Lightweight Message Logging Protocol for Distributed Sensor Networks

Jinho Ahn[1]

Dept. of Computer Science, Kyonggi University, Suwon-si, Gyeonggi-do, Korea
jhahn@kgu.ac.kr

Abstract

Among a lot of rollback-recovery protocols developed for providing fault-tolerance for long-running distributed applications, sender-based message logging with checkpointing is one of the most lightweight fault-tolerance techniques to be capable of being applied in this field, significantly decreasing high failure-free overhead of synchronous logging by using message sender's volatile memory as storage for logging its corresponding message. However, attempting to apply this technique into large-scale and geographically distributed systems such as broker-based sensor networks, etc., the following problems should be addressed; reducing the number of control and data messages passing on core networks occurring during its fully message logging and recovery procedures. In this paper, we present a lightweight message logging protocol for distributed sensor networks to solve all of them by employing the current and future distributed systems' architectural features mentioned above.

Keywords: *Distributed computing, Scalability, Reliability, Message logging.*

1. Introduction

Ubiquity, low-cost but, high capacity of small to medium size of computing devices highly easy to connect with others in wired, wireless or both ways are triggering not a few of architectural variations in distributed computing systems. This technological trend enables an arbitrarily connected set of nodes to provide low-cost high performance computing environments for their general users. However, this type of technical diversity tends to cause reliability issues for them to be difficult to be achieved [1, 2, 6, 7]. There are a lot of rollback-recovery algorithms developed for providing fault-tolerance for long-running distributed applications on small-scale, cluster, grid computing platforms [1, 2, 6, 11]. Among them, sender-based message logging [1, 5, 10] with checkpointing [2, 3, 8] is one of the most lightweight fault-tolerance techniques to be capable of being applied in this field. It may considerably lower high failure-free overhead of receiver-based message logging [9, 11] resulting from synchronously logging each message into stable storage, which can be realized by using volatile memory of its sender as storage for logging [1, 4, 5, 10]. Figure 1 shows four processes p, q, r, s where p is sending several messages to the others. In this example, sender-based message logging has process p maintain the log information of each sent message in its volatile memory. This beneficial feature can be obtained at the expense of extra communication required for allowing message senders to get receive sequence numbers(RSNs) of the messages from their receivers and confirm them with the receivers and slowness and complexity of recovery of each failed process coming from its obtaining message log from the corresponding senders. As architectural aspects

[1] Corresponding author: Tel.: +82 31 249-9674, FAX: +82 31 249-9673

of current and future distributed computing systems are changing to geographically group-based and peer-to-peer based, many of these systems are adopting broker-based architecture to accommodate these topological features well. Thus, this change is making a lot of issues about their fundamental building blocks reconsidered to work well for these newly fashioned systems in highly effective manners. Sender-based message logging abbreviated by SBML should also be examined properly before its application to accommodate this architectural change, which we are focusing on in this paper. In this point of view, the two drawbacks mentioned earlier most of sender-based message logging protocols [1, 5, 10] have may be amplified and highlighted greatly if applied into these new systems, being capable of significantly diminishing the practical value coming from their common advantageous features. This paper designs a lightweight message logging protocol for distributed sensor networks to address the critical problems by employing the current and future distributed systems' architectural features mentioned above.

The remainder of the paper is organized as follows. In sections 2 and 3, we explain failure-free and recovery operations of our broker-based SBML algorithm in detail. Finally, section 4 concludes this paper.

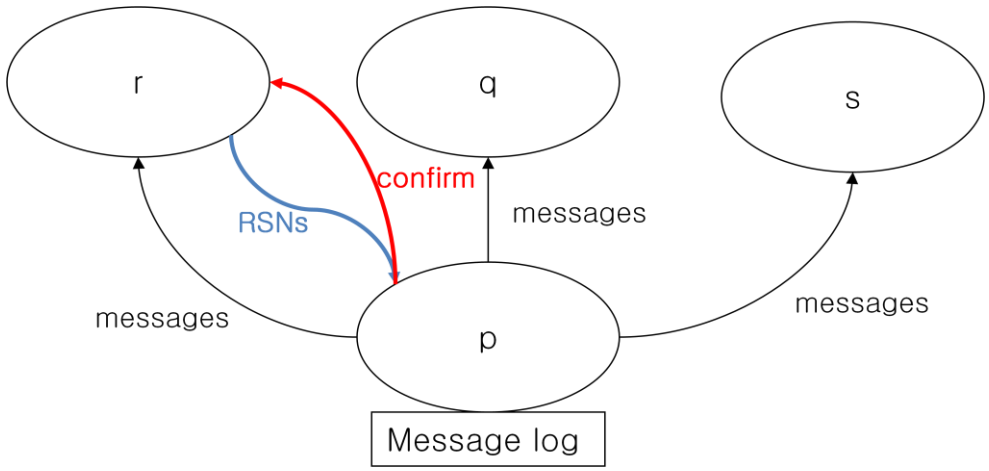

Figure 1. Basic Operations of SBML

2. Normal Operation Algorithm

First of all, let us assume the following system model; a distributed sensor networked system is composed of a finite, but arbitrary set of cells. A certain number of nodes reside on a cell among which one having generally, but not mandatorily, the highest capacity of resources such as CPU, memory, storage, network, etc., is elected as cell broker. Figures 2 and 3 illustrate the same instance of this system model assumed having three cells with 8 processes. In these figures, a gray-shaded ellipse on each cell indicates its cell broker. When a sender sends a message to its corresponding receiver, traditional sender-based message logging protocols require the following three steps; partially logging the message with its SSN (Send Sequence Number), IDs of its sender and receivers into its volatile storage, saving its RSN returned from its receiver into its

log element and informing the receiver of the success of fully logging the message on the sender's volatile storage. However, should they be applied into broker-based sensor networks, the previous SBML protocols may incur high failure-free overheads in terms of communication cost. This disadvantage occurs if inter-process communications frequently perform across cells. For example, in Figure 2, process P_{c1}^{1} sends message M1 to process P_{c2}^{2}, which sends M2 to P_{c3}^{3}. Then, P_{c3}^{3} sends M3 to P_{c3}^{2}, sending M4 to P_{c1}^{2}, which transmits M5 to P_{c1}^{1}. In this case, the protocols force their three steps with all three messages M1, M2 and M4 to be executed using cell-crossing interaction primitives.

In contrast to the previous protocols, our protocol enables the broker to manage the volatile storage for fully logging all the messages sent to every cell member, not including itself, from outside its managing cell like in Figure 4. This feature allows the second and the third steps to perform locally with each message receiver's broker. For example, figure 3 shows how our proposed protocol executes in the same scenario as Figure 2. In this case, every sender of each message sent to outside its cell like P_{c1}^{1}, P_{c2}^{2} or P_{c3}^{2} has only to maintain the partial log information of the message. Instead, each cell broker like P_{c2}^{1}, P_{c3}^{1} or P_{c1}^{1} keeps fully logged messages M1, M2 and M4 on its volatile storage respectively. From this figure, we can see that the failure-free steps excluding the first-step with messages M1, M2 and M4 are executed only inside each local cell of their receivers respectively.

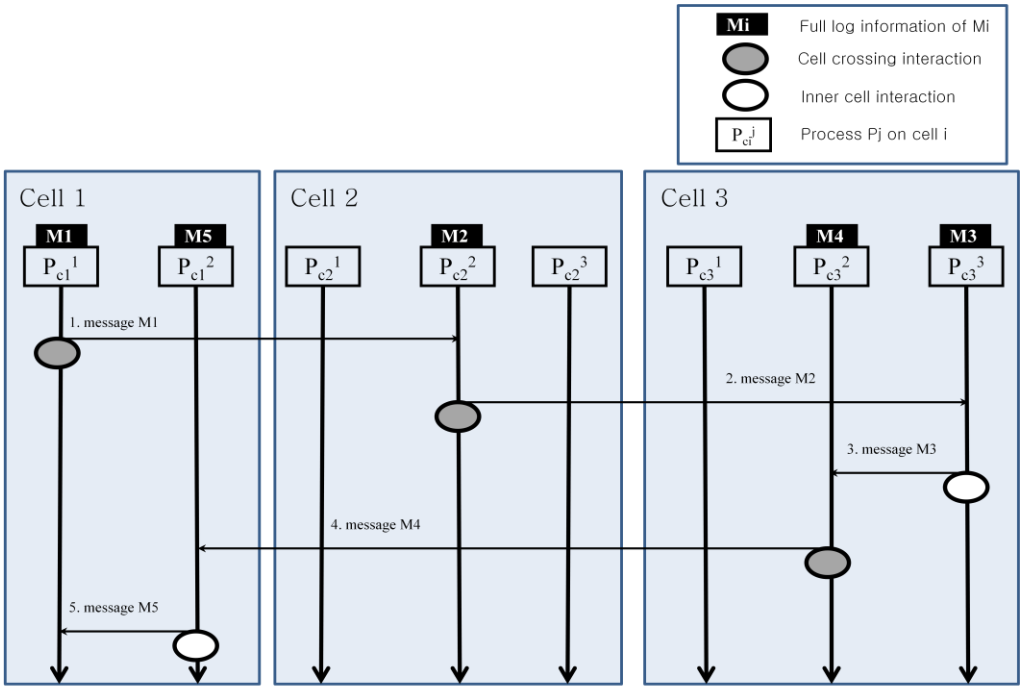

Figure 2. An Example of Traditional Protocols with 8 Processes on Three Cells

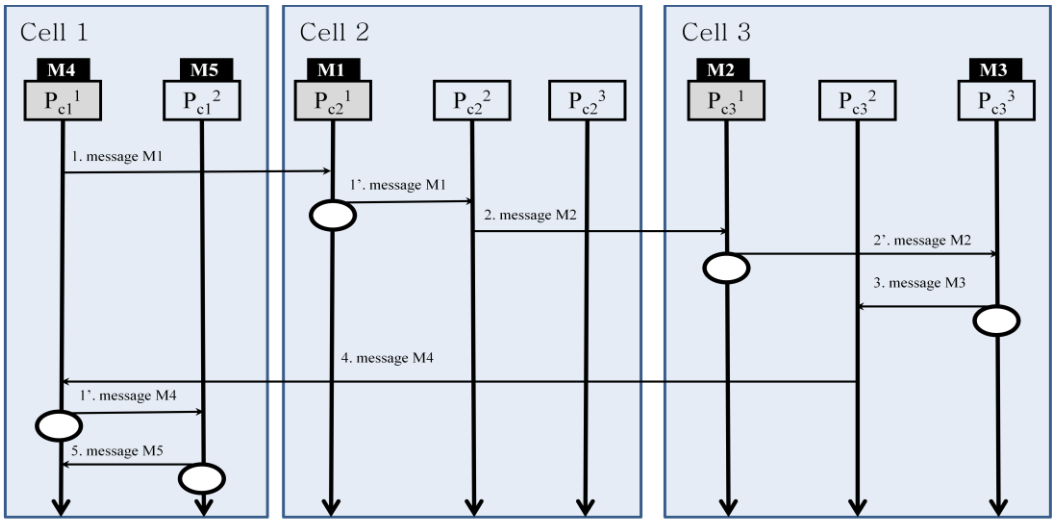

Figure 3. An Example of our Protocol with 8 Processes on Three Cells

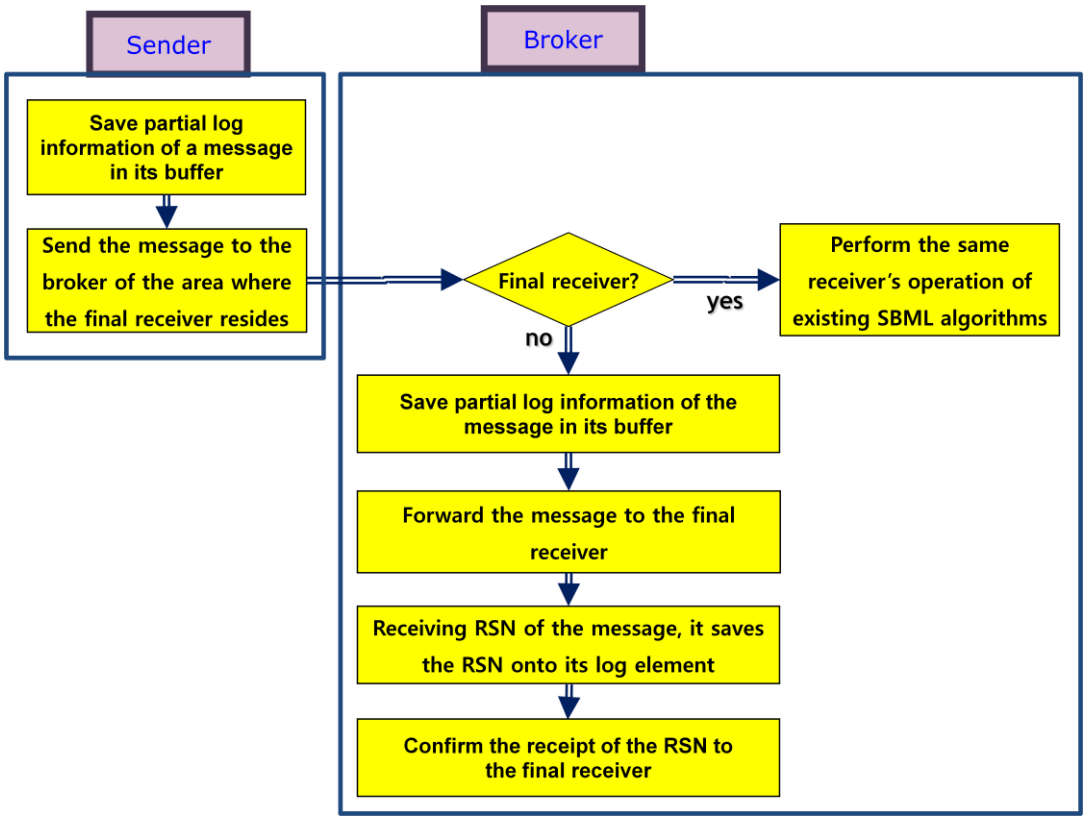

Figure 4. Normal Operation Steps of our SBML Protocol

3. Recovery Algorithm

Suppose a process crashes. In order to recover this process, sender-based message logging protocols have it restore a checkpointed state from the stable storage, obtain all the messages received from others before failure and replay them in their RSNs and then FIFO order. Although these previous protocols suffer from their high communication overhead occurring during recovery, our protocol can localize the recovery procedure of each process like in figure 5 as follows; First, if a process, not cell broker, fails according to the crash failure model, it can be recreated on an available node, restore a pre-failure state from its latest checkpoint and broadcast a recovery request only inside its local cell. Whenever it receives a reply including logged messages sent to itself from another process, it puts fully logged messages into its replay buffer in RSN order and then partially logged ones in FIFO order. After it has gotten replies from all the other nodes, it replays all the messages sent before its failure after its latest checkpoint accordingly.

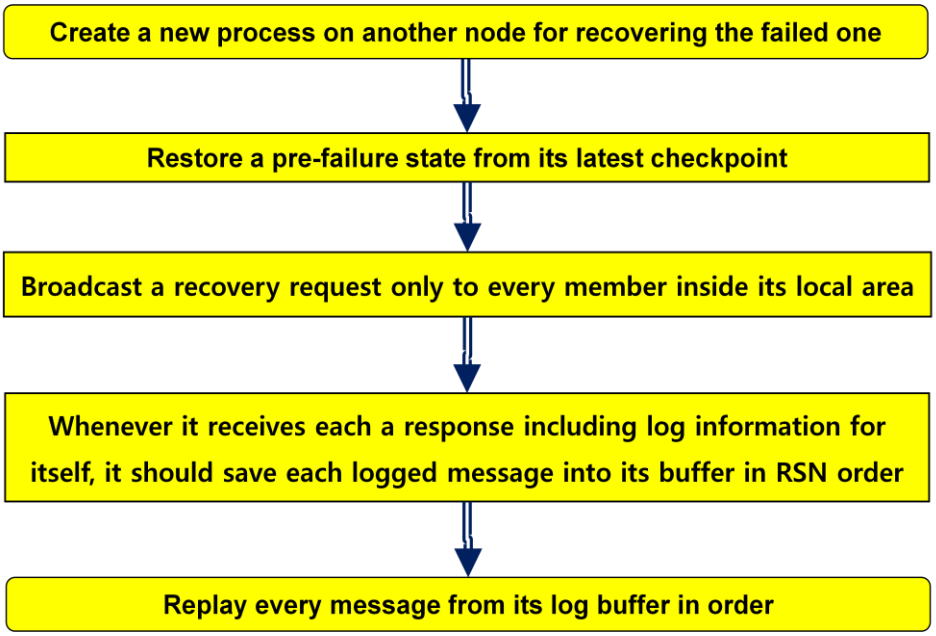

Figure 5. Recovery Steps of our SBML Protocol

4. Conclusion

The traditional SBML protocols may suffer from the following two overheads in common that may make it very difficult to use them in large-scale distributed sensor networks. Extra communication cost is required for allowing message senders to get RSNs of messages from their receivers and confirm their receipt to the receivers. Second, slowness and complexity of recovery procedure are coming from each failed process' obtaining message log from the corresponding senders. In this paper, we presented a lightweight message logging protocol for distributed sensor networks to employ the current and future distributed systems' architectural features for considerably alleviating these two drawbacks.

References

[1] A. Bouteiller, F. Cappello, T. Herault, G. Krawezik, P. Lemarinier and F. Magniette, "MPICH-V2: a Fault Tolerant MPI for Volatile Nodes based on Pessimistic Sender Based Message Logging", In Proc. of the Int'l Conf. on High Performance Networking and Computing, **(2003)**.

[2] D. Buntinasd, C. Coti, T. Herault, P. Lemarinier, L. Pilard, A. Rezmerita, E. Rodriguez and F. Cappello, Blocking vs. non-blocking coordinated checkpointing for large-scale fault tolerant MPI Protocols, Future Generation Computer Systems, 24, **(2008)**.

[3] K. M. Chandy and L. Lamport, Distributed Snapshots: Determining Global States of Distributed Systems, ACM Transactions on Computer Systems, 3, 1, **(1985)**.

[4] E. N. Elnozahy, L. Alvisi, Y. M. Wang and D. B. Johnson, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems", ACM Computing Surveys, 34, 3, **(2002)**.

[5] D. Johnson and W. Zwaenpoel, "Sender-Based Message Logging", Int'l Symp. on Fault-Tolerant Computing, **(1987)**.

[6] T. LeBlanc, R. Anand, E. Gabriel and J. Subhlok, "VolpexMPI: An MPI Library for Execution of Parallel Applications on Volatile Nodes", Lecture Notes In Computer Science, 5759 **(2009)**.

[7] H. F. Li, Z. Wei and D. Goswami, "Quasi-atomic recovery for distributed agents", Parallel Computing, 32, **(2009)**.

[8] Y. Luo and D. Manivannan, "FINE: A Fully Informed aNd Efficient communication-induced checkpointing protocol for distributed systems", J. Parallel Distrib. Comput., 69, **(2009)**.

[9] M. Powell and D. Presotto, "Publishing: A reliable broadcast communication mechanism", In Proc. Of the 9th International Symposium on Operating System Principles, **(1983)**.

[10] J. Xu, R.B. Netzer and M. Mackey, "Sender-based message logging for reducing rollback propagation", In Proc. of the 7th International Symposium on Parallel and Distributed Processing, **(1995)**.

[11] B. Yao, K. Ssu and W. Fuchs, "Message Logging in Mobile Computing", In Proc. of the 29th International Symposium on Fault-Tolerant Computing, **(1999)**.

Authors

Jinho Ahn

He received his B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University, Korea, in 1997, 1999 and 2003, respectively. He has been an associate professor in Department of Computer Science, Kyonggi University since 2003. He has published more than 70 papers in refereed journals and conference proceedings and served as program or organizing committee member or session chair in several domestic/international conferences and editor-in-chief of journal of Korean Institute of Information Technology and editorial board member of journal of Korean Society for Internet Information. His research interests include distributed computing, fault-tolerance, sensor networks and mobile agent systems.