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 behind fault-tolerant distributed systems. However, and in spite of the intensive research done in the area, performance considerations (mainly the high latency) have prevented its widespread use in practical systems such as databases. Performance problems are particularly acute in the cases of total ordered multicast and uniform multicast, the versions that provide higher fault-tolerance. Fortunately, it has been recently shown that it is possible to hide the cost of the multicast operation behind the processing of the message. Such optimistic protocols can be used very effectively in a wide variety of applications. In this paper, we discuss the benefits of optimistic multicast and describe its potential use in data replication and atomic commitment.

1 Introduction

Computer clusters are the hardware platform of choice for many types of information system. At the same time, reliable multicast [Bir96] has become one of the main abstractions for building fault-tolerant distributed systems. Since reliable multicast is very appropriate for cluster architectures, it should be a key building block in modern information systems. Unfortunately, there is a lot of reluctance in the information systems community to use such protocols on account of their performance. Typical measures of efficiency in an information system are throughput and response time. The latency overhead introduced by reliable multicast, especially when total order and/or uniformity are provided, can have a severe effect on these efficiency measurements. Thus, in practice, designers opt for protocols that provide less guarantees but have a a more stable behavior under load.

The ideal would be to have protocols that provide the same strong guarantees as reliable multicast but without the performance penalty typically associated with the implementation of these guarantees. There are several ways to do this. One is to design faster protocols. This has been explored in [PGS98] where 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 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

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

delivery and the definitive delivery can be used to process the message. The caveat of such an optimistic approach is that it can result in a high number of rollbacks.

In the rest of the paper we describe two scenarios where optimistic delivery of reliable multicast messages has been successfully used to implement high performance information systems. The first scenario involves an eager data replication protocol based on totally-ordered multicast. In this protocol, the calculation of the total order is overlapped with the optimistic execution of transactions. The second scenario is a replicated atomic commitment server that provides low latency non-blocking atomic commitment. The protocol implements the non-blocking property by means of uniform multicast and uses optimism to hide its cost. In this way, valuable locked resources can be released earlier thereby decreasing the overall latency.

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 in the target group delivers a message, then every correct process in the target group will also deliver the same message. 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. Uniformity, for instance, guarantees that if a process, correct or not, delivers a message, then every correct process will also deliver the same message. This is useful in cases involving persistent actions like writing to disk 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 new processes, that is, the set of processes perceived as correct. Virtual synchrony 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 avoid losing messages when a process fails or a new process joins the group. 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 guarantees associated to the primitive used hold, the message is definitively delivered (def-delivered). What to do with messages that are opt-delivered but not yet def-delivered is up to the application. One option is to optimistically process messages right away. Another is to 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, often mutually exclusive purposes. On the one hand, it has been used to provide high availability, by masking the failure of one or more replicas. This high availability has always been provided at the cost of scalability, with the whole system performing worse than a equivalent system with no reliability. On the other hand, data replication has also been used to increase throughput. Typically, the increase in throughput is achieved at the cost of availability.

Postgres-R [KA00] is the first system that succeeds in combining replication to provide both full 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. Data are partitioned into non-intersecting subsets called conflict classes. Transactions access one of these classes and it is known in advance which one it 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. This can be done when 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. As a result, transactions do not pay for the total ordering. Additionally, this protocol applies a reordering technique that allows, in some cases, to disregard the total order without compromising consistency, thereby reducing the number of aborts due to misordering of optimistic deliveries.

4 Atomic Commitment

A great deal of work has been invested in non-blocking atomic commitment (NBC) protocols, 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 use optimism to reduce the average latency of NBC. This approach has been followed 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.

Optimistic decisions might be wrong, in which case an optimistically committed transaction might need to be aborted. When that happens, all those transactions that acquired the optimistically released locks will also be aborted. To prevent the possibility of cascading aborts, transactions that acquire optimistically released locks are not allowed to commit until the transaction that released the locks finally commits. The odds of aborts, however, are very 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.

This situation happens very infrequently. Thus, the protocol delays conflicting transactions only for the duration of two rounds, exactly the same delay as in 2PC but with the advantage of non-blocking behavior in case of failures.

5 Conclusions

Reliable multicast is a powerful building block for fault-tolerant 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 messages. In this paper, this approach has been shown in the context of protocols for 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.