Coordination Algorithms

Reference for study:

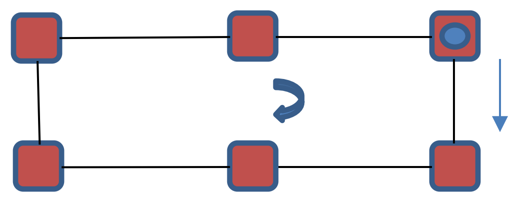
Van Steen, Tanenbaum, "Distributed Systems", chapter 6

Mutual Exclusion

- Problem Statement
  - Guarantee mutual exclusive access to shared resources by multiple processes in a DS
- Different types of algorithms
  - Token-based
  - Permission-based
    - centralized
    - decentralized

Token Ring Mutual Exclusion

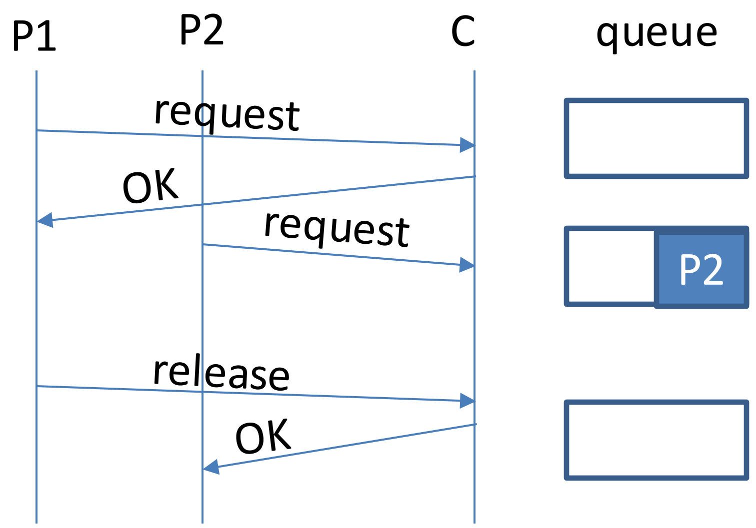
- Processes organized in a ring overlay
- 1 token in the system, continually circulating on the ring



- A process P willing to access the resource waits for the token. When the token arrives at P:
  - P starts accessing the resource and keeps the token until its access is finished
  - Then, P passes the token to the next process in the ring

Centralized Mutual Exclusion

- Central manager (C) of shared resource(s)
- A process P willing to access a resource sends request to C and waits for permission response
- C delays permissions while the resource is engaged



Decentralized Mutual Exclusion

- Based on Lamport clocks (totally ordered multicast)
- Requester sends request to all processes (including itself) and waits for permission from every process
  - responses to requests received while accessing the resource are delayed
  - in case of conflict (receiver also wants to access the resource), the request with the lower timestamp wins

Performance Comparison

Algorithm		#Messages/access	delay before entry (# messages)
Permission based	Centralized	3	2
	Decentralized	2 (N-1)	2 (N-1)
Token based	Token ring	1...	0 ... N-1

N: number of processes

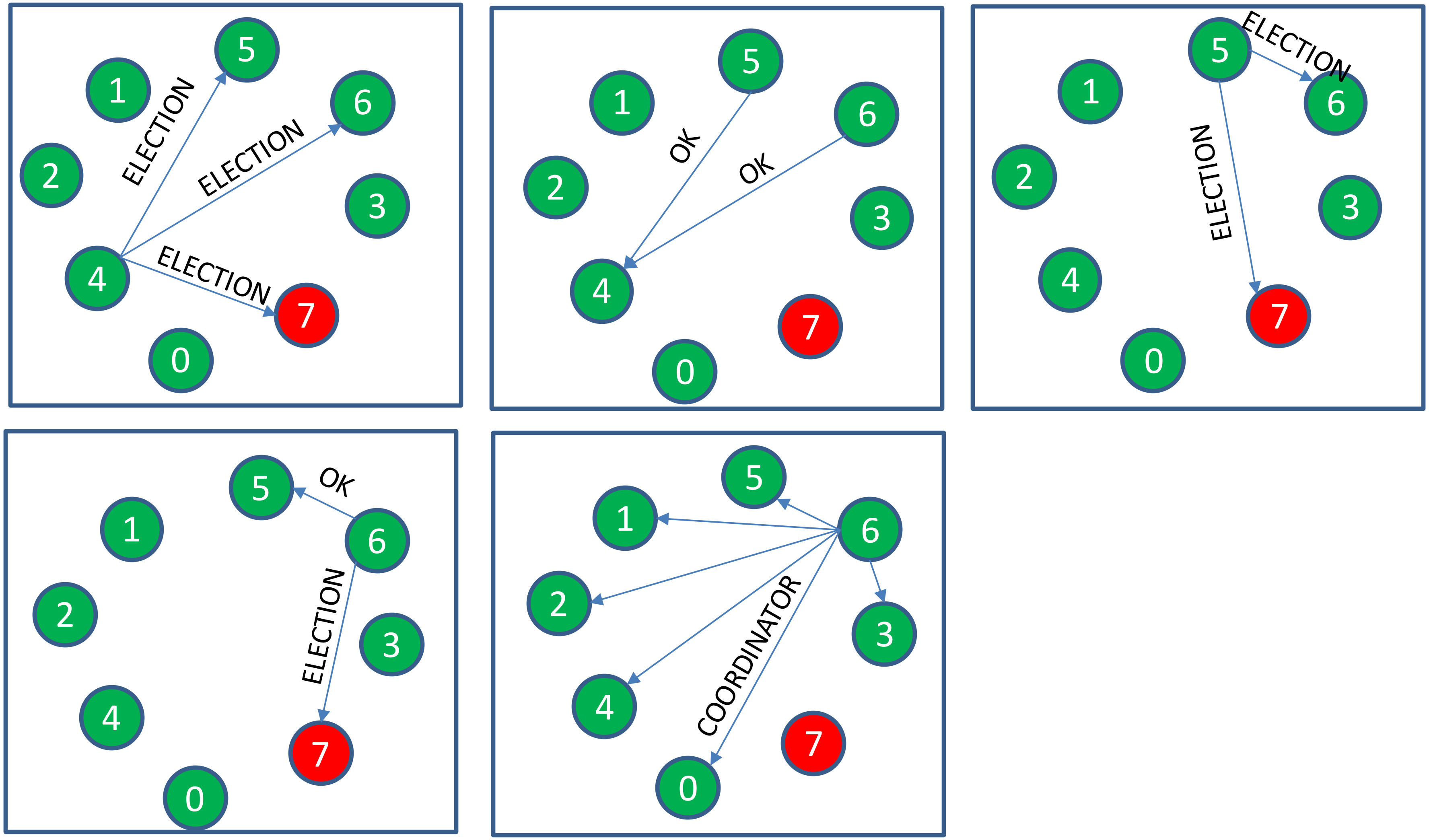
Election

- Problem Statement:
  - Elect a process (e.g., a coordinator) in a group of processes
- Assumptions:
  - Each process has a unique identifier id(P) (it can be obtained by means of a naming system)
  - Each process knows all the other processes in the group
  - Processes in the group can be up and running or down, but channels are reliable
- Requirements
  - The algorithm must elect the up process having the highest id
  - At the end of the algorithm, all processes agree about who is the elected process

Election: The Bully Algorithm

- Let $id(P_k)=k$
- The algorithm starts when a process P_k detects the coordinator is missing and decides to hold an election:
  - P_k sends ELECTION message to $P_{k+1}, P_{k+2}, \ldots, P_{N-1}$
  - If no one responds, P_k wins the election, else P_k gives up
- Whenever P_i receives an ELECTION,
  - P_i responds with OK (means it is alive)
  - P_i holds an election (if it was not already holding one)
- Eventually all but one processes give up and one wins
- The winner finally informs the other processes

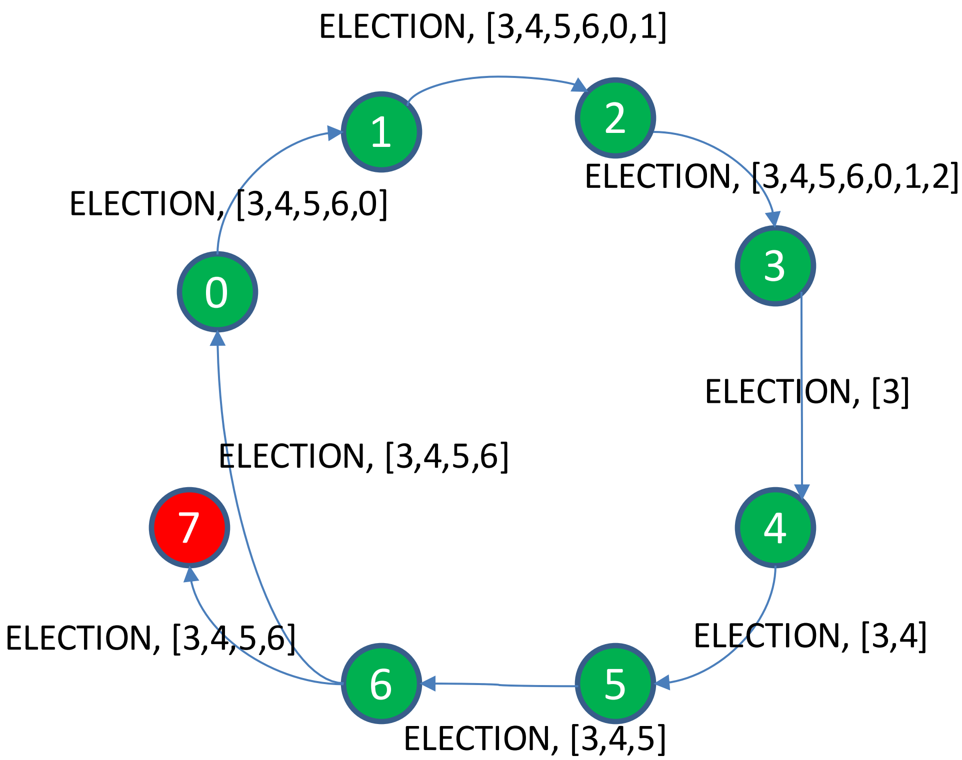
Election: The Bully Algorithm

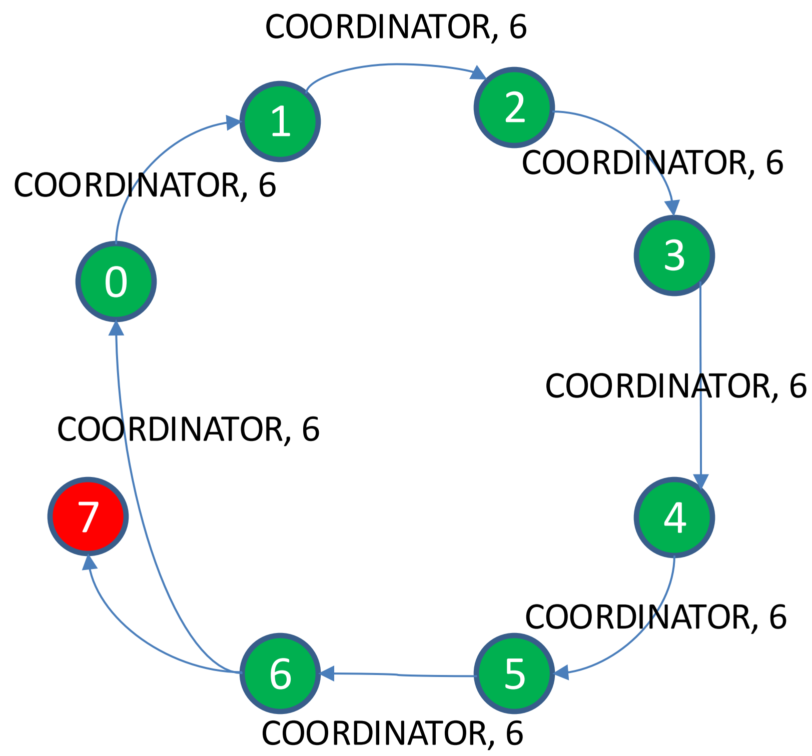


Election: Ring Algorithm

- Processes are ordered (e.g., by id) in a logical ring
- The algorithm starts when a process P_k detects the coordinator is missing and decides to hold an election:
  - P_k sends ELECTION message to its successor in the ring
  - if no response is received, P_k sends the message to the next successor in the ring and so on until one responds
- Each ELECTION message carries the list of senders
- When eventually the message gets back to the starter P_k
  - P_k stops the circulation of the message
  - P_k computes the winner and circulates a COORDINATOR msg on the ring (in the same way) to inform about the winner

Election: Ring Algorithm





Consensus

- A more general coordination problem
- Problem statement
  - Let n processes, each one proposing an input value, agree on the same output value
- Properties
  - Mutual exclusion, leader election: special cases of consensus
  - But consensus can be solved by using leader election or mutual exclusion algorithms