Is Reliable Multicast too Expensive? Let's be optimistic*

R. Jiménez-Peris, M. Patiño-Martínez
Technical University of Madrid (UPM)
Facultad de Informática
E-28660 Boadilla del Monte
Madrid, Spain
{rjimenez, mpatino}@fi.upm.es

G. Alonso
Swiss Federal Institute of Technology
Department of Computer Science
CH-8092 ETH-Zentrum
Zürich, Switzerland
alonso@inf.ethz.ch

Abstract

Reliable multicast is one of the main abstractions to build fault-tolerant distributed systems. However, there is still a lot of reluctance to use it in many applications, e.g. in the database industry, due to its poor latency. There are some practical reasons for this reluctance, in particular, when talking about total ordered multicast, and more especially, uniform multicast. Fortunately, there is a wide class of applications that can use optimistic approaches to hide the cost of reliable multicast by overlapping the time it takes a message to be definitively delivered with the optimistic processing of the message. In this paper, we discuss the benefits of using optimism in the context of some applications from the information systems field, namely, data replication and atomic commitment.

1 Introduction

Nowadays, computer clusters are widely used as the platform of modern information systems. At the same time, reliable multicast [Bir96] has become one of the main abstractions for building reliable distributed systems. Thus, it might seem natural to use reliable multicast as a building block to build current information systems. However, there is a lot of reluctance in the information systems community to use reliable multicast. There are some practical reasons for this reluctance. Since, one of the crucial aspects of modern information systems is the response time of transactional requests, the latency introduced by reliable multicast, especially, when total order and/or uniformity are provided, is not negligible. This increase in latency is not admissible for the high performance requirements of these systems.

There are two alternatives to ease the applicability of reliable multicast to information systems. The first alternative would be to elaborate faster reliable multicast protocols. This has been explored in [PGS98]. In this work, the spontaneous total ordered in a LAN network is exploited to reduce the cost of total-ordered multicast. However, reliable multicast with some additional ordering (e.g. total order) or reliability properties has an inherent cost that imposes some limits to this approach.

The second alternative is more aggressive and exploits the application semantics to hide the cost of reliable multicast. In this approach [KPAS99], multicast messages are delivered optimistically as soon as they are received. Thus, the application can start their processing in an optimistic fashion. When the rest of the (reliability and ordering) properties of the multicast message are met, the message is definitively delivered to the application. The time elapsed between the optimistic delivery and the definitive delivery can be masked off by processing optimistically the message. However, one must be careful with the use of optimism, being too optimistic can result in a high

*This research has been partially funded by the Spanish National Research Council CICYT under grant TIC98-1032-C03-01.

number of rollbacks. The optimism should be such that the odds to be wrong are very small, what can be achieved by providing some safeguards when designing the protocol.

In the rest of the paper we describe two successful approaches of the use of optimistic delivery of reliable multicast messages in the context of modern information systems. The first approach that we will show, it is an eager data replication protocol based on totally-ordered multicast. In this approach the calculation of the total order is overlapped with the optimistic execution of transactions. The second approach that will be presented is a replicated atomic commitment server. This server provides a low latency non-blocking atomic commitment. The non-blocking property is provided by means of uniform multicast. Optimistic delivery is used to hide the inherent cost of this multicast. In this way, valuable locked resources can be released earlier, resulting in a lower latency of the commit.

The paper is structured as follows. Section 2 discusses the different properties of interest of reliable multicast. Section 3 discusses a data replication protocol based on optimistic delivery of total ordered multicast. Section 4 presents an atomic commitment server based on optimistic delivery of uniform multicast. Finally, Section 5 summarizes our conclusions.

2 Reliable Multicast and Optimistic Delivery

Reliable multicast [Bir96, VKCD99] can provide different ordering and reliability guarantees. Three events are usually distinguished, the multicast of the message, its reception, and its delivery. Reliable multicast guarantees that if a correct process of the target group delivers a message every correct process will deliver it. Reliability provides atomicity in the sense that a multicast request will be process by all correct group members or by none of them.

Sometimes some ordering guarantees are helpful. In particular, total order multicast ensures that every group member will deliver messages in the same order. Total ordering provides a way to schedule conflicting actions in the same order at all the group members.

In other occasions, stronger reliability guarantees are needed, like uniformity, that guarantees that if a process, correct or not, delivers a message every correct process will deliver it. This is useful in cases where a process can perform a persistent action like storing data on secondary storage or multicast a message to another group.

Reliable multicast is usually enriched with virtual synchrony [Bir96, VKCD99]. In a virtual synchronous multicast environment, two kinds of events are delivered, messages and view changes. View changes inform about process failures and joins, that is, the provide the set of processes perceived as correct. Virtual synchrony (in its strong version) ensures that all the processes transiting to a new view, will deliver the same set of messages in the previous view. This property is helpful to agree in which requests have already been provided and which ones are still pending. Additionally, it provides a means to ensure a state synchronization among all the group members.

Optimistic delivery consists in delivering messages in two steps. As soon as the message is received by a process, the message is optimistically delivered (opt-delivered). This delivery does not provide any ordering or reliability guaranty. When the associated guarantees to the multicast messages hold, the message is definitively delivered (def-delivered). It is up to the application, what can be done with a message opt-delivered and not def-delivered. The application can optimistically process it right away, or wait until it has the certainty that the probability of rollback is low enough.

3 Data Replication

Data replication has been used for two different and exclusive purposes. On one hand, it has been used to provide high availability, tolerating faults of some replicas. However, this high availability has always been provided at the cost of no scalability, the whole system performs worse than a single site. On the other hand, data replication has been used to increase the throughput of the system. However, most successful approaches have been based on lazy replication that does not provide full consistency, and in the advent of failures some updates can be lost.

Postgres-R [KA00] is one of the few successful approaches to provide fully consistent and scalable replication. This approach is based on total order multicast, and thus it has to pay for its latency. A newer approach that builds on this work [PJKA00], hides the cost of total order multicast behind the optimistic processing of transactions.

This approach takes advantage of a priori knowledge of concurrency control. That is, data are partitioned into non-intersecting subsets called conflict classes, and before processing a transaction is known which of these conflict classes the transaction will access. Since in a local area network, multicast messages are spontaneously total ordered, it is worth to process a transaction as soon as it is opt-delivered, provided that no other conflicting transactions are known to be optimistically executing, or optimistically executed, to prevent cascading aborts. This means that most of the transaction processing time is overlapped with the calculation of the total ordering, therefore transactions do not pay for the total ordering. Additionally, this protocol applies a reordering technique that even when the optimistic order does not conform the total order, it results sometimes possible to commit transactions definitively in the optimistic order violating the total order without compromising consistency and hence, reducing the number of aborts due to misordering of optimistic deliveries.

4 Atomic Commitment

There have been a lot of research to provide non-blocking atomic commitment (NBC), such as three phase commit [Ske81, KD95, GLS95]. Despite all this work, the standard atomic commitment protocol in the database industry is two phase commit (2PC), because non-blocking alternatives are too expensive in terms of latency. A NBC protocol can be easily implemented on top of uniform multicast [BT93]. However, three rounds are still needed (due to uniformity that has two implicit rounds), and what is more, these three rounds are inherent [DS83].

Although, it seems that there is no hope to reduce the latency of NBC, it is possible to incorporate the use of optimism to reduce the average latency of NBC. This approach has been taken in [JPAA01]. This protocol is triggered by a prepare message multicast to all the participants in the commit protocol. In response to the prepare message, a participant uniform multicasts its vote to the commit server. One member of the commit server will act as coordinator of the commit. This member will optimistically commit the transaction as soon as it opt-delivers all the votes (and if they are all yes). In this way, blocked transactions awaiting locks by the optimistically committed transaction will be able to progress at the end of the second round of the commit protocol. When the definitive delivery of all votes takes place, the transaction is definitively committed.

This optimism can be wrong, in which case an optimistically committed transaction can be aborted, and thus those transactions that acquired the optimistically released locks will also be aborted. To prevent the possibility of cascading aborts, transactions that get optimistically released locks are not allowed to enter the commit protocol until the transaction that released them commits definitively. However, it must be noticed that the odds for this optimism to be wrong are extremely low. The only situation under which an optimistically committed transaction can be aborted is characterized as follows:

- All the votes were affirmative.
- At least one participant, let us say p, fails after multicasting its vote.
- p's vote is opt-delivered by the transaction coordinator in the commit server.
- The transaction coordinator commits optimistically the transaction.
- Before p´s vote reaches any other member of the commit server group, p fails and the same happens with the transaction coordinator.

- Another member of the commit server group takes over when the transaction coordinator failure is noticed, and it will abort the transaction due to the missing vote from the failed participant.

As it can be seen, this situation is very unlikely, and so it will happen very infrequently, what means that the use of optimism in this protocol will mean in practice, that the optimistic NBC delays conflicting transactions only for the duration of two rounds. This means that optimistic delivery of uniform multicast combined with optimism at the application level (i.e., the commit protocol) has been able to mask off the latency of uniform multicast.

5 Conclusions

Reliable multicast is a powerful building block for reliable distributed systems. Its main shortcoming is its latency. However, in a wide class of applications this latency can be effectively hidden by overlapping the definitive delivery with the optimistic processing of the multicast messages. In this paper, this approach has been shown in the context of two important protocols for current information systems, such as data replication and non-blocking atomic commitment.

References

[Bir96] K.P. Birman. *Building Secure and Reliable Network Applications.* Prentice Hall, NJ, 1996.

[BT93] O. Babaoglu and S. Toueg. Understanding Non-Blocking Atomic Commitment. In *Distributed Systems.* Addison Wesley, 1993.

[DS83] C. Dwork and D. Skeen. The Inherent Cost of Nonblocking Commitment. In *Proc. of ACM PODC*, pages 1–11, 1983.

[GLS95] R. Guerraoui, M. Larrea, and A. Schiper. Non-Blocking Atomic Commitment with an Unreliable Failure Detector. In *Proc. of the 14th IEEE Symp. on Reliable Distributed Systems*, Bad Neuenahr, Germany, September 1995.

[JPAA01] R. Jiménez Peris, M. Patiño Martínez, G. Alonso, and S. Arévalo. A Low-Latency Non-Blocking Commit Server. In *Int. Conf. On Distributed Computing, DISC'01*, 2001.

[KA00] B. Kemme and G. Alonso. Don't be lazy, be consistent: Postgres-R, A new way to implement Database Replication. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Cairo, Egypt, September 2000.

[KD95] I. Keidar and D. Dolev. Increasing the Resilience of Atomic Commit at No Additional Cost. In *Proc. of ACM PODS*, 1995.

[KPAS99] B. Kemme, F. Pedone, G. Alonso, and A. Schiper. Processing Transactions over Optimistic Atomic Broadcast Protocols. In *Proc. of 19th IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, pages 424–431, 1999.

[PGS98] F. Pedone, R. Guerraoui, and A. Schiper. Exploiting Atomic Broadcast in Replicated Databases. In D. J. Pritchard and J. Reeve, editors, *Proc. of 4th International Euro-Par Conference*, volume LNCS 1470, pages 513–520. Springer, September 1998.

[PJKA00] M. Patiño Martínez, R. Jiménez Peris, B. Kemme, and G. Alonso. Scalable Replication in Database Clusters. In *Proc. of Distributed Computing Conf., DISC'00. Toledo, Spain*, volume LNCS 1914, pages 315–329, October 2000.

[Ske81] D. Skeen. Nonblocking Commit Protocols. *ACM SIGMOD*, 1981.

[VKCD99] R. Vitenberg, I. Keidar, G. V. Chockler, and D. Dolev. Group Communication Specifications: A Comprehensive Study. Technical Report CS99-31, Hebrew Univ., September 1999. To be published in ACM Comp. Surveys.