EDA387: Computer Networks - Lab 2.3
Self-Stabilizing Maximum Matching

Group 5: Haitham Babbili and Olalekan Peter Adare

4th October 2020

Assumptions

- The number of processors is **n**, labelled from P_0 to P_{n-1}, and are all connected in a ring.
- All processors are activated infinitely often.
- The directed ring represents a unidirectional communication among the processors, in a clockwise manner starting from the root processor, P_0. The root processor, P_0, has a fixed value in its register that does not change. This is built on the premise that it is running a different program from all the other processors.
- The execution starts from processor P_0 to P_{n-1}. Also, at any given time, only a single processor is executing a function at a time.i.e. it is an asynchronous shared memory system.
- Each processor has its register separated into two parts, $l.r_{i-1}$, where it can write the value from the previous processor, and r_i, where we it can read its value from. A processor can only read from its previous neighbour on the ring.
- The algorithm is treated as a vertex coloring case, where adjacent nodes connected by an edge cannot have the same color number.
- Matching only occurs when the color value in previous processor, P_{i-1}, is **0**. This translates to the condition that the color value in P_i is **1**, while the color value in P_{i-1} is **0**.

Pseudo-code

---

if $p_i := p_0$ **then**
    do forever
    $l.r_{n-1}$:=**read**(r_{n-1})
    if $l.r_{n-1} := 0$ **then**
        write $r_0 := 2$
    else
        write $r_0 := 2$
    end if
    end
else $p_i :\neq p_0$
    do forever
    $l.r_{i-1} :=$ **read**(r_{i-1})
    if $l.r_{i-1}$=0 **then**
        write $r_i = \mathbf{1}$
    else $l.r_{i-1} > 0$
        write $r_i = \mathbf{0}$
    end if
    if $r_i mod_2$=0 **then**
        r_i match with r_{i+1}
    else
        r_i match with r_{i-1}
    end if
end if

Algorithm

Building on the fact that the root processor, P_0, is running a different program, it is taken that it will not be involved in any matching process on the ring. However, the coloring process of the ring starts from P_0, and it has a color number **2**, which will never change irrespective of the changes on the ring. Intuitively, the color number of P_0, is needed to be set in this manner to always have a correct solution for vertex coloring. Also, P_0 is different, program-wise, when compared to the other processors, P_i.
Going forward, in the matching process, then P_0 is not pointing to either of its neighbours. Since a ring network has a maximum degree of two (2), then the two neighbours of P_0 will not form any matching with it. This will in turn reduce the number of the maximum matching the ring network, or graph, can offer.
Furthermore, this is treated as a vertex coloring case, where no two neighbouring processors on the ring have the same color number. All processors, P_i have a shared register, r_i , where every processor can read the value of the previous register, r_{i-1}, and then determine its own color number to be either 0 or 1. The color number of the P_0 will never change. Since P_0 will also not match with any other processor, P_i, in the ring, this then gives us the ability to use only 2 other color numbers, labelled as 0,1, to achieve the coloring of the graph, irrespective of the size, or diameter, or the number of processors (vertices), on the ring.
Matching occurs only when the color number of the previous processor is **0**. Here, this implies that the color number of P_{i-1} in r_{i-1}, is 0 and color number of P_i in r_i, is 1. This can simply be explained that if the color number of a processor is 1, it can only match with its previous processor, if its color number is 0.
Intuitively, two (2) rounds are necessary for this algorithm to self-stabilize. The first round is the coloring of the processors, while the second round is to determine the maximum matching number for the ring. Therefore, for efficient implementation of maximum matching in this case, we take the possibility of two (2) states in the finite machine: Matched or unmatched. Therefore, the time complexity of the algorithm will be reduced.

Proofs

Theorem 1.1: In a ring network, the number of vertices (processors) is equal to the number of edges. This means that for a ring network the number of processors is equal to the number of neighbour-to-neighbour connections on the ring.

Theorem 1.2: If **N** vertices (processors) form a ring connection, a maximum of three (3) colors are needed for coloring the corresponding graph.
This implies that the highest chromatic number we can use, is 3. For even number vertices (processors) on a ring, only 2 colors are actually needed for coloring the graph, while 3 colors are needed for coloring an odd number vertices (processors) connected in a ring.

Lemma 1.1: A ring network that consists of even number processors, **n**, will have its corresponding maximum matching to be $\frac{n}{2}$ (Note: a ring network has a maximum degree of 2).

Lemma 1.2: A ring network that consists of odd number processors, **n**, will have its corresponding maximum matching to be $\frac{n-1}{2}$.

Lemma 1.3: A ring network that consists of even number processors, **n**, with a special processor P_0 that has a fixed color number, will have the corresponding maximum matching to be $\frac{n-2}{2}$, and $\frac{n-1}{2}$ otherwise, i.e if **n** is an odd number.

Safe Configuration

We say that the configuration **c** is safe, c_{safe}, with regard to a legal execution, if for every fair execution of the algorithm, that starts from an arbitrary configuration belongs eventually to the legal execution. In vertex coloring of a ring network, the color of a processor depends only of its own state and that of its predecessor. In simple words, configuration $c = \{r_0,r_1,r_2,...,r_{n-1}\}$ will reach a safe configuration state when every two neighbouring processors, P_i and P_{i-1}, do not have the same colour number in their register, r_i and r_{i-1}, respectively. This is built on the constraint that $r_0 > 0$ and $r_i < 2$. This offers the possibility of chromatic number 3, at most, with color numbers {0,1,2}. A safe configuration for the coloring will always be reached, with a special processor, P_0, having fixed color number **2** and sequential coloring of the graph from P_1 to P_{n-1}, using color numbers **0** and **1**, repeatedly. Any transient fault will simply result in a new round to re-allocate the colors, which will still give a safe configuration eventually.
Then, for determining the maximum matching, a safe configuration is always reached because a sequentially numbered vertices (processors) in a ring, starting from index number 1, ends up with an even-and-odd pattern of arrangement. This means that numbering nodes on a ring will be a continuous loop of color numbers **0** and **1**, which will lead back to the special processor, with color number **2**. The imposed condition for matching is that the color value in P_{i-1}, the previous processor, should be always be **0**, then this satisfies the matching process . This follows that we will end up with color numbers arranged as {2,0,1,0,1,0,1,...}, {2,0,1,0,1,...}, for every round.
Based **Theorem 1**, it is easier to see by deduction that, for a ring network, the number of vertices is equal to the number of edges, and matching will occur between only two neighbouring processors in one degree, or direction, only, where P_i and P_j are pointing to each other respectively. This leaves the other degree of the connection in a state that will never match. Hence, for an even number of processors on a ring network, the maximum matching is half of the **n** processors, or edges, implying $\frac{n}{2}$. This means $P_i mod 2$ is zero, which agrees with Lemma 1.1, when n is even.
On the other hand, for an odd number of processors on a ring network,$P_i \ mod_2$ is 1, which implies there will always be a processor, P_{n-1}, that will not match with any other processor. This gives a bound to the maximum matching value of such a ring network to be $\frac{n-1}{2}$, and $\frac{n-2}{2}$ otherwise. Therefore, Lemma 1.2 and Lemma 1.3 are also correct. This is represented in the figure below:

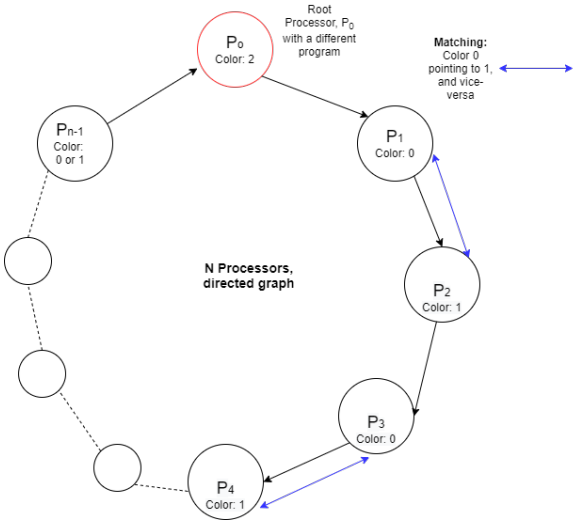

Figure 1: Matching of N processors in a ring

Convergence

Given ring network of **n** processors with a special processor P_0, that will always have in its own register the color value **2**. This P_0 is connected to other processors P_i (P_1 to P_{n-1}), which can have in there registers **0 or 1**. Then, the maximum matching that will be gotten, after arriving to self-stabilizing state, will take a maximum of two (2) rounds.
To prove this, let us start with $\mathbf{n} = 5$. The execution processes will start from P_0, who will take the colour number **2**, and then P_1, will read the previous processor, P_0, and save it in $l.r_0$. After comparing $r_1 = \mathbf{0}$ and in 5 asynchronous executions, the congratulation will be **c**={2,0,1,0,1}, and the value in the register will be fixed so that $r_0 \neq r_i$ even if there is a change in the ring, or in the number of processors. The maximum matching process will start depending on our assumption that **0** will match with **1** and we will have the maximal number of matching to be **2** and maximum number of matching to be $\frac{n-1}{2}$ =2.
In a similar way, if $\mathbf{n} = 8$, after 8 asynchronous cycles, the self-stabilizing configurations c will be c={2,0,1,0,1,0,1,0} and the colour values will be fixed, so $r_0 \neq r_i$, even if there is a change in the ring, or in the number of processors. Then, the maximum matching is be determined based on our assumption that **0** will match with **1**. This gives us the maximal number of matching to be **2** and maximum number of matching to be $\frac{n-2}{2}$ =3.
In the case of **n** number of processors with execution E from P_k, $l.r_{k-1}$ will read r_{k-1}, then r_k will decide to take **0** or **1** based on the value in $l.r_{k-1}$. After P_k finishes its round, the next processor P_{k+1} is activated and it will do the same. The process will continue until P_{n-1}, then next execution will start from P_0, which will still take color number **2**. After P_0, P_1 will be activated, and after n asynchronous cycles the congratulation will be c_{safe}, c={2,0,1,0,1,...} and the colour number will be fixed, so that $r_0 \neq r_i$, even if there is change in the ring, or in the number of processors, which will still result in a self-stabilizing configuration. After this, the matching process will start and maximal number of matching is **2** but the maximum number of matching depends on number of **n**, if it is even or odd.

Closure

P_0 is the special processor and it takes colour number **2**, while P_i processors in the ring will be activated. After P_0 finishes its cycle, where P_i will read the value in $l.r_{i-1}$ then write in its own register the opposite number. After **n** asynchronous cycles and maximum 2 executions of E, the system will arrive at a safe configuration, which means the system is self-stabilized, and it is in the legal execution stage. After that, the matching process will start by matching every processor with color number **1** to the previous processor which has in own its register color number **0**. Hence, the maximal number of matching is **2**, while maximum number of matching is $\frac{n-2}{2}$, if **n** is even and two processors are single (P_0, P_{n-1}), or maximum number of matching is $\frac{n-1}{2}$ if **n** is odd and one processor is in the single state. The variant function vector, after a maximum of 2 executions, will therefore be $VF(c) = \{n,0,0,0\}$.

Optimality of the Algorithm

Therefore, to have a deterministic self-stabilizing algorithm, we must create a synchronous system. This leads to a bounded asynchronous self-stabilizing algorithm. With two executions only (coloring and maximum matching), the algorithm offers a legal execution and self-stabilizes. This means that the time complexity is based on **n** number of asynchronous cycles, in each round, of the ring. This gives us a bound in the order of $\mathbf{O}(n^2)$. Although, it may still be a little less since the executions of the special processor, P_0, is predetermined that its state never changes with time and it will never match with any of its neighbours.

Number of States

The algorithm only requires two (2) possible states in its finite state machine. They are **matched state** and **unmatched state**.So the total number of states is in the order of $\mathbf{O}(n^2)$.

Conclusion

Based on all the clearly stated proofs above, we can conclude that starting this algorithm from an arbitrary position, it will always reach a self-stabilizing state. The maximal matching number is **2**, and the maximum matching number depends on **n**, the number of processors, which can be even or odd.