From Spontaneous Total Order to Uniform Total Order: different degrees of optimistic delivery*

Luís Rodrigues
Universidade de Lisboa
ler@di.fc.ul.pt

José Mocito
Universidade de Lisboa
jmocito@lasige.di.fc.ul.pt

Nuno Carvalho
Universidade de Lisboa
nunomrc@di.fc.ul.pt

Abstract

A total order protocol is a fundamental building block in the construction of distributed fault-tolerant applications. Unfortunately, the implementation of such a primitive can be expensive both in terms of communication steps and of number of messages exchanged. This problem is exacerbated in large-scale systems, where the performance of the algorithm may be limited by the presence of high-latency links.

Optimistic total order protocols have been proposed to alleviate this problem. However, different optimistic protocols offer quite distinct services. This paper makes an overview of different optimistic approaches and shows how they can be combined in a single adaptive protocol.

1 Introduction

A Total Order Broadcast protocol is a fundamental building block in the construction of distributed fault-tolerant applications [3]. The purpose of such a protocol is to provide a communication primitive that allows processes to agree on the set of messages they deliver and also on their delivery order. Uniform Total Order Broadcast is particularly useful to implement fault-tolerant services by using software-based replication [5]. A particular case, relevant for this work, is the application of total order protocols to implement database replication services [7].

Unfortunately, the implementation of such a primitive can be expensive both in terms of communication steps and of number of messages exchanged. This problem is exacerbated in large-scale systems, where the performance of the algorithm may be limited by the presence of high-latency links. To alleviate this performance problem, Optimistic Total Order protocols have been proposed. Such protocols provide to the application an early indication of the estimated uniform total order. The application can use this estimate to perform a number of actions optimistically, which are later committed when the final definitive order is established. The goal is to execute some application steps in parallel with the communication steps of the total order algorithm.

Different Optimistic Total Order protocols can be found in the literature [11, 4, 12]. Although most of these protocols share the same designation, they offer quite different services.

 This work has been partially supported by the project IST-STREP 004758, GORDA: Open Replication of Databases.

RTO1 - Total order: Let m_1 and m_2 be two messages that are *RTO-broadcast*. Let p_i and p_j be any two correct processes that *RTO-deliver*(m_1) and *RTO-deliver*(m_2). If p_i *RTO-delivers*(m_1) before *RTO-delivers*(m_2), then p_j *RTO-delivers*(m_1) before *RTO-delivers*(m_2), and we note $m_1 < m_2$.

RTO2 - Agreement: If a correct process in Ω has *RTO-delivered*(m), then every correct process in Ω eventually *RTO-delivers*(m).

RTO3 - Termination: If a correct process *RTO-broadcasts*(m), then every correct process in Ω eventually *RTO-delivers*(m).

RTO4 - Integrity: For any message m, every correct process delivers m at most once, and only if m was previously broadcast by some process $p \in \Omega$.

Table 1: Regular total order properties

In this paper, we clarify the different possible definitions of optimism in the context of total order protocols. We show that it is possible to establish a classification of *optimism*, that can rank existing protocols from very optimistic to conservative. Furthermore, we show that the different grades of optimism are useful in different operating conditions and that they can be combined in a single adaptive protocol.

The rest of the paper is structured as follows. Section 2, clarifies the differences among the different optimistic approaches previously proposed. Section 3 shows how different protocols can be combined in a single adaptive protocol. Early experimental results from the combined protocol are provided in Section 4. Section 5 concludes the paper.

2 Ranking Optimistic Approaches

At this point it is important to distinguish two alternative definitions of total order: *regular* total order and *uniform* total order.

Regular total order broadcast is defined on a set of processes Ω by the primitives (1) *RTO-broadcast*(m) which issues message m to Ω, and (2) *RTO-deliver(m)* which is the corresponding delivery of m. When a process p_i executes *RTO-broadcast*(m) (resp *RTO-deliver(m)*), we say that p_i *"RTO-broadcasts m"* (resp *"RTO-delivers m"*). Regular total order is characterized by the properties depicted in Table 1. Informally, a regular total order protocol ensures that two correct processes (i.e., processes that never crash) deliver exactly the same set of messages in the same order.

The uniform version can be obtained by replacing properties *RTO1* and *RTO2* by properties *UTO1* and *UTO2* presented in Table 2. Uniform total order is stronger as it ensures that, if a processes p_i delivers two messages in a given order, all processes will deliver the same messages in that order, *even* if p_i fails. Uniform total order is the desired consistency criteria in applications such as database replication, given that certain messages may cause a transaction to be aborted or committed. If the delivery of a message to a process causes this process to commit a transaction before crashing, all other processes need also to deliver the same message to ensure a consistent outcome of the transaction.

An *optimistic* total order protocol includes an additional primitive *UTO-opt-deliver(m)*. When a process p_i executes *UTO-opt-deliver(m)*, we say that p_i *"UTO-opt-delivers m"*. The

UTO1 - Uniform Total order: Let m_1 and m_2 be two messages that are *UTO-broadcast*. Let p_i and p_j be any two processes that *UTO-deliver*(m_1) and *UTO-deliver*(m_2). If p_i *UTO-delivers*(m_1) before *UTO-delivers*(m_2), then p_j *UTO-delivers*(m_1) before *UTO-delivers*(m_2), and we note $m_1 < m_2$.

UTO2 - Uniform Agreement: If a process in Ω (correct or not) has *UTO-delivered*(m), then every correct process in Ω eventually *UTO-delivers*(m).

Table 2: Uniform total order properties

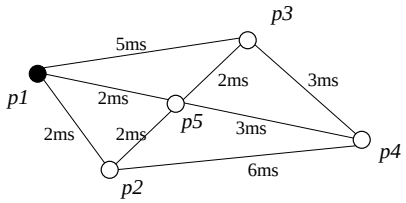

Figure 1: Network with 5 nodes.

order by which a process p *UTO-opt-delivers* messages is an estimate of the order by which p will *UTO-deliver* the same messages. Note that in some cases the estimate may be wrong, *i.e.*, the order by which messages are *UTO-opt-delivered* may differ from the order by which they are *UTO-delivered* (although in stable periods it is desirable that it is the same). Note also that it is possible that a message is directly *UTO-delivered* without ever being *UTO-opt-delivered*.

The purpose of the optimistic delivery is to allow the application to execute some steps in parallel with the communication steps of the total order protocol. These steps can later be committed or aborted when the final definitive order is established. Naturally, the earlier the optimistic delivery can be provided, the more steps can be executed in parallel. However, the speedup gains obtained from this parallelism, can be compromised by the need to abort steps, when the estimate proves to be inaccurate.

To illustrate the trade-offs involved in an optimistic total order protocol, we will use the most intuitive algorithms to establish total order: the sequencer based algorithm. Note that a similar discussion could be made using other total order algorithms, but the simplicity of the sequencer approach makes the text more clear. In a sequencer based algorithm, one of the processes in the system, designated the sequencer, has the onus of assigning a sequence number to every message it receives. All processes, including the sequencer, deliver messages according to these sequence numbers.

To better describe the steps involved in an uniform total order protocol, we will use the network illustrated in Figure 1. The figure shows a network with five nodes $p_1 - p_5$ connected by point-to-point links. The average delay of each link is also depicted (for instance, the average delay in the link $p_1 - p_3$ is $5ms$).

Let us now consider a particular run using the network above. This run is depicted in Figure 2. At time $t_0 = 0$ process p_2 sends a message m_2 and process p_3 sends a message m_3. Assume that process p_4 receives message m_3 at time $t_3 = 3$ and message m_2 at time $t_6 = 6$. We name the order by which messages are received at each process from the underlying

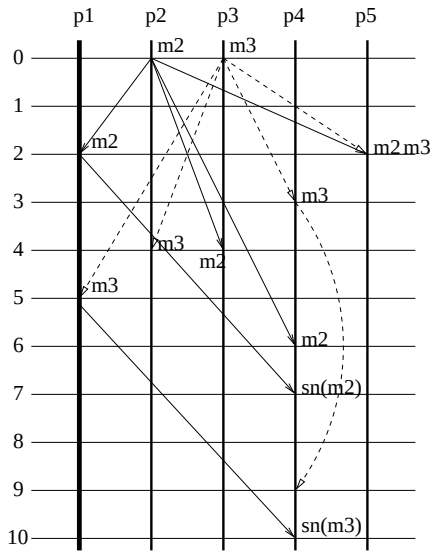

Figure 2: A run.

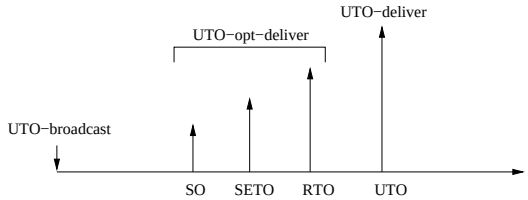

Figure 3: Optimism rank.

transport protocols the *spontaneous order* (SO).

In the same example, assume that m_2 is received by process p_1, the sequencer, at time $t_2 = 2$ and m_3 at time $t_5 = 5$. Assume that the sequencer assigns sequence number to messages in the order it receives them. Clearly, in this example, the spontaneous order observed at p_4 would be different than the final order as assigned by the sequencer. Sequence numbers assigned by the sequencer will be received at process p_4 at times $t_7 = 7$ and $t_{10} = 10$. In this case, as soon as the sequence number is received we have assured *regular total order* (RTO). Note that if both p_1 and p_4 fail, it is still possible that the remaining processes assign a difference order to messages m_2 and m_3.

The final *uniform total order* (UTO) can only be guaranteed when p_4 is sure that the sequencer numbers have been received by all the remaining processes (or, at least, a majority). In our example this would happen at time $t_{13} = 13$.

Note that, if p_4 can estimate that the delay between p_1 and p_3 is $3ms$ higher than the delay between p_1 and p_2, it could attempt to reproduce the order by which the sequencer receives messages m_2 and m_3 by artificially delaying the delivery of m_3 by a delta of $6ms$ (i.e., by delivering m_3 at time $t_9 = 9$). A clever scheme inspired in this insight has been proposed to establish a *statistically estimated total order* (SETO) before the regular total order is known [12].

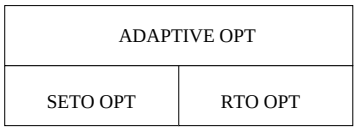

Figure 4: Adaptive protocol.

Figure 3 shows a time line of total order delivery. Naturally, spontaneous order occurs first in the time line and uniform total order occurs last. The question now is to decide which of the intermediate orders should be used to support optimistic delivery.

Clearly, spontaneous total order is only an accurate indication of the final delivery if all nodes use the same local area network segment, as assumed in [11]. On the other hand, the statistically estimated total order is a good choice in stable networks, where the network delays have small variance and can be accurately estimated. In unstable networks, regular total order is the best choice for optimistic delivery, as it only provides inaccurate information in the rare cases where a node crashes.

3 An Adaptive Protocol

Given that SETO is the best choice to provide optimistic delivery in stable networks while RTO is the best choice in unstable networks, it is interesting to design an adaptive protocol that can dynamically commute from one scheme to the other as a function of the measured network stability.

It is interesting to observe that such protocol can be obtained as a modular composition of two different total order protocols, as depicted in Figure 4. In steady-state, the adaptive protocol would simply receive TO-broadcast/ TO-deliver requests/indications and forward them to the most appropriate protocol.

Obviously, the key aspect of the adaptive protocol is the algorithm used to commute from one underlying total order protocol to another. To our knowledge, there is little work in the literature on how to efficiently perform this sort of transition. Previous work [8] requires messages to be buffered during the reconfiguration. Here we propose a generic transition protocol that does not require the traffic to be stopped, allowing a smooth adaptation to changes in the underlying network. Let us assume that the adaptive protocol is using protocol TO-A to order messages and pretends to commute to protocol TO-B.

The transition protocol operates as follows. A control message is sent to all nodes to initiate the reconfiguration. When a node receives this control message, the adaptive protocol starts broadcasting all messages using both TO protocols. If a node has no message to be sent, it should send a special null message to ensure a faster termination of the reconfiguration process. When a node starts receiving messages from both TO protocols it performs the following steps: messages received from TO-A are delivered as normally; messages received from TO-B are buffered in order.

As soon as a message from each and every node is received from both TO protocols the reconfiguration is concluded using the following "sanity" procedure. Firstly, from the buffer of messages received from TO-B are removed all messages that have already been delivered by TO-A. Secondly, all messages remaining in the buffer are delivered in order. Finally, from this point on, all messages received from TO-A are simply discarded and messages

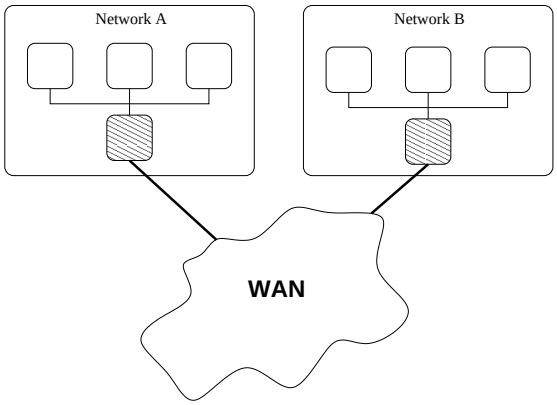

Figure 5: Network used in the simulation.

received from TO-B are delivered instead. No further message is sent using TO-A (until a new reconfiguration is needed).

Clearly, the above protocol is not fault-tolerant as, in order to terminate the reconfiguration, a message is awaited from each and every node. Different strategies can be used to make the protocol fault-tolerant, depending on the fault-model. To illustrate one of these strategies, probably the simplest, we assume that both total order protocols are implemented on top of a view-synchronous [2] group communication service, a typical implementation in systems such as [13, 6, 9]. In such case, when a failure occurs, all correct processes receive a new view, with updated group membership information. Furthermore, the view-synchronous communication ensures that all processes have delivered exactly the same set of messages prior to the view delivery. Therefore, if a crash occurs during the reconfiguration, each process can perform the "sanity" procedure described in the previous paragraph as soon as the first view is delivered.

In addition to the reconfiguration protocol, another important aspect of the operation of the adaptive protocol is the definition of the threshold that triggers the reconfiguration. This threshold is application dependent, and should be set by the application.

4 Experimental results

We have performed a number of experiments to validate the comparative behavior of *Statistically Estimated Total Order* (SETO) and *Regular Total Order* (RTO) protocols. The experiments were made using the SSFNet network simulator [10, 1]. The network topology used consists on a wide area network with two clouds, connected by one link. Each cloud contains one router that is used to connect the clouds. The other nodes form a group membership. Figure 5 shows the network used in the experiments.

The average latency in the long-haul connecting the two networks is $20ms$; this is the major source of latency in this experimental setting. In order to simulate instability in the network the standard deviation of the transmission delays of the long-haul link is made variable between 0% to 10%.

Every node in the simulation receives messages from the sending group members. Senders transmit messages at a variable, uniformly distributed rate. In Network A we have two nodes that actively send messages while in Network B there is only one sender. The sequencer is

Type of Delivery	Error Rate
Spontaneous	66%
SETO	13%

Table 3: Error Rate in Stable Network

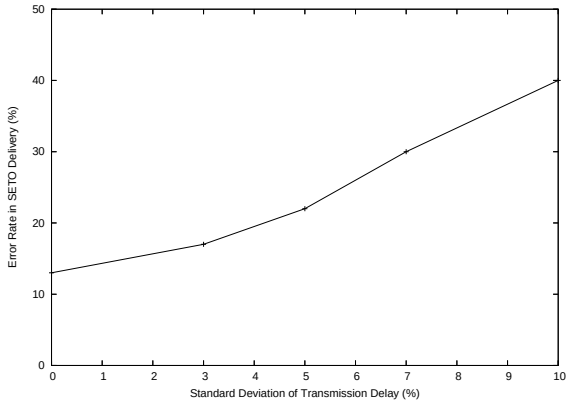

Figure 6: Error Rate in SETO Delivery

located in Network A. All values depicted in the figures and tables below were measured at a node located in Network B.

This configuration was chosen to illustrate the behavior of SETO protocol in particular conditions. As said before, spontaneous order is only accurate when all nodes execute on the same local area network segment. By having sender nodes on both networks the spontaneous delivery will become an inaccurate estimation of the final delivery. This scenario thus make the case for other types of optimistic deliveries. The second aspect that we want to illustrate is the impact of the variation in the transmission delays in the performance of the SETO protocol. By having two senders in Network A we will be able to observe that some messages sent by these nodes will exchange their delivery order when traversing the long-haul link in their way to Network B producing, once more, an inaccurate estimation of the final delivery at B.

As described in Section 2, optimistic delivery is only useful if highly accurate, i.e., if the application is not required to rollback the execution steps performed optimistically very often. Table 3 clearly shows that in heterogeneous networks (i.e., in networks were nodes have different distances to the sequencer) the spontaneous order in an unacceptable source for optimistic delivery, as the error rate is extremely high even in a stable network where the standard deviation is 0%. On the other hand, the SETO approach, in a stable network, can provide a significantly smaller error rate and provides an interesting source of optimistic delivery.

In Figure 6, we show the impact of network instability on the accuracy of SETO. As can be seen, an increase in the standard deviation of the transmission delay in the long haul link makes the error rate of the SETO protocol increase, reaching 40% for a $\sigma = 10\%$. These results confirm the results published in [12] and clearly show that SETO accuracy is highly dependent on the network stability.

Type of Delivery	Time of Delivery
Spontaneous	13429 μs
SETO	20891 μs
Regular	40835 μs
Uniform	42528 μs

Table 4: Type of TO Delivery *vs* Time of Delivery

Table 4 shows the time line of deliveries in our experiment. The values are averages of all messages received by our target node in Network B. Naturally, spontaneous order provides the smaller latency, given that messages are delivered as soon as they are received from the network. The value depicted in the table can be explained as follows. We recall that the measurements are made in a received in Network B. There are two senders in Network A and their messages suffer and average delay of $20ms$. The messages from sender located in Network B suffer a negligible delay. Since all senders transmit at approximately the same average rate, the average delay becomes approximately $13ms$. A similar reasoning explains the figures depicted in the remaining rows. The interesting aspect is that SETO offers significantly less latency than regular delivery. Therefore, in stable networks SETO is the source of choice for providing optimistic delivery. Also, regular delivery is still faster than the final uniform delivery and provides room for optimistic execution of application steps.

5 Conclusions

SETO protocol provides a fast and accurate estimation of the final total order in stable networks. Unfortunately, in unstable networks, SETO delivery is highly inaccurate, producing a very high rate of rollbacks. In such networks a RTO delivery is more suited, because of its robustness regarding variability in the transmission delays.

An adaptive protocol was proposed to capture the requirements of today's wide area networks, where the stability of the transmission links may vary very frequently. This protocol commutes from a SETO to a RTO as a source for optimistic delivery. In periods of stable network operation, the protocol commutes back again to a SETO delivery, thus allowing early message processing. Early experimental results were obtained to validate the approach.

Future work will address the use of our novel adaptive protocol in a concrete wide-area database replication system and to seek for the optimal commuting threshold for that application (as we have mentioned, this parameter is application dependent). Also, we plan to further improve the reconfiguration algorithm and study the potential advantages of providing multiple intermediate delivery indications to the application.

6 Acknowledgments

The authors are grateful to J. Pereira and to the anonymous reviewers for their comments on earlier versions of this paper.

References

[1] The SSFNet project: http://www.ssfnet.org.

[2] K. Birman and T. Joseph. Exploiting virtual synchrony in distributed systems. Technical Report 87-811, Department of Computer Science, Cornell University, Ithaca, New York, February 1987.

[3] D. Powell (Guest Ed.). Special issue on group communication. *Communications of the ACM*, 39(4):50–97, 1996.

[4] P. Felber and A. Schiper. Optimistic active replication. In *Proceedings of 21st International Conference on Distributed Computing Systems (ICDCS'2001)*, Phoenix, Arizona, USA, April 2001. IEEE Computer Society.

[5] R. Guerraoui and A. Schiper. Software-based replication for fault tolerance. *IEEE Computer*, 30(4):68–74, 1997.

[6] M. Hayden. *The Ensemble System*. PhD thesis, Cornell University, Computer Science Department, 1998.

[7] Y. Lin, B. Kemme, M. Patiño-Martnez, and R. Jiménez-Peris. Consistent data replication: Is it feasible in wans? In *Proceedings of Europar Conf. (EUROPAR)*, 2005.

[8] X. Liu and R. van Renesse. Fast protocol transition in a distributed environment. In *Proceedings of the 19th ACM Conference on Principles of Distributed Computing (PODC 2000)*, page 341, Portland, OR, July 2000.

[9] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 707–710, Phoenix, Arizona, April 2001. IEEE.

[10] D. Nicol, J. Liu, M. Liljenstam, and G. Yan. Simulation of large-scale networks using ssf. In *Proceedings of the 2003 Winter Simulation Conference*, 2003.

[11] F. Pedone and A. Schiper. Optimistic atomic broadcast. In *Proceedings of the 12th International Symposium on Distributed Computing (DISC'98)*, 1998.

[12] A. Sousa, J. Pereira, F. Moura, and R. Oliveira. Optimistic total order in wide area networks. In *Proc. 21st IEEE Symposium on Reliable Distributed Systems*, pages 190–199. IEEE CS, October 2002.

[13] R. van Renesse, K. Birman, R. Cooper, B. Glade, and P. Stephenson. Reliable Multicast between Microkernels. In *Proceedings of the USENIX workshop on Micro-Kernels and Other Kernel Architectures*, pages 27–28, April 1992.