# Let's conduct an election

Many distributed algorithms require one process to act as coordinator, initiator, or otherwise perform some special role. In general, it does not matter which process takes on this special responsibility, but one of them has to do it. In this exam, you will implement an algorithm for electing a **coordinator**.

If all processes are exactly the same, with no distinguishing characteristics, there is no way to select one of them to be special. Consequently, you will assume that each process $\mathbf{P}$ has a unique identifier $\mathbf{id(P)}$.

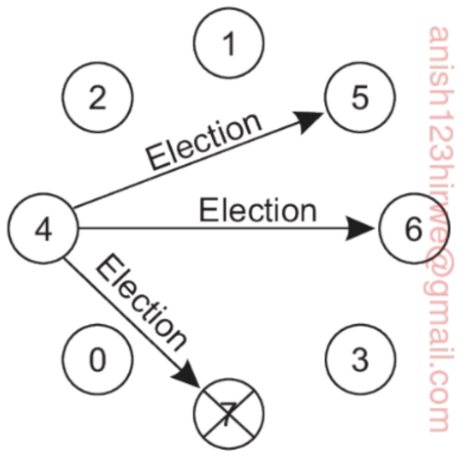
Furthermore, you also assume that every process knows the identifier of every other process. In other words, each process has complete knowledge of the process group in which a coordinator must be elected. What the processes do not know is which ones are currently up and which ones are currently down. The goal of an election algorithm is to ensure that when an election starts, it concludes with all processes agreeing on who the new coordinator is to be.

A well-known solution for electing a coordinator is the **bully algorithm**. Consider $\mathbf{N}$ processes $\mathbf{\{P_0, \ . \ . \ . \ , \ P_{N-1}\}}$ and let $\mathbf{id(Pk) = k}$. When any process notices that the coordinator is no longer responding to requests, it initiates an election. A process, $\mathbf{P_k}$
, holds an election as follows:

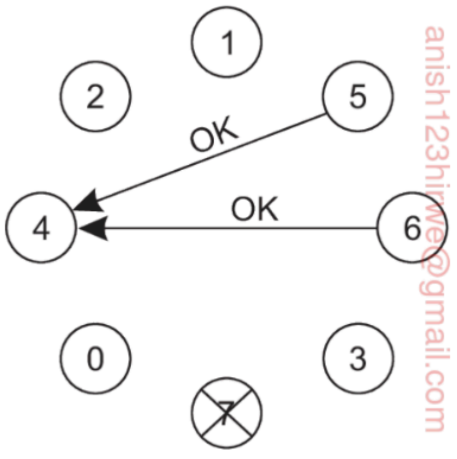
1. $\mathbf{P_k}$ sends an **ELECTION** message to all processes with higher identifiers: $\mathbf{P_{k+1}}$, $\mathbf{P_{k+2}, \ldots, P_{N-1}}$.
2. If no one responds, $\mathbf{P_k}$ wins the election and becomes coordinator.
3. If one of the higher-ups answers, it takes over and $\mathbf{P_k}$'s job is done.

At any moment, a process can get an **ELECTION** message from one of its lower-numbered colleagues. When such a 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.

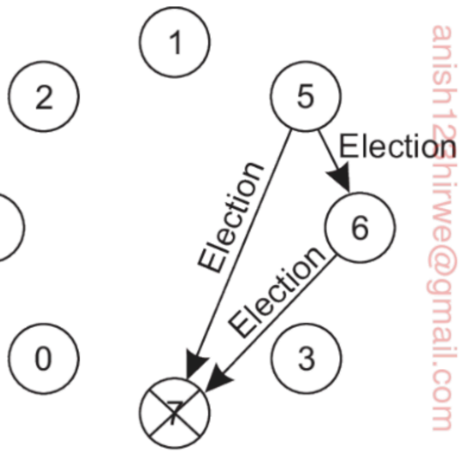
**Implement the scenario shown in the following figure:**



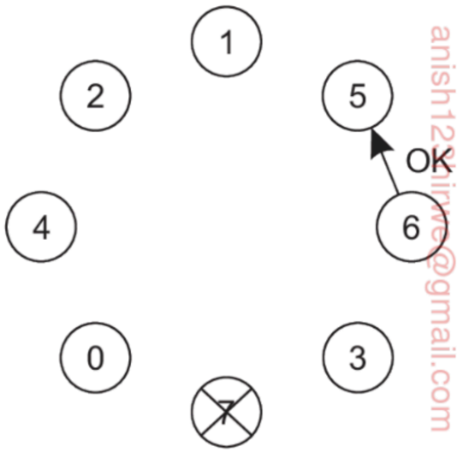
(a)



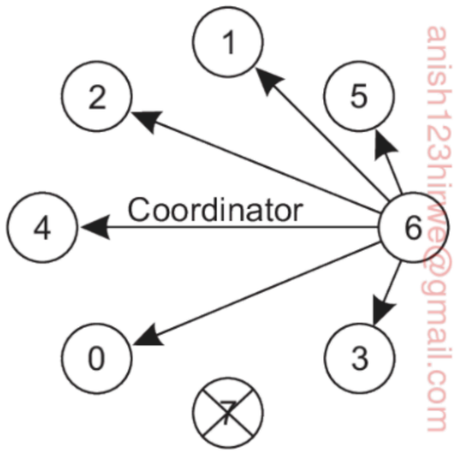
(b)



(c)



(d)



(e)

- Write a socket-based program where each process is represented as a separate socket client.
- The group consists of eight processes, with identifiers numbered from **0 to 7.**
- Previously process $\mathbf{P_7}$ was the coordinator, but it has just crashed.
- Process $\mathbf{P_4}$ is the first one to notice this, so it sends **ELECTION** messages to all the processes higher than it, namely **P5, P6, and P7**, as shown in above Figure(a).
- Processes **P5 and P6** both respond with OK, as shown in Figure(b)
- Upon getting the first of these responses, **P4** knows that its job is over, knowing that either one of **P5 or P6** will take over and become coordinator. Process **P4** just sits back and waits to see who the winner will be.
- In Figure(c), both **P5 and P6** hold elections, each one sending messages only to those processes with identifiers higher than itself.
- In Figure (d), **P6** tells **P5** that it will take over. At this point **P6** knows that **P7** is dead and that it (**P6)** is the winner.
- **P6** takeover by sending a **COORDINATOR** message to all running processes.