# Reducing the Latency of Non-Blocking Commitment using Optimism and Replication*

R. Jiménez-Peris†, M. Patiño-Martínez†
School of Computer Science
Technical University of Madrid (UPM)
Madrid, Spain
{rjimenez, mpatino}@fi.upm.es

G. Alonso
Department of Computer Science
Swiss Federal Institute of Technology (ETHZ),
Zürich, Switzerland
alonso@inf.ethz.ch

S. Arévalo
Escuela de Ciencias Experimentales
Universidad Rey Juan Carlos, Madrid, Spain
s.arevalo@escet.urjc.es

## 1. The Latency of Non-Blocking Protocols

Atomic commitment protocols are used to ensure the atomicity of atomic transactions. The best known and most widely used atomic commitment protocol is *two phase commit* (2PC). The main idea behind 2PC is to perform two rounds of voting under the guidance of a site acting as *coordinator*. The coordinator asks first for votes on whether to commit (*yes* vote) or abort (*no* vote) the transaction. The *participants* send their vote and the coordinator decides to commit if all participants voted yes. Otherwise the decision is to abort. The main drawback of 2PC is that it might block. This happens when the coordinator fails after having received yes votes from all participants but before sending the commit message. In that situation, the participants cannot make a decision among themselves because they do not know what the coordinator decided before it failed. By deciding to commit or abort, they could be doing exactly the opposite of what the coordinator did, thereby violating the atomicity of the transaction.

Non-blocking commit protocols, e.g., *3 phase commit* (3PC) have been proposed to overcome this drawback. However, and in spite of the work in the area, the standard atomic commitment protocol is still 2PC. The main reason is that most transactional systems pay as much attention to performance as they do to consistency. For instance, in most systems, if a transaction has not committed after a given period of time, it is summarily aborted regardless of whether the coordinator is up or down. There are good practical reasons for doing this. Atomic commitment is an expensive procedure that adds to the latency of transactions. It does not only increase the response time of individual transactions. Since transactions waiting to commit are locking valuable resources, it also results in a lower throughput if transactions take too long to commit. Existing non-blocking commit protocols solve the problem of consistency in case of failures but typically introduce unacceptable delays by requiring more message rounds.

## 2. Reducing the Latency

To obtain a consistent non-blocking behavior (with the absence of partitions), it is enough for the 2PC coordinator to use a virtual-synchronous uniform multicast message to propagate the outcome of the transaction. This guarantees that either all or none of the participants know about the fate of the transaction and the coordinator status. However, uniformity is very expensive in terms of the delay it introduces.

In contrast to previous work in which the latency of the commitment is increased due to the use of uniformity or an additional round of messages, in this proposal to minimize this delay, we resort to a novel technique based on optimistic delivery that overlaps the commit processing with the uniform delivery of the multicast. The idea is to hide the latency of the multicast behind operations that need to be performed anyway. This is accomplished by processing messages in an optimistic manner and hoping that most decisions will be correct although in some cases transactions might need to be aborted. This idea builds upon recent work in optimistic multicast [4], which was proposed to reduce the latency of total ordered multicast. In here, we follow



† This work has been partially supported by the Spanish National Research Council CICYT, grant #TIC98-1032

the more aggressive version of optimistic delivery proposed in the context of Postgres-R [2] and later used to provide high performance eager replication in clusters [3].

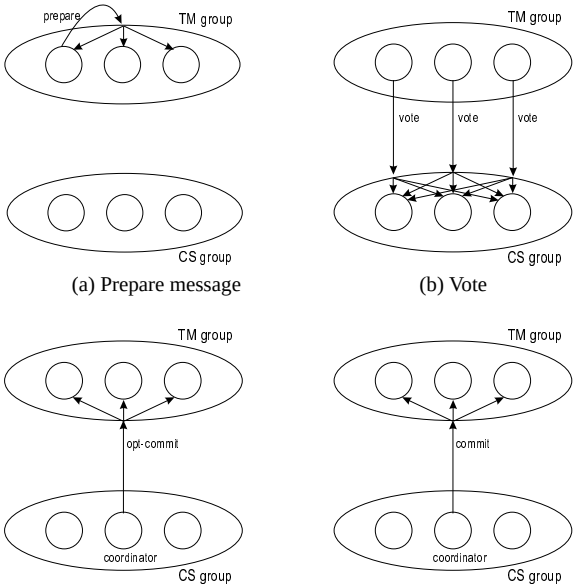
In addition, in any atomic commitment protocol participants are required to flush to disk a log entry before sending their vote. This log entry contains all the information needed by a participant to recall its own actions in the event of a crash. The coordinator is also required to flush the outcome of the protocol before communicating the decision to the participants. Flushing log records adds to the overall latency as messages cannot be sent or responded to before writing to the log. In the protocol we propose, this delay is reduced by allowing sites to send messages instead of flushing log records. The idea is to use the main memory of a replicated group as stable memory instead of a mirrored log with careful writes.

Finally, to minimize the waiting time of transactions, locks are released optimistically. The idea is that a transaction can be optimistically committed pending the confirmation provided by the uniform multicast. By optimistically committing the transaction, other transactions can proceed although they risk a rollback if the transaction that was optimistically committed ended up aborting. In our protocol, the optimistic commit is performed in such a way that *cascading aborts* are confined to a single level. In addition, transactions are only committed optimistically when all their participants have voted affirmatively, thereby greatly reducing the risk of having to abort the transaction. This contrasts with other optimistic commit protocols, e.g., [1], where transactions that must abort (because one or more participants voted no) can be optimistically committed.

## 3. Protocol Overview

The system is organized in two groups of processes (see Fig. 1), a replicated group providing the commit service that acts as coordinator (the CS group), and the group of the participating transaction managers (the TM group).

The protocol is triggered by the transaction manager of the client transaction that multicasts a *prepare* message to the TM group. Participants react to the prepare message by sending a vote message to the CS group. A participant does not wait to flush its log, instead it uniformly multicasts its vote together with its log entry. When the message is received by the coordinator, the message is *optimistically delivered* right away without waiting for the stabilization of the message (e.g., waiting for the message to be received by all the members of the group). A member from the CS group will coordinate the transaction. It will be chosen among the available ones based on the transaction identifier (tid). If the message is a no vote, the transaction is aborted and the decision is multicast to the TM group. If the message corresponds to the last vote, and all were yes votes, the



(a) Prepare message (b) Vote

(c) Optimistic decision (d) Outcome

**Figure 1. Protocol steps**

transaction is optimistically committed, and the fact is communicated to the TM group. The optimistic commit changes the locks held by the transaction to opt-mode, which is compatible with any other lock. Opt-locks do not allow to the holding transaction to commit until the transaction that released them, definitively commits. In this way, conflicting transactions do not pay for the cost of uniformity. When the last yes vote is uniformly delivered, the coordinator definitively commits the transaction multicasting the decision to the TM group. Additionally, the protocol preserves the consistency in the advent of partitions and limits the duration of the commit protocol to prevent unbounded resource contention.

With these properties the protocol we propose satisfactorily addresses all design concerns related to non-blocking atomic commitment and can thus become an important contribution to future distributed applications.

## References


[1] R. Gupta, J. Haritsa, and K. Ramamritham. Revisiting Commit Processing in Distributed Database Systems. In *Proc. of the ACM SIGMOD Conference*, 1997.

[2] 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.

[3] 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, Oct. 2000.

[4] F. Pedone and A. Schiper. Optimistic Atomic Broadcast. In *Proc. of 12th Distributed Computing Conference*, volume LNCS 1499, pages 318–332. Springer, Sept. 1998.