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

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

Polling algorithm

- 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?

Polling algorithm

```
var recCIg for each q in Nq, do send (tok) to q; (* recCIg is true if q has received message from q *)
begin while q in Gq, do
  begin if recCIg is false then
    send (tok) to q with recCIg is false;
    receive (tok) from q with recCIg is true;
    decide; (* informs other processes of decision *)
    forall q in Nq, q not in Gq do
      send (tok) to q; (* )
end
```

Tree algorithm

- 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 the forall statement?
- How many messages are sent in the algorithm?
- Is this a wave algorithm?

Polling algorithm

```
var recCIg for each q in Nq do send (tok) to q; (* recCIg is true if q has received message from q *)
begin for all q in Gq do
  if recCIg is false then
    send (tok) to q;
    receive (tok) from q;
    recCIg = recCIg + 1;
    decide;
end
```

Chang's Echo algorithm

- 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?

Chang's Echo algorithm

```
var recCIg; integer Init 0; (* for initiator only *)
begin for all q in Gq do
  send (tok) to q;
  recCIg = recCIg + 1;
  receive (tok) from q;
  decide;
end
```

Ring algorithm

```
begin send (tok) to Nq; receive (tok); decide
```

Polling algorithm

```
begin for all q in Gq do send (tok) to q;
```

Polling algorithm

```
begin for all q in Gq do send (tok) to q;
```

Polling algorithm

```
begin for all q in Gq do send (tok) to q;
```

Ring algorithm

```
begin send (tok) to Nq; receive (tok); decide
```

Polling algorithm

```
begin for all q in Gq do send (tok) to q;
```

Polling algorithm

```
begin for all q in Gq do send (tok) to q;
```