HS 2009 Prof. Dr. Roger Wattenhofer, Thomas Locher, Remo Meier, Benjamin Sigg

Distributed Systems
Theory exercise 3

Assigned: November 20, 2009
Discussion: November 27, 2009

1 Authentication

In the lecture we talked about an algorithm using authentication to reach consensus in an environment with Byzantine processes. See chapter 6, slide 132 ff for a detailed discussion about this algorithm.

a) Modify the algorithm in such a way that in handles arbitrary input. Write your algorithm as pseudo-code. The processes may also agree on a special "sender faulty"-value.

Hint: implement `value` as a set, work with the size of the set.

b) Prove the correctness of your algorithm.

2 Randomization

In the lecture we talked about a randomized algorithm reaching consensus in an asynchronous system with Byzantine failures. See chapter 6, slides 137 ff for a detailed discussion about this algorithm.

We assume that there are only crash failures but no Byzantine failures. A crash can happen anytime and broadcasts may not be completed. Crashed processes do not recover.

a) How many crash-failed processes can the algorithm handle?

Hint: Have a close look at the proofs for the validity condition, agreement, and termination.

b) Modify the algorithm to handle more crash failures.

c) How many crash failed processes can your modified algorithm handle?

3 Three Phase Commit

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, 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) In five of these steps processes have to wait, 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 yet been made, then a newly elected coordinator may choose either to commit or to abort. Once the protocol started it cannot be restarted.