Brief Announcement: Optimal Atomic Broadcast and Multicast Algorithms for Wide Area Networks[*]

Nicolas Schiper
University of Lugano
Switzerland
nicolas.schiper@lu.unisi.ch

Fernando Pedone
University of Lugano
Switzerland
fernando.pedone@unisi.ch

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Algorithms, Performance, Reliability, Theory

Keywords

fault-tolerance, atomic broadcast/multicast, lower bounds, wide area networks

1. INTRODUCTION

Distributed applications spanning multiple geographical locations have become common in recent years. Typically, each geographical site, or *group*, hosts an arbitrarily large number of processes connected through high-end local links; a few groups exist, interconnected through high-latency communication links. As a consequence, communication among processes in the same group is cheap and fast; communication among processes in different groups is expensive and orders of magnitude slower than local communication. Application data is replicated both locally, for high availability, and globally, usually for locality of access. In this paper, we investigate the atomic broadcast and multicast problems, two communication primitives that offer adequate properties, namely agreement on the set of messages delivered and on their delivery order, to implement data replication [3].

As opposed to atomic broadcast, atomic multicast allows messages to be sent to a subset of processes in the system. More precisely, messages can be addressed to any subset of the system's groups Γ, each message possibly being multicast to a different subset. The set of groups to which a message m is multicast is denoted by $m.dest$.[1] Atomic broadcast and multicast are defined by the primitives A-XCast, i.e., A-BCast or A-MCast whether the message is broadcast or multicast, and A-Deliver. Moreover, we define the relation $<$ on the set of messages that processes (correct or faulty) A-Deliver as follows: $m < m'$ if and only if a process A-Delivers m before m'.

[*]The work presented in this paper has been partially funded by the SNSF, Switzerland (project #200021-107824).

[1]In the case of atomic broadcast, for every message m, $m.dest = \Gamma$.

Copyright is held by the author/owner(s).
PODC'07, August 12–15, 2007, Portland, Oregon, USA.
ACM 978-1-59593-616-5/07/0008.

Atomic broadcast and multicast satisfy the following properties[2]:

- *Uniform Integrity:* For any process p and any message m, p A-Delivers m at most once, and only if $p \in m.dest$ and m was previously A-XCast.
- *Validity:* If a correct process p A-XCasts m, then all correct processes $q \in m.dest$ eventually A-Deliver m.
- *Uniform Agreement:* For any message m, if a process p A-Delivers m, then all correct processes $q \in m.dest$ eventually A-Deliver m.
- *Uniform Total Order:* The relation $<$ is acyclic.

From a problem solvability point of view, atomic multicast can be easily reduced to atomic broadcast: every message is broadcast to all the groups in the system and only delivered by those processes the message is originally addressed to. Obviously, this solution is inefficient as it implies communication among processes that are not concerned by the multicast messages. To rule out trivial implementations of no practical interest, we require multicast algorithms to be *genuine*, i.e., only processes addressed by the message should be involved in the protocol. A genuine atomic multicast can thus be seen as an adequate communication primitive for distributed applications spanning multiple geographical locations in which processes store a subset of the application's data (i.e., *partial replication*).

2. OPTIMAL ALGORITHMS FOR WIDE AREA NETWORKS

Ideally, we would like to devise algorithms that use inter-group links as sparingly as possible, saving on both latency and bandwidth (i.e., number of messages). To measure the latency cost of broadcast (multicast) algorithms, we use their *latency degree*. Consider an algorithm $\mathcal{A}$ and a run R of $\mathcal{A}$ in which messages in a set M are A-XCast. Informally—a precise definition is presented in [4]—the latency degree of R, $\Delta(R)$, is the minimum number of inter-group message delays, among all messages m in M, between the A-XCast of m and the last delivery of m among the processes that A-Deliver m.

In [4], we present a fault-tolerant genuine atomic multicast algorithm that has a latency degree of two for messages that are multicast to at least two groups, i.e., there exists a

[2]Note that, by abuse of notation, we write $p \in m.dest$ instead of $\exists g \in \Gamma : g \in m.dest \land p \in g$.

run R in which a message is multicast to at least two groups such that $\Delta(R) = 2$. Our algorithm is inspired by Fritzke *et al.*'s [2], a fault-tolerant version of Skeen's algorithm, originally described in [1].

We show the optimality of our algorithm by establishing a lower bound on the latency degree of genuine multicast protocols. A corollary of this result is that Skeen's algorithm is also optimal—a surprising result not for its technical difficulty, but for the fact that it has apparently been left unnoticed by the scientific community for more than 20 years.

Theorem 1. *In a failure-free system with reliable links*[3] *and at least two groups, for any algorithm $\mathcal{A}$ solving genuine atomic multicast, there does not exist runs R_1, R_2 of $\mathcal{A}$ in which one message is multicast to at least two groups such that $\Delta(R_1) = \Delta(R_2) = 1$.*

Note that our algorithm achieves a lower latency degree for messages that are multicast to a single group. In such a case, our protocol has a latency degree of one, if the multicaster process is outside the destination group, and zero otherwise.

We demonstrate that atomic multicast is inherently more expensive than atomic broadcast by presenting a fault-tolerant broadcast algorithm with a latency degree of one. To achieve such a low latency, the algorithm is proactive, i.e., it may take actions even though no messages are broadcast. Nevertheless, we show how it can be made *quiescent*: provided that a finite number of messages are broadcast, processes eventually cease to communicate. In runs where the algorithm becomes quiescent too early, that is, a message m is broadcast after processes have decided to stop communicating, m will not be delivered in a single inter-group message delay, but in two. We show that this extra cost is unavoidable, i.e., no quiescent atomic broadcast algorithm can hope to always achieve a latency degree of one.

Definition 1.

1. *Let $\mathcal{R}(\mathcal{A})_{qs}$ be the runs of a quiescent atomic broadcast algorithm $\mathcal{A}$ in which a finite number of messages are A-BCast. Since $\mathcal{A}$ is quiescent, in every run of $\mathcal{R}(\mathcal{A})_{qs}$, there is a time t after which no message is received.*
2. *Let the restarting runs $\mathcal{R}(\mathcal{A})_{rs}$ be the extensions of runs $\mathcal{R}(\mathcal{A})_{qs}$, starting after time t, where one message is A-BCast.*

Theorem 2. *In a failure-free system with reliable links*[3] *and at least two groups, for any quiescent algorithm $\mathcal{A}$ solving atomic broadcast, there does not exist runs $R_1, R_2 \in \mathcal{R}(\mathcal{A})_{rs}$ such that $\Delta(R_1) = \Delta(R_2) = 1$.*

Interestingly, our broadcast algorithm performs better under a high frequency of broadcasts, i.e., if the time between two consecutive broadcasts is smaller than the largest inter-group message delay, the algorithm will not become quiescent prematurely. In wide area network settings where the inter-group message delay can easily reach 100 milliseconds, ten broadcasts per second is thus enough.

In addition to their latency degree optimality, the proposed algorithms ensure a *locality* property, i.e., the underlying consensus abstraction they use is always run inside groups. This property is of significant importance for one main reason: the oracle used to solve consensus does not span groups. In the case of failure detectors, this means sparing the bandwidth of inter-group links. Furthermore, as intra-group links have a lower and more predictable message delay than inter-group links, the speed and accuracy of failure detection will be improved, thus reducing the time to execute consensus.

Compared to existing solutions, our algorithms either achieve a lower latency degree or equal the best latency degree but send fewer intra-group messages.

3. CONCLUDING REMARKS

These results help better understand the difference between atomic broadcast and multicast. In particular, they point out a tradeoff between the latency degree and message complexity of these two problems. Consider a partial replication scenario where each group replicates a set of objects. If latency is the main concern, then every operation should be broadcast to all groups, and only groups concerned by the operation handle it. This solution, however, has a high message complexity: every operation leads to sending at least one message to all processes in the system. Obviously, this is inefficient if the operation only *touches* a subset of the system's groups. To reduce the message complexity, genuine multicast can be used. However, any genuine multicast algorithm will have a latency degree of at least two.

The design principles of the algorithms, their correctness proofs as well as the optimality results can be found in [4].

4. REFERENCES

[1] K. P. Birman and T. A. Joseph. Reliable communication in the presence of failures. *ACM Trans. Comput. Syst.*, 5(1):47–76, 1987.

[2] U. Fritzke, Ph. Ingels, A. Mostéfaoui, and M. Raynal. Fault-tolerant total order multicast to asynchronous groups. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, pages 578–585, October 1998.

[3] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[4] N. Schiper and F. Pedone. Optimal atomic broadcast and multicast algorithms for wide area networks. Technical Report 2007/004, University of Lugano, 2007.

[3]Reliable links do not corrupt, duplicate or lose messages.