CSC2/458 Parallel and Distributed Systems
Consensus and Failures

Sreepathi Pai
April 10, 2018
URCS
The FLP theorem
The FLP theorem
The Consensus Problem: Informal

A set of processes must decide on 0 or 1 as output starting from 0 or 1 as input.

- All processes must decide same value
- The decision making procedure must allow both 0 and 1 as possible outputs
  - Can’t have “always output 1” as the algorithm
Processes

- $N \geq 2$ processes
- Each process $p$ has:
  - input register $x_p$
  - output register $y_p$
  - program counter, internal storage
- Values for $x_p$, $y_p$ can be in $\{b, 0, 1\}$
- $y_p = b$, initially
- $p$ has decided when $y_p = 0$ or $y_p = 1$
  - $y_p$ is write-once
Abstracts network communication

- \texttt{send}(p, m), adds \((p, m)\) in to the buffer
- \texttt{receive}(p), removes some message \((p, m)\) from buffer
  - but can also receive \(\emptyset\) (why?)
  - leads to event \(e(p, m)\) or \(e(p, \emptyset)\)
• Total global state of system
  • All register values, internal storage, etc.
• Definition of initial configuration
  • All processes are in initial state and message buffer is empty
• An event $e(p, m)$ or $e(p, \emptyset)$ moves a configuration from $C$ to $e(C)$
  • $e$ applied to $C$, i.e. a step
• A schedule is a sequence of events (i.e. the run).
Configurations: Definitions

- **bivalent** configuration - can reach *either* 0 or 1
  - "has not made up its mind"
- **univalent** configuration - can reach *one* of 0 or 1
  - 0-valent can reach only 0
  - 1-valent can reach only 1
- Note, at some point, the protocol must switch from a bivalent configuration to a univalent configuration
Partial Correctness: Informal

- A configuration has a decision value $v$ if some process $p$ has $y_p = v$
  - Note: some
- A consensus protocol is partially correct if:
  - No configuration reachable from an initial configuration has more than one decision value
  - For $v \in \{0, 1\}$, some configuration reachable from an initial configuration has decision value $v$
A protocol that is partially correct:

- in spite of one faulty process (i.e. a process that does not take infinitely many steps)
- if all messages are eventually delivered to non-faulty processes
- always reaches a decision in all runs
- is said to be *totally correct*
Constructing a non-deciding run: sketch

- Start with bivalent initial configuration
- Construct a series of steps to reach another bivalent “middle” configuration
- Rinse and repeat
Questions

- Start with a bivalent initial configuration
  - Is there always one available?
- Reach a bivalent “middle” configuration by a series of steps
  - Can we always do this?
- Rinse and repeat
  - Can we keep doing this?
There is always a bivalent initial configuration

- Lemma 2 in the paper, proof by contradiction
- Consider two initial configuration $C_0$ and $C_1$ that are univalent
  - $C_i$ is $i$-valent
    - must exist by partial correctness
- Find $C_0$ and $C_1$ that are adjacent
  - differ only in process $p$
- Find a deciding run (and its schedule $\sigma$) to $C_0$ where $p$ takes no steps
- Apply $\sigma$ to $C_1$, where $p$ does take steps
- By total correctness:
  - ?
Main idea: avoid univalent configurations

Let a process \( p \) have a \textit{waiting} event \( e(p, m) \) in bivalent configuration \( C \)

- if applying \( e \) to \( C \) leads to another bivalent configuration, apply \( e \)
- if not, delay \( e \) until configuration \( C' \) where \( e(C') \) leads to a bivalent configuration
There is always a bivalent configuration in the “middle”

- See Lemma 3 in the paper