Assignment 3

Theorem 1. *It is impossible to solve the Byzantine agreement deterministically in synchronous message passing network when up to f processes can be Byzantine, if the network is not $(2f+1)$-connected.*

Proof. Suppose for a contradiction that there exists an algorithm $\mathcal{A}$ which solves the Byzantine agreement problem deterministically in a synchronous message passing model when the network is not $(2f+1)$-connected. Consider the complete network on three processors where one processor is Byzantine. With $f = 1$ observe that this network is not $(2f+1)$-connected. By Theorem 5.7 of Attiya (stated as Theorem 2 here) no algorithm solves this consensus problem. This is a contradiction since $\mathcal{A}$ was suppose to be such an algorithm. *If the question further requires that the that the number of processors $n > 3f$, then the complete graph on four nodes with one edge removed serves as a counter example. The proof is similar to the impossibility result on the triangle.* □

Theorem 2. ***(Theorem 5.7 in the Attiya Textbook.)*** *In a complete system with three processors and one Byzantine processor, there is no algorithm that solves the consensus problem.*

Theorem 3. *There exists a deterministic algorithm tolerating up to f Byzantine processes that solves Byzantine agreement in a $(2f+1)$-connected network with $n > 3f$ processors.*

Proof. We will describe such a deterministic algorithm and prove its correctness. This algorithm is inspired by the exponential bit complexity algorithm for Byzantine agreement in a complete network. Even though our network is no longer complete, we can simulate completeness by taking the majority of messages along node disjoint paths. *We assume that global information about the graph is known to all processors; there is no indication in the question that this is not the case.*

Let $G = (V, E)$ be a $(2f+1)$-connected network where $|V| = n$, $|E| = m$, up to f processors can be Byzantine, and $n > 3f$. Further, suppose each processor p_i receives input v_i and stores a private knowledge tree K_i. This tree will have the same structure as the tree for the exponential BA consensus algorithm in the complete network (i.e. on the first level of the tree there is a node associated with *every* index $\langle i \rangle$ where $1 \le i \le n$, on the second level there exists a node associated with every pair of indices $\langle i, j \rangle$ for $1 \le i, j \le n$, etc.)

The algorithm is divided into two stages: first p_i builds K_i by sending requests along $(2f+1)$-node disjoint paths in G and decorating K_i with the appropriate information. Then p_i analyzes the contents of K_i by taking the recursive majority function starting at the root.

(**Detailed description of the tree-building stage.**) p_i receives a message from p_j about level l of K_i as follows: p_i decomposes G into $(2f+1)$ internal-node-disjoint paths from p_i to p_j. This is possible since G is $(2f+1)$-connected. For each path $t : p_i = p_{k_0} \rightsquigarrow p_{k_1} \rightsquigarrow \cdots \rightsquigarrow p_{k_c} = p_j$, p_i sends message $\langle l, i = k_0, k_1, ..., k_c = j \rangle$ to p_{k_1}. This message is passed along from p_{k_l} to $p_{k_{l+1}}$ until it reaches p_j. If p_j is a non-faulty processor it must return information about level l of its knowledge tree back along t to p_i. All processors will wait $2n$ rounds to deliver information about one level since this is the farthest distance between two processors. Once p_i receives the $(2f+1)$ responses along all the node disjoint paths, it decorates the associated node in K_i with the majority of the responses. This is the same procedure as for the complete network algorithm except for the $2n$ expansion in the number of rounds.

The analysis stage for each processor is identical to that of the BA consensus algorithm in a complete network. Each processor p_i starts at the root and recursively takes the majority function. The majority of a leaf is simply the value of the leaf. The majority of any internal node is the majority of the values on its children. If a node does not have a majority or has an invalid value, its value is replaced with $\perp$ which is ignored by the majority function.

This algorithm terminates since the algorithm for BA consensus in a complete network terminates (this algorithm only takes $2n$ rounds longer). Validity and agreement both follow from validity and agreement of the algorithm for BA consensus in a complete network. These results transfer since the knowledge tree for each processor is decorated in the same way in both type of networks ($(2f+1)$-connected and complete). Since the paths chosen were node-disjoint and at most f processors can be Byzantine, $f+1$ paths contain only non-faulty nodes. The results delivered on these paths form a majority and it is this majority value which decorates K_i for processor p_i.

$\square$