Diffusing Computation

Using Spanning Tree Construction for Solving Leader Election

- Root is the leader
- In the presence of faults,
  - There may be multiple trees
    - Multiple leaders
- After recovery
  - There is a unique tree
  - There is a unique leader
- A spanning tree maintenance algorithm is a nonmasking leader election algorithm
  - Make it masking fault-tolerant

Using Spanning Tree Construction for Solving Leader Election

- Spanning tree provides a nonmasking fault-tolerant solution
  - Eventually, there is one leader
  - During recovery,
    - The safety specification could be violated because there could be multiple leaders

Designing Masking Fault-Tolerant Solution for Leader Election

- Need to ensure that there is at most one leader during recovery
- Can be achieved if a node makes sure that before it declares itself to be the leader, there is no other leader
  - If this node were to become the true leader, the tree would be reconstructed to be rooted at that node
  - Check if the tree has been reconstructed
    - I.e., check if all nodes are in the tree rooted at the current node

How?

- When j changes P.j to j
  - Check if all nodes have set their root value to j?
  - Achieved through diffusing computation
    - Similar to one of the termination detection algorithms (next topic)

Diffusing Computation

- Initiation
  - The node that initiates the diffusing computation sends the diffusing computation message to all its children
- Propagation
  - A node propagates the diffusing computation when it receives it (for the first time)
    - To propagate, the node sends the diffusion message to all its children
    - A node can check some predicate (condition) during propagation

Diffusing Computation

- Completion
  - A node completes the diffusing computation iff
    - All its children complete the diffusing computation
    - All its neighbors have received (propagated) that diffusing computation
    - A node can also check some predicate during completion
      - The results from this condition are propagated towards the root;
- The diffusing computation completes when the root completes the diffusing computation
  - If the condition checked by the root evaluates to true, we say that the diffusing computation completes successfully
  - Else, we say that the diffusing computation fails/completes unsuccessfully

Using Spanning Tree for Leader Election

- When a node is about to become leader
  - It starts a diffusing Computation
  - If it completes successfully, declare itself to be the leader
- Goal of the diffusing computation is to check if the tree is formed at the initiator node
  - In other words, check if root value of all nodes is equal to that of the initiator

Property

- If the tree is not formed then
  - There is at least one node j such that
    - j is not in the tree
    - But j has a neighbor that is in the tree

Issues

- Issues
  - Multiple nodes may start diffusing computation
    - Node with higher ID should win (I.e., its diffusing computation should complete successfully)
    - Node with lower ID should lose.
  - Failures during diffusing computation
    - Fail the current diffusing computation

Upon completion

- Upon completing an unsuccessful diffusing computation
  - Start another one if the node is still likely to be the leader (P.j = j)
    - The diffusing computation may have failed due to faults

- Use sequence number to distinguish between different diffusing computations initiated by the same node

Actions

- Init

  P.j = j

  →

  - Phase.j = prop
  - Sn.j = newseq()
  - Res.j = true
    - (Result = result of the diffusing computation)
    - Currently set to true, to be read only after the diffusing computation completes

Actions

- Propagation

  Root.j = root.(P.j) /\ sn.j ≠ sn.(P.j)

  →

  - If (phase.(P.j) = prop)
    - Phase.j = prop, res.j = true
  - Else
    - Res.j = false
      - » // Fail this diffusing computation

Actions

- Completion

  Phase.j = prop /\

  ∀k : k ∈Neighbors.j : root.j = root.k /\ sn.j = sn.k /\

  ∀k : k ∈Ch.j : phase.k = comp

  →

  res.j = ∀ k : k ∈Neighbors.j ∪ {j} : res.k

  Phase.j = comp

  If (P.j = j /\ ¬res.j) Init(j) // Initiate a new diffusing computation

Actions

- Aborting Diffusing Computation

  Phase.j = comp, res.j = false

  If (P.j ∈Neighbors.j) res.(P.j) = false

  // Last part when j changes its parent. With respect to `old’ parent.

Combining with Tree Algorithm

- When j executes tree correction action 1 (red color propagation)
  - Abort(j)

- When j executes tree correction action 2 (changing color to green)
  - Init(j)

- When j changes tree
  - Abort(j)

Application in Mutual Exclusion

- Allow only tree root to generate the new token after faults
  - When a node becomes a root, set h.j = j thereby permitting it to possibly generate a token
  - Don’t send this token to other processes unless you ensure that you can actually generate the token
- If token is to be generated (safely) at the root then
  - All nodes must be in the same tree // done already with current task
  - For all nodes: h.j = P.j // condition to be checked at each node

- Init/Prop actions same
- Complete action changed as:

Actions

- Completion

  Phase.j = prop /\

  ∀k : k ∈Neighbors.j : root.j = root.k /\ sn.j = sn.k /\

  ∀k : k ∈Ch.j : phase.j = comp

  →

  res.j = ∀ k : k ∈Neighbors.j ∪ {j} : (res.k & h.k = P.k)

  Phase.j = comp

  If (P.j = j /\ ¬res.j) Init(j) // Initiate a new diffusing computation

Diffusing Computation

- Can be performed in the absence of a tree
  - Tree is formed during diffusing computation
    - The initiator sends the first diffusion message
    - If node, say j, receives a diffusion from k then
      - If this is NOT the first diffusion message
        - » Send a reply to k
      - If this is the first diffusion message then
        - » Send the diffusion to all neighbors
        - » Send a reply to k after replies from ALL neighbors is received
        - » A condition can be checked based on the replies received

Diffusing Computation (continued)

- The node from which the diffusion message is received for the first time is the parent of that node.
  - Several problems can be solved by taking appropriate actions during propagation and completion of diffusion
    - Detecting global state
    - Termination detection

Applications of Diffusing Computations and other Terminology

- Route Discovery of protocols such as DSR
- Viral computations
- Global snapshots

Scalpel vs Hammer