J. Parallel Distrib. Comput. 77 (2015) 26–40

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Time hybrid total order broadcast: Exploiting the inherent synchrony of broadcast networks

Daniel Cason, Luiz E. Buzato *

Institute of Computing, University of Campinas, Av. Albert Einstein 1251, 13083-852, Campinas, São Paulo, Brazil

HIGHLIGHTS

- We give experimental evidence of the inherent synchrony exhibited by broadcast networks.
- We investigate the implications of synchrony to the design of total order broadcasts.
- We detail a novel synchronous total order protocol (THyTOB) and its implementation.
- We implement and assess a rounds protocol that serves as the substrate for THyTOB.
- THyTOB's performance is on a par with protocols designed for purely asynchronous systems.

ARTICLE INFO

Article history:
Received 12 March 2014
Received in revised form
21 July 2014
Accepted 29 October 2014
Available online 6 November 2014

Keywords:
Total order broadcast
Broadcast networks
Asynchronous systems
Synchronous systems
Cluster environments
Fault tolerance
Consensus

ABSTRACT

Total order broadcast is a fundamental communication primitive for the construction of highly-available systems. Informally, the primitive guarantees that messages sent by a group of processes are delivered to all processes in the same order. This paper investigates the design and performance of a very simple synchronous total order broadcast that is built atop of an asynchronous distributed system based on a broadcast network. Our Time Hybrid Total Order Broadcast (THyTOB) explores the inherent synchrony of the broadcast network to build a total order for the messages, while ensuring safety under asynchrony and in the presence of process failures. We assess the performance of THyTOB in an Ethernet-based commodity cluster, and show that it is on a par with the performance of other well-known, and more complex total order broadcast protocols inherently designed for the asynchronous model.

1. Introduction

Informally, total order broadcasts allow processes to send messages with the guarantee that all processes deliver messages in the same order. In this paper, we investigate the design and performance of a very simple *synchronous* total order broadcast that is built atop of an *asynchronous* distributed system based on a broadcast network. We show that the exploration of the inherent synchrony of the network can lead to a Time Hybrid Total Order Broadcast (THyTOB) protocol with good performance and reliability.

Broadcast networks, e.g., Ethernet or Infiniband, are adopted by the vast majority of current datacenter clusters and high-performance computers [1]. However, even with the algorithmic [23] and practical [11,13,19,46] importance of total order broadcast protocols, to the best of our knowledge, no one has tried to assess the performance of a synchronous total order protocol built atop broadcast networks. In this context, the main contributions of our work are: (i) a characterization of the inherent synchrony presented by broadcast networks; and (ii) the description and assessment of THyTOB, a novel total order broadcast that uses the inherent synchrony of the network as the key feature to simplify its design and implementation.

The characterization of the inherent synchrony of a broadcast network is carried out through experiments that systematically show that asynchronous distributed systems can be used as a foundation to emulate synchronous computations. A simple protocol is used to generate rounds of synchronous communication atop of the asynchronous distributed system. The accuracy and efficiency of the rounds generated are measured under varying workloads and scales to ensure that they meet the synchronization requirements of THyTOB. The picture that emerges from the experiments

* Corresponding author.
E-mail addresses: cason@ic.unicamp.br (D. Cason), buzato@ic.unicamp.br (L.E. Buzato).

http://dx.doi.org/10.1016/j.jpdc.2014.10.012
0743-7315/

allows us to conclude that it is possible to organize the computations of the asynchronous distributed system as a sequence of synchronous rounds.

The description of THyTOB – its specification, analysis of its progress and safety properties, and its performance evaluation – shows that it is on a par with the performance of other well-known total order broadcast protocols. THyTOB is latency optimal, requiring only two communication steps to deliver totally ordered messages. When set side by side with other implementations of total order broadcast, some based on Paxos [37], such as Treplica [48] or Ring-Paxos [43], other based on virtual synchrony [9], such as Spread [3] or LCR [29], THyTOB fares well both in terms of throughput and latency. Our study compares THyTOB to six other total order broadcast protocols: THyTOB ranks third in terms of throughput and first in terms of latency. In our comparison, the top performer in terms of throughput can sustain 950 Mbit/s with an average latency of 4.6 ms while THyTOB can deliver 525 Mbit/s but with a very precise latency of 2.3 ms. The results presented in the paper show that THyTOB fills an important gap in the throughput-latency spectrum of total order broadcast protocols designed for broadcast networks.

THyTOB adds to a trend towards the revision of the TCP/IP protocol stack for use in datacenters and clouds because of the benefits the revised protocols can bring to the performance and scalability of distributed applications. For instance, the synchronous nature of the datacenter network environment has allowed the implementation of a time-division multiple access (TDMA) MAC layer for commodity Ethernet that allows end hosts to dispense with TCP's congestion control [47]. The performance and simplicity of THyTOB makes it a good alternative to more complex protocols for several classes of cluster applications. For example, it can be used to implement an MPI Broadcast (`MPI_BCAST`) primitive equivalent to the one proposed by [31], but based on a much simpler and more efficient algorithm.

The design of THyTOB explores the fact that the processes and the broadcast network of asynchronous distributed systems have an inherent tendency to behave as synchronous distributed systems during reasonably long periods of time; Section 2 contains the definitions of both models of computation. Section 3 discusses in detail the design and implementation of THyTOB; special attention is given to the role of synchrony and asynchrony in the behavior of the protocol. Section 4 assesses the performance of THyTOB, and compares it with the performance of other well-known total order broadcasts. This section also discusses the different design principles that were used in each of the protocols and analyzes the influence each of them had on the performance and fault-tolerance of the protocols. From the perspective of THyTOB, the first three sections are self-contained because a complete understanding of the protocol can be obtained without looking into the experiments that demonstrate the inherent synchrony exhibited by broadcast networks. Nevertheless, the feasibility of the sub-systems that implement both models is verified experimentally in Section 5. Section 6 surveys the main classes of total order protocols designed so far and compares them with THyTOB; the section also comments on the relation of our time hybrid system and the concept of network synchronizers. Section 7 closes the paper; it summarizes our results and discusses briefly why we consider the principles that guide THyTOB's design an interesting contribution to the engineering of total order broadcast protocols, given the vast increase in the use of broadcast networks as key components of distributed systems.

2. Models of computation

In this section we define the asynchronous and synchronous models of computation, including their failure assumptions.

2.1. The asynchronous model

An asynchronous distributed system is composed of a fixed set of n processes connected by a broadcast network. Processes are asynchronous because there is no bound in the time they take to perform each of their computing steps. Processes can fail by crashing (crash-stop), but they never perform incorrect actions. Communication is accomplished by message passing, with the broadcast network offering a best-effort broadcast primitive. So, communication is one-to-all, asynchronous and unreliable: messages can be lost, duplicated, received out of order, or arbitrarily delayed, but we assume that they cannot be corrupted.

Processes have access to local clocks, which display monotonically increasing values. The clocks are not synchronized, so the deviations between the processors' clocks can be arbitrarily large, but we assume they do progress most of the time within a narrow envelope of real time. Such clocks are useful to set timeouts, but we are particularly interested in using them to schedule the execution of periodic tasks: given a period Δ, the processes are then expected to be awakened to execute a task approximately every Δ units of real time. As the processes are asynchronous, there is no guarantee that periodicity is maintained at every moment of the computation; but when considering sufficiently long periods of computation, the average interval of real time between successive invocations of the scheduled task is expected to be reasonably close to Δ.

In addition to this clock assumption, we aggregate to the model an empirical hypothesis regarding its timing behavior. Given that the load applied to the system is controlled, the time it takes to complete a broadcast of a message with up to S bytes is *likely* to be bound by a constant δ_S. This delay includes the time the process takes to perform a processing phase, which results in the broadcast of a message, and the latency to deliver the message to all non-faulty processes. Observe that this assumption does not impose bounds to the processing delays or network latencies: it only enables us to specify maximum delays for the actions to take place in the system, that are respected with high probability by their components.

Whenever the one-way timeout delay δ_S is violated we say that there was a *performance failure*. There are no bounds on the frequency with which performance failures can occur, or in the number of processes or channels that are affected by such failures. However, for the sake of progress, we assume that there is a time beyond which all broadcasts initiated by non-faulty processes are completed within δ_S. What this means is that after a period of instability there is always a time interval of some given minimum length in which the system behaves *stably*, that is, a period in which the number of performance failures has an upper bound.

Thus, we have complemented the well-known asynchronous crash-stop model with unreliable broadcast channels using two assumptions borrowed from the timed-asynchronous system model [20]: access to local clocks with bounded drift rates, and probabilistic maximum delays δ_S for the messages as a function of their size. The validity of these additional assumptions is restricted to periods of stability during which processes and channels exhibit a (predominantly) synchronous behavior.

2.2. The synchronous model

In the synchronous model the processes have access to what we call a *logical global clock*. It is a clock, in the sense that it periodically ticks at discrete instants of time, and it is global, as it provides a common source of time to the processes. It is a logical mechanism that generates ticks based on the processes' local physical clocks, and on the exchange of synchronization messages.