Programming Assignment 1:

Implementing a reliable and ordered multicast protocol

Chalmers University of Technology

By:
Julia Gustafsson, 19920807-2026
Gebrecherkos Haylay, 19890911-2511

Introduction

In this lab a multicast protocol have been designed. The protocol meets several requirements regarding the reliability, ordering and fault tolerance. The design of the protocol is based on well-known algorithms for multicasting in distributed systems, that are described in the course book "Distributed systems concepts and design" 5th, by George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair, and in the lecture slides of the course. The protocol is written in Java and used by a program called mcgui, developed by staff at Chalmers University of Technology.

Assumptions

Several assumptions are being made regarding the system. One assumption is that a node that crashes never comes back again to the system. Another assumption is that TCP connection is a reliable one-to-one operation.

Background

The protocol has several requirements regarding its design. The first requirement was the reliability. In order to send reliable broadcast the protocol should satisfy 3 properties; integrity, validity and agreement. The integrity property states that "each correct process P delivers a message from a correct process Q at most once and only if Q actually broadcasted it.". The validity property states that " A set of messages delivered by correct processes includes all messages broadcasted by correct processes.". The agreement property states that "all correct processes eventually deliver same set of messages". These description quotes are from the lecture slides by Philippas Tsigas. Another requirement was ordering. The order should be a total order which preserves causality. Causal order is defined by "If a process delivers a message m before broadcasting a message m', then no correct process delivers m' unless it has previously delivered m" . As mentioned, the protocol should implement total ordering, which means that "If correct processes P and Q both deliver messages m and m', then p delivers m before m' if and only if q delivers m before m'." These quotes are also from lecture slides by Philippas Tsigas.

Technical description

Different algorithms are implemented in order to meet the requirement of the multicast protocol. The protocol guarantees a total order of the messages with preserved causality, reliable multicasting and handles crashing nodes.

In order to keep track of each message, the messages are identified by combination of the sending node's id and the message id. The sending node id is the same number as the node's place in the setup file, minus 1. So node number 1 in the setup list has id 0 and so on. The message id is unique for a message within the node, in each node, that starts from 1 up to the number of messages the node has sent. Each node has its own vector clocks, which include a logical clock for each of the nodes in the system. Each node maintains a list of all the received messages that waits to get their final sequence number and to eventually be delivered. In order to be able to keep track of messages that already has been delivered each node keeps track of all the delivered messages. A node that has sent messages that haven't been delivered also maintains the list of proposed sequence numbers as well.

In order to implement total ordering with preserved causality the ISIS algorithm for total ordering is used together with vector clocks. ISIS algorithm assures that the messages will be delivered in a total ordering. This is done by using sequence number. The originate sender of the message send the message to all the other nodes. The message contains a proposed sequence number, which is sum of the largest sequence number that the node has received and 1. When the other nodes receive the message, they will propose a sequence number as well in the same manner. This message containing the new proposed the sequence number, will be sent back to the node the message originated from. When the node receives all responses from the other nodes it calculates the maximum of the proposed sequence number and send the message again to all nodes. Now the message contains the maximum sequence number, now called final sequence number. This sequence number is the actual sequence number that will be used for the message. The message can now, when a node has received it, be delivered.
As mentioned above, using ISIS the sending node decide the final sequence number of the message by taking the maximum proposed sequence number. The proposed sequence number is the local maximal vector clock plus 1 for a node, and the chosen sequence number will be the new vector clock value for the sending node in each of the nodes. When the final sequence number arrives the nodes update their local vector clock value for the node that send the originate message to the final sequence number- which might be a larger number than node's proposed sequence number.

The protocol is designed to send reliable multicast; that it satisfies the 3 properties: integrity, validity and agreement. These properties are satisfied by the protocol in a numerous of ways. Missed messages will be retransmitted and by controlling that the message is accurate the protocol meets the requirement for being a reliable multicasting protocol. The control of a message's accuracy is done by checking that it is a message that we haven't got it before in case of a new message from another peer without a final sequence number, and by checking if the node actually is waiting for the received message, in case of a message with a final sequence number. If a node receives a message, which it is not waiting for or a message it already has received, it will just discard it. These checks could be done by checking the return value of add, remove functions of the used data structure. In addition, additional propositions from a node that already has proposed a value will not be added to the list of proposed values. This is guaranteed by the data structure used for the proposed values and their corresponding nodes. In case of a new message without a final sequence number from another peer or a message with a final sequence number that the node is waiting for, the node perform flooding. It sends that message further to all nodes except from the originate sender and the one that sent it to the node itself. In addition retransmission is used. The retransmission is done by a peer node, when a node receives a message with a proposed sequence x from a peer it will check if the sequence number is lower than it's own proposition. If it is it will retransmit all the delivered messages with sequence number from x to the last delivered. In those ways the protocol guarantees to meet the integrity, validity and agreement properties, and hence provide reliable multicast.

Crashing nodes are handled in several ways. The crashed nodes are saved into a list, which is checked before sending a message, so that no messages are sent to crashed nodes. Additionally, message without a final sequence number is discarded if the sender has crashed as such a message will not be delivered. Crashing processes and their sequence number propositions will also be removed from the list of sequence number propositions, in order to not get faulty sequence numbers. An example of a faulty sequence number is if the crashed node has sent a message m with proposed sequence number z, increased its the vector clock, sent a proposed priority number z+1 and then crashed. The message m with sequence number z doesn't exist and hence the priority number z + 1 is not valid. Messages that was only waiting the crashed nodes sequence proposal will be able send the message with the final sequence number to the remaining nodes and deliver the message itself, after a peer down message is received. Additionally, messages from the peer that is down without final sequence number is removed from the list of messages to wait for and also some additional data that belong to those messages. Retransmission and flooding, that was mentioned before, solves the problem of missing messages in case of a node crash in the middle of sending a message with the final sequence number set. The retransmission of messages is done by a peer if it discovers that a node misses messages(by looking at the proposed sequence number). This means that all nodes eventually will receive and deliver all messages with the final sequence number set. This hold as long as a message has been received and delivered by at least one correct process. So even if a node dies before it has sent a message with the final sequence number to all of the nodes, it will eventually be sent to all other nodes.

Conclusion

The multicast protocol that have been designed is reliable, implements total ordering with preserved causal ordering and also handles crashing nodes. The algorithm used for total ordering, the ISIS algorithm, didn't support causal ordering. But that was easily solved by simply adding vector clocks to the protocol, and then the causal ordering was maintained. To ensure that the messages are multicasted in a reliable way -satisfies the 3 properties; agreement, validity, integrity - was done by making use of used data structure's ability to not add duplicate and to check if a specific message already or is missing. In order to both meet the requirement for reliable multicast and to handle crashing nodes, retransmission and flooding are used. Also, several action regarding crashed nodes are taken, as they no longer participates and will no longer be waiting for. As described above, several of the requirements implementation are overlapping or make it simpler to implement another part of the protocol.