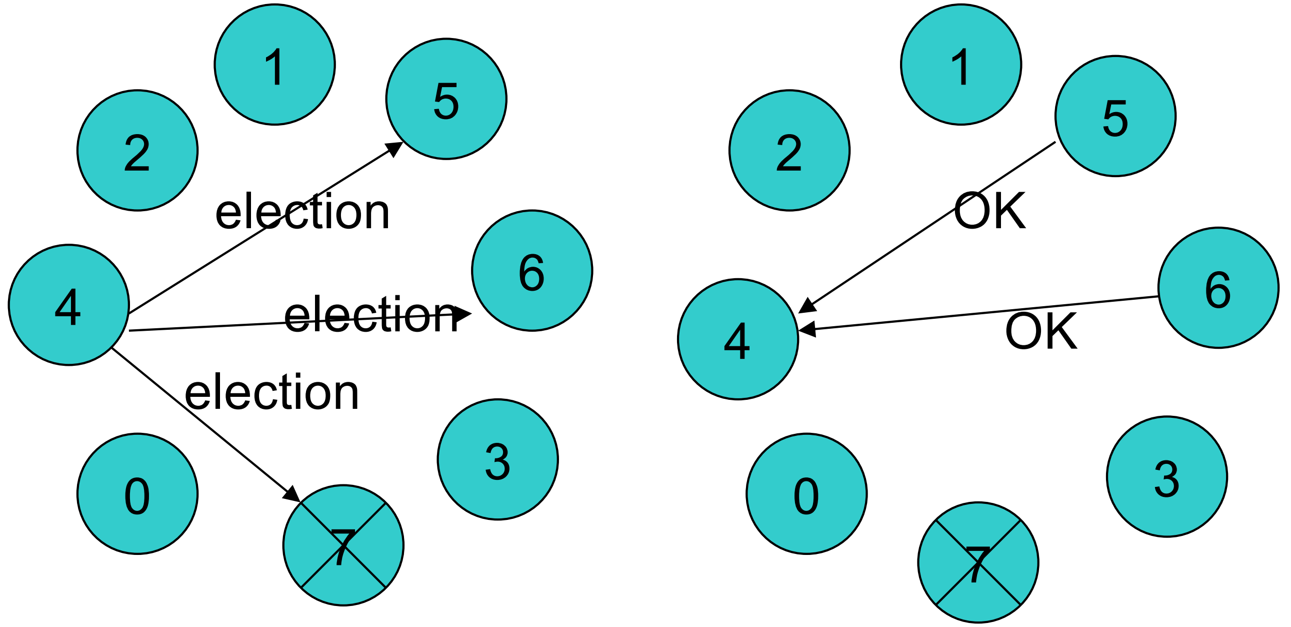
UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Leader Election Algorithms

The Bully Algorithm - Overview

- Process *p* calls an election when it notices that the coordinator is no longer responding.
- High-numbered processes “bully” low-numbered processes out of the election, until only one process remains.
- When?
  - When a crashed process reboots, it holds an election. If it is now the highest-numbered live process, it will win.
  - When some process recognize that the current leader is no longer active, I calls an election



Process *p* sends an election message to all *higher-numbered* processes in the system.

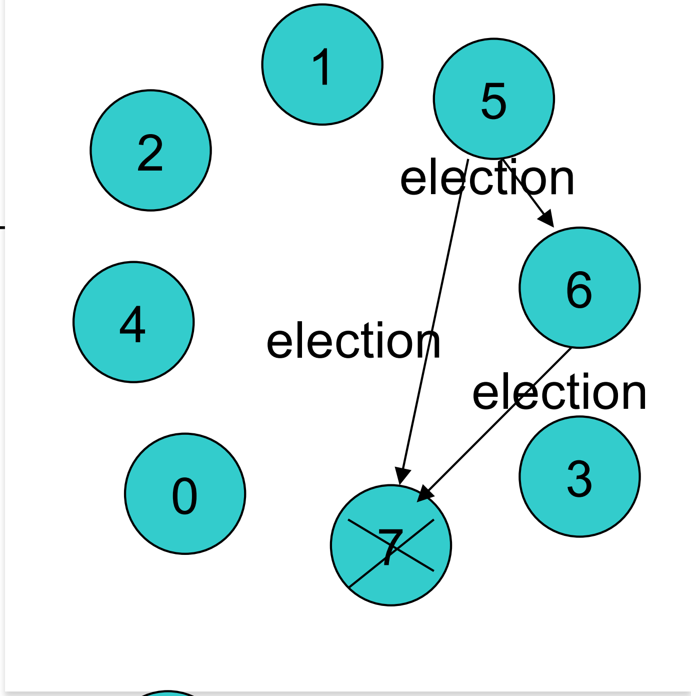
If no process responds, then *p* becomes the coordinator.

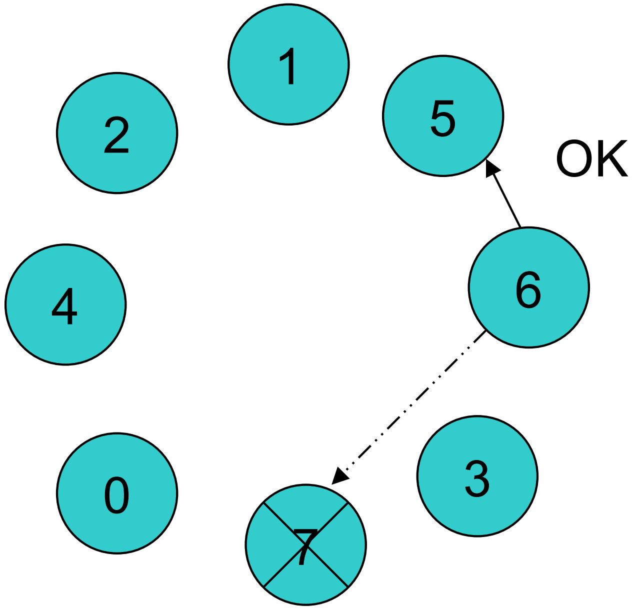
If a higher-level process (*q*) responds, it sends *p* a OK message that terminates *p*' s role in the algorithm

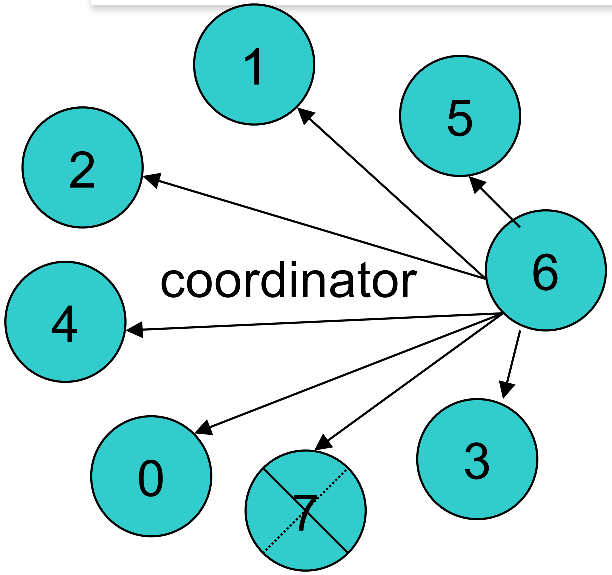
The process *q* now calls an election (if it has not already done so).

Repeat until no higher-level process responds. The last process to call an election "wins" the election.

The winner sends a message to other processes announcing itself as the new coordinator.







If 7 comes back on line, it will call an election

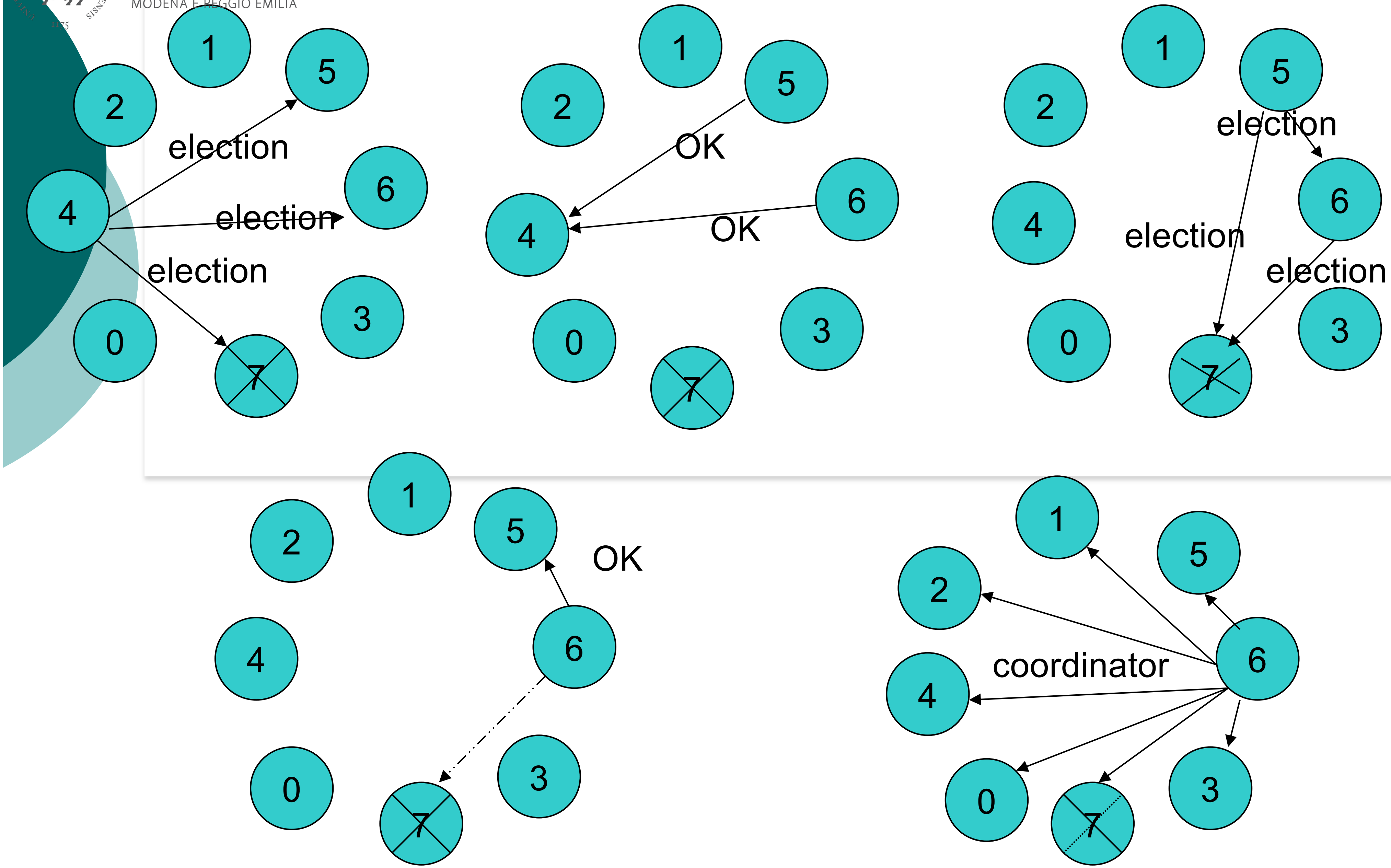

1
5
2
election
6
4
election
election
0
3
7
1
5
2
OK
6
4
OK
0
3
7
1
5
2
election
6
4
election
election
0
3
7
1
2
5
OK
4
6
0
3
7
1
5
2
6
coordinator
4
0
3
7

Figure 6-20

Analysis

- Works best if communication in the system has bounded latency so processes can determine that a process has failed by knowing the upper bound (UB) on message transmission time (T) and message processing time (M).
  - UB = 2 * T + M
- However, if a process calls an election when the coordinator is still active, the coordinator will win the election.

Complexity

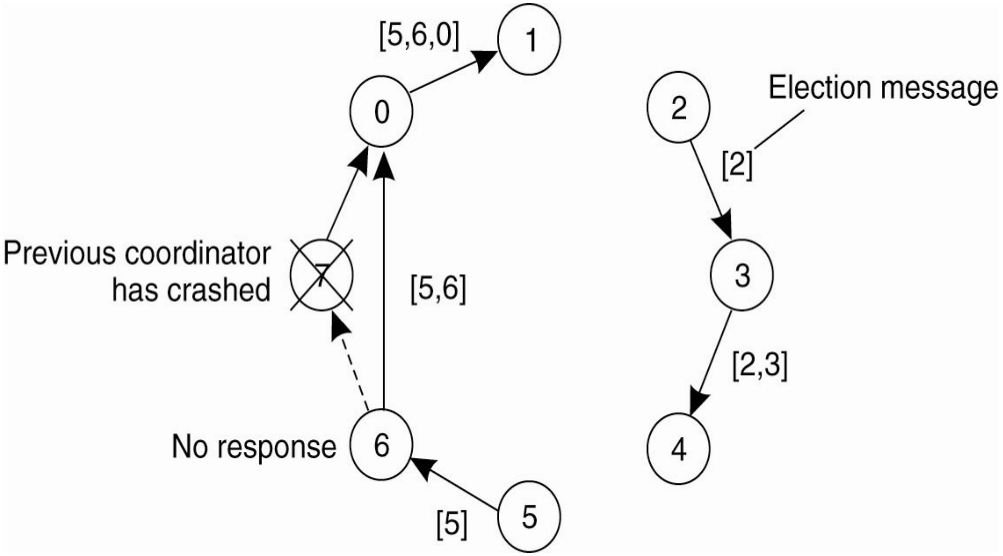
- The complexity (i.e., the number of messages required to complete) depends on who starts the election
- Given we have **n** processes
- If the election is started by the highest ID process
  - **n-1** messages are enough ("I will be the leader" said to everyone else!!)
- If the election is started by the lowest ID process
  - **n(n-1)** messages

The Ring Algorithm - Overview

- The ring algorithm assumes that the processes are arranged in a logical ring and each process is knows the order of the ring of processes.
- Processes are able to “skip” faulty systems: instead of sending to process j, send to j + 1.
- Faulty systems are those that don’t respond in a fixed amount of time.

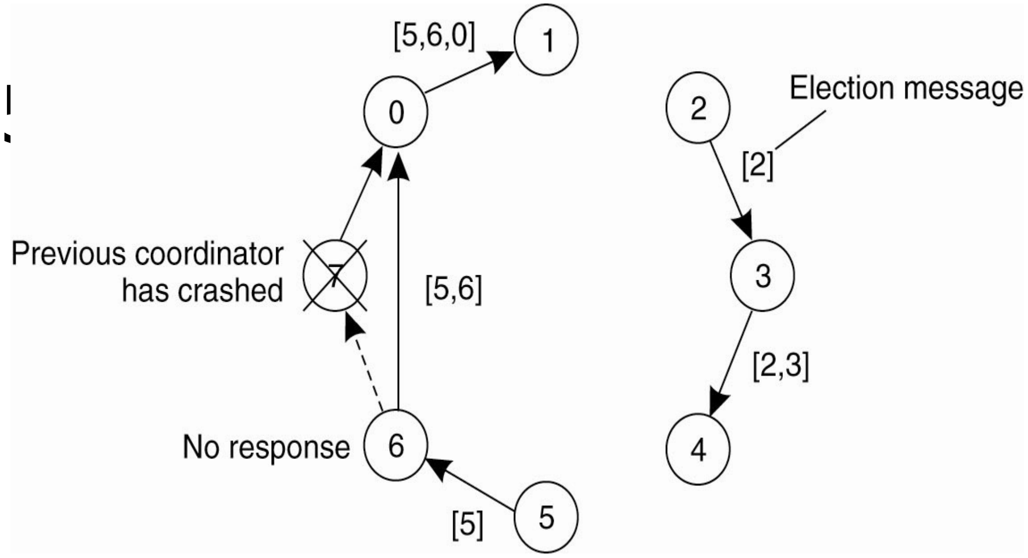
A Ring Algorithm

- P thinks the coordinator has crashed; builds an ELECTION message which contains its own ID number.
- Sends to first live successor
- Each process adds its own number and forwards to next.
- OK to have two elections at once.



Ring Algorithm - Details

- When the message returns to *p*, it sees its own process ID in the list and knows that the circuit is complete.
- P circulates a COORDINATOR message with the new high number.
- Here, both 2 and
  elect 6:
  [5,6,0,1,2,3,4]
  [2,3,4,5,6,0,1]



Complexity

- Given we have **n** processes
- The complexity is always **2n**