Groupy: a group membership service

Chuan Su

October 2, 2018

1 Introduction

In group communication, the essential feature is that a process issues only one multicast operation to send a message to each of a group of processes instead of issueing multiple send operations to individual processes. Reliable and ordering in multicasting are therefore one of the major challenge in implementing a group communucation service.

Futuremore, processes may join, leave the group or even fail at any time during group communication. Maitaining an accurate view of the group membership is therefore another important implmentations issues for group communication services.

This report will walk through the problems we encountered while implmenting groupy - a group communication sytem and our solutions in related to these issues.

2 Multicasting Reliability and Ordering

In our assignment, message multicasting is achieved by *Leader & Slave* pattern, that is all processes that wish to multicast a message will send the message to the leader and leader will in turn multicast the message to each member of the group.

While multicasting a message that is either forwarded by a *Slave* or sent from a process outside the group, leader will tag each message with a sequnce number indicating the message delivery order and each other process in the group keeps track of the sequence number it received. Process will deliver the message to its application layer only if the next sequence number (sent from leader) is the one expected to receive, that is:

```
ExpectedSeqenceNumber == CurrentHoldSequenceNumber + 1
```

The sequence number can also be used to detect duplcations of messages, which aligns with Integrity of Reliable Multicasting - *A correct process delivers a message m at most once.*

```
NextSequenceNumber < CurrentHoldSequenceNumber + 1
```

You may have noticed that it is a message missing indication if

```
NextSequenceNumber > CurrentHoldSequenceNumber + 1
```

Furthermore the difference between `NextSequenceNumber` and `CurrentHoldSequenceNumber + 1` indicates the amount of missing messages multicasted from group leader.

To handle the possible missing messages we,in our assignment,implemented a history queue that piggyback on the each message sent from leader process. When missing messages are detected (by the formula above), the slave process hold the delivery of the current message and tranverse through history queue delivering the missing messages to the Applicaton Layer in FIFO order and then deliver the current message.

```
MissedMsgs = lists:sublist(Queue, (N+1), (S - (N + 1))),
lists:foreach(fun(MissedMsg) -> Master ! MissedMsg end, MissedMsgs),
Master ! CurrMsg.
```

The original idea was that every incoming messges whose sequence number is greater than expected sequnce is placed in a hold-back queue in the *Slave* process. Slave process then communicates with *Leader* process asking for the missing messages, which will save the bredbands carrying the history queue on every outgoing messages.

However the drawback of that approach is that missing messages will never be recovered if the *Leader* process crashes during group communication since none of the *Slave* process holds a copy of the history messages.

When the *Leader* process crashes just before sending out the message we do have a situation that the message will not be delivered by any *Slave* process to their Application layer (Master Process). But this situation doesnot violate the atomic multicast *(All or None)* agreement since no node has once delivered the message.

3 Conclusion

While reading reliable multicast, I misunderstood the differences between *message is received at process* and *message is deliverd to a process.* In fact

multicasting reliability and ordering in Group Communication applies solely to the *Delivery* that is the interface between protocal layer and the application. In contrast *receive* (and send) is the interface between the protocol layer and underlying network channels.

From this seminar, I have gained a better understanding of Group Membership Service especially multicasting in Group Communication.