Lab

Total Order Broadcast

Through this work, we are going to study Total Order Broadcast (TO-Bcast) algorithms. In a nutshell, TO-Bcast is a communication primitive that ensures that messages sent to a set of processes are delivered (TO-Delivered) by all those processes in the same order.

1 Description of the work

1.1 In short

- Implementation of TO-Bcast algorithms: fixed sequencer, moving sequencer and privileged-based.
- Focus on the message ordering mechanisms.
- Use of a simulator for distributed algorithms (EasySim).
- Study performance as a function of the number of processes (latency and throughput)
- Write a short report on your work (**deadline:** February, 19)
- The work should be done by groups of at most 2 students.

The solutions cannot be shared between groups.

1.2 Goals

The goal is to implement and compare several TO-Bcast primitives. Namely, we are going to focus on fixed sequencer, moving sequencer and privilege-based algorithms. You can find a detailed description of these algorithms in [1][1] and [2][2].

TO-Bcast can be defined by the following properties:

Validity : If a correct process TO-Broadcasts a message m, then it eventually TO-Delivers m.

Uniform agreement : If a process TO-Delivers a message m, then all correct processes eventually TO-Deliver m.

Uniform integrity : For any message m, every process TO-Delivers m at most once, and only if m was previously TO-Broadcast by the sender of m.

Uniform total order : If processes p and q both TO-deliver messages m and m', then p TO-delivers m before m' if and only if q TO-delivers m before m'.

We are interested in the mechanisms used to ensure that messages are delivered by all processes in the same order. **We assume that no failure occur**: processes do not crash and channels are reliable. We would like to compare the performance of the three types of algorithms. More specifically, we would like to compare their latency and throughput.

We recall that latency and thoughput are defined a follows:

Latency : It is the time elapsed between the call of the TO-Bcast function by the source, and the time at which the message is TO-Delivered by all destinations.

[1] `http://infoscience.epfl.ch/record/88271/files/DSU04.pdf`
[2] `http://infoscience.epfl.ch/record/83648/files/paper.pdf`

Throughput : It is the number of messages TO-Delivered per unit of time.

You are asked to study for each algorithm how latency and throughput evolve with the number of processes.

Note that we consider a configuration where the set of sending processes is the same as the set of destination processes.

1.3 The simulator

EasySim is a simplified version of the Peersim simulator written in Java. You should use EasySim to implement the TO-Bcast protocols.

To keep things simple, we want to evaluate the protocols under the following conditions: channels are FIFO (First in, First out) and the latency of all messages is 1 (option `constantLatency`). Also, we assume that a process can send and receive at most 1 message in each round (options `maxMessagesToSend` and `maxMessagesToReceive` set to 1).

For this work we always consider a fully connected network topology (option `WireEntireNetwork`).

1.4 Evaluation of your work

You are asked to write a **short** report on your work. This report should include:

- A description of each of the algorithm you implemented (include the important pieces of code).
- An explanation of how the throughput and the latency of the protocols is evaluated.
- A discussion of the performance results comparing the different protocols.

The report in pdf format should be sent by email to `thomas.ropars@imag.fr` with the tag `[M2SCCI_DistAlg]` by February, 19. The report should be named based on the last name of the co-workers (name1-name2_report.pdf).

2 Detailed steps

You are strongly encouraged to follow the steps defined next.

2.1 Step 1: Privilege-based TO-Bcast

We suggest you to start by implementing a privilege-based TO-Bcast algorithm.

While designing and implementing this first algorithm, you should have the following things in mind:

- Think carefully about the design of your protocol to be able to re-use part of the work for the other protocols.
- Include mechanisms to check the correctness of your protocol: Is the same set of messages delivered in the same order by all processes?
- Think about meaningful tests to evaluate latency and throughput.

2.2 Step 2: Implementation of other algorithms

You should now implement and test other algorithms:

1. Fixed Sequencer TO-Bcast (UB variant): In the Unicast-Broadcast (UB) variant of fixed sequencer algorithms, a sender that wants to TO-Bcast a message, sends it to the sequencer. The sequencer is in charge of assigning sequence numbers and broadcasting messages to be TO-Delivered. See [1] for details.
2. Fixed Sequencer TO-Bcast (BB variant): In the Broadcast-Broadcast (BB) variant of fixed sequencer algorithms, the sender broadcast the message to be TO-Delivered. The sequencer assigns sequence numbers to messages and broadcasts these sequence numbers.
3. Moving Sequencer TO-Bcast.

2.3 Step 3: Bonus

If you have time, consider the case where channels are not FIFO and latency can vary. Re-implement the algorithms to deal with this case.

References

[1] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR)*, 36(4):372–421, 2004.

[2] Rachid Guerraoui, Ron R Levy, Bastian Pochon, and Vivien Quema. High throughput total order broadcast for cluster environments. In *International Conference on Dependable Systems and Networks*, pages 549–557. IEEE, 2006.