Homework

Robust Distributed Algorithms, Master QDCS M1

to submit in pairs on the 28th of October (with the exam)

1. Consider the basic LCR solution to leader election seen in class (see Leader Election slides 4 - 6). **Prove** that this algorithm terminates and it is also correct at termination (the definitions of these terms can be found in Model slides of the course).

2. Consider the Time-Slice algorithm (see Leader Election slides 20 - 24). Propose a **slight** change to the algorithm to improve its **bit complexity** (in terms of O). Explain why the changed algorithm is correct and analyze the new bit complexity.

3. Consider the consensus problem as is defined in slide 4 of Consensus slides. Assume a synchronous network with only two always correct processes (knowing they are only two and correct). Assume that the communication links **can** lose a non-bounded, but **finite** number of messages (notice the difference with the conditions in slide 17 of Consensus slides).
    a. Propose an algorithm solving consensus in such a network. Explain the algorithm informally and why it is correct.
    b. Write down the pseudo-code using the formalism defined in class (define M, $states_i$, $start_i$, $trans_i$, $msgs_i$, … for a process i).
    c. **Prove** the correctness (all 3 conditions given in slide 4 have to be proven).

4. Consider the Flood-Set algorithm and assumptions in slide 3 of Consensus slides. Assume that n=4 and f=2. Present an execution scenario where every two correct processes have different values in variable W up to round 2 (including round 2). Whether these values become equal in round 3? Explain why.

5. Consider 4 processes distributed within a complete network (known to all processes), of which at most one (unknown) is Byzantine (n= 4, f= 1). We consider the following agreement protocol, composed of 2 synchronous rounds (the transmission of messages is synchronous) and the messages are the sets of pairs (identifier, initial integer value):

Round 1*: each process i sends a pair consisting of its identifier i and its initial integer value v_i, (message (i, v_i)), to all the other processes and receives messages from the others (a message (j,v_j) from a process p≠j is ignored).*

Round 2: *each process i relays to all other processes the (j, v_j) pairs received during round 1 (only one pair at most for each j), and receives the pairs from the others (a process ignores a (j, v_j) pair from process j)*

At the end of this round, process i assigns *to process j the strict majority value of the values received for j (in pairs (j, v)) during these two rounds (if such a majority value exists, and otherwise it assigns the smallest majority value in case of a tie).*

The decided value of process i is the majority value among the values it has assigned (the smallest majority value in case of a tie).

We assume that a node can receive (and treat) at most one pair (j, v), for a given identifier j, on a link during a round (otherwise it ignores all duplicates). Also, remember that each process knows all its neighbors (in fact, the whole communication graph), and a correct process ignores any message with an unknown identifier in the given solution.

1) Show that at the end of the first round, the majority values that could have been calculated (from the set of received values) by correct processes may be different (so such a solution with the decision at the end of the 2nd round, may not be correct).
2) Show that at the end of the second round, a correct process cannot receive, for the same *j*, 2 pairs *(j, v_j)* and 2 pairs *(j, v'_j)* with $v_j \neq v'_j$.
3) Show that, at the end of the second round, given a process j, all correct processes assign to j the same value.
4) Deduce that the executions of this protocol verify the agreement condition. **Prove**.
5) Do they verify the other conditions of the Byzantine consensus? **Prove**.