Computer Networks

EDA387/DIT663

Fault-tolerant Algorithms for Computer Networks

Spanning tree construction (Ch.2)

Goal

- We would like to understand how to design self-stabilizing network protocols
- At the end of these three lectures and after the home assignments you should be able to:
  - Define network tasks
    - leader election, token circulation, spanning tree construction (BFS) and network topology update (routing)
  - Propose methods for solving such tasks
    - with an emphasis on self-stabilizing methods
  - Argue about the correctness of their proposals

Today

- Quick review of the system settings
- Showing the solution for spanning tree construction
- Spending some time on the review questions

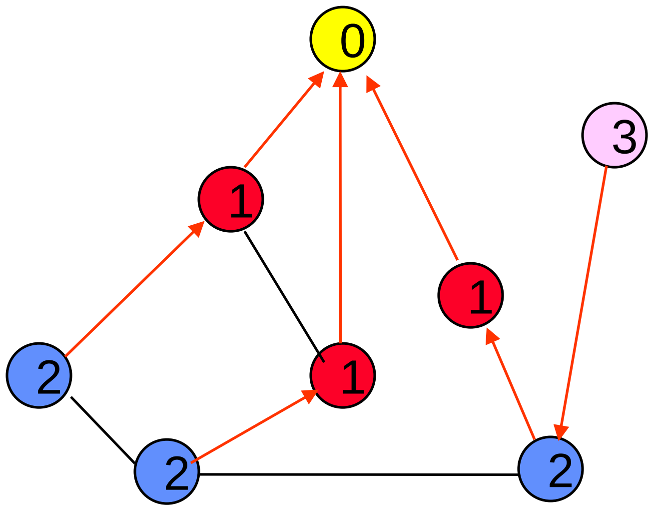
BFS Construction: Background

- Breadth-first search (BFS) is an algorithm for traversing a graph.
- It starts at the tree root (or a general graph) and explores the neighboring nodes first, before moving to the next level neighbors.
- The task here is to construct a directed tree, i.e., a subgraph that its edges are directed towards the root.
- Moreover, the directed path from every node to the root is the shortest path on the graph.

BFS Construction: Motivation

- Routing Information Protocol (RIP)
- Based on the distance-vector routing protocol
- The protocl’s algorithm can be seen as an extension to the BFS construction algorithm (in which every router is the root of the BFS tree).

Spanning-Tree Construction



The root writes 0 to all it's neighbors

The rest – each processor chooses the minimal distance of it's neighbors, adds one and updates it's neighbors

Spanning-Tree Algorithm for P_i

Root: **do** forever
    for $m := 1$ to δ **do write** $r_{im} := \langle 0,0 \rangle$
  od
Other: **do** forever
      for $m := 1$ to δ **do** $lr_{mi} :=$ **read**(r_{mi})
      FirstFound := *false*
      $dist := 1 + min\{lr_{mi}.dis \mid 1 \leq m \leq \delta\}$
      for $m := 1$ to δ
      do
        if not *FirstFound* **and** $lr_{mi}.dis = dist - 1$
          write $r_{im} := \langle 1, dist \rangle$
          FirstFound := *true*
        else
          write $r_{im} := \langle 0, dist \rangle$
      od
  od

δ= # of processor's neighbors
i= the writing processor
m= for whom the data is written

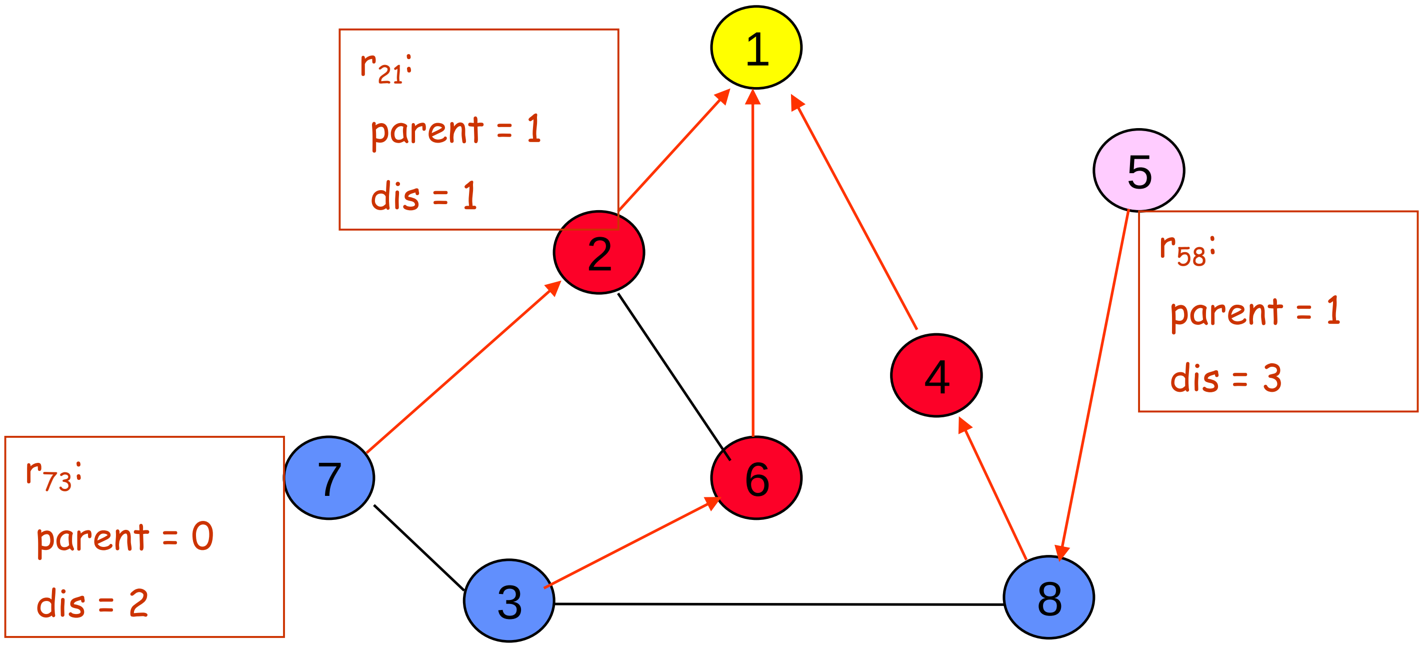
lr_{ji} (local register ji) the last value of r_{ji} read by P_i

Spanning-Tree, System and code

- Demonstrates the use of our definitions and requirements
- <u>The system</u> –
  - We will use the shared memory model for this example
  - The system consists n processors
  - A processor P_i communicates with its neighbor P_j by writing in the communication register r_{ij} and reading from r_{ji}

Spanning-Tree, System and code

- The output tree is encoded by means of the registers



See a java app demonstration

http://www.cs.bgu.ac.il/~dolev/book/chapter2/SpanningTreeGo.htm

Spanning-Tree, is Self-Stabilizing

- The legal task ST - every configuration encodes a first BFS tree of the communication graph
- <u>Definitions</u> :
  - A floating distance in configuration c is a value in $r_{ij}.dis$ that is smaller than the distance of P_i from the root
  - The smallest floating distance in configuration c is the smallest value among the floating distances
  - Δ is the maximum number of links adjacent to a processor

Spanning-Tree, is Self-Stabilizing

- For every k > 0 and for every configuration that follows $\Delta + 4k\Delta$ rounds, it holds that: (Lemma 2.1)
  - If there exists a floating distance, then the value of the smallest floating distance is at least k
  - The value in the registers of every processor that is within distance k from the root is equal to its distance from the root

- Note that once a value in the register of every processor is equal to it's distance from the root, a processor p_i chooses its parent to be the parent in the first BFS tree, this implies that :

- The algorithm presented is Self-Stabilizing for ST

Proof

- Note that in every 2Δ successive rounds, each processor reads the registers of all its neighbors and writes to each of its registers. We prove the lemma by induction over k.

Base Case: Proof for k=1

Distances stored in the registers and internal variables are non-negative; thus the value of the smallest floating distance is at least 0 in the first configuration.

During the first 2Δ rounds, each non-root processor p_i, computes the value of the variable dist (line 7).

The result of each such computation must be greater than or equal to 1.

Let C2 be the configuration reached following the first computation of the value of dist by each processor.

Base Case: Proof for k=1

Each non-root processor writes to each of its registers the computed value of dist during the 2Δ rounds that follow c_2.

Thus, in every configuration that follows the first 4Δ rounds there is no non-root processor with value 0 in its registers. The above proves assertion 1.

To prove assertion 2, note that the root repeatedly writes the distance 0 to its registers in every Δ rounds.

Let c be the configuration reached after these Δ rounds.

Base Case: Proof for k=1

- Each processor reads the registers of the root and *then* writes to its own registers during the 4Δ rounds that follow c_1.
- In this write operation the processor assigns 1 to its own registers.
- Any further read of the root registers returns the value 0; therefore, the value of the registers of each neighbor of the root is 1 following the first $\Delta + 4\Delta$ rounds.
- Thus, assertion 2 holds as well.

Induction Step

We assume correctness for $k \geq 0$ and prove for $k + 1$.

Let $m \geq k$ be the smallest floating distance in the configuration c_{4k} that follows the first $\Delta + 4k\Delta$ rounds.

During the 4Δ rounds that follow c_{4k}, each processor that reads m and chooses m as the smallest value assigns $m + 1$ to its distance and writes this value.

Therefore, the smallest floating distance value is $m + 1$ in the configuration $c_{4(k+1)}$.

This proves assertion 1.

Induction Step

Since the smallest floating distance is $m \geq k$, it is clear that each processor reads the distance of a neighboring processor of distance k and assigns $k + 1$ to its distance.

(By the algorithm, it will not choose a neighbour that has a distance larger than k and by the assumption that k is the shortest distance of a neighbour the algorithm cannot choose a neighbour with shorter distance.) ■

Proof, cont.

Note that once the value in the registers of every processor is equal to its distance from the root, a processor p_i chooses its parent to be the parent in the first BFS tree — p_i, chooses the first neighbor according to its internal link ordering, with distance smaller than its own.

Corollary 2.1: The algorithm presented above is self-stabilizing for *ST*.

Fair Composition

... and mutual exclusion for general communication graphs

Fair Composition -Some definitions

- The idea – composing self-stabilizing algorithms $AL_1, \ldots, AL_k$ so that the stabilized behavior of $AL_1, AL_2, \ldots, AL_i$ is used by AL_{i+1}
- AL_{i+1} cannot detect whether the algorithms have stabilized, but it is executed as if they have done so

The technique is described for k=2 :

- Two simple algorithms server & client are combined to obtain a more complex algorithm
- The server algorithm ensures that some properties will hold to be used by the client algorithm

Fair Composition -more definitions

- Assume the server algorithm AL_1 is for a task defined by a set of legal execution T_1, and the client algorithm AL_2 for T_2
- Let A_i be the state set of P_i in AL_1 and $S_i = A_i \times B_i$ the state set of P_i in AL_2, where whenever P_i executes AL_2, it modifies only the B_i components of $A_i \times B_i$
- For a configuration $c \in S_1 \times \ldots \times S_n$, define the A-projection of c as the configuration $(ap_1, \ldots, ap_n) \in A_1 \times \ldots \times A_n$
- The A-projection of an execution - consist of the A-projection of every configuration of the execution

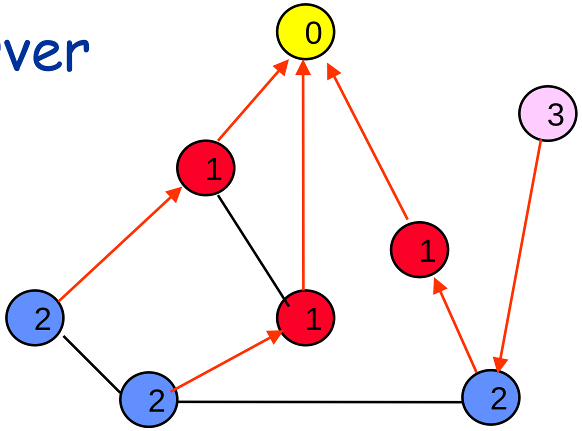
Fair Composition -more definitions …

- AL_2 is self-stabilizing for task T_2 given task T_1 if any fair execution of AL_2 that has an A-projection in T_1 has a suffix in T_2
- AL is a fair composition of AL_1 and AL_2 if, in AL, every processor execute steps of AL_1 and AL_2 alternately
- Assume AL_2 is self-stabilizing for task T_2 given task T_1. If AL_1 is self-stabilizing for T_1, then the fair composition of AL_1 and AL_2 is self-stabilizing for T_2 (Theorem 2.2)

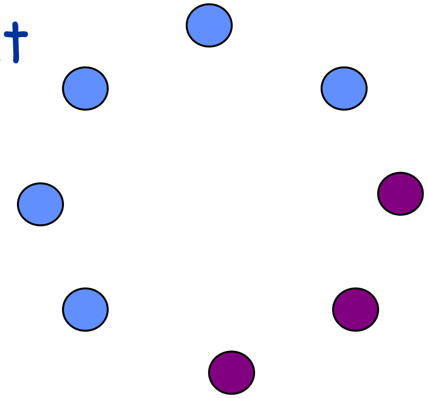
Example : Mutual exclusion for general communication graphs

- Will demonstrate the power of fair composition method
- What will we do ? Compose the spanning-tree construction algorithm with the mutual exclusion algorithm

Server

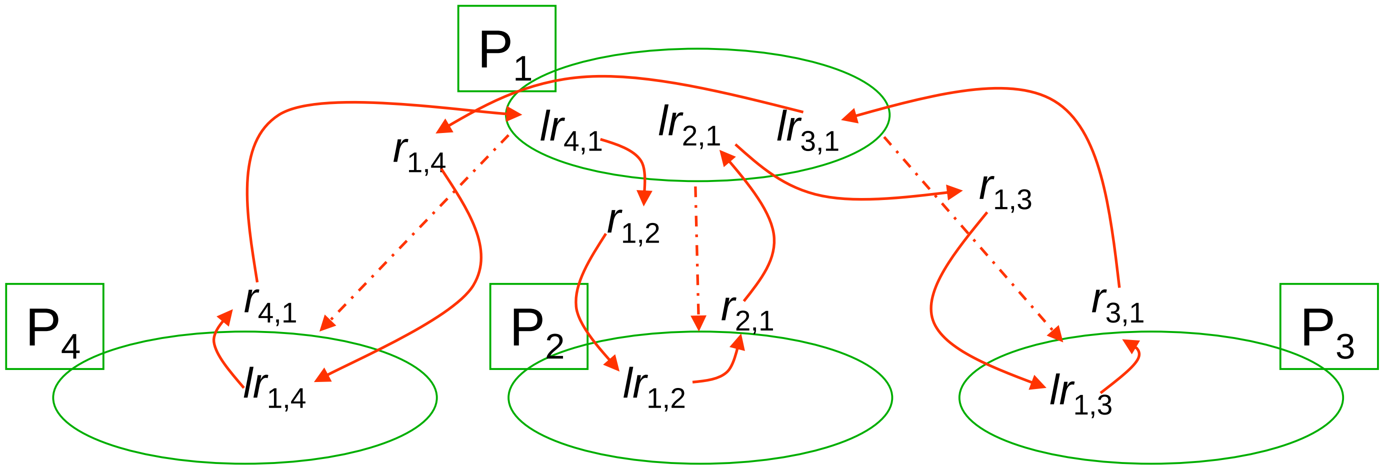


Client



Modified version of the mutual exclusion algorithm

- Designed to stabilize in a system in which a rooted spanning tree exists and in which only read/write atomicity is assumed
- Euler tour defines a virtual ring



Mutual exclusion for tree structure (for P_i)

Root: **do** forever
$lr_{1,i} :=$ **read** $(r_{1,i})$
if $lr_{\delta,i} = r_{i,1}$ **then**
(* critical section*)
write $r_{i,2} := (lr_{1,i} + 1 \bmod (4n - 5))$
for $m := 2$ to δ **do**
$lr_{m,i} =$ **read**$(r_{m,i})$
write $r_{i,m+1} := lr_{m,i}$
od
od
Other: **do** forever
$lr_{1,i} :=$ **read** $(r_{1,i})$
if $lr_{1,i} \neq r_{i,2}$
(* critical section*)
write $r_{i,2} := lr_{1,i}$
for $m := 2$ to δ **do**
$lr_{m,i} :=$ read $(r_{m,i})$
write $r_{i,m+1} := lr_{m,i}$
od
od

mutual exclusion for communication graphs

- Mutual-exclusion can be applied to a spanning tree of the system using the ring defined by the Euler tour
- Note – a parent field in the spanning-tree defines the parent of each processor in the tree
- When the server reaches the stabilizing state, the mutual-exclusion algorithm is in an arbitrary state, from there converges to reach the safe configuration

Summary

- Revised our system settings
  - Added message passing
  - Added asynchrony
- Presented solution for BFS construction

Review Questions

1. Let N be a network with n processors. Suppose that each node can write to a single shared variable (register) and that its program uses a single non-shared (internal) variable. Moreover, suppose that both the shared and non-shared variables are of $(\log_2 n + 1)$ bits. How many different values can be encoded in this system? Could you use the pigeonhole principle for bounding from below the number of values that the system does not encode?

Review Questions

2. We would like to have a spanning forest in which there are several BFS tree
   - Each tree is rooted by a different root node and every root node is the root of a tree
   - Any non-root node selects a single tree to be a member of

Review Questions

- The node selects its closest root
  - i.e., considers the number of links on the shortest path between the node and the root
- In case there is more than one root, the node uses a technique for breaking symmetry
  - Namely, suppose that there are two roots in the system and the distance of node p_i is d to both of them, then node p_i uses some technique for deciding which tree to belongs to

Review Questions

- Please propose a technique for breaking the symmetry
  - Specify the precise assumptions regarding the system settings and describe in detail how the technique works
- Moreover, if you propose a randomized technique, please prove that there is no deterministic one
  - Please note that some symmetry breaking techniques have easer correctness proofs than others; choose wisely

3. Please modify the algorithm and the proof according to the new requirements