Diffusing Computations

- Given a distributed system represented by an undirected graph in which the vertices represent agents and the edges represent message channels.
- An agent is either *active* or *idle*.
- An idle agent remains idle until it receives a message.
- An active agent can change state, send messages and receive messages, and an active process may become idle.
Problem

- **Initially**: all agents other than one – called the initiator – are idle. All channels are empty. The initiator is active.

- **Termination condition**: all agents (including initiator) are idle and all channels are empty.

- Problem: Devise an algorithm by which the initiator detects that the computation has terminated. Initiator has a variable claim (for claim terminated) where

**Safety**: Invariant: claim IMPLIES termination condition

**Progress**: Termination condition leads-to claim
An approach to the problem

- We want a distributed data structure such that:
  1. All active processes and the initiator are in the structure, and if a message is in transit from u to v then u or v (or both) are in the structure.
  2. termination condition leads-to the structure becoming empty.

What distributed data structure do you recommend we use?
Rooted directed tree with initiator as root

Variables

- For each process $p$, $p$.parent.
- Invariant: $p$.parent = null if and only if $p$ is not on the tree
- Invariant: $p$.parent = $v$ (for some $v$) if and only if $p$’s parent is $v$ where $v$ is on the tree.
Invariant: Rooted Tree Requirements

Invariant: Every vertex is either

1) Unconnected (i.e., has no parent) or part of a tree rooted at the initiator (i.e., it has a unique directed path along parent edges to the initiator)

2) q is active implies q is on the tree

3) Message from w to u implies w is on the tree
Prove the invariant.
Actions: Sending a Message

- Given: Only active processes can send messages.
- From the invariant it follows that a process p sends a message only if p is on the tree.
Prove the invariant: Receiving a Message

- When a process receives a message it becomes active.
- To maintain the invariant, this process must join the tree if it is not already part of the tree.
- To maintain the invariant, a process q that is not on the tree and that receives a message from a process p and thus becomes active, does what?

Sets q.parent = p

Is that enough?
Maintaining the invariant: Becomes idle

When a process \( w \) becomes idle it must remain on the tree if:

1. There exists a message from \( w \) in a channel to an agent, or
2. \( w \) is the root of a subtree
Maintaining invariant when becoming idle

- Associated with each process is an ack.
- An idle process on the tree doesn’t change its parent while it has an outstanding ack.
When should an agent send an ack?

- When an agent $q$ that is not on the tree becomes active because it got a message from an agent $p$, then it sets
  $$q.\text{parent} := p$$
  and holds on to the ack for that message.

- Invariant: $q.\text{parent} = p$ IMPLIES $p$ has at least one outstanding ack.

- Send acks for all other messages immediately.
Progress

- After computation terminates the tree must shrink to including just the initiator.
- Variant function: size (number of nodes on the tree).

- Assume computation has terminated. How do we ensure that the tree shrinks?
When can a vertex drop off the tree?

- From the invariant, all active processes are on the tree so only idle processes can drop off.

- To maintain the rooted tree, only leaves of the tree can drop off.

- Therefore, the invariant tells us that only idle leaves can drop off the tree.
How does a process determine if it’s a leaf?

- From the invariant, q.parent = p IMPLIES “q has received more messages from p than it has acked to p,” it follows that:

- **Invariant**: p has no children if p has received acks for all the messages it has sent.

- So, p can drop off the tree if p has received acks for all the messages it has sent and p is idle.