Byzantine Consensus

Definition

- Agreement: No two correct processes decide on different values
- Validity:
  - (a) Weak Unanimity: if all processes start from the same value \( v \) and all processes are correct, then \( v \) is the only possible decision
  - (b) Strong Unanimity: if all correct processes start from the same value \( v \), then \( v \) is the only possible decision value
- Termination:…
Structure of Consensus algorithms

• Throughout an execution, processes learn about initial values of other processes
• If failures occur, some values are learnt indirectly:
  – i sends 1 to j and fails: j knows that init_i=1
  – j sends 1 to k and fails: k knows that j knows that init_i=1
  – etc…

EIG Tree

• In general, such a chain can be constructed for every initial value
• We can design an algorithm that maintains these chains explicitly
• Maintain a tree to hold all possible value propagation chains
  – Each path from a leaf to the root represents a propagation chain
EIG Tree

Initial values of 1, 2, 3 learnt through one intermediate process

EIG Algorithm

- **Round 1:**
  - decorate root with init$_i$
  - send init$_i$ to all processes
  - decorate level 1 with received values: value from j
    decorates label j
- **Round r, 2\leq r \leq t+1:**
  - relay level r-1 values in the form (label, value)
  - for every (x,val) received from j, decorate level r node
    x:j with val
- **W=\{values in the tree\}, if |W|=1, decide v \in W,**
  Otherwise, decide a default value
SilentConsensus

• Round 1:
  – If initi=1, send 1 to all processes;
• Round r+1, 1 ≤ r ≤ t:
  – If received 1 in round r && has not yet broadcast a message:
    \[ W := W \cup \{1\}; \]
    relay 1 to all processes;
• At the end of round t+1:
  – If \(|W|>0\), decide 1, otherwise decide 0

Proof

• Let \(r \geq 1\) be a failure-free round
  – \(\exists\) non-failed process \(p\) that has received 1 in one of the rounds 1,…,r-1 \(\Rightarrow\) \(p\) sends 1 to all processes in the beginning of \(r\) the latest
  – No such process exists \(\Rightarrow\) no messages are sent
• After a failure-free round either all processes either have 1, or remain silent forever
Tolerating omissions

• Round 1:
  – If initi=1, send 1 to all processes;
• Round r+1, 1 ≤ r ≤ t:
  – If received (x,1) from j && |x|=r && has not yet broadcast a message:
    W := W ∪ {1}
    relay (x·j, 1) to all processes;
• At the end of round t+1:
  – If |W|>0, decide 1, otherwise decide 0

Authenticated Byzantine

• Round 1:
  – If initi=1, send [1]si to all processes;
• Round r+1, 1 ≤ r ≤ t:
  – If received [m]sj from j &&
    – m is correctly signed by j &&
    – m is correctly signed by r different processes &&
    – has not yet broadcast a message:
      W := W ∪ {1}
      relay [m·sj]si to all processes;
• At the end of round t+1:
  – If |W|>0, decide 1, otherwise decide 0
A simpler solution

- **Round 1:**
  - If init_i=1, broadcast [1]_si to all processes;
- **Round r+1, 1 ≤ r ≤ t:**
  - If received [1]_sj from at least r different processes &&
  - has not yet broadcast a message:
    \[ W := W \cup \{1\} \]
    broadcast [1]_si and relay all messages that
    caused it to be broadcast

At the end of round t+1:
- If |W|>0, decide 1, otherwise decide 0

Consistent (Echo) Broadcast

- **Correctness:** if correct process p broadcasts a message (p,m,k) in round k, then every correct process accepts (p,m,k) in the same round
- **Unforgeability:** if correct process p does not broadcast (p,m,k), then no correct process ever accepts (p,m,k)
- **Relay:** If a correct process accepts (p,m,k) in round r ≥ k, then every correct process accepts (p,m,k) by round r+1
Proof of CB algorithm (Relay)

- Message \((i,m,k)\) is accepted by non-faulty process \(j\) at round \(r'\) ➔
- \(j\) receives \(n-t\) \((echo,i,m,k)\), at least \(n-2t>t\) of which are from correct processes
- At \(r'\), \(t+1\) correct processes sent \((echo,i,m,k)\) to all correct processes ➔
- Every one of \(n-t\) correct processes will echo \((i,m,k)\) at the round \(r'+1\)

Implementing CB with \(n>3t\)

- Broadcast \((i,m,k)\) at round \(k\): send \((init,i,m,k)\) to all processes
- if process \(j\) receives \((init,i,m,k)\) at round \(k\), it sends \((echo,i,m,k)\) to all processes
- if before any round \(r' \geq r+1\), \(j\) has received \((echo,i,m,k)\) from at least \(t+1\) processes, it sends \((echo,i,m,k)\) to all processes
- if by the end of any round \(r' \geq r\), \(j\) has received \((echo,i,m,k)\) from at least \(n-t\) processes, \(j\) accepts \((i,m,k)\)
Consensus using CB

- **Round 1:**
  - If init$_i$=1, broadcast (i,1,1) to all processes;

- **Round r+1, 1 ≤ r ≤ t:**
  - If accepted 1 from at least r different processes &&
  - has not yet broadcast a message:
    - $W := W \cup \{1\}$
    - broadcast (i,1,r+1)

At the end of round t+1:
- If $|W|>0$, decide 1, otherwise decide 0

Impossibility with n≤3t

- We show impossibility for strong unanimity
  - Fischer, Lynch and Merritt
  - Found in textbooks

- Impossibility for weak unanimity can be proved using a similar approach:
  - Fischer, Lynch and Merritt
  - Section 6.6, Theorem 6.30, Distributed Algorithms, by N. Lynch
n=3