exam question for distributed systems

Claudio Ferrari

November 21, 2017

1 Randomized Consensus revisited

This question is intended to show that even small changes, that intuitively should have no effect, in one of the algorithms presented in the lecture can prevent it from achieving its purpose.
The Questions haven purposely been tried to be formulated and presented in a way such that the examiner can modify the extent and difficulty of the question. The algorithm and the modifications should be printed next to the question for reference.

1.1 Question

Consider Algorithm 2.15 for randomized consensus from the lecture notes.

a) Suppose we wanted to add the possibility of not only having binary input but ternary. For simplicity we assume those inputs are $\{1, 2, 3\}$. The only change we make is that in line 23, we decide on a new value with probability $\frac{1}{3}$ for all three values. Does the algorithm still solve consensus, meaning do validity, termination and agreement still hold? Justify your answer.

Possible Hints/simplifications:
- consider validity
- give Lemma 2.16 to use without proof
- say that things from the lecture notes can be used without proof

Possible added difficulty:
- in case termination is given, what's the expected time
- what is the strongest form of validity given

b) Sin, Quino et al (read: "think we know it all") published a paper saying

that line 13 of the Algorithm is redundant and should be left out. Their argument was that if you have decided then you broadcasted the value at the end of the earlier round already. Do you agree with that claim? Does the algorithm still solve consensus? Justify your answer.

Possible Hints:
- consider termination or agreement
- to prove validity in binary case it is enough to prove all-same validity

If you don't want to confuse students by falsely claiming the existence of a paper, start the question off like this: Your friend Reeljih Nius suggests... And if you don't like my admittedly bad humor at all, then just leave that part out.

c) Bonus: Can you think of a slight modification that allows to generalize the algorithm to an arbitrary number of inputs?

1.2 Answer

a) Agreement and termination still hold by the same argument as for the binary input case (see proofs of Lemma 2.18 and 2.19). The proof for termination has to be changed a slight bit, the chance that all nodes have the same value in a given round is at least $\frac{1}{3^n}$, therefore the Algorithm terminates in expected time $O(3^n)$. However validity does not hold any more, consider the following case:
We have three nodes with inputs 1, 2 and 2 respectively. Initially they all broadcast their value and wait for a majority to arrive. By chance they all see the values 1 and 2 as the first two (represents a majority). Therefore all nodes will propose nothing. In line 16 they must all get two nothing proposals (nothing else has been proposed) and thus they all choose their next input randomly. As luck would have it, they all choose 3. In the next round they all broadcast it and because of that must also all propose it. Then they all will decide on 3 and terminate in the next round. Note that 3 was not an input of any correct node. We have shown that there is a possible execution in which the output was not originally an input to any node, therefore validity is violated.
Note that all-same validity is still given. If all nodes start with the same value, then one can easily see that they must all decide on that value in the first round (analogous to the proof of Lemma 2.17).

b) Consider the following scenario: We have three nodes. Node 1 decides on a value v in round r. That means it saw a majority (two) proposals for v. Therefore in the same round the other nodes saw at least one proposal for v, if they saw two, they decide too. If they saw only one, they set their value to v. Assume node 2 also saw two proposals for v and node 3 saw only 1 proposal for v. At the end of round r all nodes broadcast v. In round r + 1, all nodes will propose v. The nodes 1 and 2 which decided in round r terminate in this round, and because of the change in line 13 they do not broadcast v anymore. Node 3 that is still running will decide in round r + 1 (since all proposals were for v), broadcast v at the end of the round and go to round r + 2. In line 6 of round r + 2 it will only hear its own myValue broadcast, therefore never get a majority and wait forever. In this scenario termination and agreement are violated and therefore line 13 is absolutely crucial and cannot be left out.
Note that validity still holds. In the binary case, all-same-validity implies validity, so if they all start with the same input, they all decide on that input in the first round (see proof of Lemma 2.17). They all broadcast the value at the end of the round, thus make progress in line 6, and then terminate in line 14 in the second round.

c) We will make the following changes: in line 10 if we don't have a majority, we Broadcast $propose(\perp, v_i, round)$. With v_i being our own value. In line 23 we choose v_i uniformly at random among all the values heard in $\perp$ proposals. If a value is heard twice we still select it with the same probability as the others.
We know prove that this solves consensus, by proving validity, agreement and termination. Note that Lemma 2.16. also applies for our modifications and can therefore be used.
Validity: We prove that no node can ever hold a value that wasn't its own input or that came from another node. In other words, a node can't change its value to something that wasn't an input to any node. That together with the fact that a proposed value is a value of a node (which is trivial), implies validity. A node can choose a value either from a proposal or by choosing u.a.r. in line 23. If it takes a value from a proposal we immediately see that this value must have been broadcasted by another node. If it is chosen in line 23, we modified the algorithm such that it is chosen among the values contained in the proposal messages therefore they are previous values of nodes. Agreement: We note that a proposal can only be made for one value (because of the majority). The rest of the proof is analogous to

Lemma 2.18. in the lecture notes and is omitted here for brevity.
Termination: Suppose we start with m distinct input values. All nodes that adapt their value from a proposal choose the same value as proven earlier. The rest chooses u.a.r. from m values, therefore we have all the same values across the nodes with probability at least $\frac{1}{m^n}$. If all nodes have the same value they terminate in finite time (see proof of Lemma 2.18.). Therefore the algorithm terminates in expected time $O(m^n)$.