CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

LEADER ELECTION

Lecture D

BULLY ALGORITHM

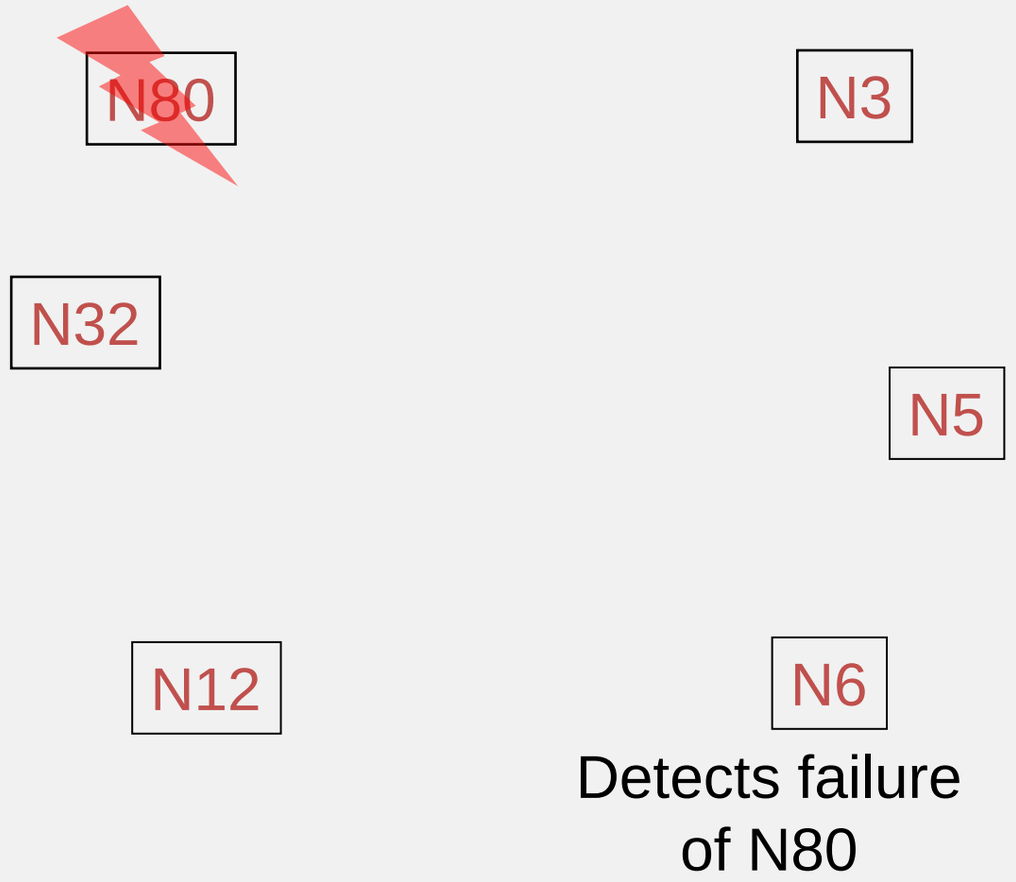
Bully Algorithm

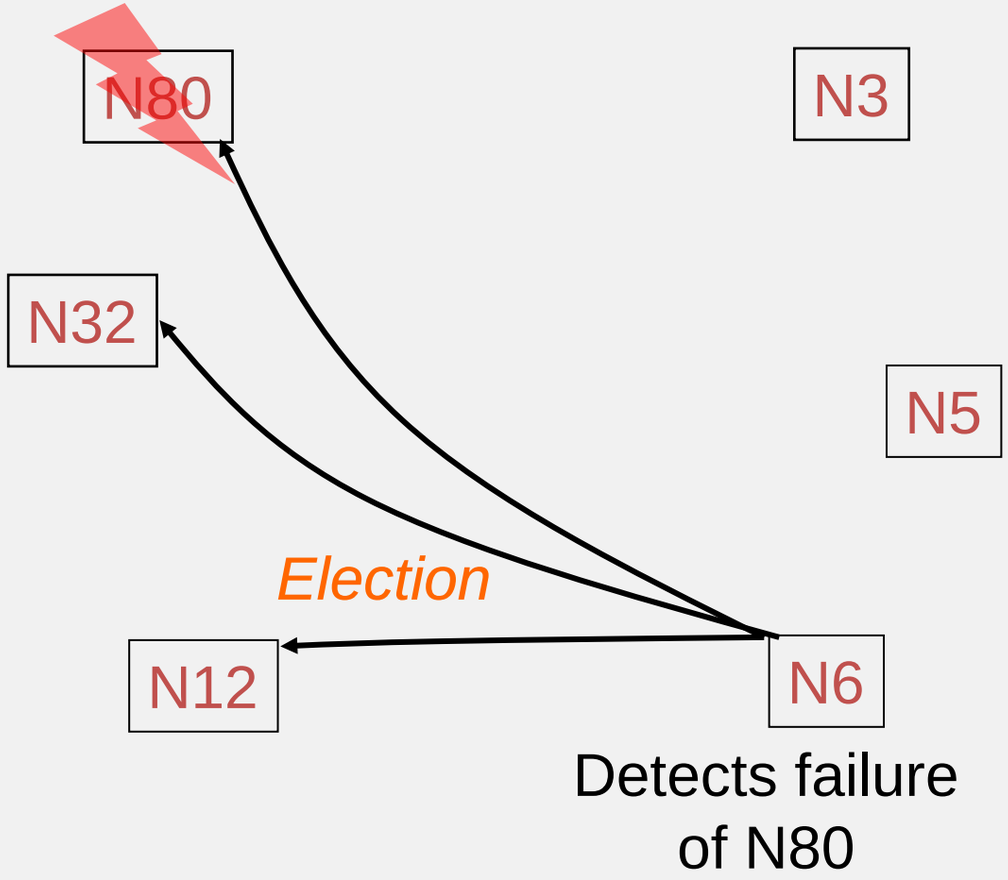
- All processes know other process' ids
- When a process finds the coordinator has failed (via the failure detector):
  - **if** it knows its id is the highest, it elects itself as coordinator, then sends a *Coordinator* message to all processes with lower identifiers. Election is completed.
  - **else** it initiates an election by sending an *Election* message
    - (contd.)

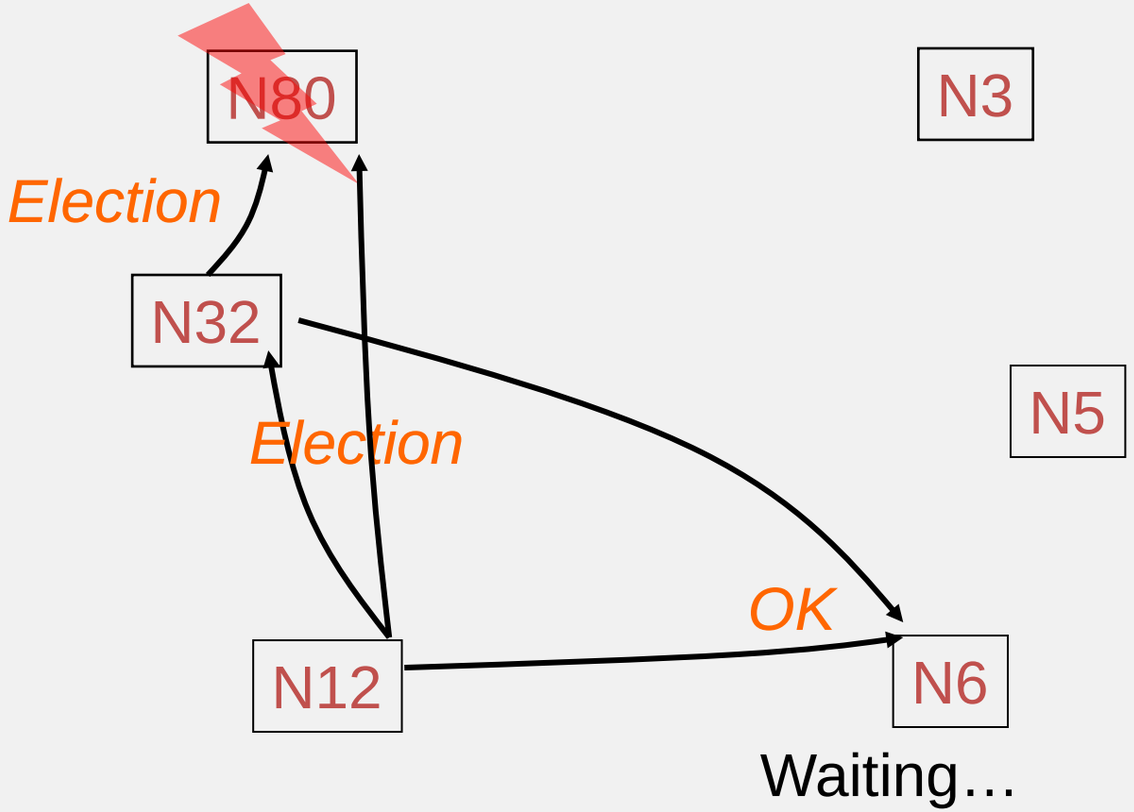
Bully Algorithm (2)

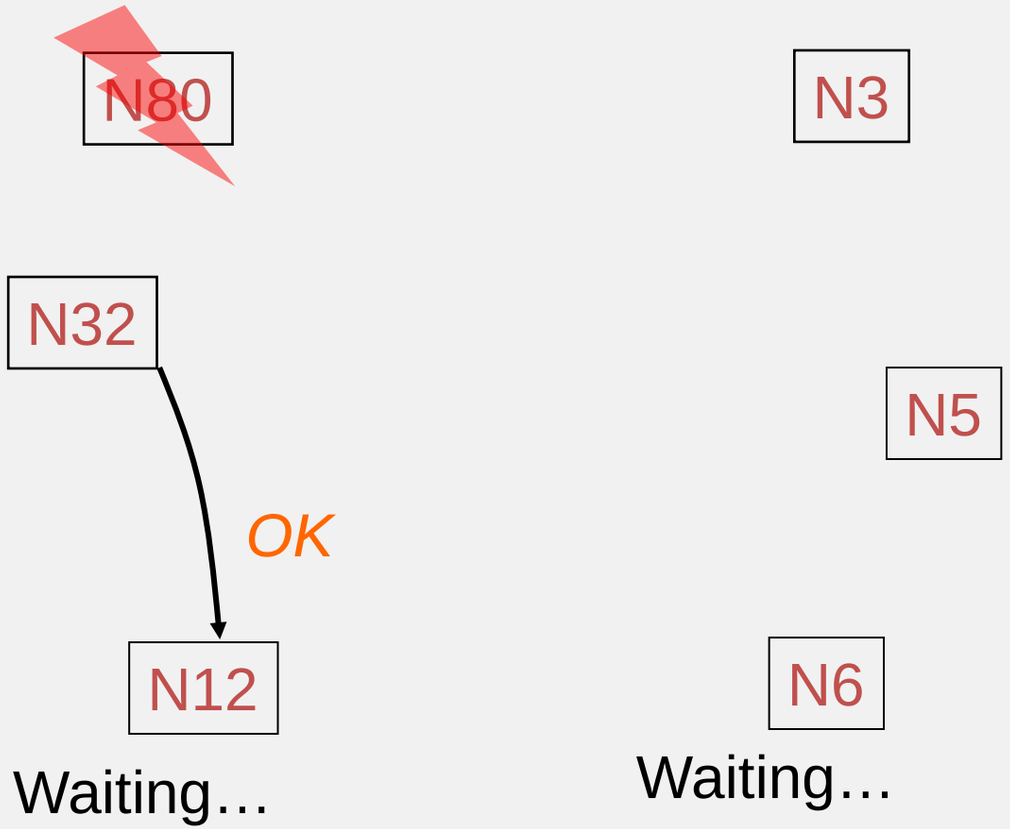
- **else** it initiates an election by sending an *Election* message
  - Sends it to only processes that have a *higher id than itself*.
  - **if** receives no answer within timeout, calls itself leader and sends *Coordinator* message to all lower id processes. Election completed.
  - **if** an answer received however, then there is some non-faulty higher process => so, wait for coordinator message. If none received after another timeout, start a new election run.
- A process that receives an *Election* message replies with *OK* message, and starts its own leader election protocol (unless it has already done so)

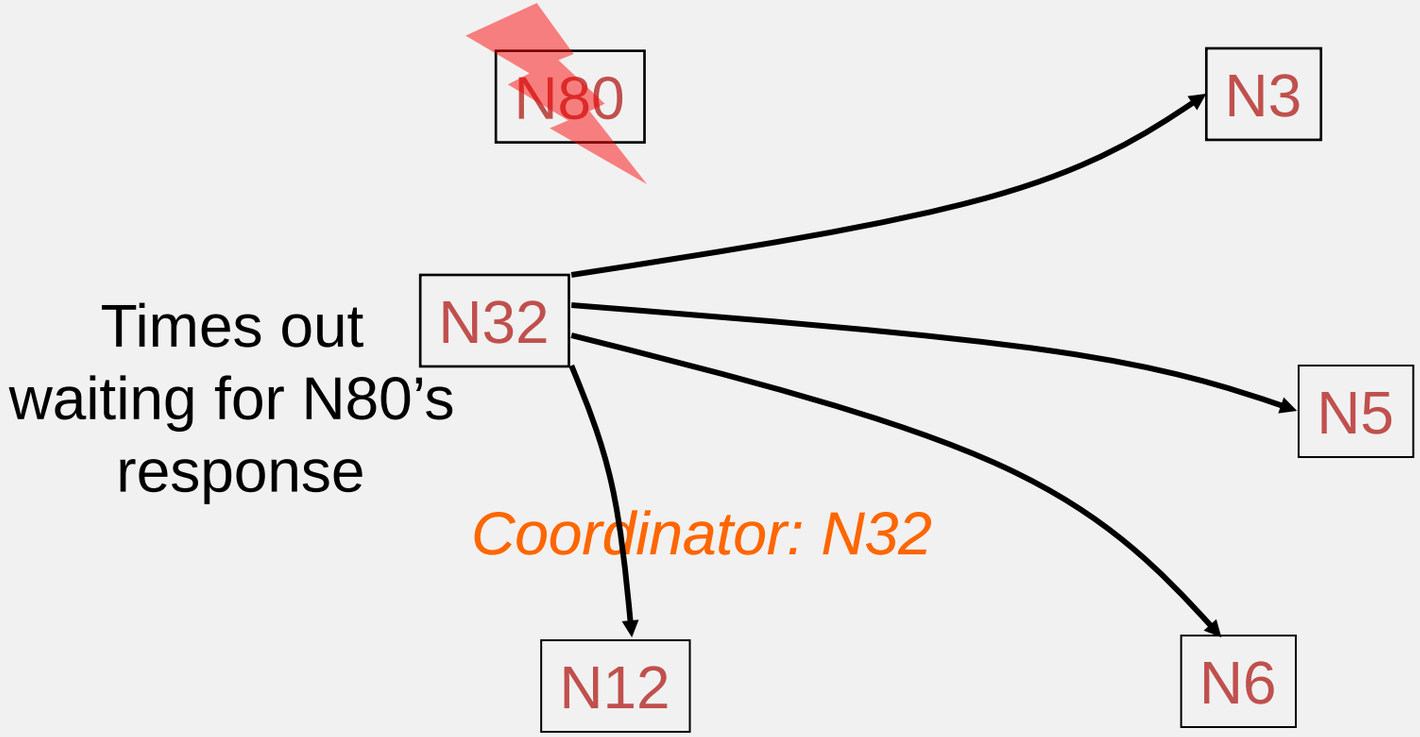
Bully Algorithm: Example




N80
N3
N32
N5
Election
N12
N6
Detects failure
of N80

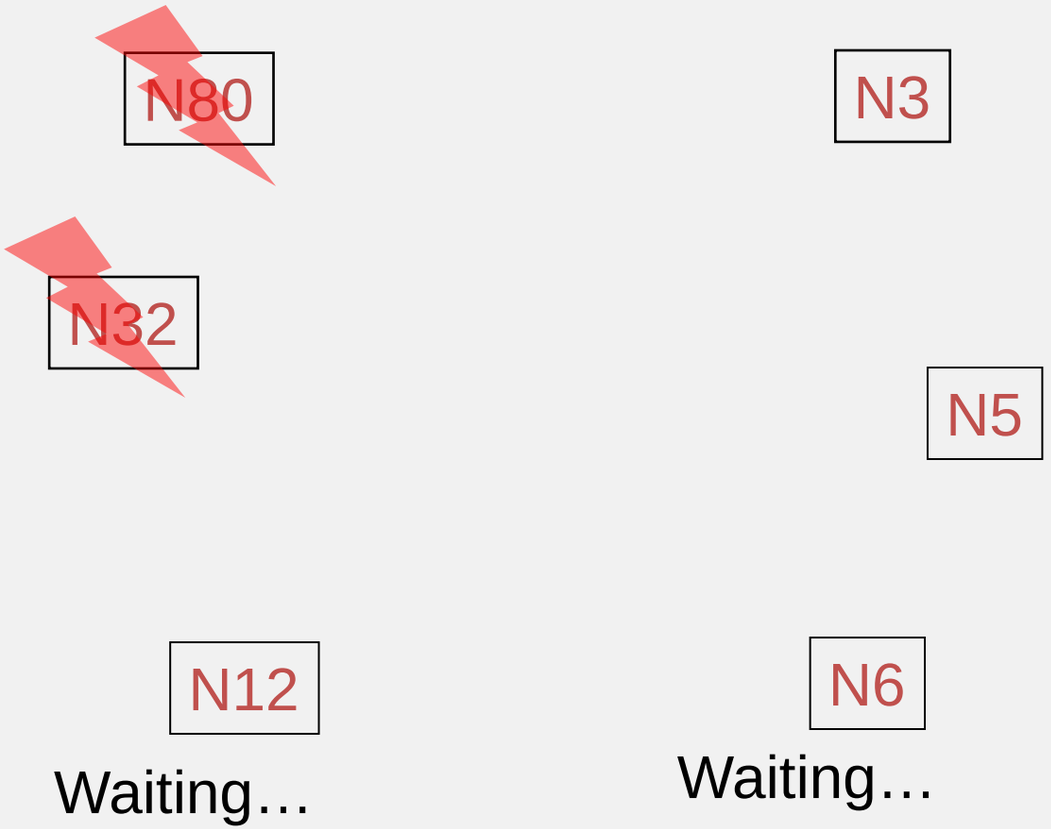

N80
N3
Election
N32
N5
Election
OK
N12
N6
Waiting…

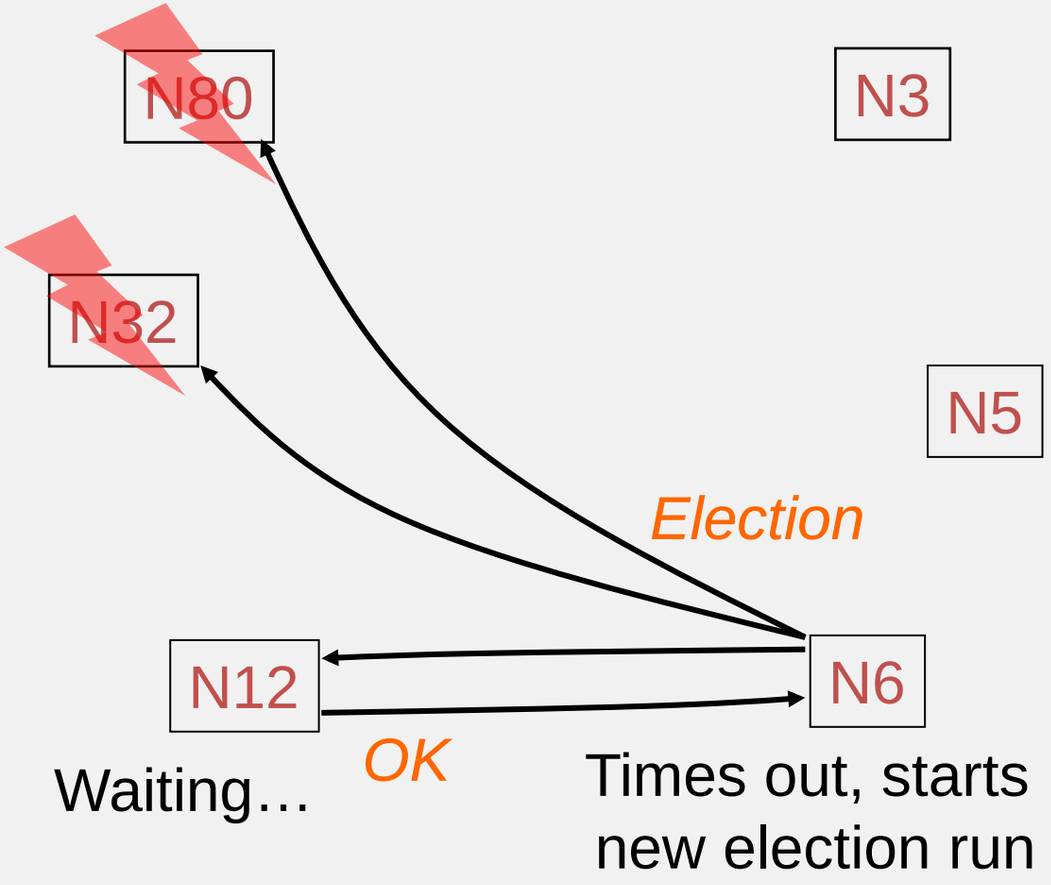

N80
N3
N32
N5
OK
N12
N6
Waiting…
Waiting…

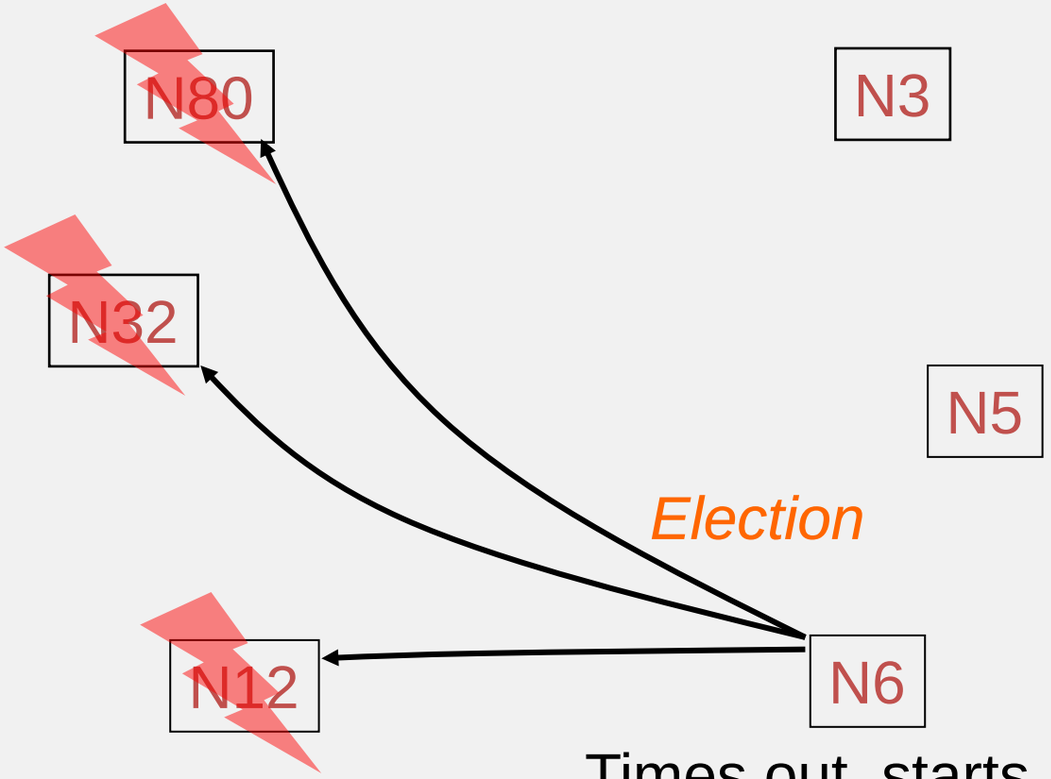


Election is completed

Failures during Election Run




N80
N3
N32
N5
Election
N12
N6
OK
Waiting…
Times out, starts
new election run



Times out, starts another new election run

Failures and Timeouts

- If failures stop, eventually will elect a leader
- How do you set the timeouts?
- Based on Worst-case time to complete election
  - 5 message transmission times if there are no failures during the run:
    1. Election from lowest id server in group
    2. Answer to lowest id server from 2nd highest id process
    3. Election from 2nd highest id server to highest id
    4. Timeout for answers @ 2nd highest id server
    5. Coordinator from 2nd highest id server

Analysis

- **Worst-case** completion time: 5 message transmission times
  - When the process with the lowest id in the system detects the failure.
    - (*N-1*) processes altogether begin elections, each sending messages to processes with higher ids.
    - *i*-th highest id process sends (*i-1*) election messages
  - Number of Election messages
    $= N-1 + N-2 + \ldots + 1 = (N-1)*N/2 = O(N^2)$
- **Best-case**
  - Second-highest id detects leader failure
  - Sends (*N-2*) Coordinator messages
  - Completion time: 1 message transmission time

Impossibility?

- Since timeouts built into protocol, in asynchronous system model:
  - Protocol may never terminate => Liveness not guaranteed
- But satisfies liveness in synchronous system model where
  - Worst-case one-way latency can be calculated = worst-case process time + worst-case message latency

Summary

- Leader election an important component of many cloud computing systems
- Classical leader election protocols
  - Ring-based
  - Bully
- But failure-prone
  - Paxos-like protocols used by Google Chubby, Apache Zookeeper