

CS138 Homework Assignment 4 Solutions

Spring 2009

1. Slide XX-8 showed a solution for a self-stabilizing token ring, in which each of the n nodes has $k \geq n$ states; assume $n > 1$. Since Dijkstra didn't bother proving it correct, it's up to you. We'll define the system to be stable when it's the case that only one node's guard is currently true, and whenever the system changes global state legally (i.e., without being zapped), it goes to a global state in which the next node's guard becomes the only guard whose value is true. Assume that the nodes are numbered such that the distinguished node is 0 and the others are numbered successively from 1 to $n-1$.

a. Explain why it is that at any particular moment, at least one guard must be true, even if the system has been zapped.

Either all nodes have the same state or there are at least two nodes with different states. If the former, then node 0's guard is true. If the latter, then there must be at least two nodes whose states are different from their predecessors. At least one of these nodes is not the distinguished node, and thus its guard is true.

b. Show that if all nodes have the same value for their states, the system is stable.

In this global state, only node 0's guard is true. Once its command is executed, its state becomes one greater (mod k) and its guard is no longer true, but its successor's guard is now true, and no other guards are true. Once this node's command is executed, its state is replaced with its predecessor's state and its guard is no longer true (and no other guards are true). This continues around the ring until we are back to node 0. At this point, all states again have equal values, and thus only node 0's guard is true.

c. Show that if node 0's state is greater than those of all other nodes, the system will necessarily reach a stable global state.

In such a global state, node 0's guard is false, but its successor's guard is true. Regardless of what else happens in the system, this condition will hold until the successor executes its command. At that point, its state becomes equal to that of node 0 and its guard becomes false. The system is now in state in which the next node's guard must be true, and this condition will hold until that node executes its command. When it does, its state becomes that of its predecessor, which is the same as that of node 0. Thus node 0's state will propagate all the way around the ring, until all nodes have the same state. Not until this global state is reached will node 0's guard become true. However, as shown in part b, the system is now in a stable state.

d. Assume now that each node's state value is an unbounded integer (i.e., k is infinite). Show that, regardless of its current state, the system will necessarily reach a global state in which node 0's state is greater than those of all others.

Assume there exists at least one node whose state is greater than or equal to that of node 0. We know from part a that there's always at least one node whose guard is true. Each time a command (associated with a true guard) is executed, either node 0's state gets larger, or one other node's state is set equal to its predecessor's state (that previously was different from it). Let node i be the first node beyond node 0 whose state is different from node 0's. After some number of executions, either node 0's state will increase by 1, or node i will set its state to be the same as node 0's (and thus the index of the first node beyond node 0 whose state is different from 0's increases by at least 1). Thus, in a finite

number of executions, either node 0's state increases by 1 or all nodes have the same state as node 0. Furthermore, the maximum of the nodes' state values does not get larger unless node 0 has the state with the maximum value. Thus, eventually, either node 0's state becomes larger than all others or the system reaches a global state in which all nodes have the same state (and thus after the next command execution, node 0's state is larger than all other's).

e. *Redo part d, this time assuming $k \geq n$.*

We can modify the proof of part d by changing all occurrences of "increase by 1" to "increase by 1 (mod k)". However, we have a problem with the last sentence because it is no longer clear that node 0's state can get larger than all others, since there is now a bound on its size (k-1). But we can safely say instead that, eventually, either node 0's state becomes 0 or the system reaches a global state in which all nodes have the same state. This clearly isn't what we're after. However, starting from this state we can repeat the argument of part d: Again, let node i be the first node beyond node 0 whose state is different from node 0's. After some number of executions, either node 0's state will increase by 1 (mod k), or node i will set its state to be the same as node 0's. In a finite number of executions, either node 0's state increases by 1 (mod k) or all nodes have the same state as node 0. However long this takes, we will call it a *round*. As before, the maximum of the nodes' state values does not get larger unless node 0 has the state with the maximum value. After no more than n rounds (and thus node 0's state has not wrapped around), since we started with node 0's state being 0, either node 0's state is larger than all others or the system has reached a global state in which all nodes have the same state.

f. *Show that the system won't necessarily ever enter a stable state after being zapped if $k < n$.*

Our argument for part e no longer holds, since after n rounds, node 0's state could have wrapped around back to 0. However, there could well be another proof that doesn't depend on this. So, what we need is a counter example. It goes like this. Put the n nodes into initial states in which node 0 is in state 0, node 1 in state k-1, node 2 in state k-2, and node k in state 0. Any nodes beyond node k are in state 0. Thus the guards for nodes 0 through k are all enabled. Each of these nodes simultaneously executes its command, which results in a right shift of their values, with node 0 entering state 1. The nodes beyond k (if any) then, one at a time shift node k's value to their successors (clockwise) until k's value is propagated to all of them. Then the entire procedure is repeated. After k iterations of this, we are back in the state in which we started, and thus it can repeat ad infinitum without ever entering a stable state.