EDA387: Computer Networks - Lab 2.4

Mats Högberg

October 8, 2019

Self-stabilizing maximum matching on a directed ring with a distinguished node

Here's the pseudo code for a self-stabilizing algorithm for maximum mathing in the given network:

```
01 p_0: do forever:
02   write r_i := 0
03 end

04 p_i, i ≠ 0: do forever:
05   lr_{i-1} := read r_{i-1}
06   if lr_{i-1} = 0:
07     write r_i := 1
08   else:
09     write r_i := 0
10   end
11 end
```

The matching $\hat{E}$ given a configuration $c = \{r_0, \ldots, r_{n-1}\}$ in the above algorithm is defined as $\hat{E} = \{(p_i, p_{i+1}) \mid r_i = 1\}$. For this task, the set of legal executions are all executions in which, in every configuration, $\hat{E}$ does not contain any edges with common endpoints. A configuration is safe with regard to the set of legal executions and the above algorithm if $\forall i : r_i = i \bmod 2$. For example, for $n = 5$ we would have $c_{\text{safe}} = \{0, 1, 0, 1, 0\}$ and for $n = 6$ we would have $c_{\text{safe}} = \{0, 1, 0, 1, 0, 1\}$. If a system is in such a configuration, no two neighbouring processes both have the value 1 stored in their registers, which means that $\hat{E}$ does not contain any edges with common endpoints. Also, in such a configuration, no register will ever change its value. c_{safe} is therefore a safe configuration with regard to the legal executions and the above algorithm.

In fact, c_{safe} is also an optimal solution, as $\hat{E}$ would contain $n/2$ edges in a system with an even number of processors, and $(n-1)/2$ edges in a system with an odd number of processors. In a ring network, no better solution than this can exist.

Let's assume that that we're dealing with an asynchronous system and that no guarantees can be made for which line in the program a processor starts its execution on. To prove that the algorithm always reaches c_{safe}, regardless of the initial configuration of the system, we will use a proof by induction. For the base case, we can clearly see that after 1 asynchronous round we will have $r_0 = 0$, and that this will hold for all succeeding rounds. For the induction step, assume that the values of all registers $r_0, \ldots, r_i$ are equal to their desired values in c_{safe} for some $0 < i < n$. If $i \bmod 2 = 0$, we will have $r_i = 0$, so p_{i+1} will set $r_{i+1} = 1$ within 3 asynchronous rounds. If $i \bmod 2 = 1$, we will have $r_i = 1$, so p_{i+1} will set $r_{i+1} = 0$, also within 3 asynchronous rounds. In both cases, r_{i+1} will be set to its desired value in c_{safe}, and after this has happened

the value of r_{i+1} will never change since the value of r_i will never change. Thus, by mathematical induction, we have proved that the algorithm always reaches c_{safe}, which is a safe and optimal configuration, regardless of the initial configuration of the system.

The algorithm needs 2 states for each processor: one for when $r_i = 0$ and one for when $r_i = 1$.