CPSC 465/565 Theory of Distributed Systems

James Aspnes

2023-09-20

Today's exciting topics

- Algorithms for synchronous Byzantine agreement.
- Impossibility of asynchronous agreement with 1 crash failure.

Byzantine agreement

Byzantine model: $\leq f$ faulty processes can do whatever they want.

- Agreement: All non-faulty nodes output same value.
- Termination: Finish in bounded number of rounds.
- Validity: If all non-faulty nodes have same input, they all output this input.

We showed last time that this requires $f + 1$ synchronous rounds and $n \geq 3f + 1$ in the worst case.

Today we'll show that these bounds are tight.

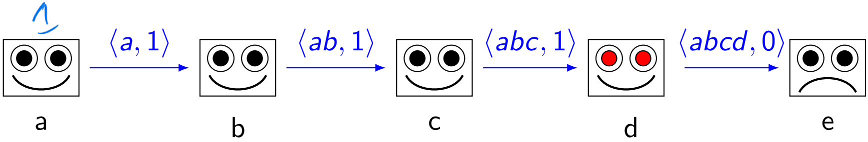
Exponential Information Gathering

(Pease, Shostak, Lamport 1980)

Idea:

- One process's reports are not trustworthy.
- But many processes' reports might be.

Strategy is to gather many secondhand reports, then use them to reconstruct "true" inputs.



Message: $\langle w, v\rangle$ means "I am reporting v from path w."

Variable: $\texttt{val}(w, i)$ = value stored by i for path w.

Only paths with no repeated processes are allowed.

EIG algorithm: gathering part

a 1 → ad 1
b 0 → bd 0
c 0 → cd 0
d

// Set my value to my input

val(⟨⟩ , i) $\leftarrow$ input

for round $\leftarrow 0 \ldots f$ **do**

  // send step for this round

  for each *non-repeating* w, $|w| = \text{round}$, $i \notin w$ **do**

    Send $\langle wi, \texttt{val}(w, i)\rangle$ to all processes

  // receive step for this round

  for each *non-repeating* w, $|w| = \text{round}$ **do**

    if j *sent* $\langle wj, v\rangle$ **then**

      // Record reported value

      $\texttt{val}(wj, i) \leftarrow v$

    else

      // Record default value

      $\texttt{val}(wj, i) \leftarrow 0$

EIG algorithm: reconstruction part

```
// Compute decision value
for each path w of length f + 1 with no repeats do
    val*(w, i) ← val(w, i)
for ℓ ← f down to 0 do
    for each non-repeating w, |w| = ℓ do
        val*(w, i) ← majority_{j∉w} val*(wj, i)
Decide val*(⟨⟩, i)
```

ab

abe 1

abd 0

abe 1

abcd 1

abce 0

abcf 1

abcg 1

- $\mathtt{val}^*(w, i)$ is i's reconstruction of value reported by w.
- But we don't trust last node in w.
- So instead take majority of reports of what that node said.
  - With $\geq 2f + 1$ reports, majority $\geq f + 1$ are honest.
- We get those reports by same reconstruction process.

Last step: Reconstruct majority input.

Proof of correctness

Start with some lemmas:

1. If i and j are both non-faulty, $\texttt{val}(wj, i) = \texttt{val}(w, j)$.
   - Proof: j sends $\langle wj, \texttt{val}(w, j)\rangle$ to i and i records it.
2. If i and j are both non-faulty, $\texttt{val}^*(wj, i) = \texttt{val}(w, j)$.
   - Proof: By induction on decreasing length of wj.
   - If $|wj| = f + 1$, $\texttt{val}^*(wj, i) = \texttt{val}(wj, i) = \texttt{val}(w, j)$.
   - If $|wj| < f + 1$,
     - Recall $\texttt{val}^*(wj, i) = \text{majority}_{k \notin wj}\, \texttt{val}^*(wjk, i)$.
     - For non-faulty k, ind hyp says $\texttt{val}^*(wjk, i) = \text{val}(wj, k) = \text{val}(w, j)$.
     - Since majority of k are non-faulty, $\texttt{val}^*(wj, i) = \text{val}(w, j)$.

Define: w is **common** if $\texttt{val}^*(w, i) = \texttt{val}^*(w, j)$ for all good i, j.

Then wk is common if k is non-faulty.

How common values propagate

$\mathrm{val}^*(w,i)$

$\mathrm{val}^*(w,j)$

w common — wa, wb, wc

- Claim: If all wk are common, so is w.
- Proof:
  1. Let i and j be non-faulty.
  2. Suppose $\mathtt{val}^*(wk,i) = \mathtt{val}^*(wk,j)$ for all k.
  3. Then i and j compute same majority value for $\mathtt{val}^*(w,i) = \mathtt{val}^*(w,j)$.
- Contraposition: If w is *not* common, $\exists k : wk$ is not common.

common

w — wk — wkl — wkl_m

Now suppose $\langle\rangle$ is not common. $\langle\rangle$ — a — ab — abc — $a_1 \ldots a_{f+1}$

1. From the contraposition, can find path $k_1 k_2 \ldots k_{f+1}$ such that each prefix $k_1 \ldots k_m$ is not common.
2. But one of those prefixes ends with a non-faulty node!
3. Contradiction $\Rightarrow \langle\rangle$ is common.
4. So decision value $\mathtt{val}^*(\langle\rangle, i)$ is the same for all non-faulty i.

This proves agreement.

EIG solves Byzantine agreement

- Agreement: already proved.
- Termination: trivial as always.
- Validity:
  - Recall $\mathtt{val}^*(j,i) = \mathtt{val}(\langle\rangle, j) = \mathsf{input}_j$ for all non-faulty i and j.
  - For all non-faulty j (a majority), $\mathsf{input}_j = v$ for some fixed v.
  - So all non-faulty i compute output $\mathtt{val}^*(\langle\rangle, i) = \mathsf{majority}_j\, \mathtt{val}^*(j,i) = \mathsf{majority}_j\, \mathtt{val}(\langle\rangle, j) = v$.

Time complexity $f+1$ is optimal.

Fault-tolerance $n \geq 3f+1$ is optimal.

Message complexity, bit complexity, local computation, local storage: all exponentially horrifying.

Phase king (Berman, Garay, Perry 1989)

Byzantine agreement with

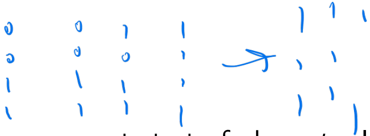
- $2(f+1)$ rounds
- $n \geq 4f+1$
- Constant-sized messages.

(We are doing simplified version from Attiya-Welch textbook; original paper gives better bounds with a more complicated algorithm.)

Phase king: pseudocode

$\mathrm{pref}_i[i]$ = input; **for each** $j \neq i$ **do** $\mathrm{pref}_i[j] = 0$
for $k \leftarrow 1$ **to** $f+1$ **do**
    send $\mathrm{pref}_i[i]$ to all processes (including myself)
    $\mathrm{pref}_i[j] \leftarrow v_j$, where v_j is the value received from process j
    majority $\leftarrow$ majority value in pref_i
    multiplicity $\leftarrow$ number of times majority appears in pref_i
    if $i = k$ **then** send majority to all processes
    if *received m from phase king k* **then**
        kingMajority $\leftarrow m$
    else
        kingMajority $\leftarrow 0$
    if multiplicity $> n/2 + f$ **then**
        $\mathrm{pref}_i[i]$ = majority
    else
        $\mathrm{pref}_i[i]$ = kingMajority
return $\mathrm{pref}_i[i]$

Phase king: phases preserve agreement

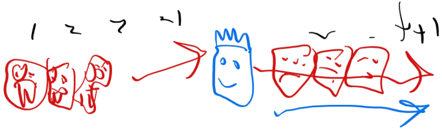


Lemma: If all non-faulty agree on v at start of phase t, all non-faulty agree on v at end of phase t.

Proof:

- $\geq n - f$ processes send v in first round of phase t.
- Let i be non-faulty.
- $\text{majority}_i \leftarrow v$
- $\text{multiplicity}_i \geq n - f > n/2 + f$
- i ignores phase king and keeps $\text{pref}_i = v$.

Phase king: proof of correctness



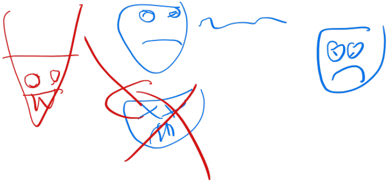
- Termination: yes.
- Validity: Use lemma to show common preference never changes.
- Agreement:
  - At least one phase $\cancel{t}$ k has a non-faulty phase king.
  - At end of phase $\cancel{t}$ k:
    - All non-faulty i that picked kingMajority $= v$ agree.
    - Any non-faulty i that picked majority $= v'$ saw $> n/2 + f$ copies.
    - But then phase king saw $> n/2$ copies of v' and $v' = v$.
    - $\Rightarrow$ all non-faulty i agree.
  - Use lemma for subsequent phases.

What happens with asynchrony?

Just one crash failure prevents agreement!

Fischer-Lynch-Paterson 1985 (FLP)

Assumptions:



- Asynchronous system.
- Up to $f = 1$ crash failure.
- Deterministic processes.
- Binary inputs and outputs.

Requirements:

- Agreement: All outputs are equal.
- Termination: All non-faulty processes finish.
- **Non-triviality**: There exist executions with different outputs.

Non-triviality is much weaker than validity.

FLP: model

Usual message-passing model except we'll assume each process receives at most one message per event.

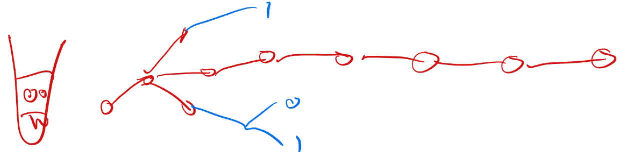
Event is

- (p, m) means p receives m
- $(p, \perp)$ means p takes a step on its own.

In each case, p can send 0 or more messages.

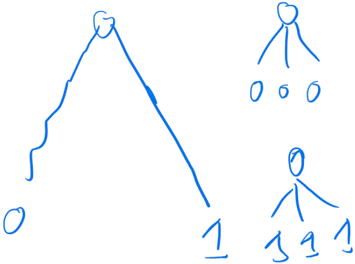
This is equivalent to our usual model since we can just split up $\mathrm{del}(i, S)$ to multiple (i, m) events.

FLP: bivalence



Adversary strategy uses classification of configurations as:

- **Bivalent**: both decision values still possible.
- **0-valent**: only 0 possible.
- **1-valent**: only 1 possible.
- **Univalent**: 0-valent or 1-valent.



Goal: Start bivalent, stay bivalent.

Bivalent ⇒ nobody decided yet.

FLP: start bivalent

0 0 1 1 1 0 → 0-valent
0 0 0 1 1 0 → ? 1-valent
0 1 0 1 1 0 →

1. Suppose all initial C are univalent.
2. Then $\exists C_0, C_1$ where
   2.1 C_i is i-valent.
   2.2 C_0 and C_1 differ only in input of some process i.
3. Consider a terminating execution $C_0\alpha$ in which i takes no steps.
4. Then $C_0\alpha \sim_j C_1\alpha$ for all $j \neq i$.
5. C_0 0-valent $\Rightarrow$ $C_0\alpha$ decides 0 $\Rightarrow$ $C_1\alpha$ decides 0, contradicting 1-valence of C_1.

So: not all initial C are univalent,.

FLP: stay bivalent

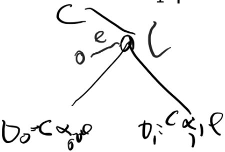
Goal: starting in bivalent C, stay bivalent forever with no failures.

Obstacle: fairness!

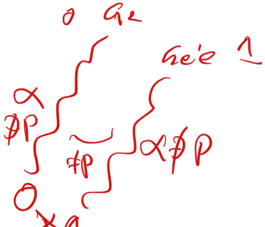
- Any pending event e must happen eventually.
- So need to find extension $C\alpha e$ that is still bivalent.
- Let $S = \{\text{all configurations } C\alpha \text{ where } \alpha \not\ni e\}$.
- If there some $D \in S$ such that De is bivalent, we win.
- If not:
  - There exist $D_0, D_1 \in S$ such that $D_i e$ is i-valent.
    - Proof: If not, C is univalent.
  - Find least common ancestor L of D_0, D_1.
  - Assumed without loss of generality that Le is 0-valent.
  - There is some pair G, Ge' on L to D_1 path such that
    - Ge is 0-valent
    - $Ge'e$ is 1-valent

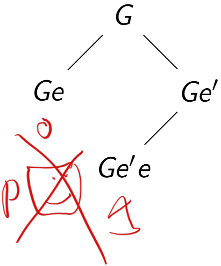
If $C\alpha e$ bivalent $\Rightarrow$ do it.

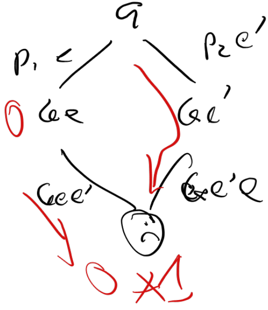
If no $C\alpha e$ bivalent



The impossible FLP diamond







- Ge is 0-valent
- $Ge'e$ is 1-valent
- What events are e and e'?
  1. e and e' are events of different processes i, j, then
     - Gee' is 0-valent.
     - $Ge'e$ is 1-valent.
     - $Gee' \sim_k Ge'e$ for all processes k.
     - So $Gee'\alpha$ and $Ge'e\alpha$ decide same value: a contradiction!
  2. e and e' are events of the same process i
     - Crash i!
     - Now $Gee'\alpha$ and $Ge'e\alpha$ again decide same value.

What went wrong?

- We assumed all $C\alpha e$ were not bivalent.
- Contradiction means some $C\alpha e$ is bivalent.

So: Starting in bivalent C, for any pending event e, can find an execution $C\alpha e$ such that

- $C\alpha e$ is still bivalent.
- e happened.

Adversary strategy: Keep doing this to oldest pending event.

This gives an infinite fair execution that never decides.

Next time.

Paxos! (Lamport 1990, Lamport 1998, Lamport 2001)

- The only asynchronous consensus protocol in actual use.
- Gives up termination.