Performance and Dependency Analysis of Data in Distributed Systems

Anitha.G.S[#1], Sivagami.V.M[*2]

[1] *PG Scholar,* [2]*Associate Professor*
Department of Information technology,
Sri Venkateswara College of Engineering,
Sriperumbudur – 602105, Chennai, INDIA.

***Abstract*— Atomic Broadcast is important in fault-tolerant distributed computing. It ensures that messages broadcast by different processes are delivered to all destinations in same order. Token circulation is one of the important ordering mechanisms in atomic broadcast. A single proposal is contained in the token. The proposal contains a batch of messages to be ordered and a decision on a batch can be taken at the earliest *f* (fault) communication steps after the batch is proposed. In the existing system, token based atomic broadcast algorithm relay on the group membership service. This algorithm helps in ordering but it does not tolerate the failure and also a wrong suspicion can lead to two costly membership operations namely addition and removal of process. To solve this problem, token-based atomic broadcast algorithm that uses an unreliable failure detector instead of a group membership service is used .It is efficiently implemented by combining a failure detector and a token-based mechanism. Overhead of a wrong failure suspicion is low when compared to group membership service. The performance of this algorithm is evaluated in both local and wide area networks. The new token-based algorithm provides the better performance of the other algorithms in most small-system settings.**
.

***Keywords*— Atomic** Broadcast, Fault tolerance, Token

I. INTRODUCTION

Grid computing is defined as applying the resources of many computers in a network to a single problem at the same time. Fault-tolerance is an important issue in computational grid. Generally faults occur when a grid resource is unable to complete its job in the given deadline. Fault-tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a natively-designed system in which even a small failure can cause total breakdown. Fault-tolerance is particularly sought-after in high-availability or life-critical systems. Atomic Broadcast is important in fault-tolerant distributed computing. It is defined by four properties namely Validity, Uniform agreement, Uniform integrity, Uniform total order. Token circulation is one of the important ordering mechanisms in atomic broadcast. A single proposal is contained in the token. The proposal contains a batch of messages to be ordered and a decision on a batch can be taken at the earliest *f*(fault) communication steps after the batch is proposed. In most traditional token-based algorithms, processes are organized in a logical ring and, for token transmission, communicate only with their immediate predecessor and successor (except during changes in the composition of the ring).Section II presents the system model which explains about atomic broadcast and problem definition. Section III explains about Token based algorithm and section IV explains about Failure detector.

II. EXISTING SYSTEM

State machine replication[6], is a well known approach for rendering services to fault tolerant. The idea is to fully replicate the service state on several servers and execute every client command in every non-faulty server in the same order. It achieves strong consistency by regulating how client commands must be propagated to and executed by the replicas. Limitation here is an overhead in service response time. Ring Paxos [2], which is a high throughput atomic broadcast protocol is used for fault tolerance. The idea is to replicate a service so that the failure of one or more replicas does not prevent the operational replicas from executing service requests. Limitation is Lost messages have a negative impact on Ring Paxos, as they result in retransmissions.

Leader follower replication technique [7], which is used in The low latency fault tolerance (LLFT) system, provides fault tolerance for distributed applications. The LLFT system achieves low latency message delivery during normal operation and low latency reconfiguration and recovery when a fault occurs. Mutual distributed algorithm on a token ring [4], is based on the token ring approach and allows simultaneous existence of several tokens in the logical ring of the network. Each process generates a unique token and sends it as request to enter the critical section that travels along the ring. The process can only enter the critical section if it gets back its own token. Drawback is, it does not work well in case of lost tokens.

III. PROPOSED SYSTEM

We consider an asynchronous system of n processes p0... pn−1. The processes communicate by message passing over reliable channels and at most f processes may fail by crashing. A process that never crashes is said to be correct, otherwise it

is faulty. The system is augmented with reliable failure detectors.

A. System Architecture

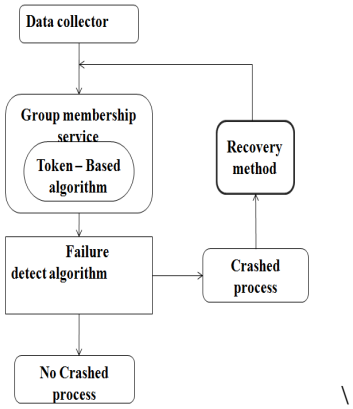

Fig 1.System architecture

B. Atomic Broadcast

Atomic broadcast or total order broadcast is a broadcasting messaging protocol that ensures that messages are received reliably and in the same order by all participants. This problem is usually considered in environments where participants can fail, for example, by crashing. Participants who never fail are called correct, the others are faulty. The following properties are usually required from an atomic broadcast protocol.

- **Validity**- If a correct process p broadcasts a message m, then it eventually a delivers m,
- **Uniform Agreement**- If a process delivers m, then all correct processes eventually deliver m,
- **Uniform Integrity**- For any message m, every process p delivers m at most once and only if m was previously a broadcast
- **Uniform total order**- If some process, correct or faulty, delivers m before m0, then every process delivers m0 only after it has delivered m.

C. Problem Defnition

In this paper, Atomic Broadcast or Total Order Broadcast problem is focused. A Total Order Broadcast ensures that processes in a distributed system deliver messages in the same order, which is essential for implementing services that require coherence between processes such as distributed databases or collaborative edition. This problem can be defined by four properties, as presented below:

Validity - If a *correct* process (a process is called *Correct* only if it does not crash during the entire execution, although even a correct process can be incorrectly suspected of crashing) broadcasts a message *m* to a list of processes, then some *correct* process in eventually delivers *m* to the application.

.

Agreement - If a *correct* process delivers a message *m*, then all *correct* processes in eventually deliver *m*.

Integrity - For any message *m*, every *correct* process *p* delivers *m* at most once and only if (1) *m* was previously broadcast by *sender(m)* and (2) *p* is a process in the set .

D. Consensus Problem

Consensus is defined by the primitives propose(v) and decide(v), where v is an arbitrary value; atomic broadcast is defined by the primitives broadcast(m) and deliver(m),where m is a message.Consensus guarantees that (i) if a process decides v then some process proposed v; (ii) no two processes decide different values; and (iii) if one (or more) correct process proposes a value then eventually some value is decided by all correct processes. Atomic broadcast guarantees that (i) if a process delivers m, then all correct processes deliver m; (ii) no two processes deliver any two messages in different orders; and (iii) if a correct process broadcasts m, then all correct processes deliver m.

E. Group membership Vs Failure detector

A group membership service provides consistent membership information to all the members of a group. Its main feature is to remove processes that are suspected to have crashed. In contrast, an unreliable failure detector, e.g., _S, does not provide consistent information about the failure status of processes. For example, it can tell to process p that r has crashed, while telling at the same time to process q that r is alive. Both mechanisms can make mistakes, e.g., by incorrectly suspecting correct processes. However, the cost of a wrong failure suspicion is higher when using a group membership service than when using failure detectors. This is because the group membership service removes suspected processes from the group, a costly operation. This removal is absolutely necessary for the atomic broadcast algorithm that relies on the membership service: the notification of the removal allows the atomic broadcast algorithm to avoid being blocked. There is no such removal of suspected processes with a failure detector.

Moreover, with a group membership service, the removal of a process is usually followed by the addition of another (or the same) process, in order to keep the same replication degree. So, with a group membership service, a wrong suspicion leads to two costly membership operations: removal of a process followed by the addition of another process. In an environment where wrong failure suspicions are

frequent, 2 algorithms based on failure detectors thus have advantages over algorithms based on a group membership service. The cost difference has been experimentally evaluated in the context of two specific (not token based) atomic broadcast algorithm. Atomic broadcast algorithms based on a failure detector have another important advantage over algorithms based on group membership: they can be used to implement the group membership service. Indeed, since a (primary partition) group membership service orders views, it seems intuitive to solve group membership using atomic broadcast: this leads to a much simpler protocol stack than implementing atomic broadcast using group membership. However, this is not possible if atomic broadcast relies on group membership.

IV. TOKEN BASED ALGORITHM

A. *Token based algorithm*

In token-based algorithms, processes are organized in a logical ring and, for token transmission they communicate only with their immediate predecessor and successor. An algorithm is said to be token-based only if 1) processes are organized in a logical ring, 2) each process must have failure detector module FD_i that provides information only about its immediate predecessor and 3) each process only sends tokens to and receives tokens from its f predecessors and successors, where f is the number of tolerated failures.

In the token based atomic broadcast algorithm, the token transports (i) sets of messages and (ii) sequences of messages. More precisely, the token carries the following information: *(round, proposalSeq, votes, adeliv, nextSet).*Messages in the sequence *proposalSeq* are delivered as soon as a sufficient number of consecutive "votes" have been collected. The field *adeliv* is the set of all consensus decisions that the token is aware of (*i.e.*, a set of pairs associating a consensus number to a sequence of messages). When a process receives the token, it can therefore, if needed, catch up with the message deliveries performed by other processes. Finally, while the token accumulates votes for *proposalSeq*, it simultaneously collects in *nextSet* the messages that have been *abroadcast*, but not *adelivered* yet. The set *nextSet* grows as the token circulates. Whenever messages in *proposalSeq* can be delivered, *nextSet* is used as the proposal for the next decision.

B.*Token Circulation*

In order to avoid the loss of the token due to crashes, process pi sends the token to its f + 1 successors in the ring, i.e., to $p_i+1, \ldots, p_i+f+1.6$ Furthermore, when awaiting the token, process pi waits to get the token from p_i-1, unless it suspects p_i-1. If pi suspects p_i-1, it accepts the token from any of its predecessors. Finally, while the token accumulates votes for *proposalSeq*, it simultaneously collects in *nextSet* the messages broadcast atomically. The set *nextSet* grows as the token circulates. Whenever messages in *proposalSeq* can be delivered, *next Set* is used as the "proposals" for the next decision.

C.*Fault tolerance for token based synchronization*

Synchronization primitives based on a privilege associated with a token provide a simple method to ensure safety properties in a distributed system. For example, in a distributed protocol for mutual exclusion it is assumed that the right to enter the critical section is a property associated with the token. In other words, only the node in possession of the token may enter the critical section. Similarly, conditional synchronization may associate the property of signaling a wait condition with a token, *i.e.*, if a node receives a token, it may wake up the first waiter suspended on the synchronization object.

In a decentralized approach based upon token passing, requests are collected in a distributed queue. New requests are sent either directly to the tail of this queue or will be forwarded along intermediate nodes before reaching the tail and being appended to this queue. During request forwarding, intermediate nodes perform path compression to log the requester as the new tail of the distributed queue, *i.e.*, later requests will be sent directly to this last requester. On the average, path compression results in an overhead of O(logn)messages per request for nodes. These protocols assume point-to-point communication (no broadcast necessary) and do not require any centralized service, which enhances the scalability of the approach. A fault-tolerant token-based synchronization protocol should preserve the properties of the base protocol, *i.e.*, instead of broadcasts or centralized services, it should rely on point-to-point communication and decentralized control. Several issues have to be addressed in this context:

- Fault detection
- Collection of fault correction information
- Election of a fault handling node
- Fault recovery
- False alarm recovery

V.FAILURE DETECTOR

Atomic broadcast algorithms based on a failure detector have important advantage. They can be used to implement the group membership service Failure-detector based algorithms have advantages over group-membership based algorithms, in case of wrong failure suspicions, and possibly also in the case of real crashes. The failure detector module of a process pi only needs to give information about the state of p_i-1.Failure detector is implemented with the help of logical ring structure. The eventual perfect failure detector P is defined by the following properties:

i. **Strong completeness**- *Eventually, every process that crashes is permanently suspected by every correct process*

ii. **Eventual strong accuracy**-*There is a time after which correct processes are not suspected by any correct process.*

For every process p_i, R ensures the following properties:

- **Completeness:** If p_i-1 crashes and p_i is correct, then p_i-1 is eventually permanently suspected by p_i,
- **Accuracy:** If p_i-1 and p_i are correct, there is a time t after which p_i-1 is never suspected by p_i.

A. Token based consensus using failure detector

Consensus is achieved by passing a token between the different processes. The token contains information regarding the current proposal. The token is passed between the processes on the logical ring p0; p1; . . . ; pn-1.To avoid the loss of the token due to crashes, process pi sends the token to its f + 1 successors in the ring,p_i+1; . . . ; p_i+f+1.The algorithm is expressed as a sequence of rounds. In each round single process sends its token.

B. Token based atomic broadcast using failure detector

It deals about the transformation of the token-based consensus algorithm into an atomic broadcast algorithm . Even though the consensus algorithm is simpler than the atomic broadcast algorithm, a two-step presentation makes it easier to understand the atomic broadcast algorithm. Note also that it is well known how to solve atomic broadcast by reduction to consensus. However, the reduction, which transforms atomic broadcast into a sequence of consensus, yields an inefficient algorithm here. The reduction would lead to multiple instances of consensus, with one token per consensus instance. A variation of the algorithm that follows is presented. The algorithm presented here is easier to understand, with processes that send regular messages and tokens .To be correct, the atomic broadcast algorithm requires the failure detector R, a number of processes n>f(f+1)+1, and a vote threshold at f +1 in order to decide as was the case in the consensus algorithm above.

VI. EXPERIMENTAL RESULTS

This work is implemented using GridSim toolkit which provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domains distributed computing systems such as clusters and Grids. Application schedulers in the Grid environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. This means that each user has his or her own private resource broker and hence it can be targeted to optimize for the requirements and objectives of its owner. Steps for implementation:

- Configure the grid using config Grid.
- Specify the number of nodes to be entered.
- Deployment of node is done.
- Specify the source.
- Specify the destination to which the packet is to be routed.
- A Shortest path is created between source and Destination.
- In case of packet loss or route failure, make any one of the intermediate node between source and destination to Sleep.
- Specify the Sleep node.
- Another shortest path is created for the same source and destination.

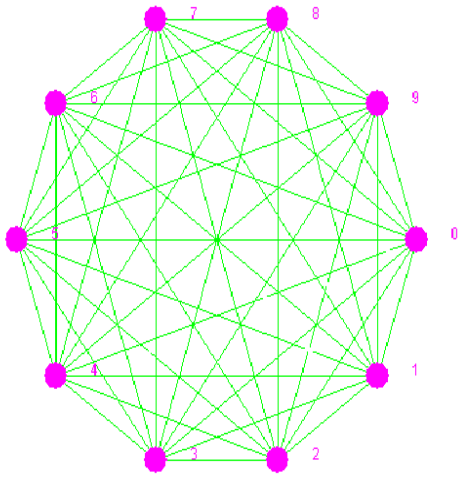

Fig 2.Topology generation

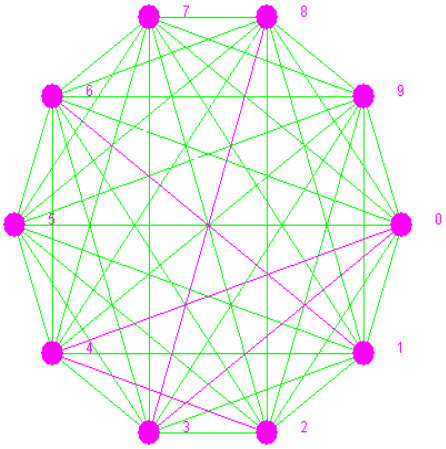

Fig 3.Deployment of nodes

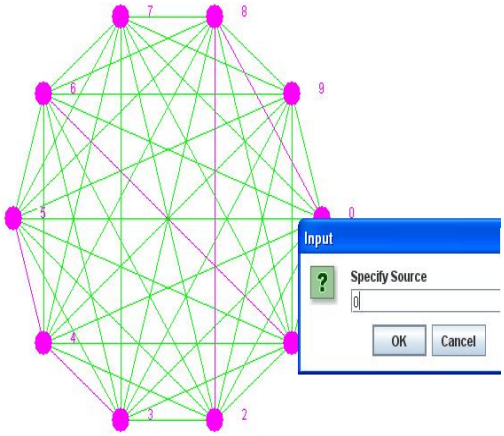

Fig 4.Source node specification

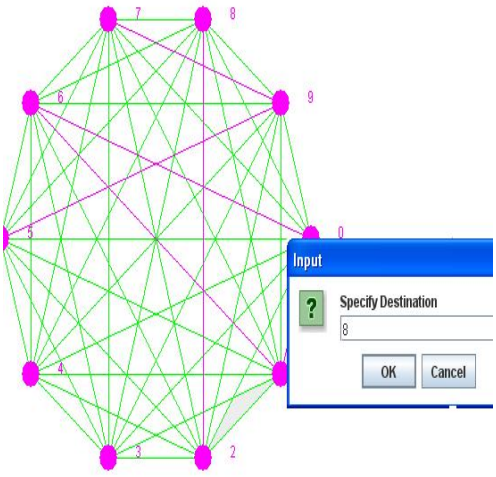

Fig 5.Destination node specification

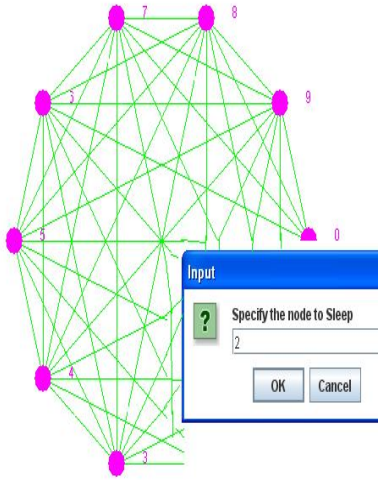

Fig 6.Sleep node specification

VII. CONCLUSION

Token-based atomic broadcast algorithms are more efficient in terms of throughput than other atomic broadcast algorithms; the token can be used to reduce network contention. However, all published token-based algorithms rely on a group membership Service none of them use unreliable failure detectors directly. This paper presents the first token-based atomic broadcast algorithm that solely relies on a failure detector. Such an algorithm has the advantage of tolerating failures directly, instead of relying on a membership service to exclude crashed processes .Thus, failure-detector based algorithms have advantages over group-membership-based algorithms, in case of wrong failure suspicions, and possibly also in the case of real crashes.

In future, dynamic routing can be deployed in distributed system by using route recovery. Fault tolerance can be improved by means of check pointing and replication.

.

REFERENCES

[1] Rachid Guerraoui, Ron R. Levy, and Bastian Pochon**,** **"**Throughput Optimal Total Order Broadcast for Cluster Environments**"** ACM Transactions on Computer Systems, Vol. 28, No. 2, Article 5, Publication date: July 2010.

[2] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Parisa Jalili Marandi et.al, "Ring Paxos: A High-Throughput Atomic Broadcast Protocol" in Dependable Systems and Networks, pp. 527-536, 2010K. Elissa, "Title of paper if known," unpublished.

[4] Luiz Angelo Steffenel et.al, "Total Order Broadcast on Pervasive System", ACM Transactions on Distributed systems, SAC'08,2008

[5] Chao-Chin Wu and Ren-Yi Sun, "An integrated security-aware job scheduling strategy for large-scale computational grids", Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua City 500, Taiwan.

[6] P. Urba´n, I. Shnayderman, and A. Schiper, "Comparison of Failure Detectors and Group Membership: Performance Study of two Atomic Broadcast Algorithms," Proc. Int'l Conf. Dependable Systems and Networks (DSN), pp. 645-654, June 2003.

[7] Marco Primi et.al, "High Performance State-Machine Replication" in Dependable Systems and Networks - DSN, pp. 454-465, 2011.

[8] Wenbing Zhaoet.al," The Low Latency Fault Tolerance System" Journal:Computing Research Repository - CORR , vol. abs/1004.1, 2010.

[9] J.M. Chang and N. Maxemchuck, "Reliable Broadcast Protocols,"ACM Trans. Computer Systems, vol. 2, no. 3, pp. 251-273, Aug. 1984.

[10] F.Cristian, S. Mishra, and G. Alvarez, "High-PerformanceAsynchronous Atomic Broadcast,"Distributed System Eng. J.,vol. 4, no. 2, pp. 109-128, June 1997.