Election Algorithms

- Determine where a new copy of the coordinator should be restarted.
- Assume that a unique priority number is associated with each active process in the system, and assume that the priority number of process P_i is i.
- Assume a one-to-one correspondence between processes and sites.
- The coordinator is always the process with the largest priority number. When a coordinator fails, the algorithm must elect that active process with the largest priority number.
- Two algorithms, the bully algorithm and a ring algorithm, can be used to elect a new coordinator in case of failures.

Bully Algorithm

- Applicable to systems where every process can send a message to every other process in the system.
- If process P_i sends a request that is not answered by the coordinator within a time interval T, assume that the coordinator has failed; P_i tries to elect itself as the new coordinator.
- P_i sends an election message to every process with a higher priority number, P_i then waits for any of these processes to answer within T.

Bully Algorithm (Cont.)

- If no response within T, assume that all processes with numbers greater than i have failed; P_i elects itself the new coordinator. P_i then sends a message to all processes with lower priority numbers to inform them about P_i being the new coordinator.
- If answer is received, P_i begins time interval T', waiting to receive a message that a process with a higher priority number has been elected.
- If no message is sent within T', assume the process with a higher number has failed; P_i should restart the algorithm

Bully Algorithm (Cont.)

- If P_i is not the coordinator, then, at any time during execution, P_i may receive one of the following two messages from process P_j.
  - P_j is the new coordinator ($j > i$). P_i, in turn, records this information.
  - P_j started an election ($j < i$). P_i sends a response to P_j and begins its own election algorithm, provided that Pi has not already initiated such an election.
- After a failed process recovers, it immediately begins execution of the same algorithm.
- If there are no active processes with higher numbers, the recovered process forces all processes with lower number to let it become the coordinator process, even if there is a currently active coordinator with a lower number.

Ring Algorithm

- Applicable to systems organized as a ring (logically or physically).
- Assumes that the links are unidirectional, and that processes send their messages to their right neighbors.
- Each process maintains an *active list*, consisting of all the priority numbers of all active processes in the system when the algorithm ends.
- If process P_i detects a coordinator failure, it creates a new active list that is initially empty. It then sends a message *elect(i)* to its right neighbor, and adds the number *i* to its active list.

Ring Algorithm (Cont.)

- If P_i receives a message elect(*j*) from the process on the left, it must respond in one of three ways:
  1. If this is the first *elect* message it has seen or sent, P_i creates a new active list with the numbers *i* and *j*. It then sends the message *elect(i),* followed by the message *elect(j).*
  2. If $i \neq j$, then P_i adds *j* to its active list and forwards the message to its right neighbor.
  3. If $i = j$, then P_i receives the message *elect(i)*. The active list for P_i contains all the active processes in the system. P_i can now determine the new coordinator process.

Reaching Agreement

- There are applications where a set of processes wish to agree on a common “value”.
- Such agreement may not take place due to:
  - Faulty communication medium
  - Faulty processes
    - Processes may send garbled or incorrect messages to other processes.
    - A subset of the processes may collaborate with each other in an attempt to defeat the scheme.


Faulty Communications

- Process P_i at site A, has sent a message to process P_j at site B; to proceed, P_i needs to know if P_j has received the message.
- Detect failures using a time-out scheme.
  - When P_i sends out a message, it also specifies a time interval during which it is willing to wait for an acknowledgment message from P_j.
  - When P_j receives the message, it immediately sends an acknowledgment to P_i.
  - If P_i receives the acknowledgment message within the specified time interval, it concludes that P_j has received its message. If a time-out occurs, P_j needs to retransmit its message and wait for an acknowledgment.
  - Continue until P_i either receives an acknowledgment, or is notified by the system that B is down.


Faulty Communications (Cont.)

- Suppose that P_j also needs to know that P_i has received its acknowledgment message, in order to decide on how to proceed.
  - In the presence of failure, it is not possible to accomplish this task.
  - It is not possible in a distributed environment for processes P_i and P_j to agree completely on their respective states.

Faulty Processes (Byzantine Generals Problem)

- Communication medium is reliable, but processes can fail in unpredictable ways.
- Consider a system of n processes, of which no more than m are faulty. Suppose that each process P_i has some private value of V_i.
- Devise an algorithm that allows each nonfaulty P_i to construct a vector $X_i = (A_{i,1}, A_{i,2}, \ldots, A_{i,n})$ such that::
  - If P_j is a nonfaulty process, then $A_{ij} = V_j$.
  - If P_i and P_j are both nonfaulty processes, then $X_i = X_j$.
- Solutions share the following properties.
  - A correct algorithm can be devised only if $n \geq 3 \times m + 1$.
  - The worst-case delay for reaching agreement is proportionate to $m + 1$ message-passing delays.

Faulty Processes (Cont.)

- An algorithm for the case where $m = 1$ and $n = 4$ requires two rounds of information exchange:
  - Each process sends its private value to the other 3 processes.
  - Each process sends the information it has obtained in the first round to all other processes.
- If a faulty process refuses to send messages, a nonfaulty process can choose an arbitrary value and pretend that that value was sent by that process.
- After the two rounds are completed, a nonfaulty process P_i can construct its vector $X_i = (A_{i,1}, A_{i,2}, A_{i,3}, A_{i,4})$ as follows:
  - $A_{i,i} = V_i$.
  - For $j \neq i$, if at least two of the three values reported for process P_j agree, then the majority value is used to set the value of A_{ij}. Otherwise, a default value (*nil*) is used.