ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2010 M. Kuhn / S. Welten / Prof. R. Wattenhofer

Distributed Systems Part II

Exercise Sheet 5

1 Three Phase Commit

This is the same problem as Problem 3 on Exercise Sheet 4!

Three phase commit allows multiple processes to commit or abort a transaction simultaneously. In this task, you have to think about error recovery for 3PC. We assume that processes can crash any time but do not recover and there are no Byzantine processes. The network is asynchronous, messages are neither lost nor reordered. A process crashing during a broadcast may send its message to a subset of receivers. A process broadcasting requests can already receive replies even if not all processes have yet received the request.
Processes are asked to make the final step together, this is called non-blocking property:

NB: if any operational process is uncertain, then no process can decide to commit.

Remember, 3PC runs in 6 steps:

1. The coordinator sends `VOTE` to all participants.
2. Participants answer with `YES` or `NO`.
3. The coordinator sends `ABORT` or `PREPARE`.
4. Participants send `ACK` or abort.
5. The coordinator sends `COMMIT`.
6. Participants commit.

a) For five of these steps, processes have to wait before they can execute them. Write down in which steps which processes have to wait for what messages.

b) If a message does not arrive because its sender has crashed, the waiting process times out. For each of the five steps where processes wait: how should the processes react to a timeout? Make sure NB is not violated. Hint: Can they safely abort or do they have to commit? Must they elect a new coordinator?

c) (optional) Open question for bored people: Assuming crashed processes recover and can remember their last actions. Give a sequence of events where 3PC blocks (meaning: where 3PC gets in a state in which it is impossible to make a decision). Note: if no decision has been made yet, then a newly elected coordinator may choose either to commit or to abort. Once the protocol started it cannot be restarted.

2 Paxos Timeline

In the following, we want to run an implementation of Paxos on a set of 3 nodes (A, B, C) that act as acceptors. Moreover, we assume that there are two more nodes (Q, R) that act as proposers. The implementation of the acceptors is exactly as shown in the lecture, Slide 49. The two proposers use the implementation given in Figure 1.

```
function suggestValue(Node N1, Node N2, Timeout t, Value x, RequestNumber n)

Send prepare(x,n) to nodes N1, N2
Wait t seconds
If within these t seconds, either N1 or N2 has not replied then
    suggestValue(N1, N2, t, x, n + 2)

Let (y, m) be the received proposal with the largest request number
if m == 0 then
    u := x
else
    u := y
Send propose(u, n) to N1, N2
Wait t seconds
if within these t seconds, either N1 or N2 has not replied with ack(u, n) then
    suggestValue(N1, N2, t, x, n + 2)

Print("value is chosen: " + u)
```

Figure 1: Code executed by the proposers.

Draw a timeline containing all transmitted messages if a user invokes `suggestValue(A, B, 1, 22, 1)` on Q at time T_0 and `suggestValue(B, C, 2, 33, 2)` on R at time $T_0 + 0.5sec$! We assume that processing times on the nodes can be neglected (i.e. is zero), and that all messages arrive within less than $0.5sec$.

3 Paxos Acceptors

In the lecture you have seen how Paxos can solve consensus without the need of a single coordinator. It lets each node execute one or more of the following roles: proposer, acceptor, and learner. In this task you will have a closer look at the purpose of the acceptor.

a) Assume, that in a network of 5 nodes there is one node with a faulty register that is used to store the value n_{max} of its acceptor programm (See slide *Paxos: Algorithm of the Acceptor*). Can this pose a problem to the Paxos algorithm? Explain what happens in the worst case scenario.

b) Paxos assumes that there are no Byzantine failures. That is, memory failures such as the one described above, are not assumed to happen. Under these original assumptions, can you explain what the purpose of the *prepare step* in Paxos is?