Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

EFFICIENT SIMULATION OF VIEW SYNCHRONY

Frej Drejhammar

Swedish Institute of Computer Science
Box 1263
SE-164 29 Kista, SWEDEN

Seif Haridi

Swedish Institute of Computer Science
Box 1263
SE-164 29 Kista, SWEDEN

ABSTRACT

View synchrony is a communications paradigm for building reliable distributed systems. Testing a protocol using view synchrony with a simulated implementation of view synchrony allows the tested protocol to be exposed to the full timing range allowed by the view synchrony model. This both reduces the complexity of the test environment and increases the confidence in the tested protocol. This paper outlines an algorithm for efficiently simulating view synchrony, including failure-atomic total-order multicast in a discrete-time event simulator.

1 INTRODUCTION

This paper outlines how view synchrony including failure-atomic total-order multicast in a partitionable network environment can be implemented efficiently in a discrete-time event simulator. View synchrony (Birman and Joseph 1987) is a communications paradigm for building reliable distributed systems (Birman 2006), it is provided by middleware such as JGroups (JGroups 2008), Spread (Amir and Stanton 1998) and FTM (Ventura Networks Inc. 2009). To the best of our knowledge, this is the first description of a view synchrony protocol tailored for a simulated environment.

Testing and debugging a complex application using view synchrony is hard and typically requires a network-level test environment. Using a custom view synchrony implementation, intended for use in a simulated environment, reduces the complexity of the test environment and can be made to exhibit the full timing range allowed by view synchrony instead of the subset occurring in a particular implementation and testing environment.

View synchrony can be implemented concisely by keeping track of view and partition states and using this information to schedule delivery of application messages and middleware events.

2 VIEW SYNCHRONY

View synchrony is a group communications abstraction in which message delivery is uniform[1] within a particular system configuration, called a *view*, and was first introduced by Birman in (Birman and Joseph 1987). The semantics of view synchrony as a group communications abstraction in partitionable systems has been formalized by Babaoglu et al. (Babaoglu, Davoli, and Montresor 2001). Friedman and van Renesse (Friedman and van Renesse 1995) extended the model to guarantee that messages are delivered in the same *view* as they were sent in or not at all. Practical implementation such as the Spread (Amir and Stanton 1998), FTM (Ventura Networks Inc. 2009) and JGroups (JGroups 2008) middleware extend the model with unicast messages, total order broadcasts and FIFO message delivery.

Intuitively the semantics of the view synchrony as a group communications abstraction can be summarized as follows: a *view* is a set of nodes. Communication occurs between nodes in a *view*. Message delivery within a *view* is uniform, meaning that all non-faulty nodes in the *view* will see the same set of messages,

[1] If a node delivers a message, all non-faulty nodes deliver the message.

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

in the same order. When a node fails or enters the system, message sending is halted and the *view* is said to be *blocked*, after which all messages currently "in flight" are delivered to the non-faulty nodes. When delivery is complete a new *view* is installed and message sending is resumed. For the full formal semantics, the reader is referred to Babaoglu et al. (Babaoglu, Davoli, and Montresor 2001) and Friedman and van Renesse (Friedman and van Renesse 1995).

3 VIEW SYNCHRONY SIMULATION

In a production-quality view synchrony middleware, FIFO ordering of point-to-point messages is handled implicitly by underlying network layers or by an explicit sequence numbering scheme. View synchronous total order broadcast is implemented by a consensus implementation agreeing on the delivery order of the messages. In a simulated environment, where the global system state is available, simpler mechanisms can be used to ensure message ordering guarantees. This section gives an outline of the simulator implementation, a complete description is available in a technical report (Drejhammar and Haridi 2012).

With the global knowledge available in a simulated environment it is no longer necessary to implement distributed consensus, instead we can schedule *view* generation and message delivery as soon as an event occurs in the discrete-time event simulator. When an event is scheduled it is placed randomly within a time window allowed by the semantics of view synchrony.

Our view synchrony simulator is split into two parts: one part that tracks message delivery within a *view*, and one part that coordinates *views*. To track messages within a *view* we use a simple network simulator. The main purpose of the network simulator is to track network connectivity and provide reliable message delivery and atomic broadcasts to correct nodes within a network partition.

When the network connectivity is changed, due to merging or partitioning, the availability of global state allows the simulator to schedule the generation of a new *view* at a point in time when all messages currently "in flight" have been delivered. Partitioning is handled by cloning the simulator state to create two new states in which the nodes belonging to the other partition are treated as if they have crashed. Merging is handled similarly by first blocking the *view* in the respective partition and, when all messages have been delivered, generating a new *view*.

4 BENEFITS OF VIEW SYNCHRONY SIMULATION

The main advantage of simulating view synchrony instead of implementing one of the several known algorithms on top of a network simulator is that it reduces the complexity of the testing environment. Additionally, simulated view synchrony allows an application/protocol to be exposed to all timing behaviours allowable under view synchrony and not just those exhibited by a particular middleware implementation.

REFERENCES

Amir, Y., and J. Stanton. 1998. "The Spread Wide Area Group Communication System". Technical Report CNDS-98-4, Johns Hopkins University.

Babaoglu, O., R. Davoli, and A. Montresor. 2001, April. "Group Communication in Partitionable Systems: Specification and Algorithms". *IEEE Transactions on Software Engineering* 27 (4): 308–336.

Birman, K. 2006. *Reliable Distributed Systems Technologies, Web Services, and Applications*. Springer.

Birman, K. P., and T. A. Joseph. 1987. "Reliable communication in the presence of failures". *ACM Trans. Comput. Syst.* 5 (1): 47–76.

Drejhammar, F., and S. Haridi. 2012. "Efficient Simulation of View Synchrony". Technical Report T2012:07, Swedish Institute of Computer Science.

Friedman, R., and R. van Renesse. 1995. "Strong and Weak Virtual Synchrony in Horus". Technical Report TR95-1537, Cornell University.

JGroups 2002-2008. "http://www.jgroups.org".

Ventura Networks Inc. 2009. "http://www.venturanetworksinc.com/".