Algorithm Lab for Computer Networks (EDA387)

Elad Michael Schiller

Please submit your typed solutions in pairs via canvas. In order to submit the assignment, please join one of the lab groups with your lab partner.

1 Value discovery in complete graphs (due: September 14, 2021)

Consider n processors, such that each processor p_i is associated with two registers r_i and s_i (both of constant size that is independent of n). Processor p_i cannot read the value in s_i, however, any other processor can! Processor p_i has both read and write access rights to r_i and any other processor has only read-only access rights to r_i. The value in s_i is unknown to p_i. The other processors can help p_i to discover this unknown value. For example, suppose that $n = 2$. Processor $p_{(i+1) \mod 2}$, can write the value of s_i to $r_{(i+1) \mod 2}$ and then p_i can discover s_i's value by reading $r_{(i+1) \mod 2}$'s value. Please solve the problem, which is to let p_i discover the secret s_i, for the case in which $n > 2$ is any finite known value. Is there a solution when there are no processor identifiers? Prove your claims.

The answer idea

Note that for the case of $n = 3$, and $r_0 := s_1 \oplus s_2$, $r_1 := s_0 \oplus s_2$, $r_2 := s_0 \oplus s_1$, we can say that p_0 can read r_1 and r_2 as well as s_1 and s_2. Therefore, $r_1 \oplus s_2 = s_0 \oplus s_2 \oplus s_2 = s_0$. The idea is to study the general case, and $r_0 := s_1 \oplus \ldots \oplus s_{n-1}$, $r_1 := s_0 \oplus s_2 \oplus \ldots \oplus s_{n-1}$, $\ldots$, $r_{n-1} := s_0 \oplus \ldots \oplus s_{n-2}$, we have that $r_1 \oplus s_1 \oplus \ldots \oplus s_{n-1} = s_0$. In general, $s_i = r_j \oplus \left(\bigoplus_{j \neq i,j} s_k \right)$, where the processor ordering is local to p_i.

2 Self-stabilizing maximum matching (due: September 23, 2021)

Let $p_0, \ldots, p_{n-1}$ be n processors on a directed ring that Dijkstra considered in his self-stabilizing token circulation algorithm (in which processors are semi-uniform, i.e., there is one distinguished processor, p_0, that runs a different program than all other processors but processors have no unique identifiers). Recall that in Dijkstra's directed ring, each

processor p_i can only read from p_{i-1} mod n's shared variables and each processor can use shared variables of constant size. Design a deterministic self-stabilizing matching algorithm that always provides an optimal solution. Please give a clearly written pseudo-code, define the set of legal executions, provide a correctness proof with all the arguments needed to convince the reader that the algorithm is correct and self-stabilizing. Show that your algorithm is always optimal after convergence. What is the number of states (in the finite state machine that represents each processor) that your algorithm needs? Prove your claims.

The answer idea

The correctness proof of Algorithm 1 shows by induction on the value of n that the value of x_0 is 0 for the case of n is even and 1 otherwise. The rest of the proof is implied but the definition of the function *output*().

Algorithm 1: Self-stabilizing maximum matching

```
1  shared variables: x ∈ {0,1,2} stores a color
2  Code for processor p_i : i = 0
3  x_i ← x_{n-1} + 2 mod 3
4  Code for processor p_i : i ≠ 0
5  x_i ← x_{i-1} + 1 mod 2

6  function output() begin
7      switch x_i do
8          case x_i = 0 do print 'p_i and p_{i+1 mod n} are matched';
9          case x_i = 1 do print 'p_i and p_{i-1 mod n} are matched';
10         case x_i = 2 do print 'p_i is single';
```

3 Self-stabilizing α-maximal-partitioning (due: September 30, 2021)

Consider a network, $T := (P, E)$, whose topology is of a tree. The nodes in the network are anonymous, i.e., they have no globally unique identifiers and they are run the same algorithm. Let $\alpha > \Delta$ be a positive integer, where Δ is an upper bound on the node degree. Let $P_1, P_2, \ldots, P_k$ be a partitioning of P, such that $\bigcup_x P_x = P$ as well as each $P_x \neq \emptyset, |P_x| \leq \alpha$, and $T_x := (P_x, E_x)$ is a (connected) tree, where $x \in \{1, \ldots, k\}$ and $E_x \subseteq E \cap (P_x \times P_x)$. In this case, we say that $P_1, P_2, \ldots, P_k$ is an α-partitioning of T. Suppose that there are no $x, y \in \{1, \ldots, k\} : x \neq y$, such that $P_1, P_2, \ldots, (P_x \cup P_y), \ldots, P_k$ is also an α-partitioning of T. In this case, we say that $P_1, P_2, \ldots, P_k$ is an α-maximal-partitioning of T. Your task is to design a self-stabilizing algorithm for constructing an α-maximal-partitioning of T.

Please provide clear and exact pseudo-code for the entire protocol that you propose. Please prove the correctness of your proposed solution. That is, show that when starting from any system state (configuration), the algorithm constructs an α-maximal-partitioning of T.

Hint: start by assuming that the network includes a distinguished processor, i.e., the network is semi-anonymous. Then, use a self-stabilizing algorithm for finding the center of a tree, such as the one by Datta, Larmore, and Masuzawa [1]. If the center includes a single processor, the task is done. If the center includes two processors, each processor is a the root of a tree. Solve the problem for each of these trees via a divide and conquer approach.

Comment: an earlier version of this question considered the case in which $|P_x| > \alpha$ rather than $|P_x| \leq \alpha$ in the definition of α-maximal-partitioning. If you prefer, you can consider the earlier version of the question, as long as you clearly this mention this in your answer.

4 Self-stabilizing minimum spanning tree (due: October 7, 2021)

Consider a computer network whose topology is of a general graph, $G := (P, E)$ and every processor has a unique global identifier, where $E \subseteq P \times P$ is a set of weighted edges, such that each connect a pair of nodes in P. Suppose G includes a set, $T \subseteq E$, of selected edges, such that T defines a spanning tree for which there is no other tree, $T' \subseteq E$, where the sum of weights for T' is smaller than T. In this case, we say T is a minimum spanning tree in E. The studied problem is to construct a minimum spanning tree (MST) for graph G. Your task is to provide, your own, self-stabilizing solution to the problem.

- In case you provide an original algorithm (or parts of your solution use an original algorithm), please provide the code of the algorithm. In case your solution uses algorithms that we learned in class (or parts of your solution use algorithms that we learned in class), please specify the exact names of these algorithms as well as the page numbers (or sub-chapter) in the book of these algorithms. In case your solution stacks several such building blocks, please explain how you put them all to work together. There is no need to copy code from the book, slides or any other sources.

- Please prove that when starting from any system state (configuration), the algorithm constructs a minimum spanning tree.

- Please state exactly the asymptotic cost of memory for your solution. Specifically, what is the maximum about of memory that any node would need? What is the largest message size that any node will send? Also, what is the maximum number of messages until the MST is guaranteed to be constructed? What is the highest number of communication rounds it would take to construct the MST? What is the stabilization time?

The following assumptions might simplify your solution. Assume that the network has a single distinguished node. Also, assume that the communication is via shared memory as well as that writing and reading to it is within a constant time.

An extra question: can be solved instead of any of the questions above

Consider a computer network that its topology is of a general graph. Assume that the network includes a single distinguished node. Also, the system is synchronous. That is, upon a pulse, all processors invoke a procedure simultaneously. These pulses mark the start of a new synchronous round.

This question considers the problem of self-stabilizing simultaneous set-and-go. The definition of the simultaneous set-and-go problem assumes that at any given round a processor may receive an external "set" signal, which is considered a request for the processors to simultaneously output "go." The problem requires that: (a) if some processor fires in round r, all processors fire simultaneously in round r; (b) if a processor receives a go input in round r', it will fire at some later round $r > r'$, and (c) a processor fires in round r only if some processor received a go input in some round $r' < r$. It is also disallowed for a single go input to induce multiple firing events.

Your task is to design a self-stabilizing algorithm for solving the problem of the simultaneous set-and-go. For the sake of a simpler presentation, assume that shared memory model. Your proposed solution should include a pseudo-code for any part that does not use a solution that was learned during the course. Please write a proof that shows recovery after the last occurrence of a transient fault. How long is the recovery period? Please write a proof that shows after the recovery period, the system satisfies the problem requirements.

References

[1] Ajoy Kumar Datta, Lawrence L. Larmore, and Toshimitsu Masuzawa. Constant space self-stabilizing center finding in anonymous tree networks. In *ICDCN*, pages 38:1–38:10. ACM, 2015.

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

[3] Shlomi Dolev, Amos Israeli, and Shlomo Moran. Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Comput.*, 7(1):3–16, 1993.