Distributed Systems – Theory
12. Election algorithms

Why? And assumptions

- Many distributed algorithms require one process to act as coordinator, initiator, sequencer, or otherwise perform some special role.
- Example: the coordinator in the centralized mutual exclusion algorithm.
- Problem: look at algorithms for electing a coordinator.
- The algorithms differ in the way they do the location.
- Remark: If all processes are exactly the same, with no distinguishing characteristics, there is no way to select one of them to be special.
- Assumption 1: each process has a unique number,
  - for example its network address (for simplicity, we will assume one process per machine).
  - In general, election algorithms attempt to locate the process with the highest process number and designate it as coordinator.
- Assumption 2: every process knows the process number of every other process.
- Assumption 3: what the processes do not know is which ones are currently up and which ones are currently down.
- Goal of an election algorithm: ensure that when an election starts, it concludes with all processes agreeing on who the new coordinator is to be.

Bully Algorithm (Garcia-Molina)

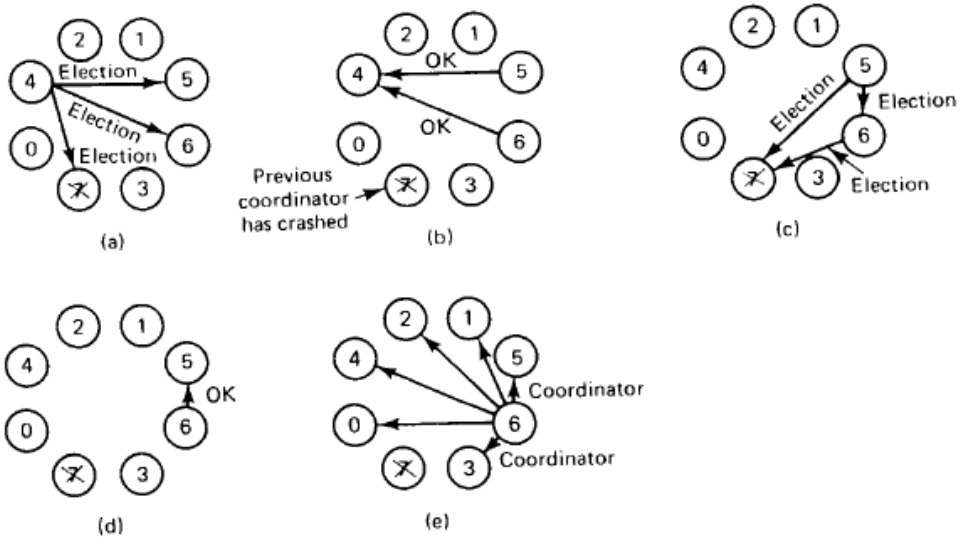
- When a process notices that the coordinator is no longer responding to requests, it initiates an election.
- A process, P, holds an election as follows:

1. P sends an ELECTION message to all processes with higher numbers
2. If no one responds, P wins the election and becomes coordinator
3. If one of the higher-ups answers, it takes over. P's job is done

- If a process can get an ELECTION message from one of its lower-numbered colleagues.
  - message arrives => the receiver sends an OK message back to the sender to indicate that he is alive and will take over.
  - The receiver then holds an election, unless it is already holding one.
- Eventually, all processes give up but one, and that one is the new coordinator.
  - It announces its victory by sending all processes a message telling them that starting immediately it is the new coordinator.
- If a process that was previously down comes back up, it holds an election.
  - If it happens to be the highest-numbered process currently running, it will win the election and take over the coordinator's job.
- Thus the biggest guy in town always wins, hence the name "bully algorithm."

Bully Algorithm - example

- The group consists of 8 processes.
- Previously process 7 was the coordinator, but it has just crashed.
- Process 4 is the first one to notice this, so it sends ELECTION messages to all the processes higher than it, namely Processes 5 and 6 both respond with OK.
- Upon getting the first of these responses, 4 knows that its job is over.
- Both 5 and 6 hold elections, each one only sending messages to those processes higher than itself.
- Process 6 tells 5 that it will take over.
- 6 knows that 7 is dead and that it is the winner.
- 6 announces this by sending a COORDINATOR message to all running processes.
- When 4 gets this message, it can now continue with the operation it was trying to do when it discovered that 7 was dead, but using 6 as the coordinator this time.
- If process 7 is ever restarted, it will just send all the others a COORDINATOR message and bully them into submission.



A Ring Algorithm

- Based on the use of a ring.
- Assume: the processes are physically or logically ordered, so that each process knows who its successor is.
- When any process notices that the coordinator is not functioning, it builds an ELECTION message containing its own process number and sends the message to its successor.
- If the successor is down, the sender skips over the successor and goes to the next member along the ring, or the one after that, until a running process is located.
- At each step, the sender adds its own process number to the list in the message.
- Eventually, the message gets back to the process that started it all.
- That process recognizes this event when it receives an incoming message containing its own process number.
- At that point, the message type is changed to COORDINATOR and circulated once again, this time to inform everyone else who the coordinator is (the list member with the highest number) and who the members of the new ring are.
- When this message has circulated once, everyone goes back to work.

Ring algorithm - example

- two processes, 2 and 5, discover simultaneously that the previous coordinator, process 7, has crashed.
- Each of these builds an ELECTION message and starts circulating it.
- Eventually, both messages will go all the way around, and both 2 and 5 will convert them into COORDINATOR messages, with exactly the same members and in the same order.
- When both have gone around again, both will be removed.
- It does no harm to have extra messages circulating – at most it wastes a little bandwidth.