Agreement Problem

- Motivation
  - Transaction commit
- Main difficulty: failures
  - process failures
  - link failures
- Different fault models
  - initially dead, fail-stop, omission, byzantine
- Surprising result: *Even in presence of one unannounced process death, agreement problem is impossible to solve.*
  - No Byzantine failures
  - Reliable messages
  - Processing is completely asynchronous
Consensus Problem

- Every process starts with an initial value of \{0,1\}
- A non-faulty process decides by entering a decision state
- Require that *some* process eventually make a decision
System Model

- Processes are modeled as automata (possibly infinite state)
- communication using messages
- Atomic step
  - attempt to receive a message
  - perform local computation
  - send a finite set of messages to other processes
Consensus Protocol

- $N$ processes
- one bit input register
- output register with values $\{0, 1, b\}$ initially $b$
- output register write-once
- unbounded storage
- message system: a buffer with
  - $\text{send}(p,m)$: places $(p,m)$ in the buffer
  - $\text{receive}(p)$: deletes $(p,m)$ and return $m$ or return $\emptyset$
- Condition on the message system
  - If $\text{receive}(p)$ is performed infinitely times, then every message is eventually delivered.
Global State

- Configuration
  - defined by local states. message buffer
  - initial configuration
  - step = primitive step by one process
    - step determined by the pair $e = (p, m)$

- Application of an event $e$ to $C$

- Schedule from $C$
  - finite or infinite sequence $\sigma$ of events
  - when $\sigma$ finite $\sigma(C')$: result of application
  - reachable configuration
Commutativity Property

- Lemma 1: If two schedules are disjoint, then they can be commuted.
- decision value of C
- Partially correct consensus protocol
  - no accessible configuration has more than one value
  - for each \( v \in \{0, 1\} \), some accessible configuration has decision value \( v \)
Faults

- faulty vs nonfaulty process
  - faulty = takes only finite number of steps

- admissible run
  - at most one process is faulty
  - all messages sent to non-faulty process eventually delivered

- deciding run
  - some process reaches a decision state
  - Totally correct protocol
  - Partially correct
  - every admissible run is deciding
Main Result

• Theorem: No consensus protocol is totally correct in spite of one fault.

• Proof: main idea. To show that there exists an admissible run which remains forever indecisive.
  • there is an initial such configuration
  • there exists a method to keep the system indecisive. The system does not take the “commit” step.

• Bi-valent vs univalent configurations
  • if univalent, 0-valent or 1-valent
Initial ambiguity

- Lemma: The protocol $P$ has a bivalent initial configuration.
  - there exist adjacent 0-valent and 1-valent configurations
  - apply schedule in which $p$ takes no steps.
Remaining indecisive

• Lemma: Let $C$ be a bivalent configuration of $P$. Let $e = (p, m)$ be applicable to $C$. Let $C$ be the set of configurations reachable from $C$. Let $D = e(C)$. Then $D$ contains a bi-valent configuration.
  
  • Pf: Assume if possible $D$ contains no bi-valent configs.
  
  • claim: $D$ contains both 0-valent and 1-valent states.
  
  • claim: exists neighbors $C0$, $C1$ such that
    • $D0 = e(C0)$ is 0-valent
    • $D1 = e(C1)$ is 1-valent
    • w.l.o.g. let $C1 = e'(C0)$, where $e' = (p', m')$
  
  • case 1: $p$ different from $p'$
    • contradiction
  
  • case 2: $p = p'$
    • consider any finite deciding run in which $p$ takes no steps
Constructing admissible non-deciding run

- Maintain a queue of processes
- maintain message buffer a FIFO queue
- in each stage the process at the head of the queue receives the earliest message
- Move the process to the back of the queue
- Now use earlier lemmas