Wave Algorithms

- ring algorithm
- tree algorithm
- echo algorithm
- polling algorithm

Wave Algorithms

- Wave algorithm satisfies the following three properties
  - termination: each computation is finite
  - decision: each computation contains at least one decide event
  - dependence: in each computation each decide event is causally preceded by an event in each process
- *initiator*(*starter*) - process that execution of its actions spontaneously
- *non-initiator*(*follower*) - starts execution only when receives a message
- wave algorithms differ in many respects, some features:
  - *centralized* (*single-source*) - one initiator; *decentralized* (*multi-source*) - multiple initiators
  - topology - ring, tree, clique, etc.
  - initial knowledge:
    - each process knows its own unique name
    - each process knows the names of its neighbors
  - number of decisions to occur in each process
- usually wave algorithms exchange messages with no content - tokens

Ring algorithm

For the initiator:
begin send ⟨ **tok** ⟩ to $Next_p$; receive ⟨ **tok** ⟩ ; *decide* **end**

For non-initiators:
begin receive ⟨ **tok** ⟩ ; send ⟨ **tok** ⟩ to $Next_p$ **end**

- Processes are arranged in a unidirectional ring (each process has a sense of direction or knowledge of one dedicated neighbor)
- initiator send message **⟨tok⟩** along the cycle
- each process passes it on
- when it returns to initiator, initiator decides
- *Theorem*: ring algorithm is a wave algorithm

Tree algorithm

```
var rec_p[q] for each q ∈ Neigh_p : boolean init false ;
    (* rec_p[q] is true if p has received a message from q *)
begin while #{q : rec_p[q] is false} > 1 do
          begin receive ⟨tok⟩ from q ; rec_p[q] := true end ;
    (* Now there is one q_0 with rec_p[q_0] is false *)
    send ⟨tok⟩ to q_0 with rec_p[q_0] is false ;
    receive ⟨tok⟩ from q_0 ; rec_p[q_0] := true ;
    decide
    (* Inform other processes of decision:
       forall q ∈ Neigh_p, q ≠ q_0 do send ⟨tok⟩ to q *)
end
```

- Operates on tree network (can work on spanning tree of arbitrary network) - no root, edges are undirected (bi-directional)
- leaves of tree initiate the algorithm
- if a process has received a message from all neighbors but one (initially true for leaves), the process sends a message to the remaining neighbor.
- If process gets messages from all neighbors - it decides
- Excluding the **forall** statement how many processes can decide? What are these processes?
- Why do we need **forall** statement?
- How many messages are sent in the algorithm?
- Is this a wave algorithm?

Polling algorithm

```
var rec_p  : integer   init 0 ;  (* for initiator only *)
For the initiator:
    begin forall q ∈ Neigh_p do send ⟨tok⟩ to q ;
          while rec_p < #Neigh_p do
                begin receive ⟨tok⟩ ; rec_p := rec_p + 1 end ;
          decide
    end
For non-initiators:
    begin receive ⟨tok⟩ from q ; send ⟨tok⟩ to q end
```

- Works on cliques (complete networks)
- one initiator (centralized)
- how many processes decide?
- How many messages are exchanged in the algorithm
- what other topology can this algorithm be used in
- Is this a wave algorithm?

Chang's Echo algorithm

```
var rec_p    : integer  init 0 ; (* Counts number of received messages *)
    father_p : P        init udef
For the initiator:
    begin forall q ∈ Neigh_p do send ⟨tok⟩ to q ;
          while rec_p < #Neigh_p do
                begin receive ⟨tok⟩ ; rec_p := rec_p + 1 end ;
          decide
    end
For non-initiators:
    begin receive ⟨tok⟩ from neighbor q ; father_p := q ; rec_p := rec_p + 1 ;
          forall q ∈ Neigh_p, q ≠ father_p do send ⟨tok⟩ to q ;
          while rec_p < #Neigh_p do
                begin receive ⟨tok⟩ ; rec_p := rec_p + 1 end ;
          send ⟨tok⟩ to father_p
    end
```

- Works on networks of arbitrary topology
- one initiator (centralized)
- initiator sends messages to all neighbors
- when non-initiator receives the first message it forwards it to all other neighbors, when it gets tokens from all other neighbors it replies back
- how many processes decide?
- How many messages are exchanged in the algorithm
- Is this a wave algorithm?