Co-rdinator Election Algorithms

George Coulouris, Jean Dollimore and Tim Kindberg,
“Distributed Systems Concepts and Design”, Fifth Edition,
Pearson Education, 2012

Election Introduction

- An algorithm for choosing a unique process to play a particular role is called an Election Algorithm.
- It is essential that all the processes agree on the choice. Afterwards, if the process that plays the role of server wishes to retire then another election is required to choose a replacement.
- An individual process does not call more than one election at a time, but in principle the N processes could call N concurrent elections
- A process pi is
  - either a participant –it is engaged in some run of the election algorithm.
  - or a non-participant –it is not currently engaged in any election

Election Introduction

- Important requirement is for the choice of elected process to be unique, even if several processes call elections concurrently.
- For instance, two processes could decide independently that a coordinator process has failed, and both call elections.
- We require that the elected process be chosen as the one with the largest identifier.
- Each process pi (i = 1, 2,...N) has a variable electedi= ⊥, which will contain the
- identifier of the elected process
- E1: (safety) A participant process pi has electedi = ⊥ or electedi = P, where P is chosen as the non-crashed process at the end of the run with the largest identifier.
- E2: (liveness) All processes pi participate and eventually either set electedi = ⊥ – or crash.
- Performance of an election algorithm by its total network bandwidth utilization and by the turnaround time for the algorithm

Election Algorithms

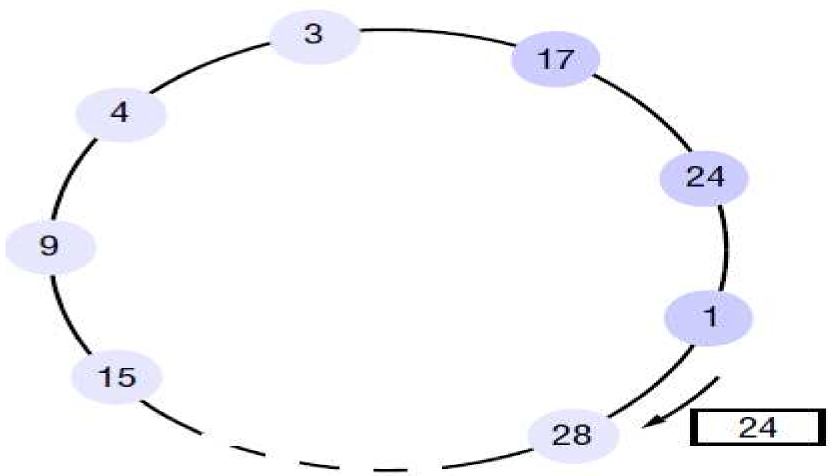
- Ring-based Election Algorithm
- Bully Algorithm

Election Algorithms

Ring-based Election Algorithm

Ring-based Election Algorithm

- The algorithm of Chang and Roberts [1979] is suitable for a collection of processes arranged in a logical ring.
- Each process pi has a communication channel to the next process in the ring, p(i + 1)mod N , and all messages are sent clockwise around the ring
- It assume that no failures occur, and that the system is asynchronous
- The goal of this algorithm is to elect a single process called the coordinator, which is the process with the largest identifier



Ring-based Election Algorithm

- Initially, every process is marked as a non-participant in an election.
- Any process can begin an election. It proceeds by marking itself as a participant, placing its identifier in an election message and sending it to its clockwise neighbor.
- When a process receives an election message, it compares the identifier in the message with its own.
- If the arrived identifier is greater, then it forwards the message to its neighbor.
- If the arrived identifier is smaller and the receiver is not a participant, then it substitutes its own identifier in the message and forwards it;
- On forwarding an election message in any case, the process marks itself as a participant
- If, however, the received identifier is that of the receiver itself, then this process's identifier must be the greatest, and it becomes the coordinator. The coordinator marks itself as a non-participant once more and sends an elected message to its neighbor, announcing its election and enclosing its identity

Ring-based Election Algorithm

- When a process pi receives an elected message, it marks itself as a nonparticipant, sets its variable electedi to the identifier in the message and, unless it is the new coordinator, forwards the message to its neighbour.
- Condition E1 is met. All identifiers are compared, since a process must receive its own identifier back before sending an elected message.
- For any two processes, the one with the larger identifier will not pass on the other's identifier. It is therefore impossible that both receive their own identifier back.
- Condition E2 follows immediately from the guaranteed traversals of the ring (there are no failures).

Ring-based Election Algorithm

- The worst-performing case is when its anti-clockwise neighbour has the highest identifier.
- A total of N – 1 messages are then required to reach this neighbour.
- It will not announce its election until its identifier has completed another circuit, taking a further N messages.
- The elected message is then sent N times,
- Making 3N – 1 messages in all.
- The turnaround time is also 3N – 1 , since these messages are sent sequentially.
- Drawback
  – It does not tolerate failures making it limited practical value.
  – Reliable failure detector is required for process crashes

Election Algorithms

Bully Algorithm

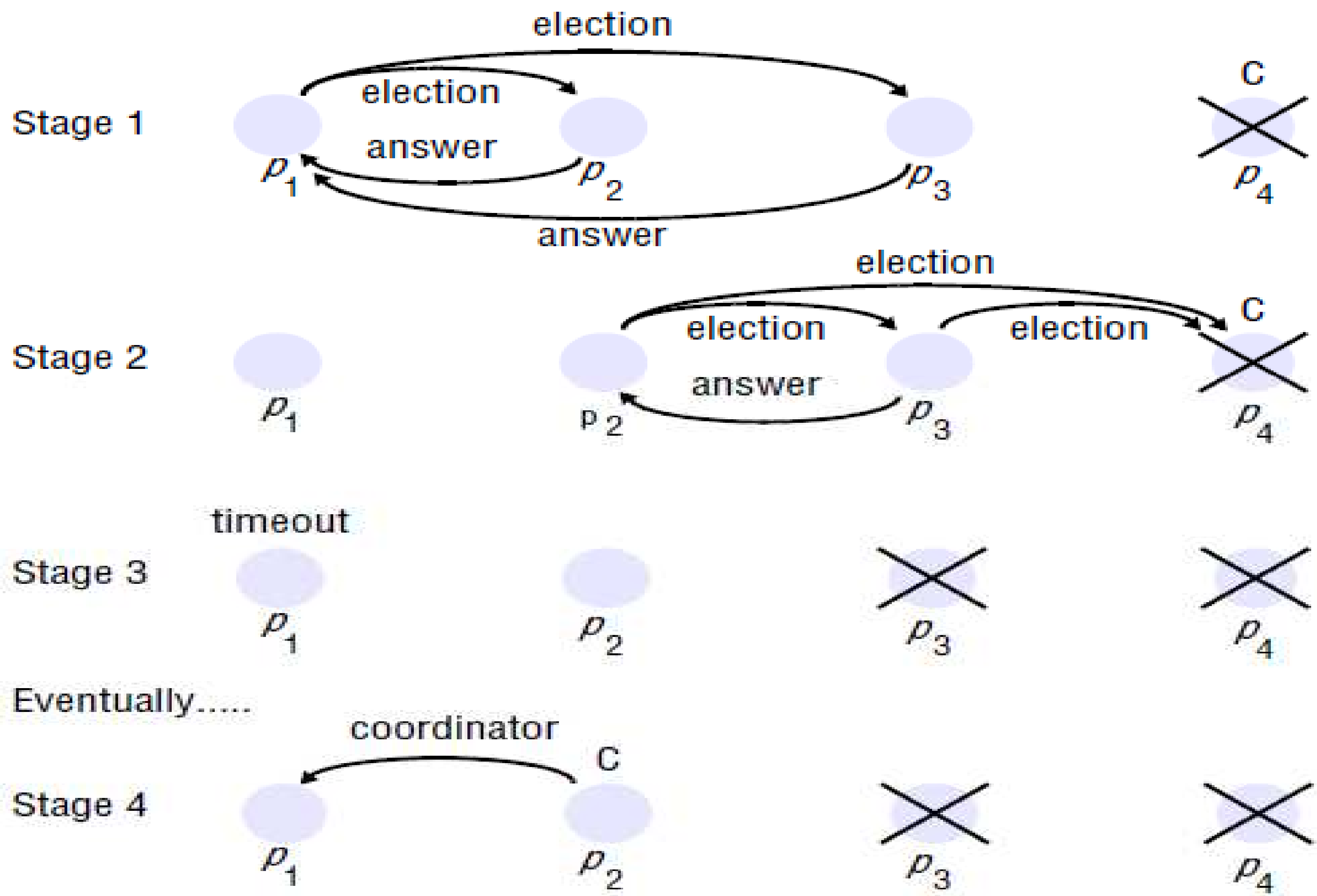
Bully Algorithm

- The bully algorithm [Garcia-Molina 1982] allows processes to crash during an election, although it assumes that message delivery between processes is reliable.
- This algorithm assumes that the system is synchronous: it uses timeouts to detect a process failure
- In Bully algorithm : Each process knows which processes have higher identifiers, and that it can communicate with all such processes. 3 message types.
- Election message is sent to announce an election;
- Answer message is sent in response to an election message and
- Coordinator message is sent to announce the identity of the elected process

Bully Algorithm

- A process begins an election when it notices, through timeouts, that the coordinator has failed.
- Several processes may discover this concurrently.
- Maximum message transmission delay, T_{trans} , and a maximum delay for processing a message $T_{process}$.
- Time $T = 2T_{trans} + T_{process}$ that is an upper bound on the time that can elapse between sending a message to another process and receiving a response.
- If no response arrives within time T, recipient of request has failed.

Bully Algorithm



Bully Algorithm

- The process that knows it has the highest identifier can elect itself as the coordinator simply by sending a coordinator message to all processes with lower identifiers.
- On the other hand, a process with a lower identifier can begin an election by sending an election message to those processes that have a higher identifier and awaiting answer messages in response.
- If none arrives within time T, the process considers itself the coordinator and sends a coordinator message to all processes with lower identifiers announcing this.
- If a process pi receives a coordinator message, it sets its variable electedi to the identifier of the coordinator
- If a process receives an election message, it sends back an answer message and begins another election

Bully Algorithm

- Algorithm clearly meets the liveness condition E2, by the assumption of reliable message delivery.
- Algorithm is not guaranteed to meet the safety condition E1 if processes that have crashed are replaced by processes with the same identifiers.
- A process that replaces a crashed process p may decide that it has the highest identifier just as another process (which has detected p's crash) decides that it has the highest identifier.
- Two processes will therefore announce themselves as the coordinator concurrently.
- That is why this algorithm is called Bully Algorithm

Bully Algorithm

- Performance of the algorithm, in the best case the process with the second-highest identifier notices the coordinator's failure.
- Then it can immediately elect itself and send N – 2 coordinator messages. The turnaround time is one message.
- The bully algorithm requires O (N^2) messages in the worst case – that is, when the process with the lowest identifier first detects the coordinator's failure.
- For then N – 1 processes altogether begin elections, each sending messages to processes with higher identifiers.

Summary

Ring-based vs Bully

	Ring Based	Bully
Asynchronous	Yes	No
Allows processes to crash	No	Yes
Satisfies Safety	Yes	Yes/No
Dynamic process identifiers	Yes	No
Dynamic configuration of processes	Maybe	Maybe
Best case performance	$2 \times N$	$N - 1$
Worst case performance	$3 \times N - 1$	$\mathcal{O}(N^2)$

Summary

- Electing process with highest identifier as a co-ordinator process.
- Ring-based election algorithm
- Bully Algorithm