
Election Algorithms 2

16 June 2010

Lecture 13

Slide credits: R. H. Mak

Topics for Today

- Leader election on a ring
 - LeLann, Chang-Roberts
 - Peterson/Dolev-Klawe-Rodeh
- Minimum Spanning Trees
 - Kruskal's Algorithm
- Leader election on an arbitrary network
 - Gallager-Humblet-Spira

Source: Tel 7

LeLann's election algorithm

```
var       $List_p$  : set of  $P$                 init {p};
         $state_p$  : (sleep, cand, leader, lost)  init sleep

begin
  if  $p$  is initiator then
    begin       $state_p := cand$ ;
              send  $\langle \mathbf{tok}, p \rangle$  to  $Next_p$ ; receive  $\langle \mathbf{tok}, q \rangle$ ;
              while  $q \neq p$  do
                begin  $List_p := List_p \cup \{ q \}$ ;
                      send  $\langle \mathbf{tok}, q \rangle$  to  $Next_p$ ; receive  $\langle \mathbf{tok}, q \rangle$ 
                end;
              if  $p = \min(List_p)$  then  $state_p := leader$ 
              else  $state_p := lost$ 
            end;
  else while true do
    begin receive  $\langle \mathbf{tok}, q \rangle$ ; send  $\langle \mathbf{tok}, q \rangle$  to  $Next_p$ ;
          if  $state_p = sleep$  then  $state_p := lost$ 
        end;
  end
end
```

Remarks about LeLann's algorithm

1. It is not necessary to keep track of the lists
2. It assumes that channels have the FIFO property. Modifications that do not require this assumption exist.
3. Non-initiators do not terminate. A simple fix exists.
4. Each initiator sends out a token that makes a complete round around the ring. For ring size N this takes N messages. Since, there can be at most N initiators the message complexity is $O(N^2)$. Since all initiators start within a time interval of N time slots and it takes N time slots for a token to make a roundtrip the time complexity is $O(N)$.

Chang-Roberts leader algorithm

```
var  $state_p$  : (sleep, cand, leader, lost)  init sleep

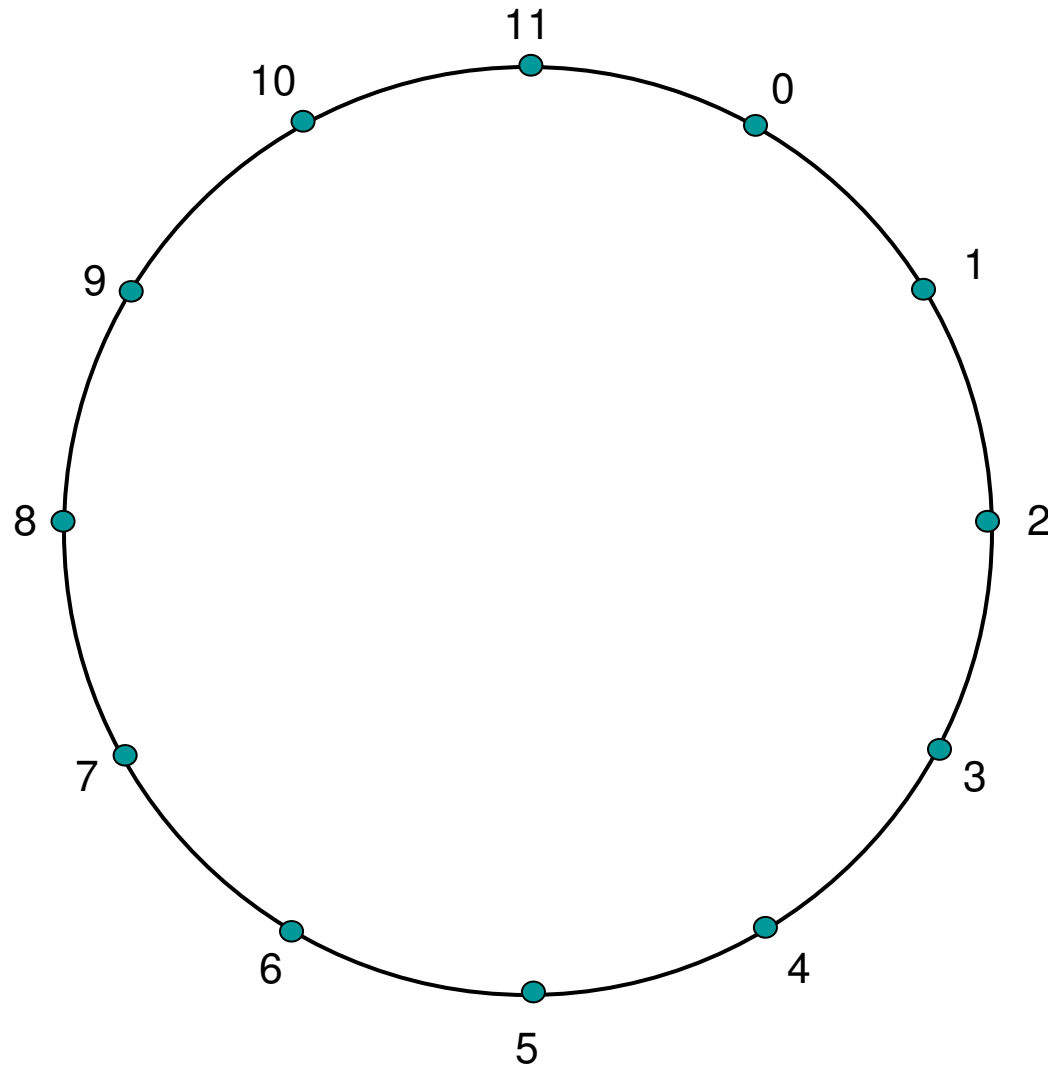
begin if  $p$  is initiator then
    begin  $state_p := cand$ ; send  $\langle tok, p \rangle$  to  $Next_p$ ;
        while  $state_p \neq leader$  do
            begin receive  $\langle tok, q \rangle$ ;
                if  $q = p$  then  $state_p := leader$ 
                else if  $q < p$  then
                    begin
                        if  $state_p = cand$  then  $state_p := lost$ ;
                        send  $\langle tok, q \rangle$  to  $Next_p$ ;
                    end
                end
            end
        end
    end
else while true do
    begin receive  $\langle tok, q \rangle$ ; send  $\langle tok, q \rangle$  to  $Next_p$ ;
        if  $state_p = sleep$  then  $state_p := lost$ 
    end;
end

end
```

Remarks about Chang-Roberts

1. The worst case message complexity is $\Theta(N^2)$
 1. at least an improvement of a factor 2 over LeLann
2. When all processes are initiators, the average case message complexity is $O(N \log N)$
3. The time complexity is $O(N)$

Worst-case scenario Chang-Roberts



Ideas for improvement

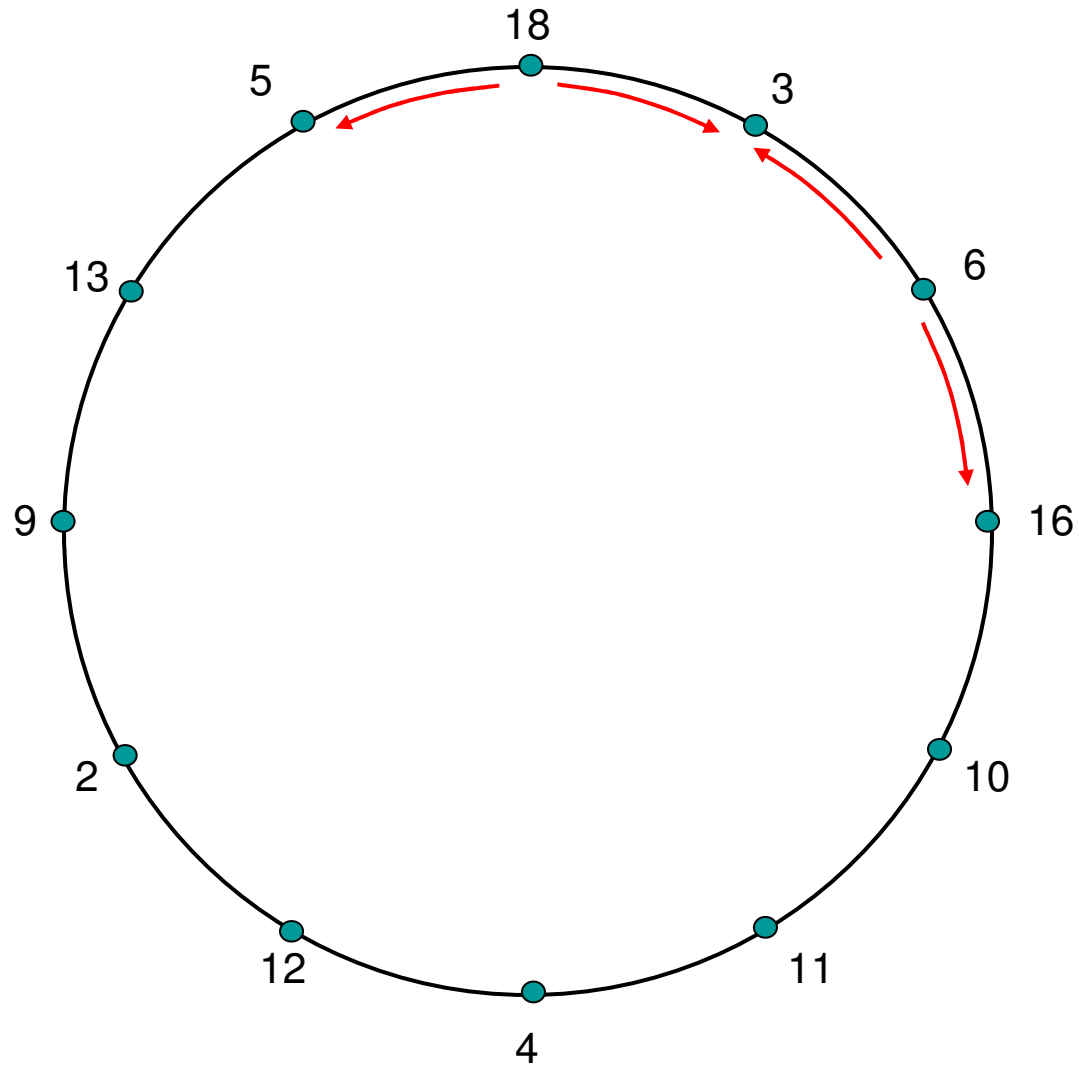
1. Restrict the distance a token can travel
 1. already present in Chang-Roberts
 2. fixed upper bound (unlike C-R)
2. Organize the election in rounds
 1. each round the number of candidates is reduced

For unidirectional rings this lead to an algorithm that has been developed independently by Peterson and by Dolev, Klawe, and Rodeh

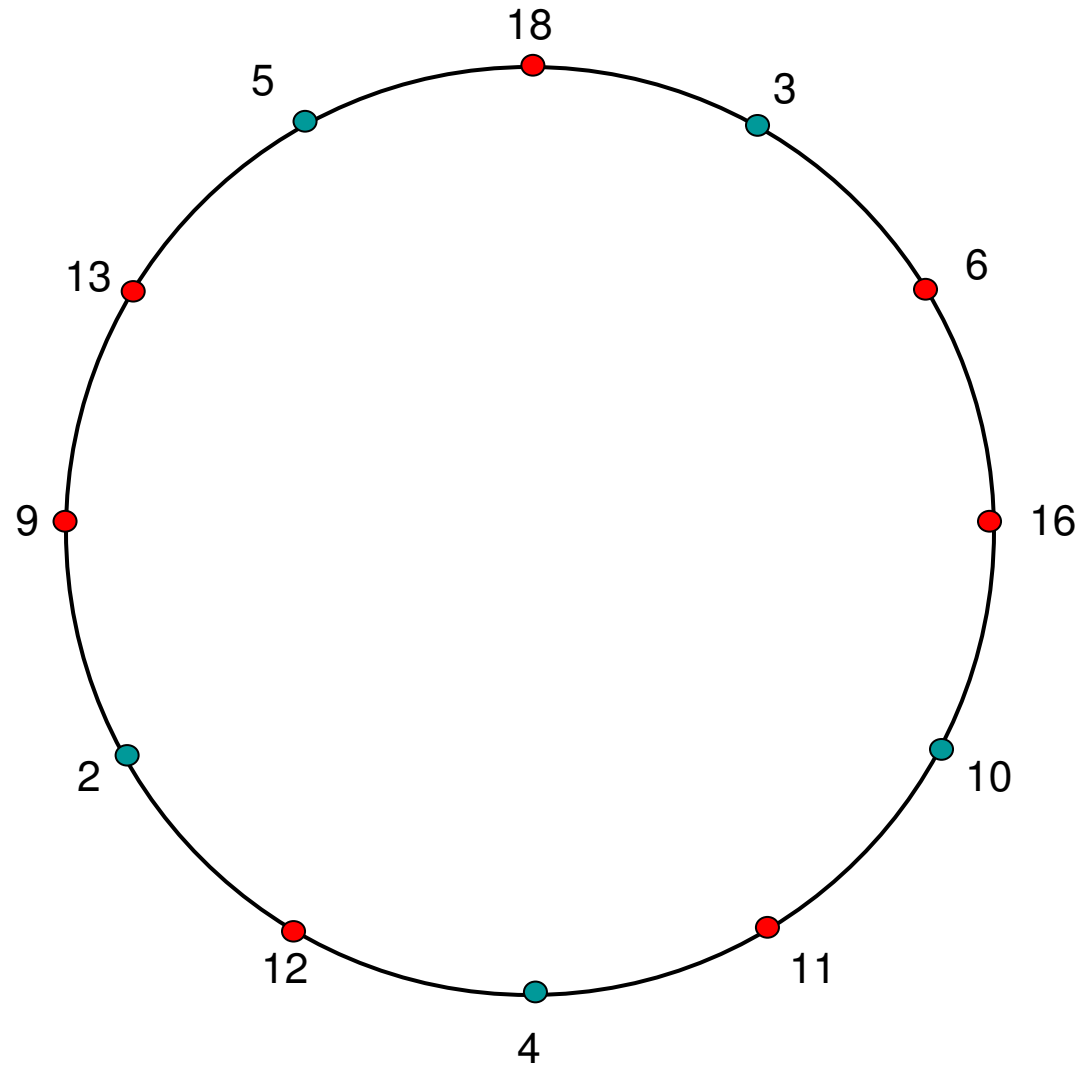
So Far

- Leader election on a ring
 - LeLann, Chang-Roberts
 - Peterson/Dolev-Klawe-Rodeh
- Minimum Spanning Trees
 - Kruskal's Algorithm
- Leader election on an arbitrary network
 - Gallager-Humblet-Spira

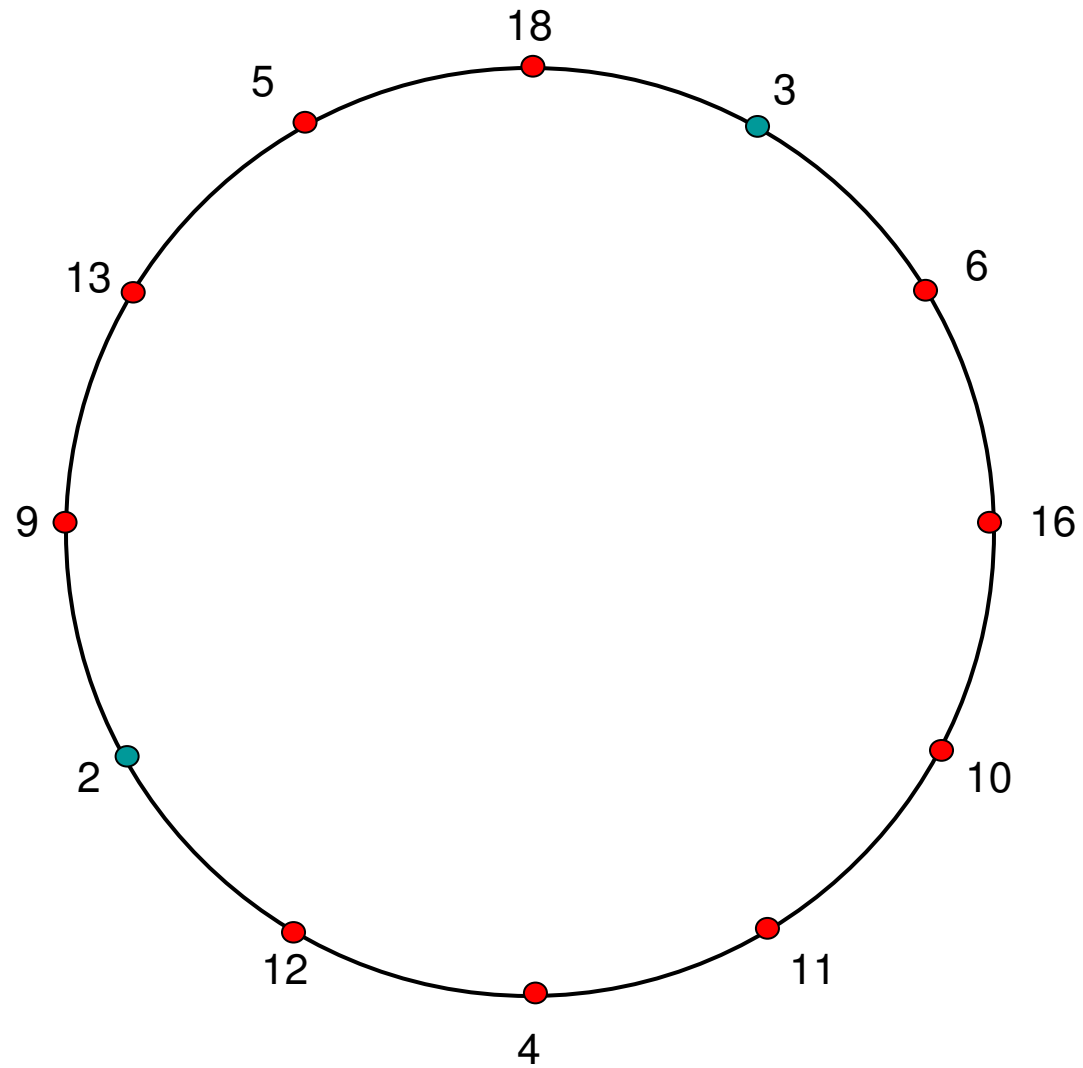
Initial configuration



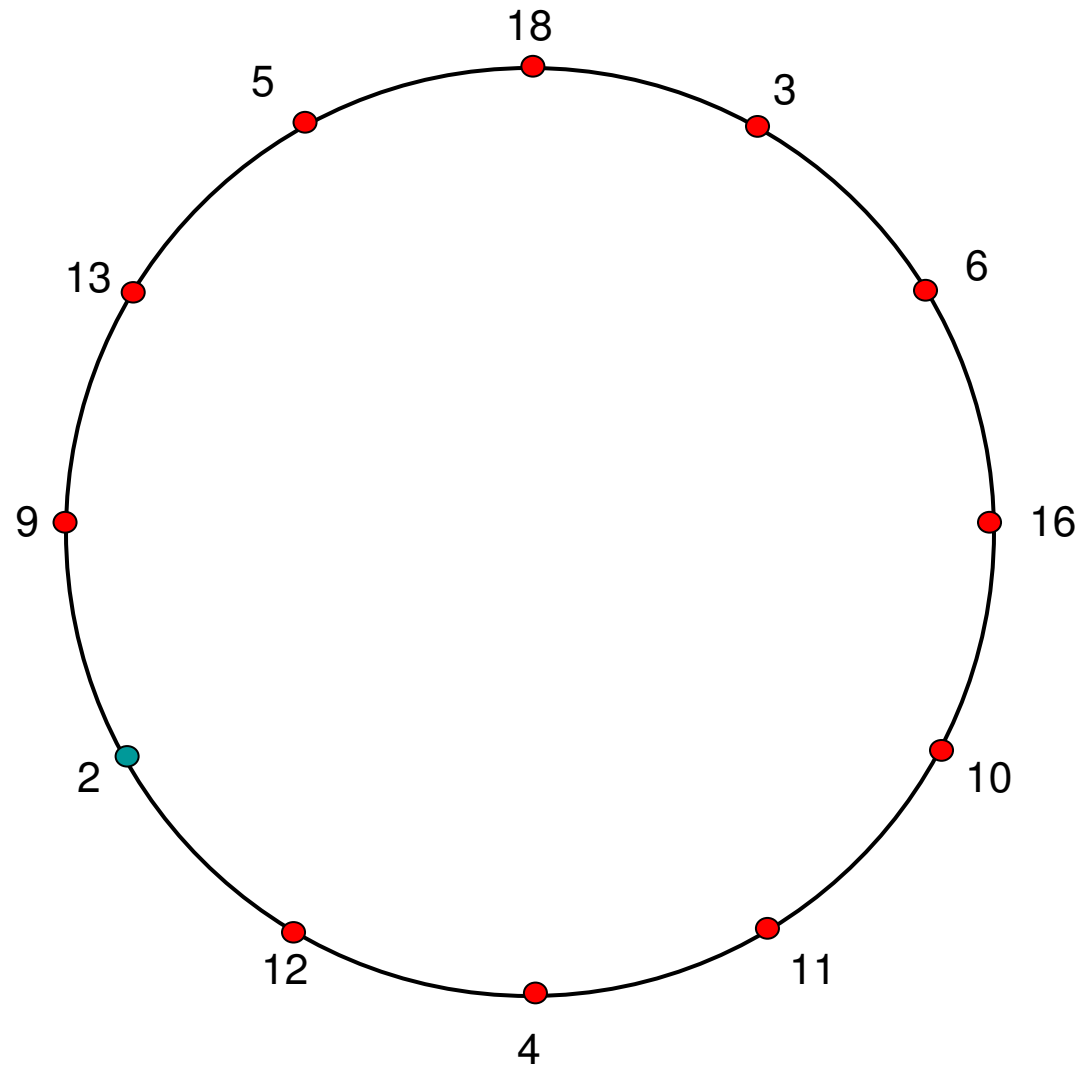
Round 1



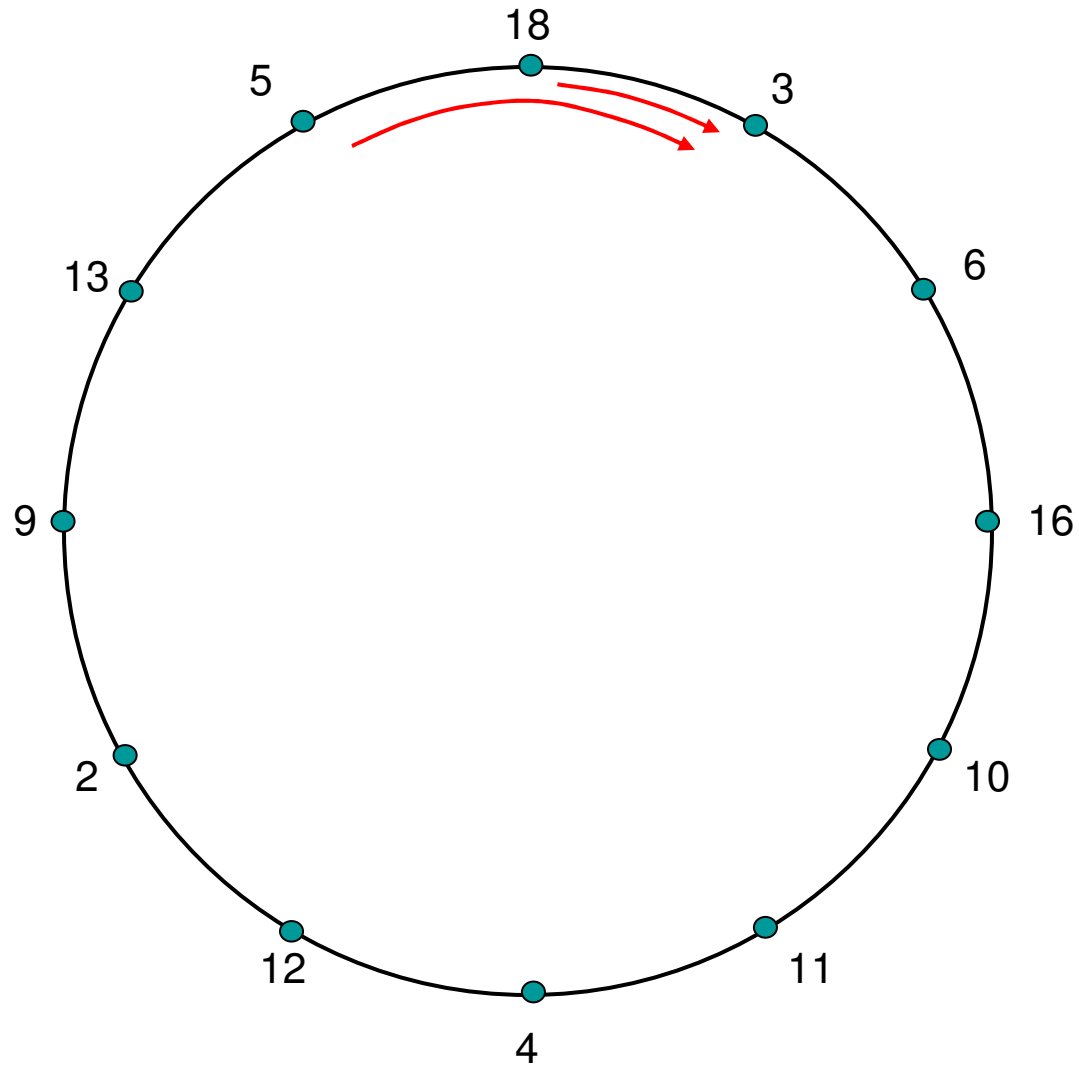
Round 2



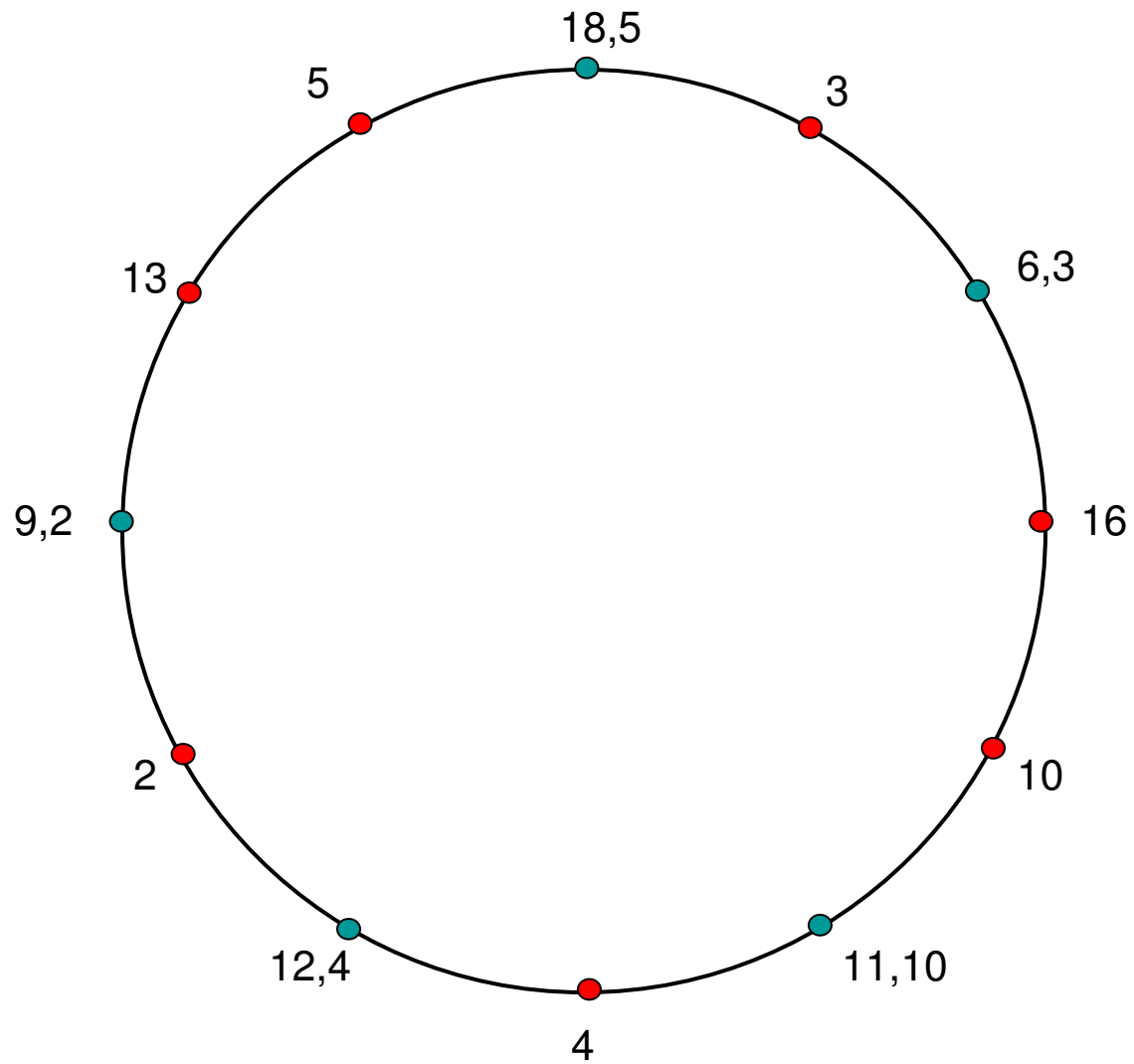
Round 3



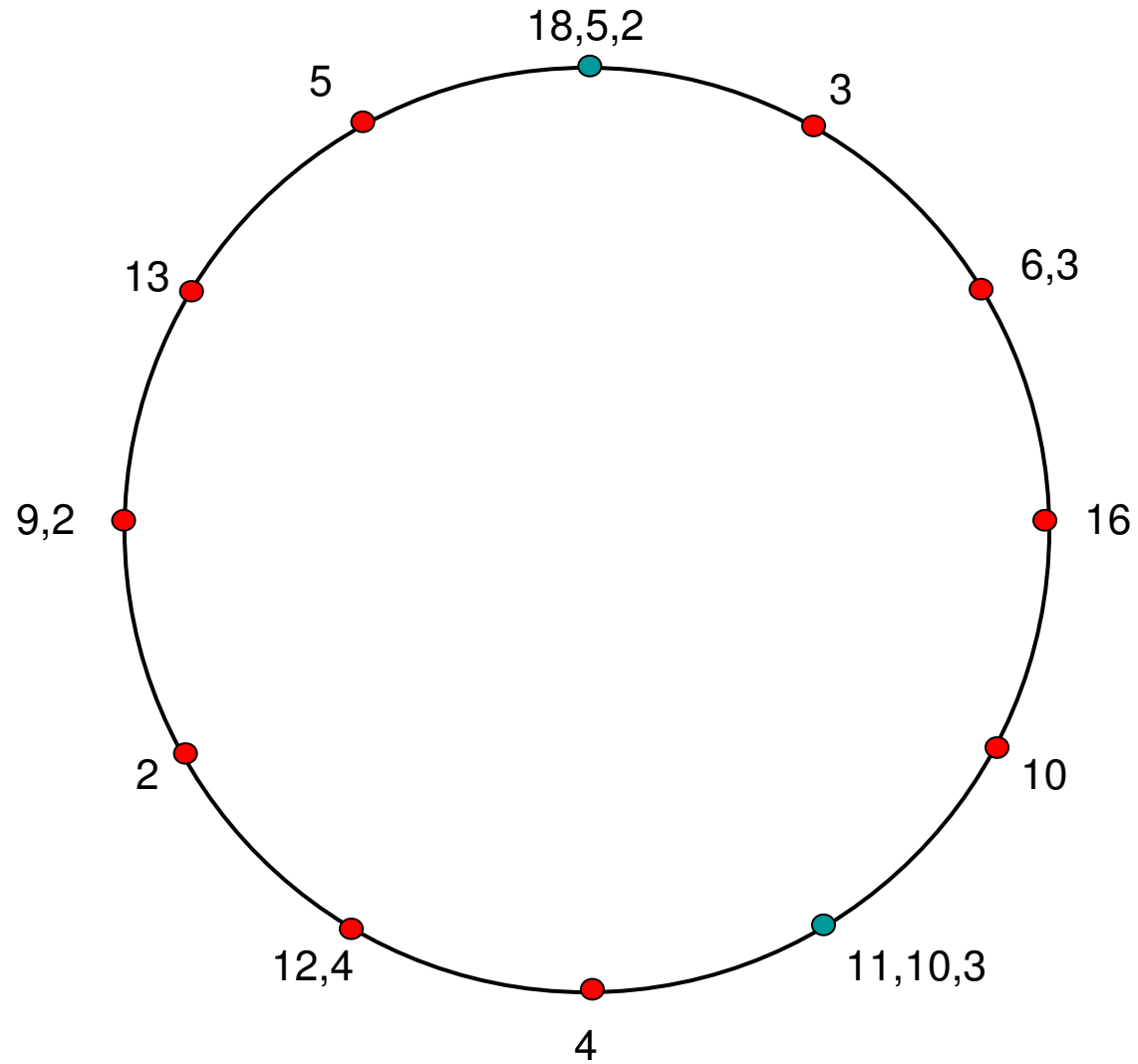
Initial configuration



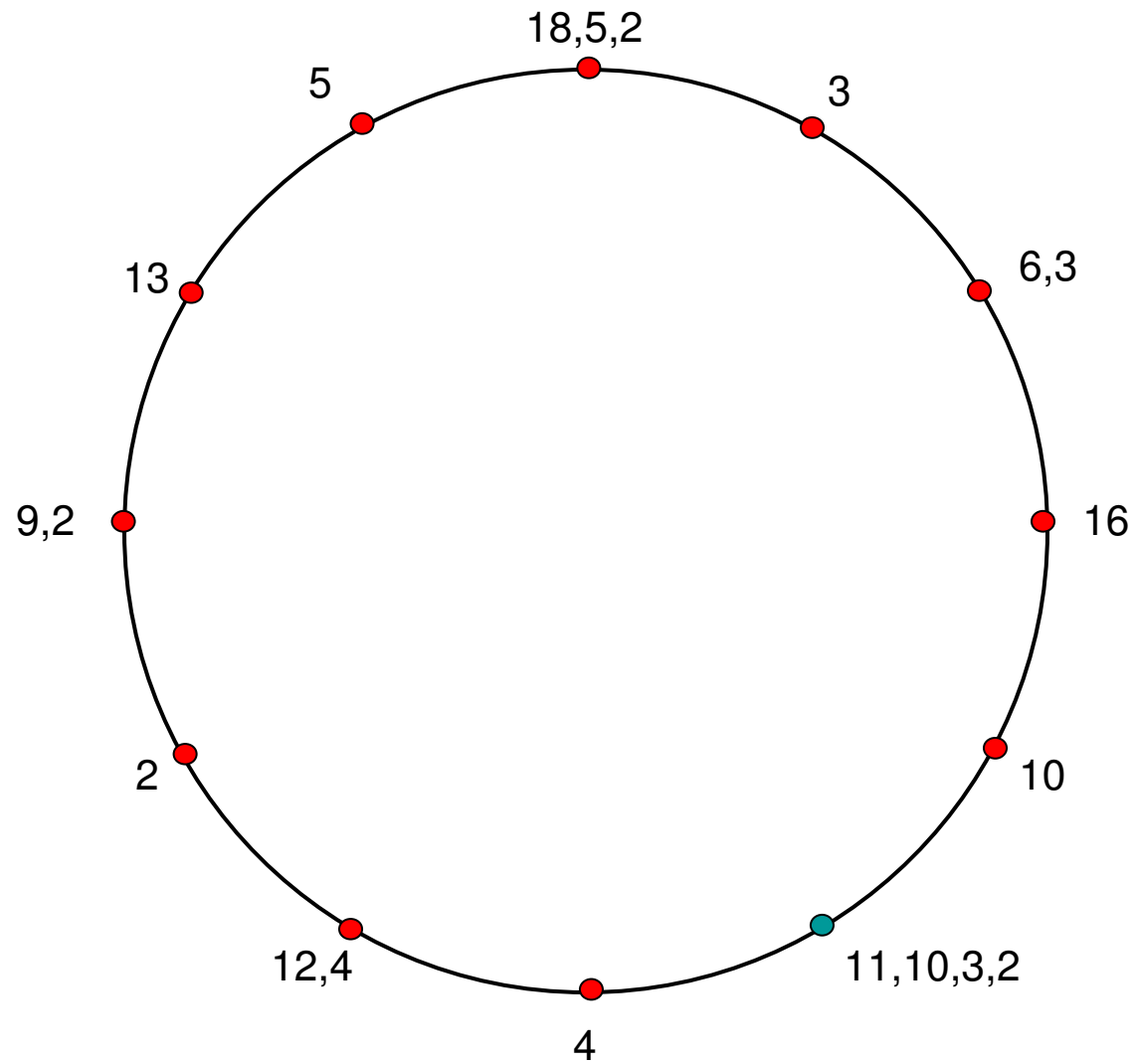
Round 1



Round 2



Round 3



Peterson/Dolev-Klawe-Rodeh

```
var  $cip$  :  $\mathcal{P}$  init  $p$ ; (* Current Id of  $p$  *)
     $acn_p$  :  $\mathcal{P}$  init undef; (* Id of active predecessor *)
     $win_p$  :  $\mathcal{P}$  init undef; (* Id of winner *)
     $state_p$  : (active, passive, leader, lost) init active;

begin if  $p$  is initiator then
     $state_p := active$ 
else
     $state_p := passive$ ;
while  $win_p = undef$  do
    if  $state_p = active$  then
         $DoActive(p)$ 
    else (*  $state_p = passive$  *)
         $DoPassive(p)$ ;
    if  $p = win_p$  then  $state_p := leader$  else  $state_p := lost$ 
end
```

Peterson/Dolev-Klawe-Rodeh

```
procedure DoActive( $p : \mathcal{P}$ )
begin  send  $\langle \text{one}, ci_p \rangle$  to  $Next_p$ 
      receive  $\langle \text{one}, q \rangle$  from  $Prev_p$ ;  $acn_p := q$ ;
      if  $acn_p = ci_p$  then (*  $acn_p$  is the minimum *)
        begin  send  $\langle \text{smal}, acn_p \rangle$  to  $Next_p$ ;  $win_p := acn_p$ ;
              receive  $\langle \text{smal}, q \rangle$  from  $Prev_p$ 
        end
      else (*  $acn_p$  is current id of neighbor *)
        begin  send  $\langle \text{two}, acn_p \rangle$  to  $Next_p$ ;
              receive  $\langle \text{two}, q \rangle$  from  $Prev_p$ ;
              if  $acn_p < ci_p \wedge acn_p < q$  then
                 $ci_p := acn_p$ 
              else  $state_p := passive$ 
        end
      end
end
```

Peterson/Dolev-Klawe-Rodeh

```
procedure DoPassive( $p : \mathcal{P}$ )
begin receive  $\langle \text{one}, q \rangle$  from  $Prev_p$ ;
      send  $\langle \text{one}, q \rangle$  to  $Next_p$ ;
      receive  $m$  from  $Prev_p$ ;
      send  $m$  to  $Next_p$ ;
      (*  $m$  is either  $\langle \text{two}, q \rangle$  or  $\langle \text{smal}, q \rangle$  *)
      if  $m :: \langle \text{smal}, q \rangle$  then  $win_p := q$ 
end
```

Complexity P/D-K-R

Lemma. At the beginning of every round all active processes have different c_i 's including the minimum. In each round the number of active processes is at least halved, and at least one active process survives.

Lemma. If a round starts with exactly one active process p with current identity c_i , then that round ends with $win_q = c_i$ for all processes q .

Thm. The worst case message complexity of P/D-K-R is $O(N \log N)$.

Lower bound result

Thm. The average case message complexity of leader election on a unidirectional ring is $\Omega(N \log N)$

Thm. The worst case message complexity of leader election on a unidirectional ring is $\Omega(N \log N)$

Assumptions:

- FIFO channels
- All processes are initiators (worst case)
- Processes don't know ring size
- Message-driven

So Far

- Leader election on a ring
 - LeLann, Chang-Roberts
 - Peterson/Dolev-Klawe-Rodeh
- Minimum Spanning Trees
 - Kruskal's Algorithm
- Leader election on an arbitrary network
 - Gallager-Humblet-Spira

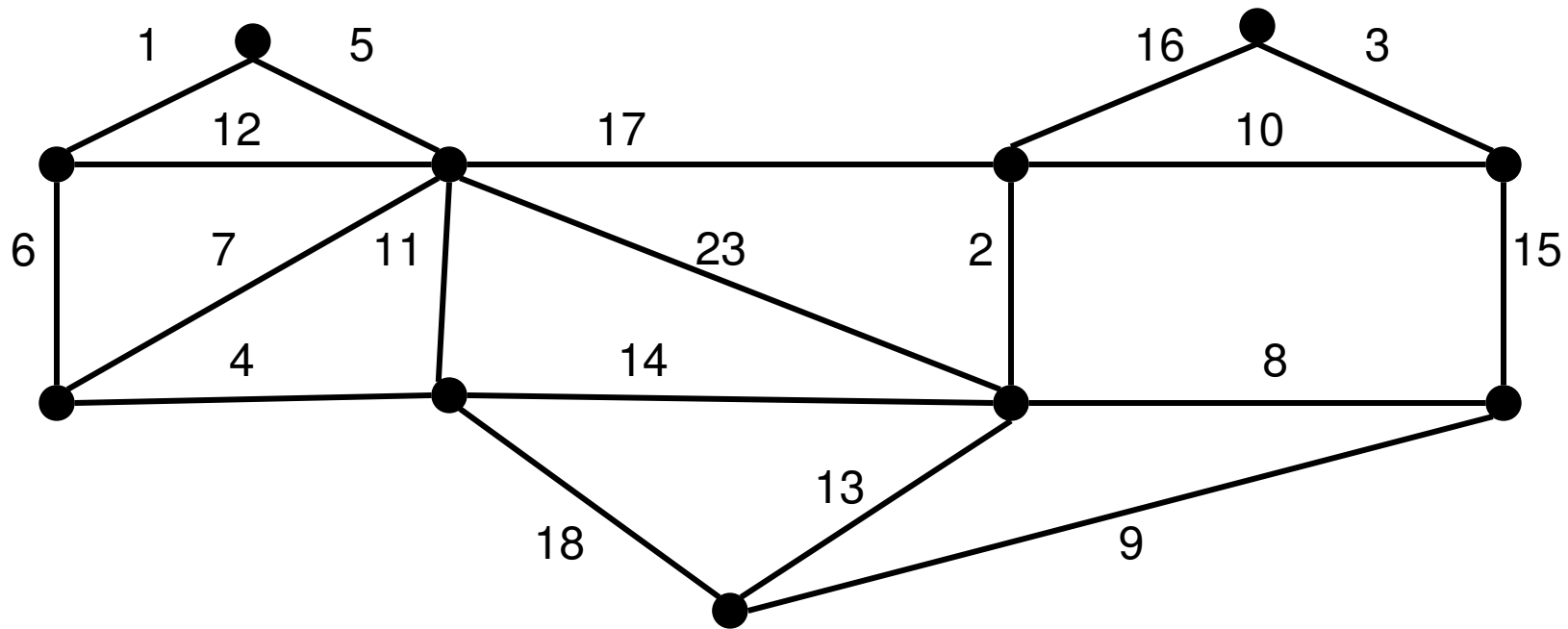
Minimal spanning trees

Def. Let $G = (V, E, \omega)$ be a weighted graph. The weight of a spanning tree T of G is the sum of the weights $\omega(e)$ of the edges $e \in E_T$. A minimal spanning tree (MST) is a spanning tree with minimal weight.

Prop. If all edge weights are different there exists a unique MST.

In case of a unique MST T any local decision that $e \in E_T$ is globally correct.

Kruskal's Algorithm

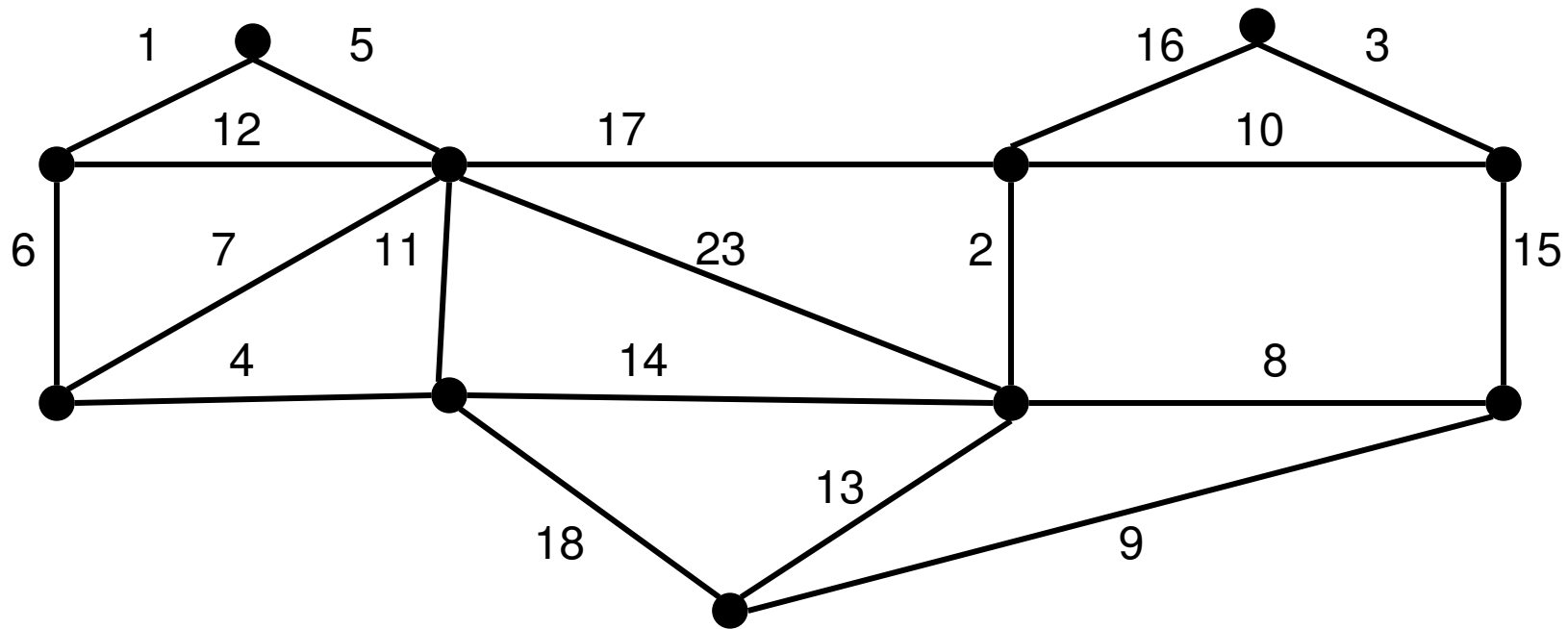


Fragments

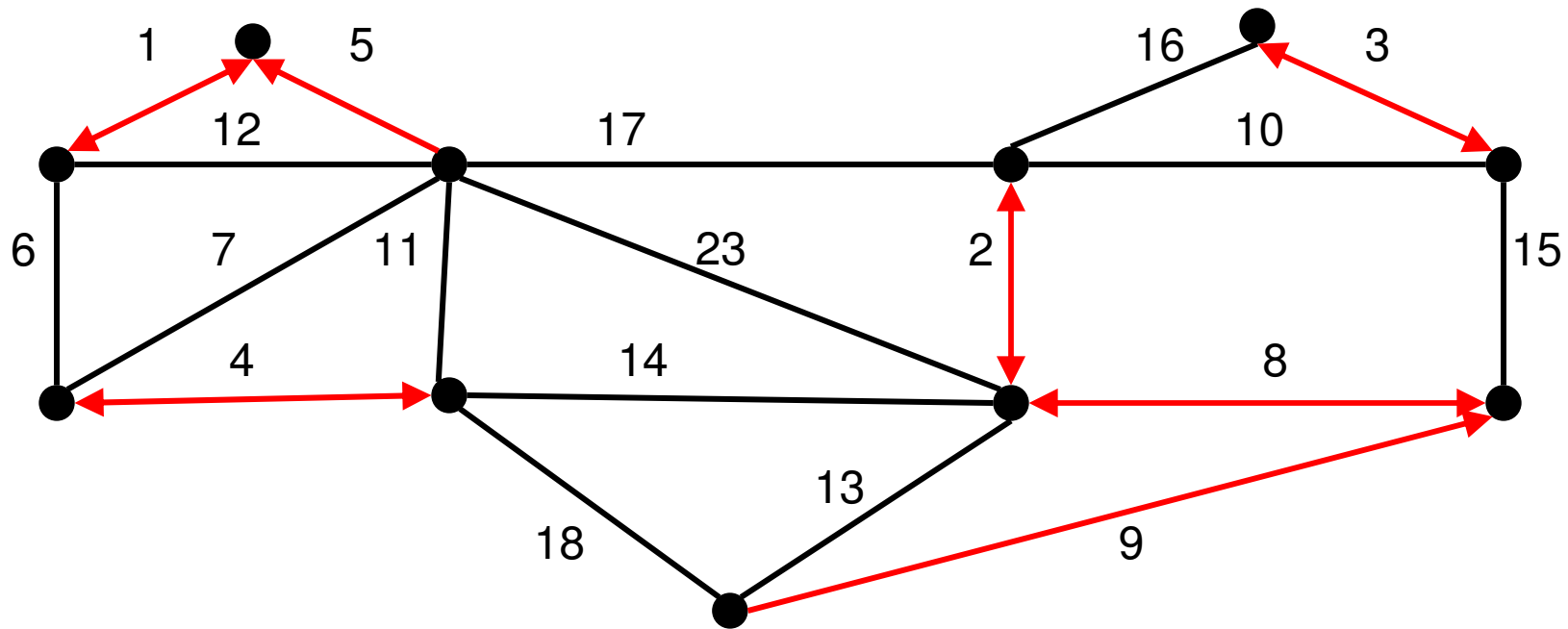
Def. Let $G = (V, E, \omega)$ be a weighted graph with MST T . Any subtree F of T is called a *fragment*. For F a fragment of T , an outgoing edge of F is an edge $e \in E$ with one endpoint in F and the other not.

Prop. If $F = (V_F, E_F)$ is a fragment of a MST T and $e = pq$ is the outgoing edge with minimal weight, then $F \cup \{e\} = (V_F \cup \{p, q\}, E_F \cup \{pq\})$ is a fragment of T .

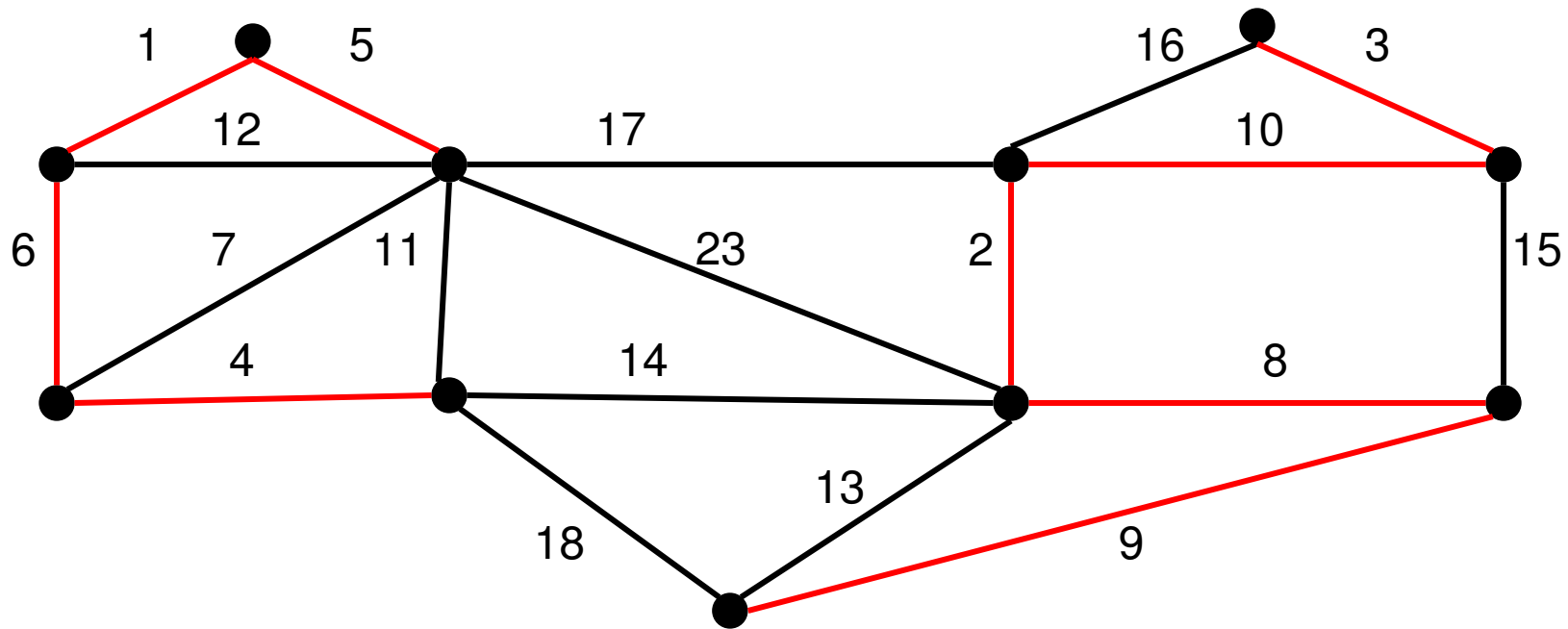
Distributed Synchronous Kruskal



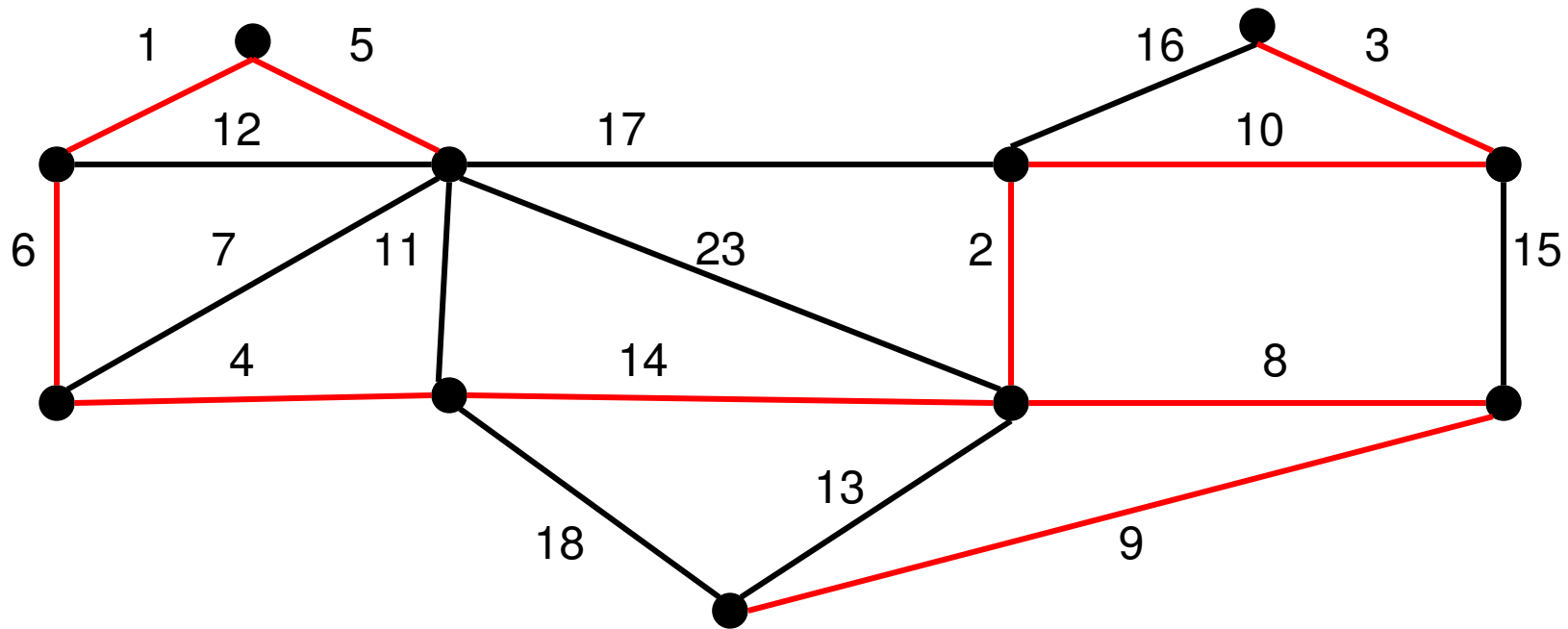
Distributed Synchronous Kruskal



Distributed Synchronous Kruskal



Distributed Synchronous Kruskal



GHS-properties

- Distributed variant of Kruskal's algorithm to compute a unique minimal spanning tree
- Combines (merges) fragments of the MST until the final tree is obtained.
- Also known as Mega-Merger (see book Santoro)
- Message complexity : $5|V| \log |V| + 2|E|$

Global description

- A collection of fragments is maintained whose union covers all nodes of the network.
- Initially each node forms a one-node fragment.
- Nodes of a fragment cooperate to find the minimal-weight outgoing edge.
- Fragments are combined using minimal-weight outgoing edges. An edge that is used to combine two fragments is called the *core edge* of the resulting fragment. Its endpoints are called *core nodes*.
- The combination process terminates with the MST when only one fragment remains.

Conclusion

- Leader election on a ring
 - LeLann, Chang-Roberts
 - Peterson/Dolev-Klawe-Rodeh
- Minimum Spanning Trees
 - Kruskal's Algorithm
- Leader election on an arbitrary network
 - Gallager-Humblet-Spira

Recitation ring

