THE UNIVERSITY *of* EDINBURGH

Distributed Systems Fall 2024

Yuvraj Patel

Disclaimer: Slides prepared using multiple sources (UW-Madison – Remzi, Andrea, Mike; Cambridge – Martin Kleepman; Distributed Systems book by MVS/ AST; University of Edinburgh – Yuvraj Patel)

Today's Agenda

Consensus

- Basics

Leader Election

Why Consensus?

Multiple use-cases

- Replication – make sure the replicated data is same on all the nodes
- Failure Detection – a machine/leader has failed/stopped responding
- Leader Election – elect a leader to initiate a snapshot, etc.
- and many more…

All the above scenarios involve

- Multiple parties
- Presence of faults
- Coordinate amongst themselves
- Need to agree to something or arrive at a decision

Consensus Problem – Single value formulation

Consensus Protocol

Consider a distributed system with n nodes

- Each node i has an input x_i
- Faults may happen at arbitrary times

Output

- All nodes agree on a single value; Value cannot change later

Guarantee the following

- Termination (Liveness): Every non-faulty node eventually decides
- Agreement (Safety): All non-faulty nodes decide on the same value
- Validity: The decided value must be the input of at least one node

Consensus Protocol (contd…)

Not democratic; Value proposed by a small minority can be decided

Consensus possible depending on multiple parameters

Most important parameters

- System Model – Synchronous or Asynchronous
- Fault types – Crash or Byzantine

Synchronous vs. Asynchronous Systems

Synchronous systems

- Process execution speeds and message delivery times are bounded
- Can detect omission and timing failures

Asynchronous systems

- No assumptions about process execution speed or message delivery times
- Cannot reliably detect crash failures

Consensus

- Challenging in Asynchronous systems
- Solvable in Synchronous systems
- Algorithm for Asynchronous systems will work for Synchronous systems

Impossibility in Asynchronous Systems

Fischer, Nancy & Paterson show it is impossible to achieve consensus in asynchronous system with a single faulty process

They prove that no asynchronous algorithm for agreeing on a one-bit value can guarantee that it will terminate in the presence of crash faults

- With no crash too, algorithm may not terminate
- Proof constructs infinite non-terminating runs

One of the most fundamental results in distributed systems.

- Interested students can check the FLP paper -- https://dl.acm.org/doi/pdf/10.1145/3149.214121

Leader Election Problem

Need to elect leader to perform tasks and broadcast leader details

If leader fails

- Someone will detect leader failed
- Initiate a leader election to elect another leader
- Only one leader elected, and everyone agrees on who is the leader

System Model & Assumptions

System Model

- N nodes in the system; each node having unique id
- Communicate via messages; messages will eventually be delivered
- Failures/crashes may happen at arbitrary time

Assumptions

- Any node can call for an election
- Any node can call for at most one election at a time
- Multiple processes can call for an election simultaneously; still lead to a single leader
- Result independent of who calls for an election

Bully Algorithm

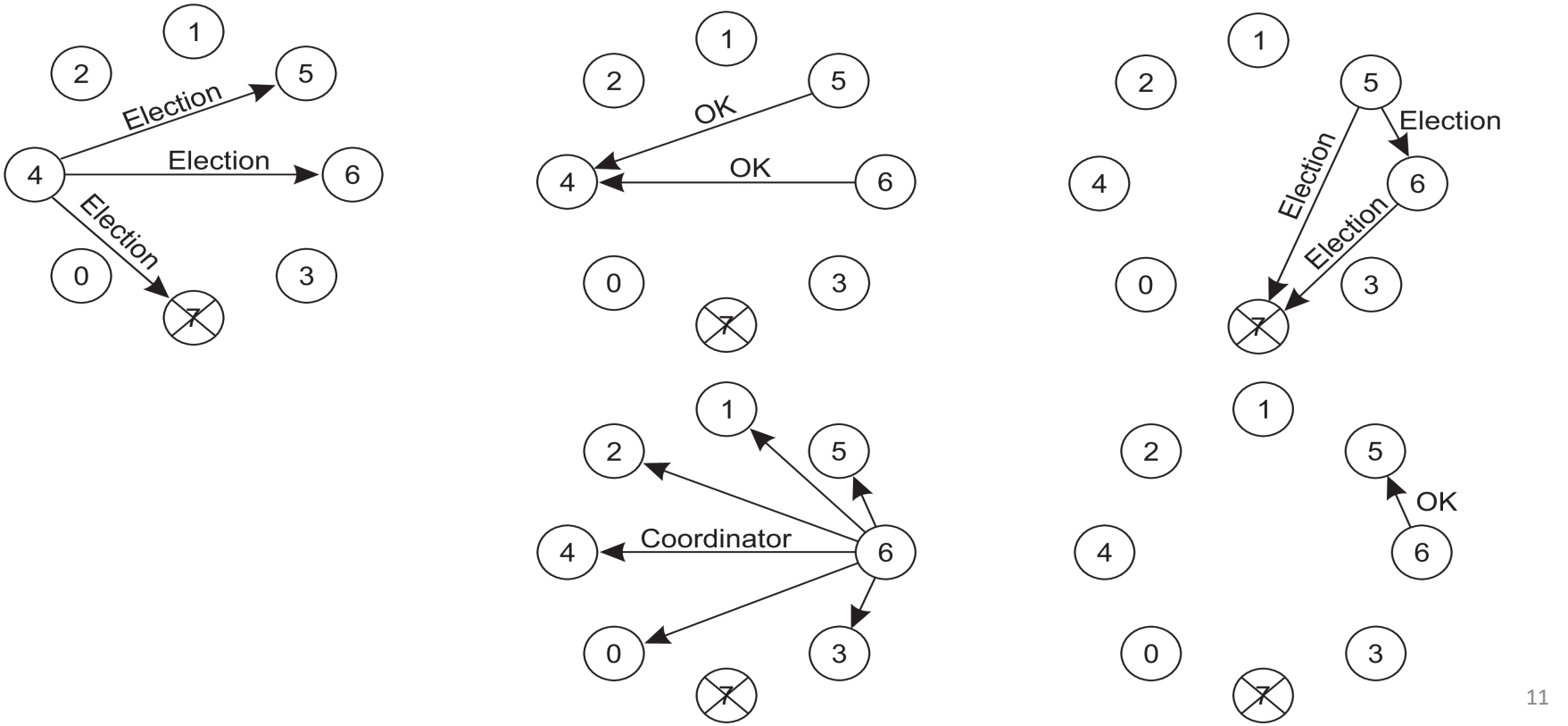
Key Idea: Node with highest ID wins

Consider N nodes {N_0, N_1, $N_{2...}$ N_n}.

Whenever a node N_k notices that the leader is unresponsive, election initiated

- N_k sends an ELECTION message to all the processes with higher IDs: N_{k+1},… N_n
- If no one responds, N_k wins
- If one of the higher-up's answers, it takes over and N_k's job is done

Example


1
2
5
Election
4
Election
6
Election
0
3
7
1
2
5
OK
4
OK
6
0
3
7
1
2
5
Election
4
Election
6
Election
0
3
7
1
2
5
Coordinator
4
6
0
3
7
1
2
5
OK
4
6
0
3
7

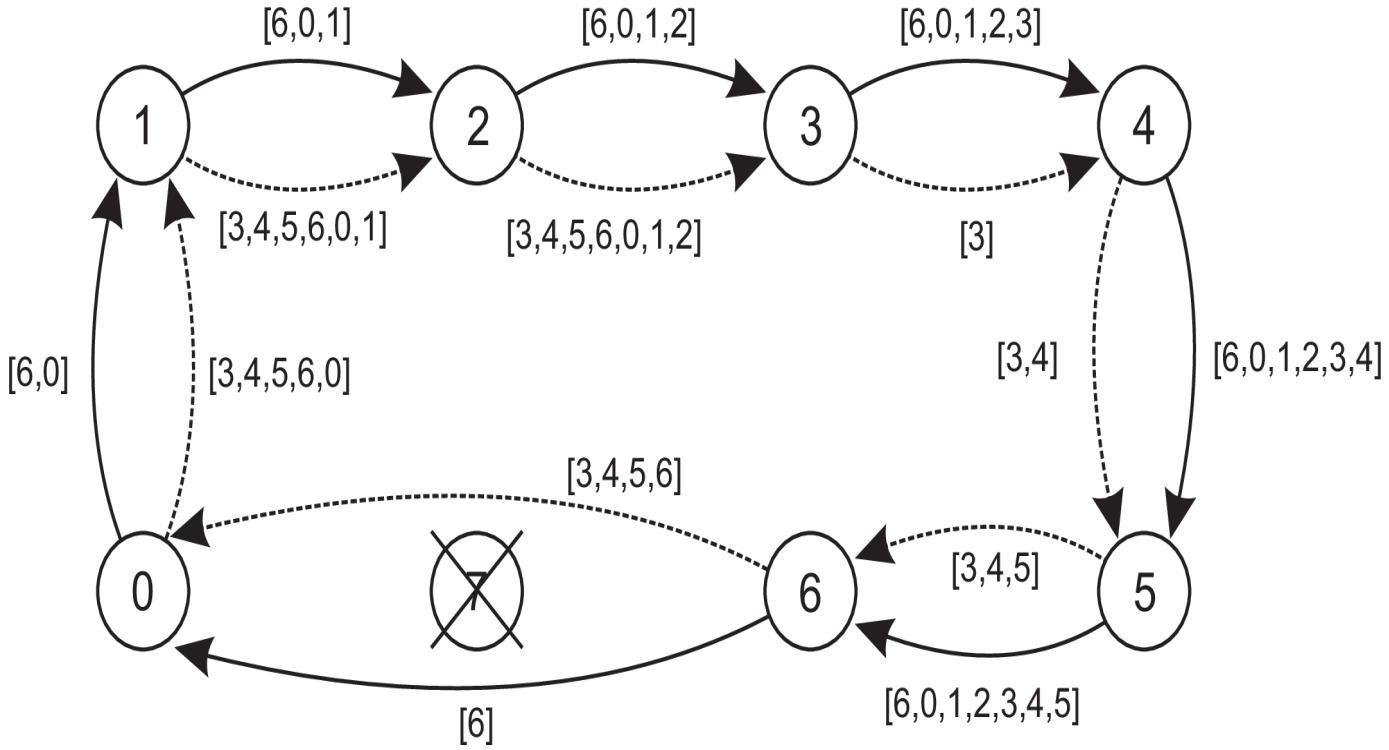
Ring Algorithm

Nodes are organized into a ring. Process with highest id is elected as coordinator

Whenever a node N_k notices that the leader is unresponsive, election initiated

- Any process can start an election by sending an election message to its successor. If a successor is down, the message is passed on the next successor
- If a message is passed on, the sender adds itself to the list.
- When the message gets back to the initiator, everyone had a chance to make its presence known.
- The initiator sends a coordinator message around the ring containing a list of all the living nodes. The one with the highest id is elected as coordinator

Example



The solid line shows the election messages initiated by N_6
The dashed one is election messages initiated by P_3

Both have the same list so it is safe to have two nodes initiating an election