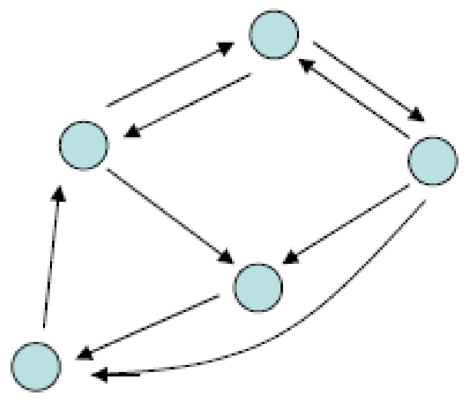
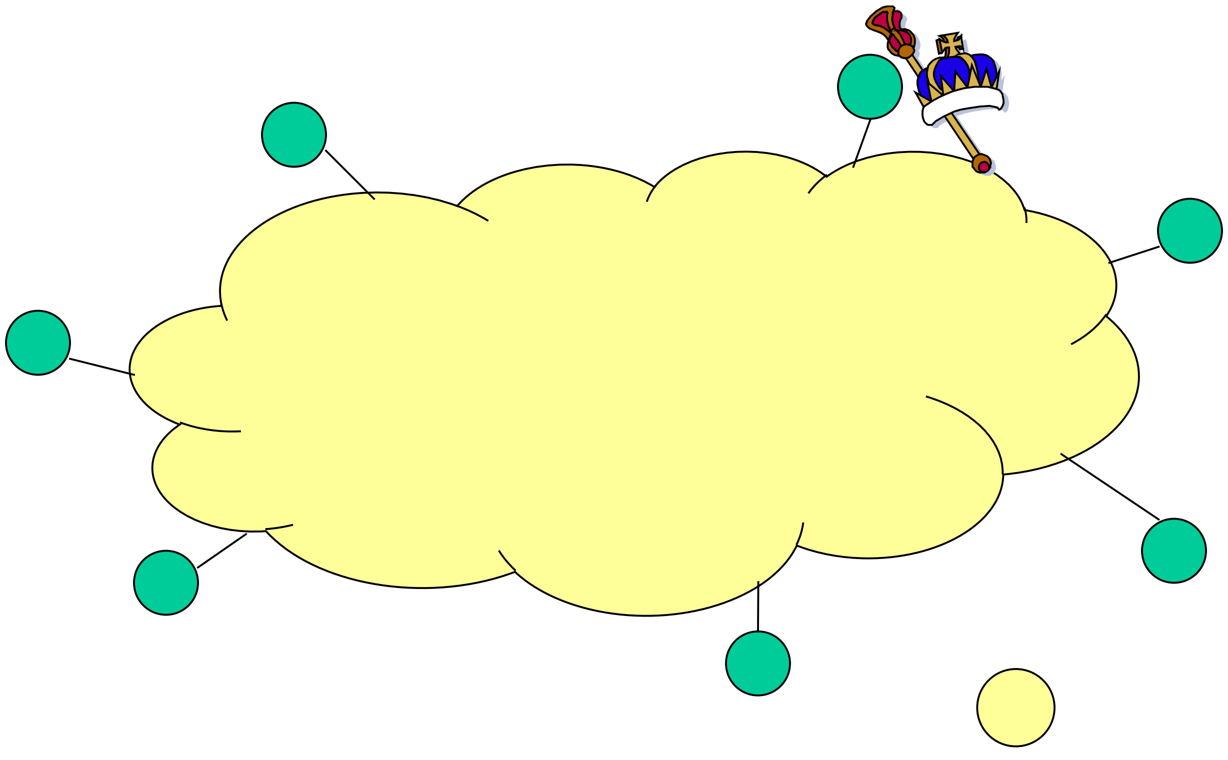
Leader election



- Network of processes.
- Want to distinguish exactly one, as the "leader".
- Eventually, exactly one process should output "leader" (set special status variable to "leader").
- Motivation: Leader can take charge of:
  - Communication
  - Coordinating data processing (e.g., in commit protocols)
  - Allocating resources
  - Scheduling tasks
  - Coordinating consensus protocols
  - …

Leader Election: the idea



- We study Leader Election in rings: Depicts basic characteristics of large family of problems in real distributed system (LeLann 1977)
- Illustrates techniques and principles
- Good for lower bounds and impossibility results

The Problem

- Final states of processes partitioned in two classes:

elected

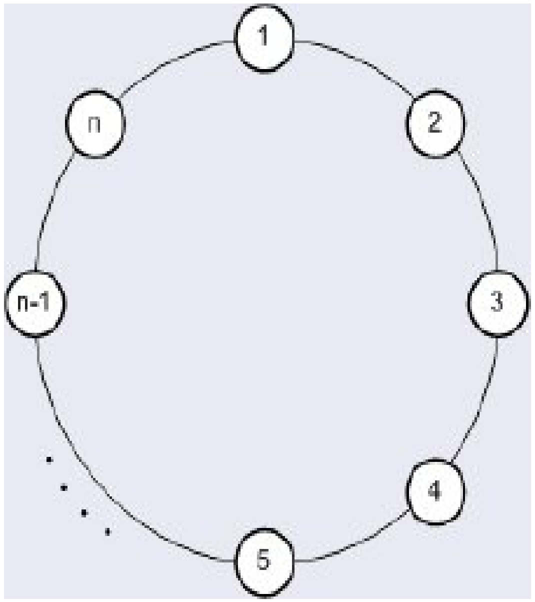
non-elected

- In every *admissible* execution, exactly one process (the *leader*) enters an elected state. All remaining enter a *non-elected* state.

Ring Networks

- We assume that the network graph G is a ring consisting of n processes.
- Numbered $1 \ldots n$ in the clockwise direction.
- We often count mod n, allowing 0 to be another name for process n, $n+1$ another name for process 1, ...
- The processes associated with the nodes of G do not know their indices, nor those of their neighbors.

Processes in a Ring



- We assume that message-generation and transition functions are defined in terms of local, relative names of their neighbors.

The LCR Algorithm

Algorithm LCR (informal)

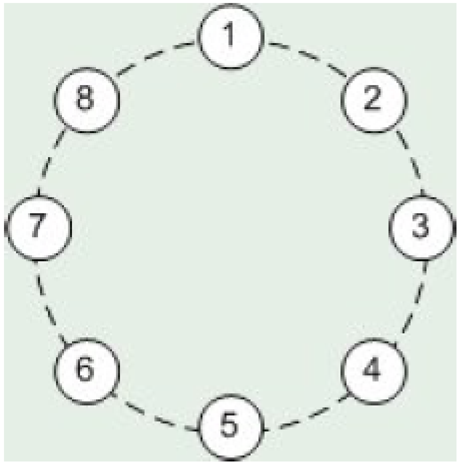
Each process sends its identifier around the ring. When a process receives an incoming identifier, it compares that identifier to its own. If the incoming identifier is greater than its own, it keeps passing the identifier; if it is less that its own, it discards the incoming identifier; if it is equal to its own, the process declares itself the leader.

- Decentralized, Uniform algorithm.
- Uses only unidirectional communication.
- Uses only comparison operations on the UIDs.
- Only the leader performs an output.

Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.

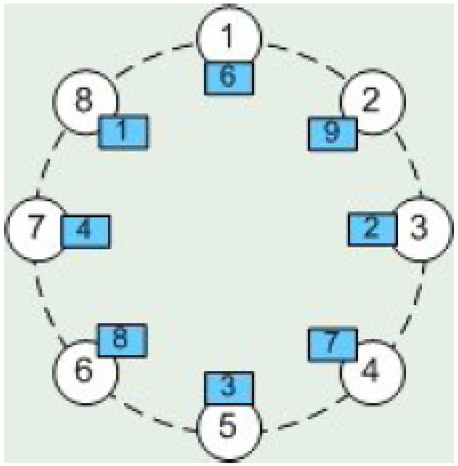
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.

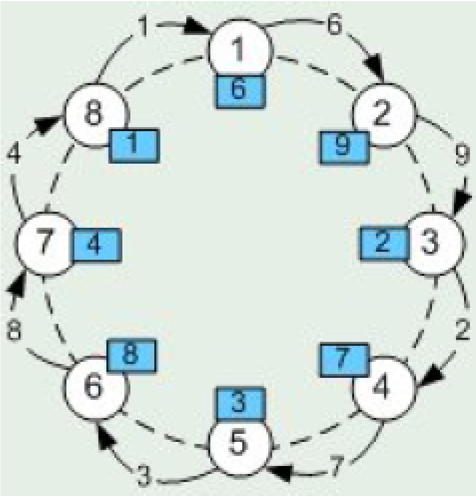
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round

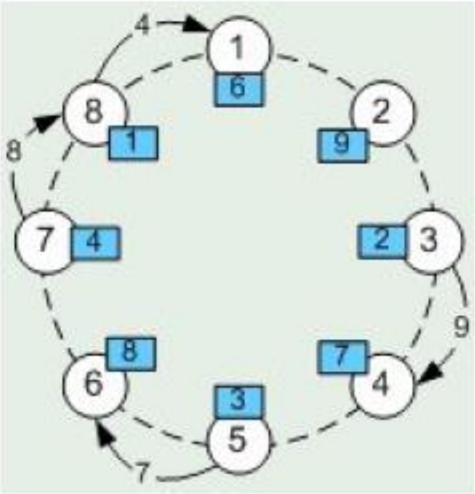
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round

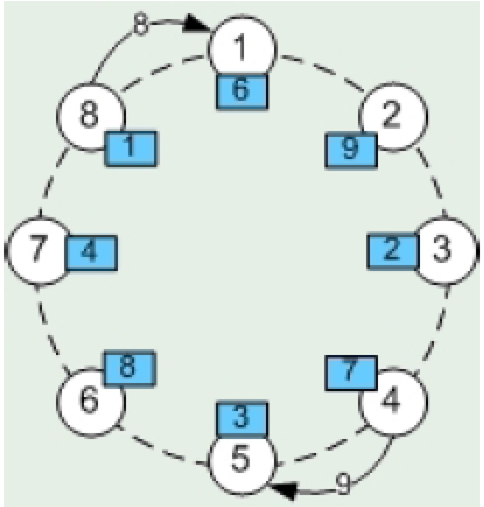
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round
- Next rounds

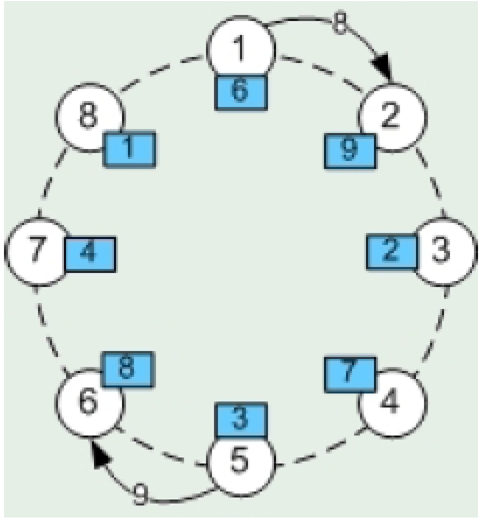
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round
- Next rounds

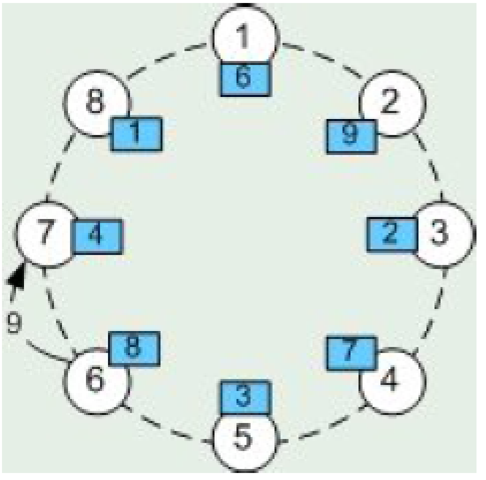
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round
- Next rounds

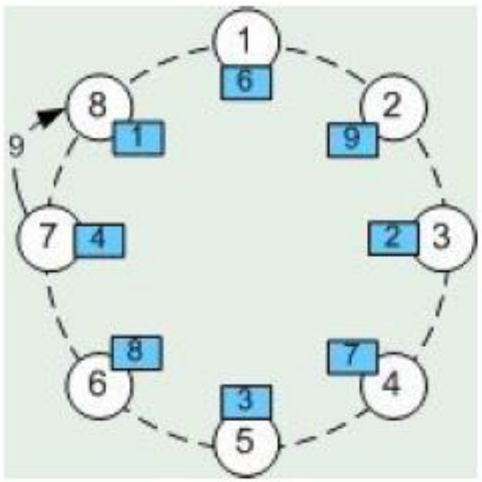
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round
- Next rounds

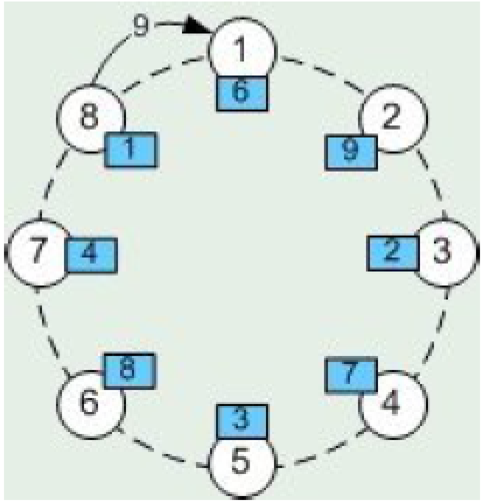
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round
- Next rounds

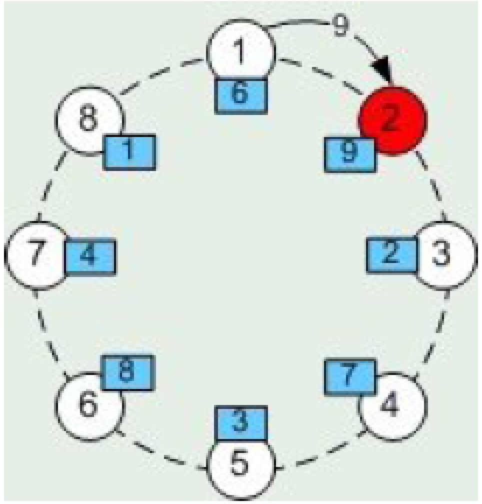
Synchronous Ring



Example Execution

- Let's assume a synchronous ring of $n = 8$ processes.
  - Processes are indexed from 1 to 8 clockwise.
- All processes have UIDs
  - Do now know the UIDs of the other processes.
- First round
- Second round
- Next rounds
- Leader election – process 2

Synchronous Ring



Variations of the problem

There are several variations of the problem:

- The ring can be either unidirectional or bidirectional.
- The number n of nodes may be either known or unknown to the processes.
- Processes may be identical or can be distinguished by each starting with a unique identifier (UID).
- It might be required that all not-elected processes eventually output the value “non-leader”.
- It might be required that all non-elected processes eventually output the UID of the leader.
- We might wish to elect k leaders.
- . . .

Algorithm Properties

Let i_{max} denote the index of the process with the maximum UID, and u_{max} its UID.

- Process i_{max} is elected leader at the end of round n.
- No other processes apart from i_{max} ends up in "elected" state.
- The time complexity is $\mathcal{O}(n)$
- The message complexity varies...
  - $\mathcal{O}(n^2)$ — worst case,
  - $\mathcal{O}(n)$ — best case,
  - $\mathcal{O}(n \log n)$ — average case.

Algorithm FloodMax

Every process maintains a record of the maximum UID it has seen so far (initially its own). At each round, each process propagates this maximum on all of its outgoing edges. After $diam(G)$ rounds, if the maximum value seen is the process's own UID, the process elects itself the leader; otherwise, it is a non-leader.

- Processes are not aware of the total number of processes (n).
- Processes are aware of the network diameter — $\delta = diam(G)$
- Comparison-based algorithm.

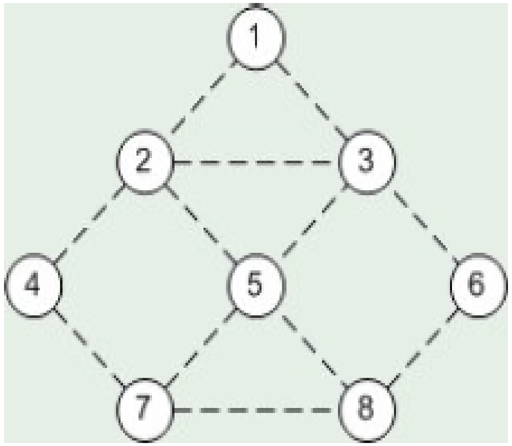
Pseudo-code for FloodMax

```
#DEFINE UID = <...>;
#DEFINE δ = <...>;
void main() {
    bool leader = false;
    int max_id = UID;
    for (int i = 0 ; i < δ; i++ ) {
        sendMessage(max_id);
        while (int new_msg = readMessage()) {
            if (new_msg > max_id)
                max_id = new_msg;
        }
    }
    if (max_id == UID)
      leader = true;
}
```

Example of Execution for FloodMax Algorithm

- Let a synchronous distributed system of $n = 8$ processes..
  - General network where $\delta = 3$
  - Processes are index $1 \ldots 8$

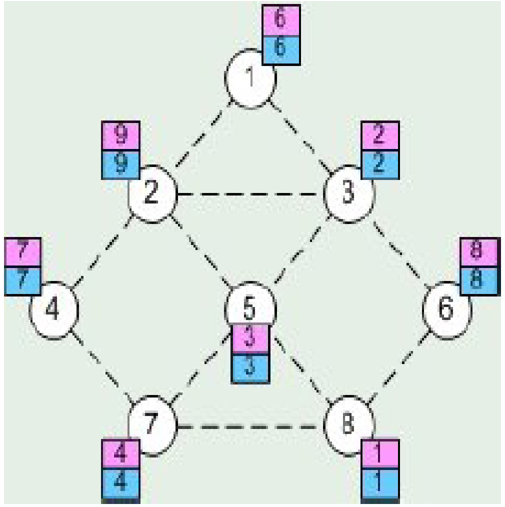
General Network



Example of Execution for FloodMax Algorithm

- Let a synchronous distributed system of $n = 8$ processes..
  - General network where $\delta = 3$
  - Processes are index $1 \ldots 8$
- The processes have UID.
  - Not aware of the UID of the other processes.

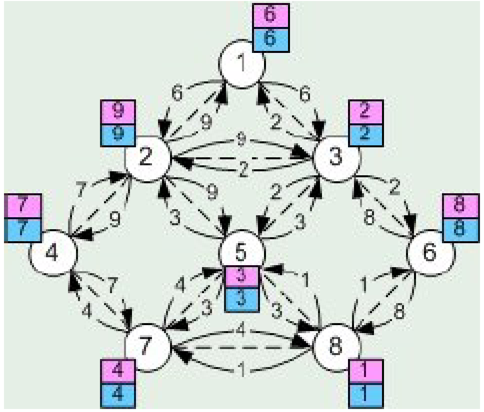
General Network



Example of Execution for FloodMax Algorithm

- Let a synchronous distributed system of $n = 8$ processes..
  - General network where $\delta = 3$
  - Processes are index $1 \dots 8$
- The processes have UID.
  - Not aware of the UID of the other processes.
- First Round

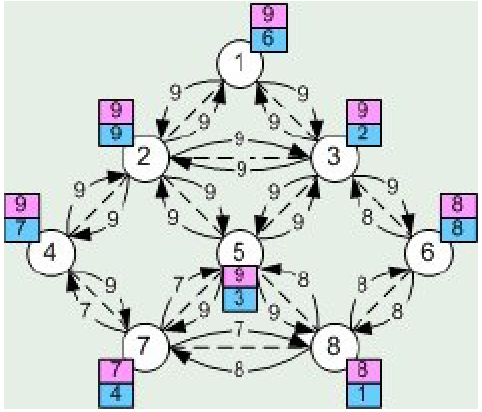
General Network



Example of Execution for FloodMax Algorithm

- Let a synchronous distributed system of $n = 8$ processes..
  - General network where $\delta = 3$
  - Processes are index $1 \dots 8$
- The processes have UID.
  - Not aware of the UID of the other processes.
- First Round
- Second Round

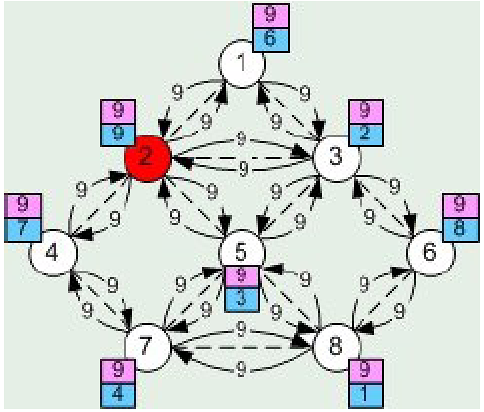
General Network



Example of Execution for FloodMax Algorithm

- Let a synchronous distributed system of $n = 8$ processes..
  - General network where $\delta = 3$
  - Processes are index $1 \dots 8$
- The processes have UID.
  - Not aware of the UID of the other processes.
- First Round
- Second Round
- Leader Election

General Network



Properties of FloodMax Algorithm

Let n processes and m channels, where the process with the highest UID is i_{max}.

- Process i_{max} is elected leader at the end of round δ.
- No other process is in state "elected".
- Time complexity is $\mathcal{O}\left(diam(G)\right)$.
- Message complexity $\mathcal{O}\left(diam(G) \cdot m\right)$.