EDA387 - Lab 4.2: Self-stabilizing maximum matching

Jinsong LI, Nathan HAUDOT

September 26, 2021

1 Processors' state (three-state machine)

In a directed ring context (Dijkstra's assumption), each p_i processor can only read shared variables of the p_{i-1} previous processor to him. All processors will have a single register that we will note r_i. The p_i processors ($i \in [1, n]$, without the p_0 root processor) can have two register states:

- $r(i) = 1$: p_i processor matches the p_{i+1} processor
- $r(i) = -1$: p_i processor matches the p_{i-1} processor

The only way to efficiently detect that there is an odd number of processors in the ring is to take advantage of the fact that the root processor p_0 has a different program from the others: it is the one that will understand if there is an even or odd number of processors. Here are its two possible states:

- $r(0) = -1$: p_0 (root) processor matches p_n (last processor in the ring)
- $r(0) = 0$: p_0 (root) processor does not match with any processor

2 Proposed algorithm

Proposed algorithm for the p_0 root processor:

while true **do**
    if $r(i-1) = -1$ **then**
        $r(0) \leftarrow 0$
    else
        $r(0) \leftarrow -1$
    end if
end while

Proposed algorithm for the p_i processors ($i \in [1, n]$)

while true **do**
    if $r(i-1) = -1$ **then**
        $r(i) \leftarrow 1$
    else
        $r(i) \leftarrow -1$
    end if
end while

3 Set of legal executions

Therefore, the set of legal executions of this algorithm is that each execution converges the processors of the ring to an alternate sequence of 1s and -1s, except for the root processor which is labelled 0, or may see its register change to -1 if the ring has an even number of processors.

4 Proof of correctness

A variant function $VF(c)$ that returns a vector $(m+s, w, f, c)$ is defined in order to prove the correctness of the algorithm where m, s, w, f and c are the total number of matched, single, waiting, free and chaining processors respectively [Dol00]. In this question, $VF(c) = (n, 0, 0, 0)$ refers to a safe configuration, which means all the processors are either matched or single, so that the optimal matching is completed.

For the program for p_0, when the assignment in line 3 is executed, p_0 changes the state from waiting to free or single. So the value of the variant function doesn't change. However, if the assignment in line 5 happens, the assignment in line 3 of p_1's program will be executed in the next step. Finally, these 2 processors are matched. Thus, the value of the variant function is increased.

During the following steps, the assignment in line 3 and 5 of the program for p_i processors ($i \in [1, n]$) will be executed in pairs, until p_{n-1} executes its program. Once a pair of assignments are executed, a pair of processors are matched, and the value of the variant function either increased or remained unchanged.

When the last processor p_{n-1} executes its program, it's either matched with the previous processor p_{n-2}, or matched with the root p_0 since p_0 is waiting at this moment. Similarly, the value of the variant function either increased or remained unchanged.

So we can conclude that, the value of the variant function $VF(c)$ will converge to $(n, 0, 0, 0)$, the system will finally reach the safe configuration, and the maximal matching will be achieved.

5 Self-stabilizing

This algorithm is self-stabilizing, we have shown above that the variance function converges to $(n, 0, 0, 0)$.

It is also fault-tolerant, if one of the p_i processors has an error in its register, the problem will propagate to the next neighbor, the algorithm will then have to re-execute itself on the whole ring to correct the problem. The root processor may eventually be "fooled" (it depends on whether the faulty register changes the pattern of matches in the ring), so the algorithm will have to run over the whole ring again to finally converge to a stable state.

References

[Dol00] Shlomi Dolev. *Self-Stabilization.* MIT Press, 2000.