Today’s Feature Presentation

- simple, not practical algorithm
- exponential number of operations per process
- BUT more practical protocols exist
  - down to $O(n \log^2 n)$ expected operations/ process
  - $n$-1 resilient

The idea

- Infinite repetition of asynchronous rounds
  - endless broadcast of $a$-values and $b$-values
  - two phases per round
  - no termination, but eventual decision
  - during round $r$, process only handles messages timestamped with round $r$

Ben Or’s Algorithm

```
1: $a_r$ := input bit; $r \rightarrow 1$
2: repeat forever
3: (phase 1)
4: send $(a_r, r)$ to all
5: Let $A$ be the multiset of the first $n$ $a'$s with timestamp $r$ received
6: if $(3r \in \{0,1\})$ : $\forall a : \exists a'$ $a = a'$ then $b_r = v$
7: else $b_r = \bot$
8: (phase 2)
9: send $(b_r, r)$ to all
10: Let $B$ be the multiset of the first $n$ $b'$s with timestamp $r$ received
11: if $(3r \in \{0,1\})$ : $\forall b : \exists b'$ $b = b'$ then decide$(v)$: $a_r = v$
12: else if $(3r \in \{0,1\})$ : $a_r = b$
13: else $a_r$ := $\$ (is chosen uniformly at random to be 0 or 1)
14: $r := r + 1$
```

Validity

- All identical inputs ($i$)
- Each process set $a$-value := $i$ and broadcasts it to all
- Since at most $f$ faulty, every correct process receives at least $n - f$ identical $a$-values in round 1
- Every correct process sets $b$-value := $i$ and broadcasts it to all
- Again, every correct process receives at least $n - f$ identical $b$-values in round 1 and decides $i$
A Useful Observation

Lemma. For all $r$, either $b_r \in \{1,4\}$ for all $p$ or $b_r \in \{0,1\}$ for all $p$.

Proof. By contradiction.
- Suppose $p$ and $q$ reach round $r$ such that $b_p = 0$ and $b_q = 1$.
- From line 8, $p$ receives $n - f$ distinct $0$s, $q$ received $n - f$ distinct 1s.
- Then, $2n - 2f$.
- But this implies $n/2$!
  
  **Contradiction**

Corollary. It is impossible that two processes $p$ and $q$ decide at round $r$ on different values.

Agreement

- Let $r$ be the first round in which a decision is made.
- Let $p$ be a process that decides in $r$.
- By the Corollary, no other process can decide on a different value in $r$.
- To decide, $p$ must have received $n - f$ from distinct processes.
- Every other correct process has received $n/2$ from at least $n - 2f + 1$.
- By lines 11 and 12, every correct process sets its new $a$-value to for round $r + 1$.
- By the same argument used to prove Validity, every correct process that has not decided $\uparrow$ in round $r$ will do so by the end of round $r + 1$.

Termination I

- Remember that by Validity, if all (correct) processes propose the same value $\uparrow$ in phase 1 of round $r$, then every correct process decides $\uparrow$ in round $r$.
- The probability of all processes proposing the same input value ($\text{landslide}$) in round 1 is

$$\Pr[\text{landslide in round 1}] = 1/2^n$$

- What can we say about the following rounds?

Termination II

- In round $r > 1$, the $a$-values are not necessarily chosen at random!
- By line 12, some process may set its $a$-value to a non-random value $v$.
- By the Lemma, however, all non-random values are identical!
- Therefore, in every $r$ there is a positive probability at least $1/2^n$ for a landslide.
- Hence, for any round $r$:

$$\Pr[\text{landslide at round } r] \geq 1/2^n$$

- Since coin flips are independent:

$$\Pr[\text{no landslide at first } k \text{ rounds}] = (1 - 1/2^n)^k$$

- When $k \rightarrow \infty$, this value is about 0; then, if $k = 2^n$:

$$\Pr[\text{landslide within } k \text{ rounds}] = 1 - (1 - 1/2^n)^{2^n}$$

which converges quickly to 0 as $c$ grows.
Unreliable Failure Detectors for Reliable Distributed Systems

A different approach

- Augment the asynchronous model with an unreliable failure detector for crash failures
- Define failure detectors in terms of abstract properties, not specific implementations
- Identify classes of failure detectors that allow to solve Consensus

The Model

General
- asynchronous system
- processes fail by crashing
- a failed process does not recover

Failure Detectors
- outputs set of processes that it currently suspects to have crashed
- the set may be different for different processes