Failure Detector-Ring Paxos based Atomic Broadcast Algorithm

Nadjette Rebouh
LAMOS, Faculty of Exact Sciences, University of Béjaïa
Algeria
nadjette_rebouh34@yahoo.fr

The Atomic broadcast problem constitutes an essential component in fault-tolerant distributed systems. An Atomic broadcast algorithm ensures that all processes deliver the same messages sequence. Many algorithms have been published. Recently, a new Paxos-based algorithm has been proposed for clustered systems, named Ring Paxos. It inherits many of its characteristics: safe under asynchronous assumptions, live under weak synchronous assumptions, and requires a majority of non faulty processes to ensure progress. However, the proposed algorithm relies on strong assumptions, and uses the group membership for faults tolerance. In this paper, we propose a new Ring Paxos-based atomic broadcast algorithm for distributed systems. It inherits some of Ring Paxos characteristics and uses failure detectors to tolerate failures. A slight comparison between Ring Paxos and the proposed algorithm shows that our algorithm surpasses the performance of the other algorithm.

Agreement Problem; Atomic broadcast; Ring Paxos; Failure Detectors

1. INTRODUCTION

An Atomic Broadcast Algorithm is an important building block in distributed fault-tolerant systems. It provides a group communication primitive that allows processes to deliver the same messages sequence Hadzilacos (1993). Atomic Broadcast is pratically useful to implement state-machine replication Verissimo and Raynal (1999). By employing this group primitive to disseminate updates, all correct copies deliver the same set of updates in the same order, and consequently ensuring the copies consistency. The Atomic Broadcast problem has been extensively studied in the literature. Therefore, several algorithms have been proposed to implement this primitive, considering a diverse set of execution environments and system models. Although many classifications are possible when considering algorithms designed for distributed systems, according to the ordering mechanism or the mechanism used to tolerate failures. The first class includes protocols that rely on consensus to decide the order and the number of messages to deliver (Chandra and Toueg (1996),Verissimo and Raynal (1999),etc.) or on the token circulation protocols schiper et al. (2004). Two fundamental mechanisms define the second class: unreliable failure detectors Chandra and Toueg (1996) and group membership Chockler et al. (2001). Recently, a new algorithm that solves the atomic broadcast problem has been proposed, Ring Paxos (RP) Parisa et al. (2012). RP is based on Paxos, a well-known consensus algorithm Lamport (1998), and inherits many of its characteristics: it is safe under asynchronous assumptions, live under weak synchronous assumptions, and resiliency-optimal, that is, it requires a majority of correct processes to ensure progress. They optimize the original version of Paxos from which they derive RP. In their paper, they look for an efficient implementation, in terms of throughput, of RP. For that purpose, RP uses a single ip-multicast stream to disseminate messages and thus benefits from the throughput that ip-multicast can provide. Unfortunately, ip-multicast is unreliable. Hence, RP places a majority of correct processes (*f+1*) in a logical ring, where *f* is the number of tolerated failures, to totally order messages. From one part, RP achieves very high throughput while providing low latency, almost constant with the number of receivers. From the other part, RP relies on very strong assumptions: it supposes one coordinator along the execution, impractical assumption. It, also, requires a partially synchronous system, that is, an asynchronous model until a global stabilization time, when it becomes synchronous. Last and not

least, RP relies on a group membership service to detect failures, a highly expensive mechanism Ekwall and Schiper (2011). The aim of this paper is to circumvent these inconvenient by considering an asynchronous system, with a variable coordinator and $\diamondsuit S$ failure detector for fault tolerance, the resulting algorithm is Failure Detector-Ring Paxos based Atomic Broadcast Algorithm (FD-RP). FD-RP is an algorithm that solves the atomic broadcast problem in a distributed asynchronous system. As, in RP, FD-RP distinguishes three roles of processes: proposers, acceptors and learners, indeed of the coordinator (that can be one among them). Its main characteristic is the use of $\diamondsuit S$ failure detector to agree on the set of acceptors forming the logical ring, hence enabling processes from waiting for responses from crashed processes, consequently, reducing the execution time. Initially, the coordinator broadcasts a request to form the ring. Correct acceptors that have not suspected the coordinator accepts the request and respond positively. When the agreement about the value decided (the sequence of messages to deliver) has been reached (between coordinator and the elected acceptors), the decision will be broadcast by the coordinator to all correct processes (proposers, acceptors and learners). The rest of the paper is structured as follows. Section 2 describes some related work. Section 3 is devoted to the system model and the definition of related agreement problems. In section 4 we give an overview of the FD-RP-based Atomic Broadcast. The correctness proof of the algorithm is provided in section 5. Section 6 concludes the paper.

2. RELATED WORK

Atomic broadcast algorithms have been widely studied in the last twenty years. They can be summarized into five interesting classes schiper et al. (2004): (1) fixed sequencer algorithms: in this class, one unique process, during the hole execution, is chosen as a sequencer (coordinator) and it is responsible for ordering messages by affecting sequence numbers to the broadcast messages Garcia-Molina and Spauster (1991),Birman et al. (1991), (2) moving sequencer class: it keeps the same principle as the previous class but allows the role of the sequencer to be transferred between several processes Chang and Maxemchuk (1984), Cristian et al. (1997), (3) privilege based class: it is characterized by the use of the token, processes can send their messages only when they receive the token and the order is fixed by the message sender Ekwall and Schiper (2011), Amir et al. (1995), (4) communication history algorithms: the messages' order is determined by the messages' senders by affecting a timestamp to each message broadcast, upon receiving the messages, the destination orders the messages according to their timestamps Lamport (1978), Bar-Joseph et al. (2002) and (5) destination agreement algorithms: in this class, the delivery order results from an agreement between the destination processes in different states Chandra and Toueg (1996), Rodrigues and Raynal (2000). However, the ordering mechanism is not the only key for atomic broadcast algorithms. The mechanism used for faults tolerance is another important characteristic of these algorithms. Two classes, in asynchronous systems, are defined: (1) algorithms that rely on unreliable failure detectors Chandra and Toueg (1996) and (2) algorithms that rely on group membership Chockler et al. (2001). It will be interesting to propose an algorithm that combines more than two classes. In Parisa et al. (2012), the authors proposed a solution that is a combination of several classes. The solution is based on the coordinator principle, similarly to the Paxos protocol Lamport (1998), to ensure the delivery order by affecting unique identifiers to broadcast messages. They also, once the value is picked by the coordinator, use the token principle for disseminating the proposed value until taking the decision. However, the solution uses ip-multicast primitives, which is unreliable communication mechanism and needs the retransmission, for unknown times, of lost messages. The solution, also, assumes very strong hypothesis: one coordinator and failure free partially synchronous model. Consequently, these hypotheses make the protocol impractical in some models and increase the time execution of the protocol.

In this paper, we propose a new solution for the atomic broadcast problem. It circumvents the different inconvenient faced in the RP and adds other characteristics from other classes to obtain a novel algorithm for the atomic broadcast. Our solution assumes a general asynchronous system. It combines two different classes: the rotating coordinator and the token ring as the ordering mechanisms and the $\diamondsuit S$ failure detector Chandra and Toueg (1996) for tolerating faults. In order to disseminate messages, our protocol implements the reliable broadcast primitives that is a broadcast message is received by all correct processes or by none of them. These characteristics make our protocol general and perform good results in terms of time and the number of messages needed to reach a decision.

3. SYSTEM MODEL AND DEFINITIONS

3.1. System Model

We consider a distributed asynchronous, there is no assumption about the relative speed of processes nor on message transfer delays, system composed

of n processes $\Pi = \{p_1, p_2, \ldots, p_n\}$. Processes communicate by message passing through reliable channel, defined by the two primitives *send(m)* and *receive(m)*. A process can only fail by crashing. At most f ($< n/2$) processes are faulty, when n is the total number of the processes in the system. By definition a correct process is a process that does not crash. A faulty process is one that is not correct. We assume that the system is augmented with $\diamondsuit S$ failure detector that provides (possibly incorrect) information about the processes that are suspected to have crashed.

3.2. Atomic Broadcast

Atomic broadcast is an agreement problem that is equivalent to consensus Chandra and Toueg (1996), defined by two primitive *abroadcast(m)* and *adeliver(m)*. Atomic Broadcast satisfies the following properties Hadzilacos (1993):(1) if a correct process abroadcasts a message m, then it eventually adelivers m (*validity*), (2) for any message m, a correct process adelivers m at most once and only if m was previously abroadcast (*integrity*), (3) if a correct process adelivers m, then all correct processes eventually adeliver m (*agreement*), (4) if a correct process adelivers m before m', then every correct process adelivers m' only after it has adelivered m (*total order*).

3.3. Failure Detector

We refer below to the $\diamondsuit S$ failure detector class introduced in Chandra and Toueg (1996) and defined by the two following properties: (i) eventually every process that crashes is permanently suspected by every correct process (*strong completeness*), and (ii) there is a time after which some correct process is never suspected by any correct process (*Eventually Weak Accuracy*). Notice that every process is equipped by a local failure detector that maintains a set of processes suspected to have crashed.

4. OVERVIEW OF THE RING PAXOS ALGORITHM

The Ring Paxos (RP) Algorithm Parisa et al. (2012) is an atomic broadcast algorithm that is based on the Paxos algorithm Lamport (1998). The computation proceeds in asynchronous rounds. Each round is composed of five tasks, distributed on two phases. A process in a round $crnd$ can take one or several roles among: (1) *proposer*, (2) *acceptor*, (3) *learner*. A proposer initiates the protocol by proposing a value. Several proposers can coexist during one round, but one value can be chosen by a coordinator. Once a coordinator (a proposer or an acceptor) receives the needed value, it begins the first phase of its round. For this purpose, it updates the variable $crnd$: the highest-numbered round that the coordinator has started to an arbitrary value that will be the greater round number picked so far and $cring$: the proposed ring for this round. It sends, using ip-multicast primitive, a *phase1A message* to a majority quorum $Q_a = (N_a+1)/2$ of acceptors (N_a is the set of acceptors) placed in a logical directed ring including the coordinator as the last acceptor, differently than Paxos. The proposed structure reduces the incoming messages at the coordinator and balances the communication among acceptors. The purpose of this message is to propose the ring that will be used in the remaining parts of the round. Upon receiving the message by an acceptor, it affects the variables rnd: the highest-numbered round, in which the acceptor has participated and $ring$: the current ring accepted by those received from the coordinator then it replays by a *phase1B message* accepting the coordinator proposition and proposing the last voted value $vval$: the value voted by the acceptor in round $vrnd$: the highest-numbered round in which the acceptor has cast a vote (it follows that $rnd \leq vrnd$ always holds). Upon receiving a majority *phase1B messages* for the same round, the coordinator calculates $cval$: the value that the coordinator has picked for round $crnd$ (chosen from the received acceptors values or a new proposed value by a proposer), affects a $cvid$: a unique identifier affected to the proposed value. It, then executes the phase 2 of its round and ip-multicasts a *phase2A message* to Q_a + N_l (the set of learners). In the remaining tasks, acceptors try to agree on the value proposed by the coordinator. So, after ip-delivering a *phase2A message* by an acceptor with the same sending information, it updates its $vrnd$,$vval$ and affects its $vvid$: a unique identifier affected to the proposed value by $cvid$. If it is not the last acceptor in the ring, it keeps sending a *phase2B message* to its successor. Otherwise, if it is the last acceptor in the ring, $i.e.$ the coordinator, then it ip-multicasts the $DECISION$ message to all the processes (Q_a + N_l).

5. FAILURE DETECTOR-RING PAXOS BASED ATOMIC BROADCAST ALGORITHM

The FD-RP Atomic Broadcast Algorithm, given in Algorithm 1, is also based on the Paxos algorithm Lamport (1998), and similarly, every process can be a proposer, an acceptor or a learner. The algorithm proceeds in asynchronous rounds. Each round is coordinated by one coordinator, it is process number $(crnd \mod n)+1$. A coordinator can be a proposer or an acceptor. In the first task of the first phase and after receiving propositions from the proposers, the coordinator launches the algorithm by broadcasting a *phase1A message* to the set of acceptors. The message contains the round

number rnd and the proposed ring number $cring$ that will be used in the remaining tasks (*Task1*). Two cases can be distinguished. The acceptor suspects the coordinator to be crashed, by requesting its associated failure detector $\Diamond S$, and proceeds to the next round. Otherwise, it replays by a *phase1B message* informing the coordinator that it abides by the proposed ring. It attaches the message by its last voted value $vval$ (*Task2*). If the coordinator hasn't received a majority of responses from the acceptors, because of acceptors' failures or its suspicion by those acceptors, it aborts the round. Otherwise, the coordinated has received a majority of messages; it chooses the value to be proposed $cval$. This value can be the voted value during the greater round executed so far, or a new value proposed by a proposer. Notice that the value proposed can be an update of a copy in a distributed database, a shared object, etc. Then, the coordinator affects an identifier $cvid$ to the proposed value and sends a *phase2A message* to the acceptors (*Task3*). An acceptor, that hasn't suspected the coordinator for the current round (by requesting its failure detector), updates its variables $vval$, $vrnd$ and $vvid$. It checks if it is not the last acceptor in the ring. If so, it keeps sending a *phase2B message* to its successor in the ring (*Task4*). Otherwise, it is the coordinator, it broadcasts a $DECISION$ message to all the processes (acceptors and learners) (*Task5*).

6. PROOF OF CORRECTNESS

Lemma1 *if a correct process adelivers a message m, then all correct processes eventually adeliver m* (Agreement property).

Proof This lemma is satisfied by the use of the reliable broadcast primitive to broadcast the decision. A reliable broadcast primitive ensures that all correct processes deliver the same set of messages broadcast by the coordinator. Let p_c be the coordinator that decides at round $crnd$ at *line 52*. Then, it rbroadcasts this decision (including all the processes in $Q_a \cup N_l$) at the same *line 52*. According to the reliable broadcast properties, all the processes will, eventually, receive the coordinator's message. Hence, they adeliver the set of messages according to a deterministic rule.

Lemma2 *If a correct process adelivers a message m then m has been abroadcast by a correct process* (validity property).

Proof Once a majority of acceptors agree on the ring $cring$ proposed by the coordinator of the round p_c (*line 10,11*), the coordinator chooses a value $cval$. This value can be a new value v received from a proposer at *line 2* or a voted value $vval$ of some processes in previous rounds calculated by p_c at *line 26*. The coordinator will propose the selected value to the majority of acceptors at *line 28*. Consequently,

Algorithm 1 FD-RP based AB Algorithm

1: **Task1** : $(Coordinator)$
2: $upon\ reception\ of\ value\ v\ from\ a\ proposer$
3: $increase\ crnd\ to\ an\ arbitrary\ unique\ value$
4: $let\ cring\ be\ an\ overlay\ ring\ with\ processes\ in\ Q_a$
5: $for\ all\ p_i \in Q_a\ do\ send$ (p_i,($phase1A$,$crnd$,$cring$))
6: **Task2** : $(Acceptor)$
7: $wait\ until\ reception\ of$ ($phase1A$,$crnd$,$cring$) $from\ the\ coord\ or\ coord \in D_i$
8: **if** $coord \notin D_i$ **then**
9: **if** $crnd > rnd$ **then**
10: $rnd \leftarrow crnd$; $ring \leftarrow cring$; $send$ ($coord$,($phase1B$,rnd,$vrnd$,$vval$))
11: **end if**
12: **else**
13: $send$ ($coord$,($phase1B$,rnd,$nack$))
14: **end if**
15: **Task3** : $(Coordinator)$
16: $wait\ until\ reception\ of$ ($phase1B$,rnd,$vrnd$,$vval$)$from\ Q_a\ such\ that\ crnd$ =rnd
17: **if** $\exists$ $one\ message$ ($phase1B$,rnd,$nack$) **then**
18: $send$ ($Q_a \cup N_l$,($phase2A$,$crnd$,$Abort$))
19: **else**
20: $k \leftarrow Maxvrnd\ received$
21: $VS \leftarrow (vrnd, vval)\ received\ s.t.\ k = vrnd$
22: **if** $k = 0$ **then**
23: $cval \leftarrow v$
24: **else**
25: $cval \leftarrow vval$, $the\ only\ vval\ in\ VS$
26: $let\ cvid\ be\ the\ unique\ identifier\ for\ cval$
27: $send$ ($Q_a \cup N_l$,($phase2A$,$crnd$,$cval$,$cvid$))
28: **end if**
29: **end if**
30: **Task4** : $(Acceptor)$
31: $wait\ until\ reception\ of$ ($phase2A$,$crnd$,$cval$,$cvid$) $from\ the\ coordinator\ or\ coord \in D_i$
32: **if** $coord \notin D_i$ **then**
33: **if** $No\ message$ ($phase2A$,$crnd$,$Abort$) $has\ been\ received$ **then**
34: **if** $crnd > rnd$ **then**
35: $vrnd \leftarrow crnd$;$vval \leftarrow cval$;$vvid \leftarrow cvid$;
36: **if** $first(ring)$ **then**
37: $send$ ($successor$,($phase2B$,$crnd$, $cvid$))
38: **end if**
39: **end if**
40: **end if**
41: **else**
42: $send$ ($successor$,($phase2B$,$crnd$,$nack$))
43: **end if**
44: **Task5** : $(Coordinator\ and\ acceptor)$
45: $upon\ reception\ of$ ($phase2B$,$crnd$,$cvid$) $from\ a\ successor\ or$ ($phase2B$,$crnd$,$nack$)
46: **if** ($phase2B$,$crnd$,$cvid$) $has\ been\ received$ **then**
47: **if** $cvid$= $vvid$ **then**
48: **if** $not\ last(ring)$ **then**
49: $send$ ($successor$,($phase2B$,$crnd$,$cvid$))
50: **else**
51: $send$ ($Q_a \cup N_l$,($DECISION$,$cvid$))
52: **end if**
53: **end if**
54: **end if**

the decided value $cval$ has a total agreement from the acceptors in $cring$ (including the coordinator) and it is a proposition of the coordinator p_c of the current round $crnd$.

Lemma3 *For each message m , every correct process adelivers m at most once and only if m was previously broadcast* (integrity property).

Proof The coordinator p_c, after choosing the value to be proposed $cval$ at *line 24* or *26* (see *lemma 2*), affects to the proposed value $cval$ a unique identifier $cvid$ at *line 27* that can be used by the processes to check if the message has been adelivered or not.

Lemma4 *If some correct process adelivers m before m', then every correct process adelivers m' only after it has adelivered m* (total order property).

Proof The property is satisfied by the use of the $\diamondsuit S$ failure detector which ensures that eventually a correct process p_c will not be suspected by any correct process, it is the coordinator. Once p_c is elected for the round $crnd$, it proposes a value $cval$ at *line 28*, affects a unique identifier $cvid$ at *line 27* to this value (for an other round, the coordinator will affect $cvid'$ to $cval'$ in $crnd'$. Consequently, the coordinator will gather the votes, firstly, to $cval$ and deliver it, then to $cval'$. Hence, the order between the values (broadcast messages) is guaranteed.

Theorem *The algorithm 1 implements the Atomic Broadcast primitives.*

7. CONCLUSIONS

The paper has presented a new algorithm for solving atomic broadcast in asynchronous system. The algorithm inherits from RP its main characteristics; coordinator principle, ring structure to decide the order of messages. They make the communication centralized and reduce the contention of the system, i.e. the number of messages transmitted. We also introduced the $\diamondsuit S$ failure detector to tolerate process failures. The use of this mechanism is considered as the best method for fault tolerance. Hence, unlike RP, many steps can be omitted using failure detectors, especially when the coordinator is crashed, so processes haven't to wait, a long time, for its message and proceed directly to the next round. The same thing for the coordinator, when its associated failure detector suspects a majority of acceptors then it proceeds directly to the next round without waiting a long time for un-received messages from crashed acceptors. Hence, a good performance in time and message number is reached, in comparison with RP. These positive points will be justified by a simulation results as future goals.

8. REFERENCES

L. Lamport (1998) The part-time parliament. *ACM Transactions on Computer Systems*, 16, 133–169, May. 1998.

P. Verissimo, M. Raynal (1999) Time in Distributed System Models and Algorithms. *In proceeding, May. 1999*. 1–32.

X. Defago, A. Schiper, P. Urbn (2004) Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys*, Vol. 36, Issue 4, Page(s):372-421, Dec. 2004.

V. Hadzilacos and S. Toueg, *Reliable broadcast and related problems* , In Distributed Systems, ACM Press, Page(s):97-145, 1993.

H. Garcia-Molina, A. Spauster, *Ordered and reliable multicast communication*, ACM Trans. Comput. Syst, Vol. 9, Issue 3, Page(s):242-271, Aug. 1991.

K. P. Birman, A. Schiper, P. Stephenson *Lightweight causal and atomic group multicast*, ACM Trans. Comput. Syst, Vol. 9, Issue 3, Page(s):272-314, Aug. 1991.

J. M. Chang, N. F. Maxemchuk *Reliable broadcast protocols*, ACM Trans. Comput. Syst, Vol. 2, Issue 3, Page(s):251273, Aug. 1984.

F. Cristian, S. Mishra,G. Alvarez *High-performance asynchronous atomic broadcast*, Distributed System Engineering Journal, Vol. 4, Issue 2, Page(s):109-128, June. 1997.

R. Ekwall, A. Schiper, *A Fault-Tolerant Token-Based Atomic Broadcast Algorithm*, IEEE Trans. Dependable Sec. Comput, Vol. 8, Issue 5, Page(s):625-639, 2011.

Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agrawal, P. Ciarfella, *The Totem single-ring ordering and membership protocol*, ACM Trans. Comput. Syst, Vol. 13, Issue 4, Page(s):311-342, Nov. 1995.

L. Lamport, *Time, clocks, and the ordering of events in a distributed system*, ACM, Vol. 21, Issue 7, Page(s):558-565, July. 1978.

Z. Bar-Joseph, I. Keidar, N. Lynch, *Early-delivery dynamic atomic broadcast*, In Proc. 16th Intl. Symp. on Distributed Computing (DISC'02), D. Malkhi, Ed. LNCS, vol. 2508, pp. 1-16, 2002.

L. Rodrigues, M. Raynal, "Atomic broadcast in asynchronous crash-recovery distributed systems". In Proc. 20th IEEE Intl. Conf. on Distributed Computing Systems (ICDCS- 20),pp. 288-295, Juin 2000.

T. Chandra, S. Toueg, *Unreliable failure detectors for reliable distributed systems*, ACM, Vol. 43, Issue 2, Page(s):225-267, 1996.

G. Chockler, I. Keidar, R. Vitenberg, *Group communication specifications: A comprehensive study*, ACM, Vol. 33, Issue 4, Page(s):1-43, Dec. 1996.

P.J. Marandi, M. Primi, N. Schiper, F. Pedone, *Ring Paxos: A HighThroughput Atomic Broadcast*

Protocol, DNS,IEEE Computer Society, pp. 112, 2012.