Homework

Robust Distributed Algorithms (Master QDCS M1)

to submit in pairs, on the 22[d] of October (before the exam)

(not compulsory, but if its grade is better than of your final exam, it will count 25% of the final grade, and 75% for the exam)

1. Consider the basic LCR solution to leader election seen in class (see Leader Election slides 5 - 7). **Prove** that this algorithm terminates and that it is also correct at termination (the definitions of these terms can be found in the Model slides of the course).

2. Assume an arbitrary communication topology and synchronous model. Propose a solution to the leader election problem; satisfying also the three variants appeared in slide 3 (Leader Election slides). Give a short textual explanation of the algorithm and provide also the complete pseudo-code (states, starts, msgs(), trans()).
   Assume that an upper bound on the network diameter Diam is given to the processes (diameter is the longest among all the shortest paths in the given network). Explain why the latter assumption is needed.

3. Consider the Time-Slice algorithm (see Leader Election slides 21 - 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.

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 of the algorithm where every two (still) alive (not yet crashed) processes have different values in variable W at least in round 1 and 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 <u>decided value</u> 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/assigned (from the set of the received values) by correct processes may be different (so such a solution with the decision at the end of the 1st 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* ≠ 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**.