The Bully Algorithm

Elections

- *Election*: algorithm to choose a process to play a particular role in a distributed system
- A process *calls* an election
- Processes *participate* or not in an election
- Vote identifier: <vote,pid>
- Requirements
  - *Safety*: a particpant process p_i has $elected_i = \perp$ or $elected_i = P$
  - *Liveness*: all processes p_i partecipate and evenutally set $elected_i \neq \perp$

The bully algorithm

- Assumptions: process can crash, message delivery reliable and synchronous, complete knowledge of vote_id of all other peers
- Synchronous: message turnaround time bound by $T = T_{transmission} + T_{process}$
- Three types of messages:
  - *election* message: to call elections
  - *answer* message: to vote
  - *coordinator* message: to announce own acting as coordinator

The Bully Algorithm (Garcia Molina, 1982)

- P broadcasts an election message (inquiry) to all other processes with higher process IDs.

- If P hears from no process with a higher process ID than itself, it wins the election and broadcasts victory.

- If P hears from a process with a higher ID, P waits a certain amount of time for that process to broadcast itself as the leader. If it does not receive this message in time, it re-broadcasts the election message.

- If P gets an election message (inquiry) from another process with a lower ID it sends an "I am alive" message back and starts new elections.

The Bully Algorithm

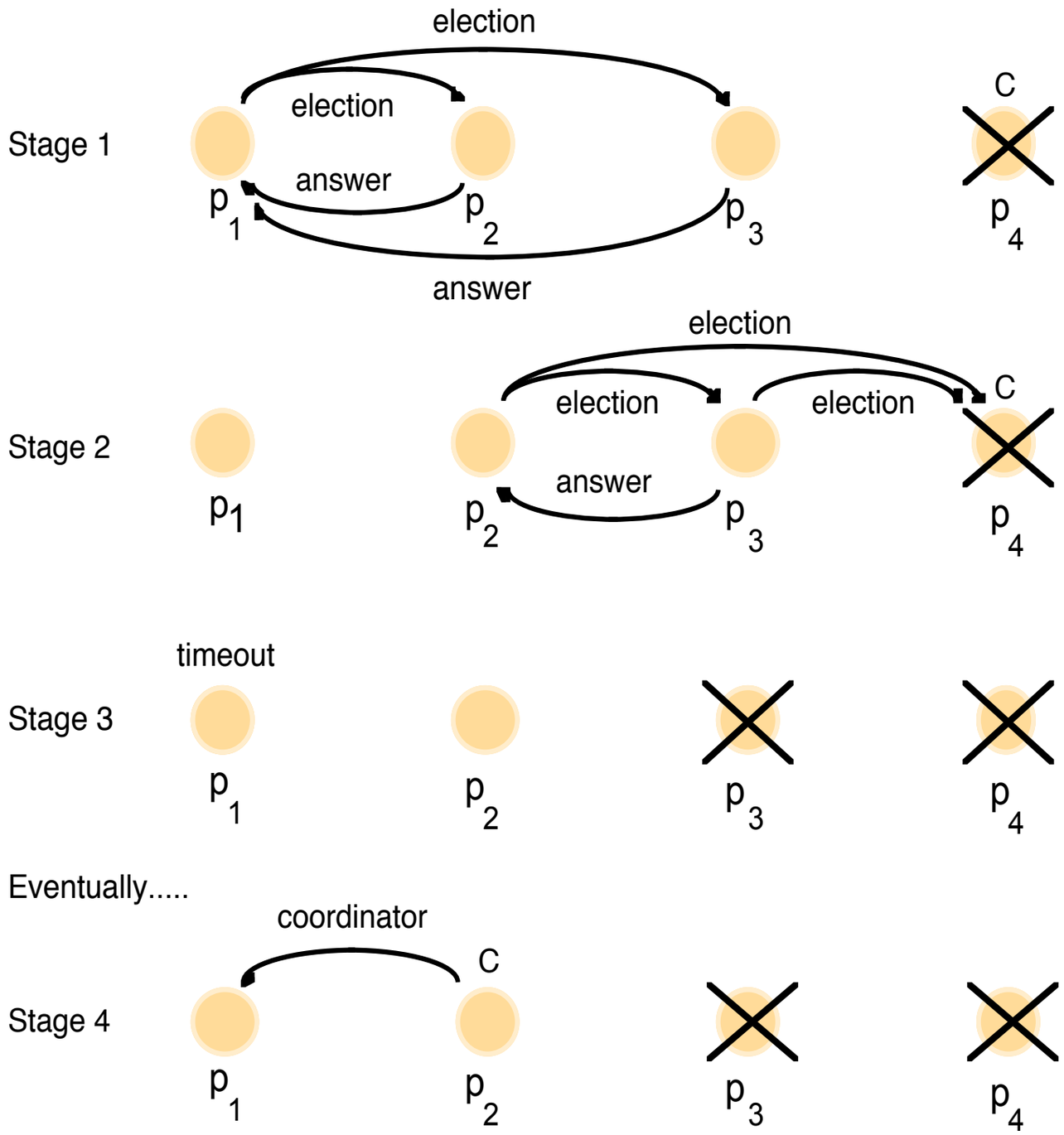
- Note that if P receives a victory message from a process with a lower ID number, it immediately initiates a new election. This is how the algorithm gets its name - a process with a higher ID number will bully a lower ID process out of the coordinator position as soon as it comes online.

The bully algorithm

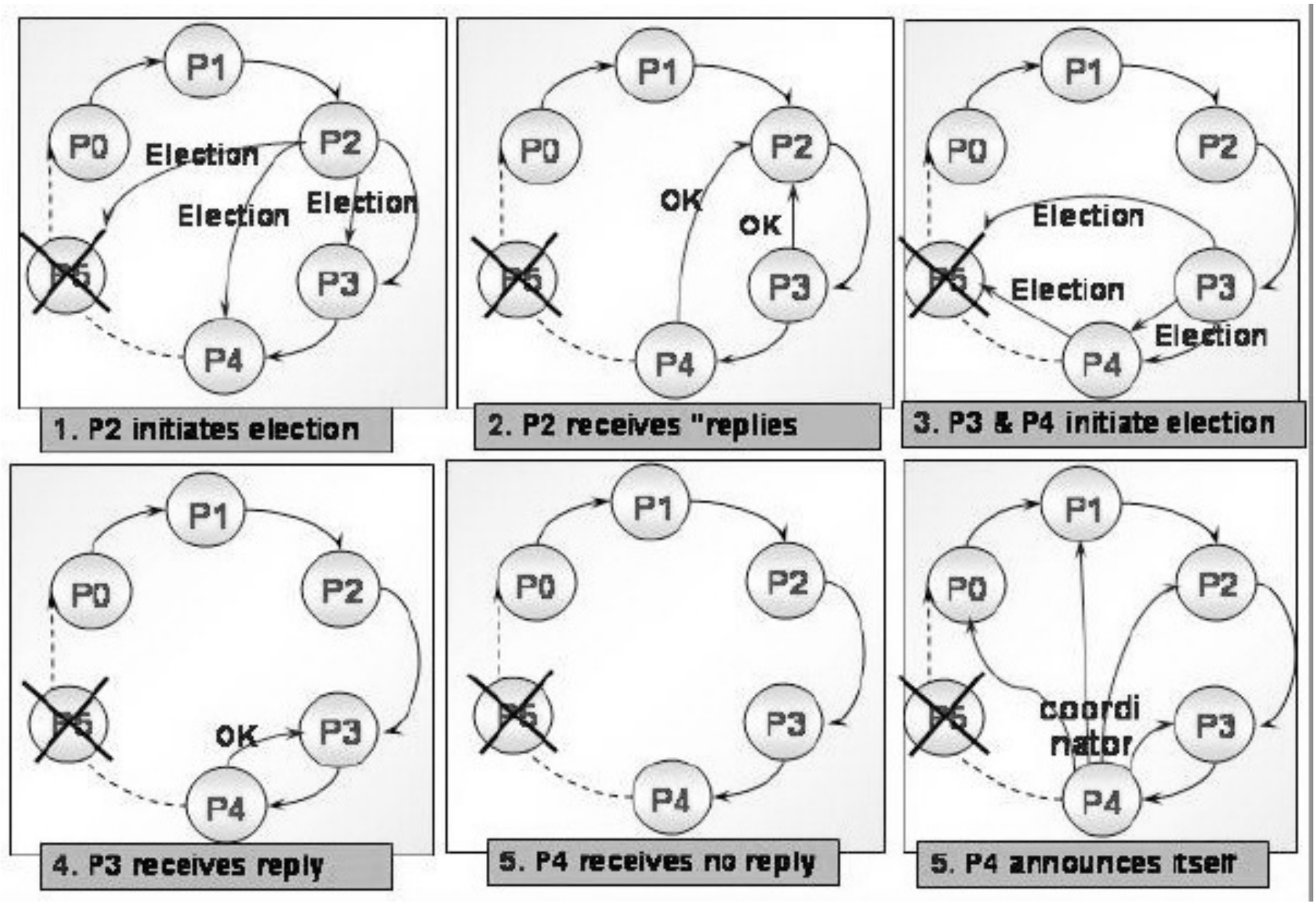
- The process with highest vote_id sends coordinator message
- A process needing a coordinator sends an election message, if no answer within time T, then it sends a coordinator message
- If a process receives a coordination message, it sets its elected variable
- If a process receives an election message, it answers and begins another election (if needed)
- If a new process starts to coordinate (highest vote_id), it sends a coordinator message and "***bullies***" the current coordinator out

The bully algorithm

The election of coordinator p_2, after the failure of p_4 and then p_3



The Bully Algorithm



The bully algorithm

- Liveness condition is met by the reliable and bounded message transmission assumption
- Safety is met only if no new process starts during an election (because there is no guarantee on order of message delivery if two processes send coordinator messages concurrently which may happen in case e new process starts)
- Bandwith
  - $O(N^2)$ worst case, i.e., the process with least vote detects the failure of the coordinator
- Turnaround
  - One message