Distributed Computing, Spring 2016,
Assignment # 1. Due: April 3rd
Prof. Yehuda Afek

Note: YOU MUST DO THE HOMEWORK BY YOURSELF. If you have difficulties in solving a question you may discuss it with friends, BUT you MUST phrase, write and formulate the answers by yourself, after you understand the solution. The language of the solution must be entirely your own.

Notice 2: Unless your hand writing is extremely clear, you should type (Word, or LaTeX, etc.) and print your homework.

Notice 3: Put your name and id number on each page of the solution.

1. In the first class we saw a simple algorithm for Broadcast and Echo (in which one initiator sends a message to all other nodes in the network and receives a signal after all nodes are known to have received a copy of the message). In that algorithm we used one type of messages, message M for both the broadcast and the echo.

   (a) Here you are required to describe a variant of the algorithm and provide its code (pseudo code) in such a way that two message types are used, type M (the original message that is being disseminated) and message type Ack (Acknowledgement) which is used to collect the acknowledgements in order to detect the termination of the broadcast process.

   (b) What is the exact complexity (messages and time) of the new algorithm.

   (c) Modify the code so at the end the root of the broadcast tree will know the total number of leaves in the created tree, and h the height of the created tree.

2. In class we saw an algorithm for Termination Detection of Diffusing computation. In that algorithm there were two types of messages, message type M (of the original diffusing computation) and message type ACK which was used to collect the acknowledgements in order to detect the termination of the diffusing process.

   (a) Here you are required to write algorithm code (pseudo code). What is the exact complexity (messages and time overhead over the diffusing computation) of the new algorithm as a function of m the number of messages sent by the diffusing computation.

   (b) Modify the code so at the end the root node of the termination detection will know the total number of times in the algorithm that a node received a message m and did not send any message in response to receiving this message (i.e., the number of leafs in the virtual tree we discussed in class).

3. Many distributed algorithms do not have a built-in termination detection mechanism. Given such an algorithm, we have to impose on it a termination detection mechanism, which will signal the nodes of the network that the algorithm has terminated.
Here is one such mechanism for non-FIFO model of computation, suggested by J.M. Helary, C. Jard, N. Plouzeau, and M. Raynal from IRISA France:

Every node $v$ maintains 2 matrices SENT and RECEIVED. Entry $SENT(i, j)$ is the number of messages sent from $i$ to $j$ in the algorithm, to the best of $v$’s knowledge. Entry $RECEIVED(i, j)$ is the number of messages received by $j$ from $i$ in the algorithm, to the best of $v$’s knowledge.

When ever a node, $x$, becomes idle (from the underlying algorithm’s point of view) it floods the network with a message containing 2 vectors: $SENT(x, *)$ and $RECEIVED(*, x)$. Upon receiving a flood message every node updates its matrices accordingly.

Prove under the following assumptions, that if in some node $SENT(i, j) = RECEIVED(i, j)$ for all pairs $(i, j)$, then there is no message of the underlying algorithm in transit on any link of the network.

Assume a non-FIFO asynchronous network. Also assume that, magically all the SENT matrices are initialized to the number of messages sent by the spontaneously starting nodes when they start (i.e., the first messages generated by the nodes which wakeup spontaneously to start the underlying algorithm, hence we assume that all start nodes start at the same time). RECEIVED matrices are initialized to 0.

4. (a) There are $n$ identical finite automata connected by $n$ unidirectional, asynchronous links to form a unidirectional ring, $n > 1$. The same automata works for any size ring. Exactly one automaton (one node) receives a WAKE signal from the environment. Design an algorithm (pseudo code of the finite automata) which informs the predecessor of the awakened automaton that it is the predecessor of the awakened automaton. Show that your algorithm has asymptotically optimal bit complexity (i.e., prove a lower bound on the total number of bits exchanged by any such solution between all the nodes).

(Hint: The lower bound should hold even for arbitrary identical processors in place of finite automata, and variable length messages, even if the processors know $n$. Can you relax these restrictions farther?)

(b) Give a tight upper and lower bound (i.e., pseudo-code and lower-bound) for this problem in the synchronous model.

5. Here we consider the population protocols model discussed in class. Consider a flock of birds, where each bird monitored with an electronic bracelet on its leg. The scientist would like to detect whether the number of birds in the flock is at least $k$. Each bracelet has a finite automata cpu in it and a communication module. The modules interact with each other in pairs: * When two birds meet, the bracelets exchange their full state. * During communication, the birds have a side assigned to them: one bird is defined as Left and the other as Right. The birds know their side.

Every two birds will meet infinitely often, and meetings are uniformly distributed (Each pair of birds has the exact same probability to meet at any given time).

There are $n$ birds, but non of the birds know $n$.  

2
Design an algorithm (a finite automata algorithm) for each bird so that they can detect if there are more than \( k \) birds in the wild for some constant \( k \). In other words: In case of more than \( k \) birds, eventually, by checking any birds bracelet state, we will receive a positive confirmation.

Here is an alternative phrasing that suppose to be equivalent:

Here we consider the population protocols model discussed in class. In that model there are \( n \) agents, each a finite automata. Every pair of two agents is connected by directed link. The execution of an algorithm proceeds by an infinite sequence of interactions. In each interaction the scheduler selects two agents uniformly at random and tell them to "talk" to each other (interact), and the agents know the direction of the link between them. The agents do not know \( n \) the total number of agents. The scheduler schedules the interactions in such a way that each pair interacts infinitely often. In other words, at any point of time during the execution each pair will eventually be scheduled again.

Design an algorithm (a finite automata algorithm) for each agent so they can detect if there are more than \( k \) agents in the network for some constant \( k \).