Election algorithms

- Definition of election algorithm, assumptions
- election algorithm on a tree
- rings
- LeLann
- Chang and Roberts
- Dolev-Klawe-Rodeh
- arbitrary networks - extinction algorithm

What is election algorithm

- Election algorithm satisfies the following properties:
  - (uniformity) each process has the same local algorithm
  - (decentralization) a computation can be initiated by arbitrary subset of processes
  - (termination) each computation terminates
  - (safety) in every terminal state there is one and only one process in the state of leader

Assumptions

- system is fully asynchronous - not necessary but useful
- each process has unique identity (name), process identities are totally ordered (can be compared) - needed to break the symmetry, smallest identity wins (becomes leader)
- the only operation allowed on identities is comparison - in asynchronous systems arbitrary (non-comparison algorithms can do no better)
- each message may contain up to a constant number of identities - so complexities of algorithms can be compared

Waves on trees and election

- Use wave tree algorithm
  - first phase - "wake up" all processes (non-initiators)
  - a wakeup message is propagated from initiators to other processes
  - second phase - do a wave on the tree
    - leaves start
    - id of lowest process in subtree is attached to token
  - message complexity: 4N-4 O(N) (two wakeup messages and two tokens are sent along each channel)
  - time complexity - 3D+1 (D - diameter) - O(N)
  - D - to send wakeup messages, in D+1 tree algorithm starts
  - D - to make first decision, D - to propagate decision

LeLann's algorithm

```plaintext
var List_p : set of P
init (p);
begin
  if p is initiator then
    begin
      List_p := P; 
      init (p) to Nhs_p; receive (tok, q);
    while q - p do
      begin
        List_p := List_p to List_p (q); 
        send (tok, q) to Nhs_p; receive (tok, q);
      end;
      if p = min(List_p) then status = leader
      else status = lost
    end;
end
```

Each initiator sends token with its id around the ring
- a process forwards foreign token and records the identity it carries
- a process decides when it receives its own token
- if processes gets a message before it sends its own - it loses
domplexity
- message - O(N)
- time - O(N)

Chang and Roberts's algorithm (CR)

```plaintext
var status ;
begin
  if p is initiator then
    begin
      status := cond; send (tok, p) to Nhs_p;
      while p - q do
        begin
          if q < p then
            begin
              if status := cond then status := leader
              else send (tok, q) to Nhs_p;
            end;
            send (tok, q) to Nhs_p;
          end;
        end;
      end;
end
```

Process propagates
- foreign message only when the original sender may become a leader
docomplexity
- time - same as LeLann's
- message - worst case - same as LeLann's
- average - O(N/log N)

Dolev-Klawe-Rodeh algorithm (DKR)

```plaintext
Worst case message complexity of Chang-Roberts is still O(N) -- in the worst case the identities that are not leaders are allowed to propagate
- idea of DKR - eliminate wrong identities as soon as possible
  - active/passive processes as in CR
    - active - propagates its current identity (stored in variable ci) to downstream processes
    - passive - forwards messages
  - algorithm proceeds in rounds, each round has two phases:
    - propagation - each active process sends message <ci, acini> to downstream active process, received id stored in acn
    - elimination - each active process sends a message <ci, acn> to downstream neighbor when <ci, acn> arrives, the receiver compares it with its can thus the following identity "catches up" with preceding
    - when process gets his own identity - it is a winner
      - winner sends message <smal, id> informing others
```

Dolev-Klawe-Rodeh algorithm (DKR)

```plaintext
Worst case message complexity of Dolev-Klawe-Rodeh is still O(N) -- in the worst case the identities that are not leaders are allowed to propagate
- idea of DKR - eliminate wrong identities as soon as possible
  - active/passive processes as in CR
    - active - propagates its current identity (stored in variable ci) to downstream processes
    - passive - forwards messages
  - algorithm proceeds in rounds, each round has two phases:
    - propagation - each active process sends message <ci, acini> to downstream active process, received id stored in acn
    - elimination - each active process sends a message <ci, acn> to downstream neighbor when <ci, acn> arrives, the receiver compares it with its can thus the following identity "catches up" with preceding
    - when process gets his own identity - it is a winner
      - winner sends message <smal, id> informing others
```
DKR message complexity estimate

- Claim: if there were \( k \) identities at the beginning of the round
  - only \( k/2 \) survive after the round
- each process (if it survives round) assumes the identity of the first upstream neighbor
- out of two neighbor identities only one survives the round
- there can be at most \( \log N \) rounds, \( 2^N \) messages are exchanged during round, algorithm completes with \( N \) messages informing processes of results
- thus message complexity of the algorithm is:
  \[ 2^N \log N \]

Election on arbitrary networks, extinction

- Election can be done on networks of arbitrary topology by using any centralized wave algorithm.
- The technique is called extinction.
- each initiator starts a separate wave. To distinguish waves, initiator’s id is attached to token.
- Only the wave of the initiator with the lowest id is allowed to finish
- A process stores the wave with the lowest id that it saw in currently active wave (caw) variable. The process ignores tokens from waves with higher id than caw
- If a process receives a token from a wave with still lower id than caw, the process ignores that id into caw
- complexity
  - message \( NM \) where \( M \) - number of messages used in the base wave algorithm
  - time \( NT \) where \( T \) - time the wave takes

Extinction applied to Echo