from process Pj to process Pk. If an active process receives a query or reply message, it discards it. When a blocked process Pk receives a query (i, j, k) message, it takes the following actions: If this is the first query message received by Pk for the deadlocked 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 num k(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 wait k(i) at process Pk denotes the fact that it has been continuously blocked since it received the last engaging query from process P i .

ig.1 shows three sites having different processes. A blocked process say P1 initiates the algorithm and sends a probe (1, 1, 2) and P2 passes it to P3 as probe(1, 2, 3) which forwards it to P4 on another site as probe(1, 3, 4). This probe keep on going forward by changing the corresponding values and at last arrives back to P1 as probe(1, 9, 1) from P9. Therefore P1 declares deadlock as the probe has reached to the same node which initiated the algorithm.This is a very general approach of diffusion computation and is known as Chandy-Misra-Haas Algorithm.

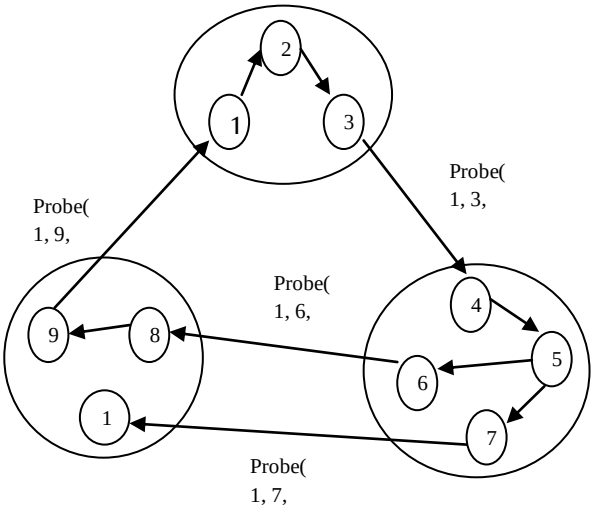

Fig 1: Example of Diffusion Computation

Many variations to this approach has been done by different researchers. Some of them are discussed in the following section

4. DEADLOCK DETECTION BASED ON DIFFUSION COMPUTATION

Usually speaking, the edge chasing and globe state detection first built blocking state table or snapshot at every node and then searched deadlocks among them, but the two method needed a lot of message exchanging to update the table or snapshot momentarily, so only several instances were proposed in this field.

Most of the proposed distributed deadlock detection algorithms belong to the path pushing and diffusion computation category. The difference of the two detection method are that the being detected node in the former method only forwarded the probes downward or upward, it fits for the cycle detection, while in the later method, the nodes have to send the echo messages upward additionally, it fits for the knot detection. Both of the path pushing and diffusion computation have the WFG assumption implicitly: the blocking state of the system is simulated as a graph, where the nodes and the direct edges represents the blocking process and the waiting relations between the processes. The detection messages (probe, label, token and deadlock detection agent) were created by the nodes and propagated along the edges to find the closed cycle or knots.

Various algorithms based on diffusion computation are discussed below:

4.1 Gabriel Bracha and Sam Toueg's Algorithm

Bracha-Toueg presented a distributed algorithm in [4] to detect deadlock in a WFG G=(V, E) modeling a static system with instantaneous communication. G represents a static "snapshot" of the system, therefore it does not changes during the execution of algorithm.

The algorithm starts when some node, called initiator, suspects that it is deadlocked. The algorithm consists of two phases: *Notify*: in which processes are notified when a deadlock detection algorithm is started, and *Grant*: in which active processes simulates the granting of requests. All the processes that are made active as a result of this also simulate the granting of requests. Deadlocked nodes are those nodes which are never made "active" by Grant. The Grant phase is nested within the Notify phase which ensures that the Notify phase terminates only after the Grant phase is over. In order to distinguish between invocations started by different nodes, all the messages that are sent are tagged with the initiator identity. This helps to manage several instances of the algorithm running concurrently in the distributed systems.

4.2 Wang J Huang's Algorithm

Wang Huang developed another algorithm [5] which also contains two phases. The difference is that in this algorithm there is an explicit termination technique is used to detect the end of the first phase. The second phase begins only after the first phase is finished.

Wang gave centralized algorithm to constructs the local Wait-For Graph (LWFG) to determine a deadlock in the system. The initiator of the algorithm in sends a probe to all processes in its reachable set exactly once and collects the replies. It then incrementally constructs the LWFG based on the information in replies. The initiator of the algorithm infers ancestor descendent relationship among the nodes in replies to build the LWFG to determine a deadlock. In addition, it uses path encoding technique to minimize the message. Although it reduces the time complexity to 3d+1, it needs 6e messages to detect the deadlock where e is the number of edges and d is the diameter of WFG.

4.3 Kshemkalyani's Algorithm

Kshemkalyani presented an efficient distributed algorithm [10-11] to detect generalized deadlocks in replicated databases. The algorithm performs reduction of a distributed wait-for- graph (WFG) to determine the existence of a deadlock. The algorithm initiated by an initiator consists of two concurrent sweeps-an outward sweep that records the WFG and an inward sweep that reduces the WFG to a deadlock. The outward sweep induces a spanning tree in the WFG. Reduction is performed by sending replies backward on cross-edges of the WFG and backward on spanning tree edges. During this distributed reduction, if sufficient information to decide the reducibility of a node is not available at that node, appropriate replies are sent, and the algorithm attempts reduction later in a lazy manner at an up-tree node in the spanning tree. The initiator receives replies on all its outgoing spanning tree edges and cross-edges because the sending of replies is never delayed, and detect termination of the distributed reduction when all such replies are received. Thus, no explicit termination detection algorithm is used. At termination, the existence of a deadlock, if any, is detected.

4.4 Shigeng Cheng's Algorithm

Chen's Algorithm [8] consists of series of similar stages. In the first stage, initiator of the algorithm sends a FORWARD message to all of its successors. On receiving the message each successor sends a BACKWARD message to the initiator containing some specified information. On receiving replies initiator expands the wait-for-graph by inserting necessary edges according to the received replies. After receiving replies from all of its successor the algorithm enters the second stage

in which, FORWARD message is sent to next level of successors and so on. This process continues stage by stage. At end of each stage, a reduction is performed in WFG to remove those edges which will not belong to any tie and those vertices which are not reachable from initiator, deadlock detection is then attempted by searching for a tie in reduced WFG. The two major advantages of this algorithm are: firstly it avoids sending messages along the edges of global wait-for-graph and secondly information required for deadlock detection is readily available at the initiator which simplifies the task of deadlock resolution and also reduces system overheads.

4.5 Srinivasan Selvaraj's Algorithm

Srinivasan Selvaraj and Rajaram Ramasamy in [16] proposed a centralized based approach for dealing with generalized deadlocks in distributed systems. In this technique, the initiator builds the Distributed Spanning Tree (DST) by propagating probes along the edges of WFG and collects the replies. As a probe is propagated, each node sends a reply that carries its unblocking condition to the initiator directly. If the initiator receives a reply from an active node, it simplifies the unblocking conditions. The initiator declares all the nodes that have not been reduced at the end of termination as deadlocked. The main advantage of this algorithm is that replies are sent back by nodes to the initiator as soon as they receive a probe message. This helps the initiator to simplify the received unblocking condition simultaneously as the algorithm is executing. This approach also uses a priority based method to handle the issues associated with the concurrent execution of algorithm. The method assigns a unique priority to each instance of the algorithm based on its identifier that comprises the initiator's identifier and the block time or sequence numbers.

4.6 Soojung Lee's Algorithm

The proposed algorithm of Soojung Lee [13] consists of two phases, probing phase and deadlock detection and resolution phase. The second phase begins only after the first phase is finished. In the probing phase, the initiator of the algorithm builds a distributed spanning tree (DST) by propagating probes. If a blocked node receives a probe for the first time, it becomes a child of the sender of the probe and forwards the probe to its successors. However, if a node active or has already joined the current instance of the algorithm upon receiving a probe, it sends a reply directly to the initiator. That is, a reply is sent either by an active or upon finding a non-tree edge. After receiving all replies, the initiator performs the second phase. The requesting conditions of nodes are carried by probes. Specifically, the initiator sends its requesting condition to one of its successors. The successor, if receiving a probe for the first time, also sends its own requesting condition along with the received one to its successor. At this time, the requesting conditions to be sent are evenly distributed, if , among the successors in order to balance the message size. The other nodes behave similarly in forwarding the probes. When a node responds to a probe, it simply delivers the exact information carried by the probe to the initiator. In addition to this information, an active node transmits its own identifier. Note that the initiator shall finally receive the identifiers of all active nodes and the requesting conditions of all blocked nodes reachable from it. Termination time of the reply receipt at the initiator is using the technique of weight distribution.

5. CONCLUSION

This paper reviews some of the existing techniques based on diffusion computation. It can be seen that Barcha's and Wang's technique almost stand same in case of delay and message length. Moreover, there is no resolution scheme provided in both of these techniques. On the other hand, slight improvement in delay, and message exchange was done by Kshemkalyani and Chen with a resolution technique in their algorithms. Recent research in this area is done by Soojung Lee and Srinivasan Selvaraj by improving all the parameters. The technique used for weight distribution in Lee's algorithm however is quite complex. This complexity is although eliminated in Srinivasan's algorithm by the cost of increasing number of messages. The comparative analysis of all the presented algorithm is given in Table 1. Although literature in this area has provided many efficient techniques but still there is a scope of research in this area addressing many other issues of distributed system like handling several instances of algorithm simultaneously in the system.

Table 1. Comparison Of Different Distributed Detection Algorithms

Algorithms	Delay	No. of Messages	Message Length	Resolution
Barcha- Toueg	4d	4e	O(1)	No scheme
Wang et.al	3d+1	6e	O(1)	No scheme
Kshemkalyani.et. al	2d	2e	O(e)	1 message
Chen.et.al	2d	2n	O(e)	3n message
Soojung Lee	d+1	<2e	O(d)	1 message
Srinivasan Selvaraj	d+1	<e+2n	O(1)	1 message

6. REFERENCES

[1] Knapp,E. 1987. Deadlock Detection in Distributed Database Systems, ACM Computing Surveys, Vol.19, No. 4, 303-327

[2] Roesler, M. and Burkhard, W.A. 1989. Resolution of Deadlocks in Object-Oriented Distributed Systems, IEEE Trans. Computers, Vol. 38, No. 8, 1212-1224

[3] Ng,W.K and Ravishankar C.V. 1994. On-Line Detection and Resolution of Communication Deadlocks, Proc. 27th Ann. Hawaii Int'l Conf. System Science, 524-533

[4] Bracha,G and Toueg,S. 1987. A distributed algorithm for generalized deadlock detection. Distributed Computing, 2:127– 138

[5] Wang,J Huang, S and Chen,N. 1990. A distributed algorithm for detecting generalized deadlocks. Tech. Rep., Dept. of Computer Science, National Tsing-Hua University.

[6] Lee, S. and Kim, J.L. 1995.An Efficient Distributed Deadlock Detection Algorithm," Proc. of the 15th Int. Conference on Distributed Computing System, 169–178

[7] Brzezinski, J. Helary, J.M. Raynal, M. and Singhal, M.1995. Deadlock Models and a General Algorithm for Distributed Deadlock Detection, J. Parallel and Distributed Computing,31(2) ,112-125

[8] Chen, S. Deng, Y. Attie, P. C. and Sun, W. 1996. Optimal deadlock detection in distributed systems based on locally constructed wait-for graphs. Proc. Int'l Conf. Distributed Computing Systems, 613–619

[9] Kshemkalyani, A.D. and Singhal, M. 1999. A One-Phase Algorithm to Detect Distributed Deadlocks in Replicated Databases, IEEE Trans. Knowledge and Data Eng., vol. 11, no. 6, 880-895

[10] Kshemkalyani, A. D. and Singhal,M. 1989. Efficient detection and resolution of generalized distributed deadlocks, IEEE Transactions on Software Engineering, 20(1):43–54

[11] Kshemkalyani,A.D and Singhal,M. 1997. Distributed detection of generalized deadlocks. Proc. 17th Int'l Conf. Distributed Computing Systems, 553–560

[12] Kshemkalyani, A.D. and Singhal, M. 1999. A One-Phase Algorithm to Detect Distributed Deadlocks in Replicated Databases, IEEE Trans. Knowledge and Data Eng., vol. 11, no. 6,880-895

[13] Lee, S. 2004. Fast, Centralized Detection and Resolution of Distributed Deadlocks in the Generalized Model, IEEE Trans. On Software Engineering, Vol. 30, NO. 9, 561-573

[14] Lee,S. and Kim, J.L.2001. Performance Analysis of Distributed Deadlock Detection Algorithms, IEEE Trans. Knowledge and Data Eng., vol. 13, no. 4,623-636,

[15] Lee, S. 2001. Efficient Generalized Deadlock Detection and Resolution in Distributed Systems, Proc. 21st Int. Conference on Distributed Computing Systems, 47-54

[16] Srinivasan Selvaraj and Rajaram Ramasamy 2010. An Efficient Detection and Resolution of Generalized Deadlock in Distributed Systems; International Journal of Computer Applications, Volume 1-No. 19