International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 6, Issue 1 (JAN-FEB 2018), PP. 56-59

STUDY OF DIVERSE MODELS OF DEADLOCK DETECTION IN DISTRIBUTED ENVIRONMENT

SAHIBA RAZA [1], JAMEEL AHMAD [2],
Department Of Computer Science
[1]M.Tech- CSE (P.T.) Integral University,
[2]Assistant Professor Integral University,
Lucknow. India

***Abstract*— A distributed system is much larger and more prevailing than typical centralized systems due to the combined capabilities of distributed components. Examples of distributed systems include computer networks, distributed databases, distributed information processing systems and real time process control systems. In this paper, we review diverse models that are used to classify deadlock detection algorithms and determine the condition for detecting deadlock.**

***Index Terms*— Distributed Deadlock, WFG, Communication Deadlock, Deadlock Models.**

I. INTRODUCTION

Deadlock can take place whenever two or more processes are competing for limited resources and the processes are allowed to attain and hold a resource (obtain a lock) thus preventing others from using the resource while the process waits for other resources.

There are three ways to deal with distributed deadlocks, namely: deadlock prevention, deadlock avoidance and deadlock detection. In deadlock prevention algorithm a process executes by forming wait-for relations with other conflicting processes in distributed systems. But deadlock prevention algorithms are inefficient and impractical for distributed systems [7]. Dijkstra has proposed the Banker's algorithm which serves as a basis for all deadlock avoidance algorithms. Banker's algorithm is widely used to avoid deadlocks in centralized systems. Since the resource requirement of a process depends on run-time inputs, complex branching and execution environment, Banker's algorithm cannot be used to avoid deadlocks in practice [8]. Many algorithms have been proposed in the literature for deadlock avoidance in specific distributed systems where the distributed WFG are non-recursive and known priori.

Deadlock Detection is more optimistic and feasible than deadlock prevention and avoidance methods. Distributed deadlock detection algorithms have maintained the WFG to represent the dynamic wait-for dependencies between distributed processes. The WFG is then analyzed periodically or continuously to detect the presence of deadlock.

Four principles, namely path pushing, edge chasing, diffusing computation and global state detection have been used to sense distributed deadlocks in the literature. Many path pushing based algorithms are disproved later. Edge chasing is mainly used to detect resource deadlocks whereas diffusing computation is widely used to detect communication deadlocks and generalized deadlocks [12]. Global State detection is used to determine the stable properties such as deadlock detection, termination detection in distributed systems.

II. RESOURCE DEADLOCK DETECTION ALGORITHMS

Deadlock Detection is more optimistic and feasible than deadlock prevention and avoidance methods. Distributed deadlock detection algorithms have maintained the WFG to represent the dynamic wait- for dependencies between distributed processes. The WFG is then analyzed periodically or continuously to detect the presence of deadlock.

Four principles, namely path pushing, edge chasing, diffusing computation and global state detection have been used to detect distributed deadlocks in the literature. Many path pushing based algorithms are disproved later. Edge chasing is mainly used to detect resource deadlocks whereas diffusing computation is widely used to detect communication deadlocks and generalized deadlocks [1]. Global State detection is used to determine the stable properties such as deadlock detection, termination detection in distributed systems.

Depending on the distributed application/environment, a process can make different type of requests Sanchez et. al.[2] have defined various request models, namely SR model, AND model, OR model, AND-OR model, P out-of Q model and Unrestricted model. Those models are used to classify deadlock detection algorithms and determine the condition for detecting deadlock.

A. Single Resource Model

SR model is the simplest request model of distributed environment. Because of its simplicity, this model has been widely used to model resource acquisition. AND model, also known as Resource deadlock model, is the generalization form of SR model. A typical resource request of a transaction in the Database System is an example of AND request. In both SR and AND model, the presence of cycle in the WFG implies

deadlock. The basic principle behind the algorithms that detect deadlocks in SR and AND model is discussed as follows.

B. AND Model

Prieto et al[3] have proposed the distributed algorithm in which a blocked process sends two types of messages in two different directions. It declares deadlock once a process receives the information about the same process in both type of messages. It simplifies a deadlock resolution by aborting a process in the message to resolve deadlock.

None of the above algorithms are able to identify deadlocks in which the initiator is not directly involved in the cycle [4]. These have proposed another algorithm in which deadlocks can be detected even if the initiator does not belong to any deadlock cycle. The initiator of this algorithm builds the Directed Spanning Tree (DST) by propagating probes among its dependents. A probe carries the dependency relationship (route string) among the processes. Based on the information about data dependency between the processes, it determines a deadlock. However, it suffers with few limitations. First, all deadlocks reachable from the initiator may not be resolved by a single execution of the algorithm. Second, deadlock detection algorithm works correctly for single execution of the algorithm, but it would detect phantom deadlocks in case of multiple executions [4].

Sanchez et al.[2] have proposed history based edge chasing algorithm to detect deadlock irrespective of whether the initiator is directly or indirectly involved in the deadlock cycle[3]. In this algorithm, a probe comprises two parameters namely the initiator id and an integer string called route string. When a process receives the first probe message, it stores a probe in its memory and forwards the probe to its dependents after appending its id in route-string. Upon receiving a message, if the route-string in the message is a prefix of the route-string in the memory in any process, this algorithm declares a deadlock. Since the deadlock resolution is initiated immediately after detecting the deadlock, it significantly reduces the deadlock persistence time. Moreover, it minimizes the messages for handling the executions of the algorithm using the priorities associated with each process and avoids the detection of same deadlock.

Sanchez et al.[3] have proposed a priority based probe algorithm to simplify the detection of multi-cycle deadlocks[8]. In this algorithm, the priority is maintained in each process. Whenever a process is blocked, it initializes the algorithm and circulates the token/probe among its dependents. Upon receiving a token, a process compares the priority in the token with its own priority. If the token is attached with highest priority, the token is forwarded to its dependents; otherwise, the priority is updated and the token is subsequently dropped. Therefore, when multiple processes initiate the algorithm, different priorities are attached with the token that belongs to different instance of the algorithm. As a result, if a deadlock exists, a token initiated by a process that has higher priority will automatically return to the initiator. Moreover, it does not circulate separate messages to clean up the storage of processes as in the existing probe based deadlock detection algorithms.

Razzaque et. al.[4] have proposed an algorithm which detects mutli-cycle deadlock detection using probe discard policy. According to the policy, if a process that has already received the probe, receives a probe of another instance, it discards the probe message. Hence, all processes in a deadlock cycle participate in the execution of single instance at a time. It also simplifies the deadlock by victimizing a single process that is waiting for more number of resources. Rahimalipour et. al.[7] have proposed another algorithm which enforces the probe discard policy for detecting multi-cycle deadlocks using daemons. It considerably reduces the message overhead and memory overhead associated with each process as in Razzaque's algorithm.

Ashfield et al [8] have used agent based edge chasing algorithm to determine deadlock in the AND model. In this algorithm, each process is aware of its predecessors and successors simultaneously. Hence, it detects a deadlock cycle with size two locally as compared to the earlier algorithms. These have proposed another agent based algorithm in which mobile agents are moved across the distributed systems for collecting adequate information to determine deadlock. Ashfield et al have proposed a mobile agent based deadlock detection algorithm, called M-Guard, which collects the resource request/ allocation information about the processes [9]. The agents are moving along the information among the distributed processes to determine deadlocks. This algorithm slightly reduced the network traffic as compared to other mobile agent based algorithms.

In a faulty distributed system, faulty processes may interrupt the execution of deadlock detection algorithms. These have proposed a distributed deadlock detection algorithm by integrating priority-based probe algorithm and fault diagnosis model. In this algorithm, the probe is propagated in a backward direction. In this algorithm, the probe message carries the information about faulty processes along with the initiator. If a process receives a probe from a non faulty process, it forwards the probe message; otherwise, it discards the probe and sends a message to clean up the storage of processes that have already received the probe messages. Ashfield et al [8] have proposed token based fault tolerant algorithm which can handle the loss of resource release messages.

III. COMMUNICATION DEADLOCK DETECTION ALGORITHMS

OR Model

The OR model, also called as communication model, is used to represent the alternative structures in programming languages such as CSP, ADA and replicated database systems.

Ashfield et al [8] have proposed a knot detection algorithm which detects a knot by checking whether all processes that are reachable from the initiator are in a cycle with the initiator. In this algorithm, a process does not know the resource requests of other processes. It also avoids the propagation of reply messages through the processes that are not reachable from the

initiator. Comparing to the other algorithms in this category, this algorithm detects all nodes that are involved in the knot. Hence, it simplifies the deadlock resolution.

The algorithms in the third category detects an OR deadlock by detecting a knot which is defined as a strongly connected subgraph. In a strongly connected subgraph, there is no edge directed outside the subgraph.

IV. Deadlock Detection Algorithms For General Model

A. AND-OR Model

The AND-OR model allows a process to specify the resource requests as a predicate by using logical AND and OR operators. In this model, deadlock is detected by applying the test for OR deadlock repeatedly. The algorithms that are used to detect deadlocks in AND-OR model are described below.

Ashfield et al[8] have presented another algorithm which uses two levels of deadlock detection procedures. In the first level, the algorithm search for a cycle in the WFG. If the cycle is found, the second- level algorithm which is similar to Harman and Chandy's algorithm is initiated by the qualified initiator. However, if the cycle does not exist in the WFG, it does not invoke the second level algorithm. Since it controls the tree computation, it reduces the communication cost as compared to the Herman's algorithm.

B. Generalized/ P out-of Q Model

P out-of Q Model is the generalization of all previous models. Although the generalized request model is equivalent to the AND-OR model, the length of the predicate in P out-of Q model is P. Moreover, the P out-of Q model defines distributed deadlock problem independent of the underlying request model. It brings more flexibility by permitting the distributed processes to change the resource request model. Since the presence of cycle or knot in the WFG is insufficient to determine a deadlock, the generalized deadlock detection algorithms have to examine some complex topology in the WFG [1]. Hence, very few generalized deadlock detection algorithm have been proposed in the literature. The working principle of all generalized deadlock detection algorithms is discussed as follows.

Ashfield et al [8] has proposed an algorithm in which the initiator constructs the LWFG by using the ancestor-descendent relationship between the processes. It uses less than 2e messages in 2d time units to detect a deadlock. It reduces the message length into O (e-n+m), where m indicates the number of nodes that are not associated with any non-tree edges in the spanning tree induced by the algorithm. However, it needs additional technique to assign a unique path string to each node in the WFG and to interpret the path strings for constructing LWFG at the initiator. In contrast to Chen's algorithm, it resolves all deadlocks reachable from the initiator.

C. Unrestricted Model

In unrestricted model, there is no assumption about underlying structure of resource requests. The deadlock detection algorithms of this model have more theoretical value from the perspective of distributed systems. Moreover, those algorithms can be used to detect other stable properties such as distributed monitoring and distributed debugging and global state detection. However, the deadlock detection algorithms developed for this model carry the additional overhead which can be avoided in the algorithms designed for previous models. Also, those algorithms have to search the whole system to construct the WFG for deadlock detection.

Designing algorithms for controlling distributed system is difficult due to the following reasons: First, there is no single site/computer that controls the entire distributed system. Second, computers/sites have to exchange messages with other computers to perform a task. In this chapter we study diverse communication and recourse deadlock detection algorithms.

V. Conclusion

Deadlock detection is cumbersome in distributed systems since no site has complete knowledge about the resource requirements of all processes. The distributed deadlock detection algorithms are classified based on the underlying deadlock models such as AND model, OR model, AND-OR model and P out-of Q model. Among the models, P out-of Q model, also called as generalized request model, has the modeling power of all other models and much more concise expressive power than other models. However, it is difficult to detect deadlock in the generalized request model. It is observed that only very few algorithms have been proposed to detect deadlock in the literature. Though the algorithms have reduced the deadlock duration through the years, they have paid very little attention on other key measures such as number of messages and message size.

References

[1] Lee, S. "Fast, Centralized Detection and Resolution of Distributed Deadlocks in the Generalized Model", IEEE Transaction on Software Engineering, Vol. 30, No. 9, pp. 561-573, 2004.

[2] Sanchez, C., Sipma, H., Manna, Z., Subramonian, V. and Gill, C.D. "On Efficient Distributed Deadlock Avoidance for Real-Time and Embedded Systems", Proceedings of the Twentieth IEEE International Parallel and Distributed Processing Symposium, IEEE Computer Society Press, pp. 133-136, 2006.

[3] Prieto, M. Villadangos, J., Fariña, F. and Córdoba, A "An O(n) distributed deadlock resolution algorithm", Proceedings of the Fourteen Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 48-55, 2006.

[4] Razzaque, M.A.,Rashid, M.M. and Hong, C."MC2DR: Multi-cycle Deadlock Detection and Recovery Algorithm for Distributed Systems", Proceedings of the High Performance Computing and Communications, pp. 554-565, 2007

[5] Hashemzadeh, M., Farajzadeh, N. and Haghighat, A.T. "Optimal detection and resolution of distributed deadlocks in the generalized model", Proceedings of the Fourteen Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 133-136, 2006

[6] Wang, Y., Kelly, T ., Kudlur, M., Lafortune, S. and Mahlke. S.A." Gadara: Dynamic Deadlock Avoidance for Multithreaded Programs", Proceedings of Eighth USENIX Symposium on Operating Systems Design and Implementation, pp.281-294, 2008

[7] Rahimalipour, Z. and Haghighat, A.T. "Daemon- Based Distributed Deadlock Detection and Resolution", World Academy of Science, Engineering and Technology, Vol. 63, pp. 339-344, 2010

[8] Ashfield, B., Deugo, D., Oppacher, F. and White, T. "Distributed Deadlock Detection in Mobile Agent Systems", Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 146-156, 2002.

[9] Zhou, J., Chen, X., Dai, H., Cao, J. and Chen, D. "M-Guard: A New Distributed Deadlock Detection Algorithm Based on Mobile Agent Technology", Proceedings of the International Symposium on Parallel and Distributed Processing with Applications, pp. 75-84, 2004.

[10] Lu, W., Yang, Y., Wang, L., Xing, W., Che, X., & Chen, L. A fault tolerant election-based deadlock detection algorithm in distributed systems. Software Quality Journal, 1-23, 2017.