Tutor: Mitja Krebs

Exercise Sheet 6

Consensus

Exercise 1 (Deterministic Random Consensus?!)**.** Algorithm 6.14 from the lecture notes solves consensus in the asynchronous model. It seems that this algorithm would be faster, if nodes picked a value deterministically instead of randomly in Line 19. However, a remark in the lecture notes claims that such a deterministic selection of a value will not work. We did it anyway! See algorithm 1 below, the only change is on Line 23 (Line 19 in Algorithm 6.14 from the lecture).

Show that this algorithm does not solve consensus! Start by choosing initial values for all nodes and show that the algorithm below does not terminate.

Exercise 2 (Impossibility of Consensus for $f \geq n/2$)**.** In the lecture, a randomized consensus algorithm (Algorithm 6.14 in the lecture notes) was presented that tolerates up to $f < n/2$ node failures. Prove that no consensus algorithm can tolerate $f \geq n/2$ node failures.

Hint: Construct an execution scenario by carefully choosing the initial values of the nodes and specifying communication links with delayed message delivery (i.e., "slow" edges). Show that if $f = n/2$ particular nodes crash before sending any messages, then any algorithm satisfying the **validity** and **agreement** conditions must violate the **termination** condition.

Exercise 3. Assume the that we are in an asynchronous setting with a register R_i for each process i, with arbitrary amount of memory. Each process i can perform the atomic operations read(R_j) for every j and write(R_i).

We want to show that consensus is impossible if we allow crashes.

The tree of possible states can be viewed as a tree as follows: The root is the state of the network in its initial state. For each state S in the tree, there is a child for each process p, which is the state $e_p(S)$ of the network after the next operation e_p (a read or write operation as described above) of p has been performed.

1. Let S be a state, and let e_j be the next operation for every process j for the state S. Now consider for some process p the state $e_p(S)$, and let e'_j be the next operation for each process j for the state $e_p(S)$. Show that $e_j = e'_j$ for all $j \neq p$.

2. Show that for a state S, if for processes p and q their next operations e_p and e_q, respectively, are either both read or both write operations, these operations commute, that is we have $e_p(e_q(S)) = e_q(e_p(S))$. (Note that we can pose this question this way only because of the previous point)

3. Show that there is an initial configuration such that the state of the system is at some point (and there fore initially) bivalent.

4. Let C be a bivalent configuration, and let e_p be the next event for process p from C. Let $\mathcal{D}$ be the set of configurations reachable from C where e_p hasn't been performed yet (meaning only processes that are not p have performed operations), including C itself. Show that there must exist $D \in \mathcal{D}$ such that $e_p(D)$ is bivalent.

5. Conclude that consensus is impossible.

Algorithm 1 Deterministic Randomized Consensus (Ben-Or)

1: $v_i \in \{0, 1\}$ ◁ input bit
2: round = 1
3: decided = false
4: Broadcast `myValue`(v_i, round)
5: **while** true **do**

Propose

6: Wait until a majority of `myValue` messages of current round arrived
7: **if** all messages contain the same value v **then**
8: Broadcast `propose`(v, round)
9: **else**
10: Broadcast `propose`($\perp$, round)
11: **end if**
12: **if** decided **then**
13: Broadcast `myValue`(v_i, round+1)
14: Decide for v_i and terminate
15: **end if**

Adapt

16: Wait until a majority of `propose` messages of current round arrived
17: **if** all messages propose the same value v **then**
18: $v_i = v$
19: decided = true
20: **else if** there is at least one proposal for v **then**
21: $v_i = v$
22: **else**
23: Choose $v_i = 1$
24: **end if**
25: round = round + 1
26: Broadcast `myValue`(v_i, round)
27: **end while**