Distributed Algorithms – TD

Problem of agreement/Byzantine consensus

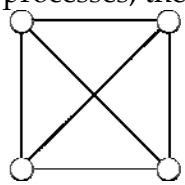
It is assumed that a network satisfies all the assumptions for synchronised (real time) round operation. Moreover, the network is complete, the links are reliable (no loss, duplication or corruption of messages), and each process knows the whole network. We propose to study possible solutions to the Byzantine Generals problem with input values in any set U.
It is assumed that n > 3f, where n is the total number of processes and f is an upper bound on the number of faulty (incorrect) processes, called Byzantine. Such processes have a behaviour that can deviate from the solution protocol (an arbitrary behaviour). Byzantine processes can start in any state, send arbitrary messages and do not follow the transition function, but cannot control other entities in the system.

The specification of the *Byzantine Generals' problem*:
Agreement: two **non-faulty/correct** processes never decide on different values.
Validity: if all **non-faulty** processes have the same initial value v, v is the only possible decision value of a **non-faulty** process.
Termination: every non-faulty process eventually decides on a value.

The decision is made only once. It is irreversible.

1 - We consider the following protocol (for a process i):
state_i et start_i:
    v_i: U; initialised to the initial value of i
    m_i: U
msgs_i: send v_i to every neighbour (including me)
trans_i:
If n messages with values (including mine) are received, then
    Compute the majority value m_i (the smallest in case of a tie) of these values
    v_i:= m_i

We choose n = 4, f = 1 (f is an upper bound on the number of Byzantine processes, the transmissions are synchronous and the network is complete).



1) Show that it is possible for a non-faulty process not to progress (its state no longer changes) and transform the algorithm to avoid this.
2) The initial values are integers. Are the variables m_i of the correct processes necessarily equal after a certain time (after the transformation of 1)?

2 - Consider 4 processes distributed in a complete network, of which at most one is Byzantine, and the following agreement protocol, composed of 3 synchronous rounds (the transmission of messages is synchronous).

In the first round, each process broadcasts its initial value to all the others and receives the values of the others. It computes the average value of the values available to it, and chooses the value closest to the computed average (if there are two close values, the higher of the two is chosen).
Rounds 2 and 3 are similar to the first round, while broadcasting the average value chosen in the previous round.
The decided value is the average value chosen at the end of the third round.

Is this protocol a correct solution to the consensus problem?

3 - We consider the problem of *reliable broadcast*, with a distinguished *marshal-process* that broadcasts its initial value, for n= 4 and f=1 (marshal is counted in the 4 processes), and the transmission of messages is still synchronous. The other processes are called *generals*. Note that this is not a problem of consensus (see the definition below), since only the marshal has an initial value, which is disseminated. Though, the goal is still that the correct processes decide (or *accept*) all the same value, within a finite time. In addition, if the marshal is not faulty, each correct process has to decide/accept the disseminated value. Formally, the problem is defined as follows:
Termination: Every non-faulty general eventually decides/accepts a value.
Approval: Any two non-faulty generals never decide/accept different values.
Validity: If the marshal is non-faulty, the common decision value is the initial marshal's value.

Consider the following algorithm, which operates in synchronous rounds:
- in the first round, the marshal broadcasts its value to all;
- in the second round, each process other than the marshal sends the value received in round 1 to all others except the marshal
- At the end of the second round, each process decides (irreversibly accepts) the median value among the values available to it.
(An empty received message – null – is considered to be a default value $u' \in U$.)

1) Prove that this algorithm is a correct solution to the problem of reliable broadcast?
2) Is this algorithm also a correct solution for all n and f? What about all n > 3f?

4 - 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 neighbours (in fact, the whole communication graph), and a correct process ignores any message with an unknown identifier in the given solution.

1) Check 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.
5) Do they verify the other conditions of the Byzantine consensus?