Fault-Tolerant Replication of High Throughput Services

José Orlando Pereira

Dep. de Informática
Universidade do Minho
4710-057 Braga Portugal
jop@di.uminho.pt

Abstract

In multicast communication systems, a single perturbed recipient can drastically affect the performance of a complete group of processes. This problem can be alleviated by allowing some messages to be omitted. We propose a multicast service that exploits semantic knowledge to select which messages can be omitted without compromising the application's correctness.

Besides summarizing initial research results [8], this text addresses the implementation of high throughput fault-tolerant services by combining virtually synchronous and totally ordered reliable multicast with semantic reliability.

1 Introduction

It has recently been pointed out that performance of reliable multicast protocols in demanding applications requiring sustained high throughput to a large number of participants can be frustrating [1].

Some protocols are inherently unable to achieve high throughput due to limitations of the retransmission mechanisms. This has been addressed by the design of more scalable protocols that implement efficient mechanisms to disseminate messages and collect stability information [5].

Furthermore, it has been shown that despite the efficiency of the protocol any single slow receiver can, due to flow control, degrade the performance of the whole system [1]. This second problem is more difficult to tackle since no protocol can force a node to execute faster than its own resources allow.

This problem is closely related to reliability itself: When messages are produced faster than some target is able to consume them, the surplus needs to be temporarily buffered for eventual delivery. However, buffering is viable only for short bursts of traffic. If message multicast rate is consistently high, unbounded buffer space would be required and message delivery by the slowest process would increasingly lag behind. Eventually, message sources must be slowed down or, alternatively, the slow recipient needs to be excluded from the multicast group. In short, either performance or fault tolerance is degraded.

This can be circumvented only by relaxing the reliability of multicast by not delivering all messages to processes that are significantly slower than the majority of group members. As summarized in Section 2, our research has focused in relaxing reliability such that performance advantages are obtained without impacting application correctness.

Currently, we are researching how our results can be combined with stronger protocols, such as total order and virtual synchrony, and used in applications which are highly dependent on multicast performance, such as in replication for fault-tolerance using both the replicated state machine and the primary-backup approaches [4]. Sections 3 and 4 summarize the intuition behind our current research in this area.

2 Message obsolescence

Relaxing the reliability of fault-tolerant multicast protocols makes it possible to accommodate perturbed members in a multicast group without impairing global performance [2].

Unfortunately, even when the semantics of the application tolerates message loss, most of the simplicity that was gained at the application level by using a reliable multicast protocol is also lost. For instance, even if some mechanism is implemented to notify receivers when a message is lost, the application might be unable to take any corrective action since it has no knowledge of that message's content, and thus, whether it is important or not.

In contrast, our proposal aims at a protocol ensuring that no important messages are ever lost. As such we aim at semantic reliability, ensuring that all current information is delivered to all receivers, either implicitly or explicitly, without necessarily delivering all messages.

The basic idea behind our approach is is that in a distributed application some messages either overwrite or implicitly convey the content of other messages sent in the past, therefore making them irrelevant. If obsolete messages have not been yet delivered, they can be safely purged without compromising the application's correctness. If a slow receiver exists but enough messages can be purged, the protocol will not need to slow down the sender, thus ensuring that the performance of fast processes remains unaffected.

Notice that, to be effective, the obsolescence property cannot be exploited solely at the application layer, since liveness or timing constraints force the application to immediately forward outgoing messages to the communication channel. Being so, messages become out of reach and cannot be discarded by the application even if immediately made obsolete.

As such sufficient semantical information must be conveyed from the application to the protocol to determine which messages can be purged. The required semantical information is formalized as a relation on messages. For each pair of related messages $m \sqsubseteq m'$, we say that m is *obsoleted* by m'. This relation is defined by each application and we assume that it is a partial order and is coherent with causal order of events, as such $m \sqsubseteq m$ for all m. The intuitive meaning of this relation is that if $m \sqsubseteq m'$ and if m' is delivered, the correctness of the application is not affected by omitting the delivery of m.

If no message is ever made obsolete, semantical reliability defaults to full reliability [6]. On the other hand, if every message is made obsolete by subsequent messages from the same sender, semantically reliable multicast results in the extension to multicast of the 1-stubborn channel [7]. In between, semantically reliable multicast simplifies the development of application while allowing enough messages to be purged when necessary to sustain high throughput. For instance, applications embodying operations with overwrite semantics, in particular, applications managing read-write items are the most obvious example of applications that exhibit message obsolescence, as any update of a given item is made obsolete by subsequent write operations.

Considering the obsolescence profile of the traffic generated by an on-line stock trading system, our initial work has shown that as much as a 40% processing delay can be tolerated without perturbing the message sender [8].

3 Semantic virtual synchrony

In order to apply the performance benefits of semantic reliability to replicated fault-tolerant services, additional guarantees such as total order and virtual synchrony are required. The safety properties of a virtually synchronous membership protocol can be classified as properties on the views or as relating views with common messages [9].

The agreement on the views is not affected by the relaxed reliability model: All it takes is that all processes agree on the elements of each view. In fact, this can be achieved using a consensus protocol on top of 1-stubborn channels [7], which is a particular case of semantic reliability.

Where the definition of virtual synchrony implicitly assumes a strict reliability model, and thus collides with semantic reliability, is in requiring agreement on messages delivered in each view. Specifically:[1]

Virtual Synchrony: If processes p and p' install the same view V in the same previous view V', then any message delivered by p in V' is also delivered by p' in V'.

In order to be compatible with semantic reliability, virtual synchrony cannot be defined as requiring that upon view change all processes have delivered the exact same set of messages, as messages delivered by one process might have become obsolete and been purged, and thus not available.

What can be ensured is that all processes have delivered the same information, either explicitly or implicitly without necessarily delivering all messages:

Semantic Virtual Synchrony: If processes p and p' install the same view V in the same previous view V', then for any message m delivered by p in V', there is m' such that $m \sqsubseteq m'$ that is delivered by p' in V'.

This definition maintains the meaning of a view change as a synchronization point in the message stream, where the state of all processes in the installed view is consistent. Notice that when the obsolescence

[1] The system model and notation assumed is the same as described in [9].

relation is empty, it defaults to standard virtual synchrony, emphasizing the nature of the property as a generalization of virtual synchrony.

In addition to agreement on the group composition, this property is sufficient for primary-backup replication, as it ensures that even though different replicas have received different messages, they all receive the latest updates and only miss those that have been rendered obsolete, for instance, by being overwritten.

4 Total order

A simple total ordering algorithm, sufficient for the implementation of the active replication approach, can be constructed on top of semantical virtual synchrony. This algorithm presented here is not concerned with efficiency: Its aim is to give an intuition of possibility and of what is the resulting service.

In fact, our proposal is similar to the algorithm in [3] in requiring the execution of an agreement protocol for each batch of messages. Specifically, we use a view change operation as the agreement protocol.

The algorithm works as follows: Processes freely multicast messages using virtually synchronous semantically reliable multicast. Upon reception by the ordering algorithm on top of virtual synchrony, messages are buffered.

Periodically, a view change is forced and a new view is installed. Upon reception of the new view, the total ordering algorithm purges all obsolete messages from its buffer. The remaining messages are then guaranteed to be the same, thus, can be ordered locally using any deterministic criterium and delivered. We are currently developing an improved totally ordering algorithm for semantically reliable multicast that should allow better performance.

5 Conclusions and future work

Our previous work has illustrated the advantages of using the notion of message obsolescence in the design of protocols for high throughput applications. As such we proposed semantic reliability as a viable approach to ensure a higher stable multicast throughput in the presence of perturbed group members. Namely, by applying a semantically reliable multicast to the obsolescence profile of an on-line stock trading system, as much as 40% processing delay can be tolerated without perturbing fast members of the group.

In contrast to solutions that admit message loss and offload responsibility of corrective action to applications, our proposal also has the advantage of providing a solid foundation for building stronger protocols such as total order and virtual synchrony, extending the performance advantages of semantic reliability to replicated fault-tolerant services.

Our current research is focused on formally specifying, applying and improving the the performance of protocols based upon semantic reliability as outlined in this paper.

References

[1] K. Birman. A review of experiences with reliable multicast. *Software Practice and Experience*, 29(9):741–774, July 1999.

[2] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[3] T. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[4] R. Guerraoui and A. Schiper. Fault-tolerance by replication in distributed systems. *Lecture Notes in Computer Science*, 1088, 1996.

[5] K. Guo. *Scalable Message Stability Detection Protocols*. PhD thesis, Cornell University, Computer Science, May 1998.

[6] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. In Sape Mullender, editor, *Distributed Systems*, chapter 5, pages 97–145. Addison Wesley, 1993.

[7] R. Oliveira. *Solving consensus: From fair-lossy channels to crash-recovery of processes*. PhD thesis, École Polytechnique Fédérale de Lausanne, February 2000.

[8] J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast protocols. Submitted to the Nineteenth IEEE Symposium on Reliable Distributed Systems, October 2000.

[9] R. Vitenberg, I. Keidar, G. Chockler, and D. Dolev. Group communication specifications: A comprehensive study. Technical Report MIT-LCS-TR-790, MIT, Laboratory for Computer Science, 1999.