Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
Juho Hirvonen

July 9, 2018

Network Algorithms, Summer Term 2018
Problem Set 10

hand in by Sunday, July 15, 2018

Exercise 1: Consensus and deterministic message passing

In this exercise we will consider consensus in an asynchronous message passing model. We will see that the impossibility of deterministic consensus does not depend on the specifics of the model in question. Consider the problem of *binary* consensus: each node v has a private input $x_v \in \{0, 1\}$, and the nodes must decide a common output in $\{0, 1\}$ such that at least one of the nodes has that output as its private input.
We will work with an *asynchronous* message passing model. The algorithms are *event based*, that is, react to receiving messages by changing their state and sending messages, and do not act otherwise. There is no shared notion of time and, in particular, the nodes cannot wait until a certain amount of time has passed. The *state* of the system consists of the states of the individual nodes and the messages that the nodes have sent, but that have yet to arrive. At each state, the adversary (called *scheduler*) decides which message arrives next (it must choose a message). The *transition* to the next state is defined by this message and the node to which it arrives. The initial state consists of the inputs and the first messages that the nodes send given those inputs.
A state is *univalent* if, independent of the choices of the scheduler, the system will decide a particular value x. Otherwise the state is *bivalent*. A state is *critical* if it is bivalent, but any choice of the scheduler will lead to a univalent state.
We will consider the case with $n = 2$ nodes. At any moment the scheduler may make one of the nodes crash (a crashed node does not participate in the computation and does not send messages). We show that the consensus problem cannot be solved in this setting.

a) Show that there is an initial state that is bivalent.
b) Show that an algorithm that always eventually stops must have a critical state.
c) Show that a critical state cannot exist.

Hint: use the same ideas as we used in the lecture to prove impossibility of consensus.

Exercise 2: Consensus with edge failures

Consider the synchronous message passing model, where at each round nodes may exchange messages with their neighbors. We only discussed node failures, but we always assumed that edges (links) never fail. Let us now study the opposite case: Assume that all nodes work correctly, but up to f edges may fail.
Analogously to node failures, edges may fail at any point during the execution. We say that a failed edge does not forward any message anymore, and remains failed until the algorithm terminates. Assume that an edge always fails completely, i.e., no message can be exchanged over that edge anymore in either direction.
We assume that the network is initially fully connected, i.e., there is an edge between every pair of nodes. Our goal is to solve consensus in such a way, that all nodes know the decision.

a) What is the smallest f such that consensus might become impossible? (Which edges fail in the worst-case)
b) What is the largest f such that consensus might still be possible? (Which edges fail in the best-case)
c) Assume that you have a setup which guarantees you that the nodes always remain connected, but possibly many edges might fail. A very simple algorithm for consensus is the following: Every node learns the initial value of all nodes, and then decides locally. How much time might this algorithm require?