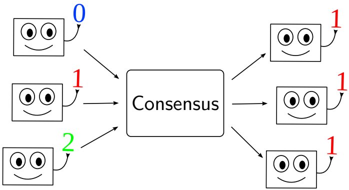
Consensus with max registers

James Aspnes and He Yang Er

DISC 2019

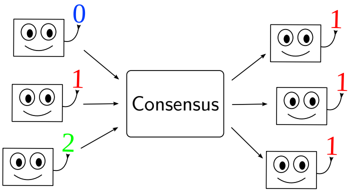
Main result



We will solve randomized, wait-free

- **consensus** for an
- **oblivious adversary** using
- **max registers** in
- $O(\log^* n)$ expected steps per process.

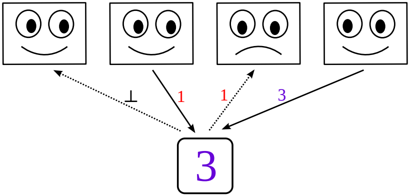
Consensus



- **Termination:** All non-faulty processes terminate (with probability 1).
- **Validity:** Every output value is somebody's input.
- **Agreement:** All output values are equal.

No deterministic solutions in message passing (Fischer, Lynch, and Paterson 1985) or shared memory (Loui and Abu-Amara 1987).

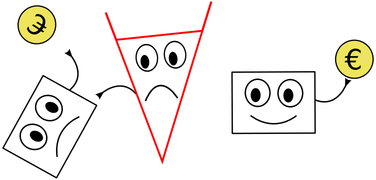
Model



Wait-free shared memory:

- Processes communicate by applying **operations** to **shared objects**.
- Each operation is one **step**.
- No fairness: adversary can choose any process to take the next step.
- Cost measure: Worst-case expected steps taken by a single process.

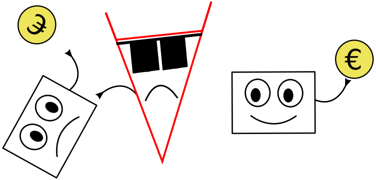
Randomization and the adversary



- Each process can flip local coins.
- Adversary chooses which process takes the next step.
  - **Adaptive adversary:** Sees coins and process actions.
  - **Oblivious adversary:** Doesn't see anything.

Adaptive adversary make consensus much harder (Attiya and Censor 2010), so we will assume oblivious.

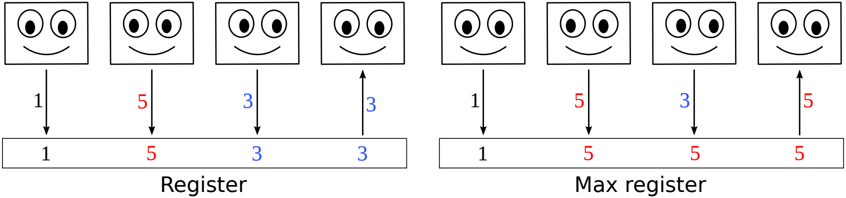
Randomization and the adversary



- Each process can flip local coins.
- Adversary chooses which process takes the next step.
  - **Adaptive adversary:** Sees coins and process actions.
  - **Oblivious adversary:** Doesn't see anything.

Adaptive adversary make consensus much harder (Attiya and Censor 2010), so we will assume oblivious.

Max registers



- **Atomic registers**: return *last* value written.
  - **Multi-writer atomic registers** allow anybody to write.
  - **Single-writer atomic registers** only allow owner to write.
- **Max registers**: return *largest* value written.
  - (Always multi-writer.)

Like atomic registers, max registers have **consensus number 1**: can't solve consensus without randomization.

Previous bounds

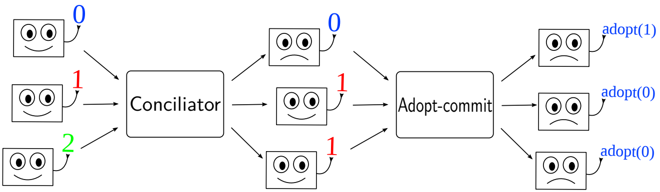
- $O(\log \log n)$ expected steps for *multi-writer* registers (Aspnes 2015).
- $O(n \log^2 n)$ expected steps for *single-writer* registers (Aspnes and Waarts 1996).

We will get:

- $O(\log^* n)$ expected steps for multi-writer *max* registers.
- $O(n \log n)$ expected steps for single-writer *atomic* registers.

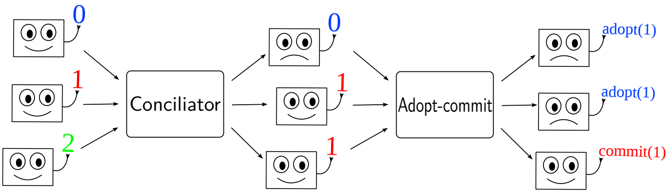
Note: *No* known non-trivial bounds on expected steps with oblivious adversary.

How to build a consensus protocol



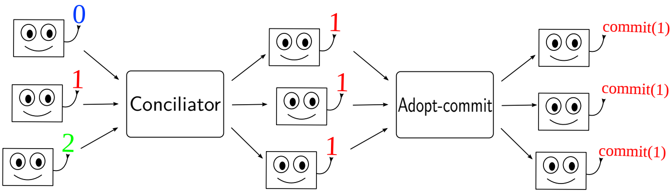
- **Conciliator** produces agreement (Aspnes 2012)
  - Inputs equal $\Rightarrow$ all outputs equal common input.
  - Inputs not equal $\Rightarrow$ outputs equal with probability $> \delta$.
- **Adopt-commit** detects agreement (Gafni 1998)
  - `adopt(`v`)` $\Rightarrow$ choose v as your new value.
  - `commit(`v`)` $\Rightarrow$ everybody else will choose v.
  - Inputs equal $\Rightarrow$ everybody commits to common input.
- Together, solve consensus after $O(1/\delta)$ expected phases.

How to build a consensus protocol



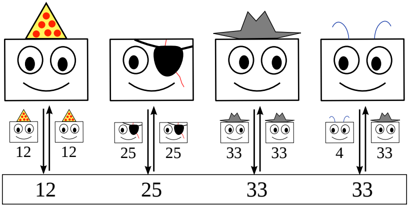
- **Conciliator** produces agreement (Aspnes 2012)
  - Inputs equal $\Rightarrow$ all outputs equal common input.
  - Inputs not equal $\Rightarrow$ outputs equal with probability $> \delta$.
- **Adopt-commit** detects agreement (Gafni 1998)
  - `adopt(`v`)` $\Rightarrow$ choose v as your new value.
  - `commit(`v`)` $\Rightarrow$ everybody else will choose v.
  - Inputs equal $\Rightarrow$ everybody commits to common input.
- Together, solve consensus after $O(1/\delta)$ expected phases.

How to build a consensus protocol



- **Conciliator** produces agreement (Aspnes 2012)
  - Inputs equal $\Rightarrow$ all outputs equal common input.
  - Inputs not equal $\Rightarrow$ outputs equal with probability $> \delta$.
- **Adopt-commit** detects agreement (Gafni 1998)
  - `adopt(`v`)` $\Rightarrow$ choose v as your new value.
  - `commit(`v`)` $\Rightarrow$ everybody else will choose v.
  - Inputs equal $\Rightarrow$ everybody commits to common input.
- Together, solve consensus after $O(1/\delta)$ expected phases.

Conciliators with max registers



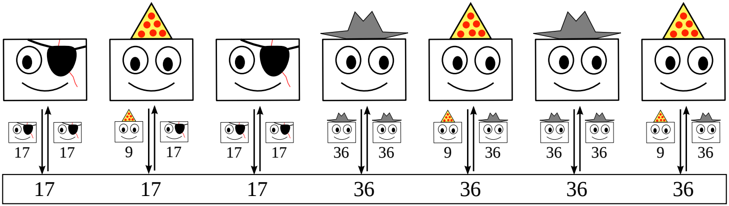
Do for $O(\log^* n)$ rounds:

- Assign a random **priority** to each value.
- Write (priority, value) to max register.
- Read new value from max register.

The idea:

- Only left-to-right maxima survive.
- So i-th value survives with probability $1/i$.
- Expected total survivors $= \sum \frac{1}{i} = H_n = O(\log n)$.

What happens after the first round?



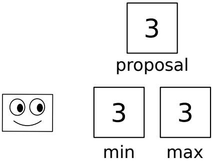
Problem:

- Same value appears in multiple processes.
- $\Rightarrow$ multiple chances to survive!

Use **personae** (Aspnes 2015):

- Generate priorities for all rounds in advance.
- Propagate priorities with values.
- v survives only if first copy of v survives.
- This gives $n \to O(\log n) \to O(\log \log n) \to \ldots$ expected survivors.
- *One* survivor with constant probability δ after $O(\log^* n)$ rounds.

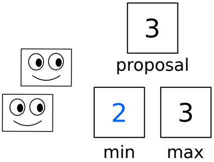
Constant-time adopt-commit with max registers



- Rules:
  - I get $\texttt{commit}(v) \Rightarrow$ you get $\texttt{commit}(v)$ or $\texttt{adopt}(v`)$
  - All inputs $v \Rightarrow$ I get $\texttt{commit}(v)$
- Algorithm:
  - Write v to $\texttt{min}$ and $\texttt{max}$
  - If $\texttt{proposal}$ is not empty, $v \leftarrow \texttt{proposal}$; else $\texttt{proposal} \leftarrow v$
  - If $\texttt{min} = v$ and $\texttt{max} = v$, $\texttt{commit}(v)$; else $\texttt{adopt}(v)$

Commit $\Rightarrow$ I wrote proposal before conflicting processes read it.

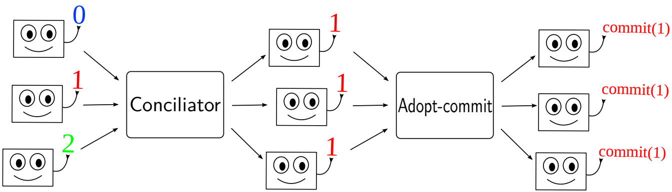
Constant-time adopt-commit with max registers



- Rules:
  - I get $\texttt{commit}(v) \Rightarrow$ you get $\texttt{commit}(v)$ or $\texttt{adopt}(v`)$
  - All inputs $v \Rightarrow$ I get $\texttt{commit}(v)$
- Algorithm:
  - Write v to $\texttt{min}$ and $\texttt{max}$
  - If $\texttt{proposal}$ is not empty, $v \leftarrow \texttt{proposal}$; else $\texttt{proposal} \leftarrow v$
  - If $\texttt{min} = v$ and $\texttt{max} = v$, $\texttt{commit}(v)$; else $\texttt{adopt}(v)$

Commit $\Rightarrow$ I wrote proposal before conflicting processes read it.

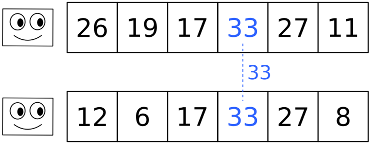
Full result



- $1/\delta$ phases on average until conciliator succeeds.
- Conciliator takes $O(\log^* n)$ steps.
- Adopt-commit takes $O(1)$ steps.

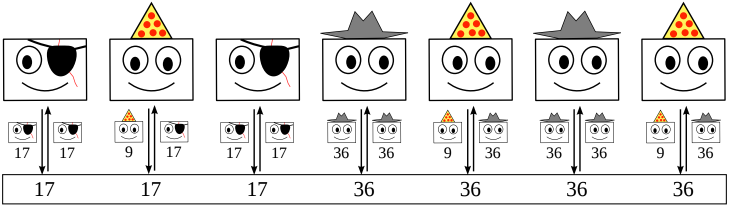
So $O(\log^* n)$ expected steps until agreement.

Max registers from single-writer registers



- For conciliator, use **double collect** snapshot.
  - **Collect** reads all n registers.
  - Repeat until max value doesn't change.
  - Repeated max value $=$ max value between collects.
- Each new max value $\Rightarrow$ one extra collect.
  - New max values $= O(\log n + \log\log n + \dots) = O(\log n)$.
  - Total cost $= O(n \log n)$ register operations.
- Beats previous $O(n \log^2 n)$ bound for (adaptive adversary) single-writer consensus.

Open problems



- Max registers give randomized consensus in $O(\log^* n)$ expected steps against an oblivious adversary.
  - But still no lower bounds other than $\Omega(1)$.
  - Can we do better with max registers?
  - Can we do as well or better with ordinary registers?
- Translating to single-writer registers gives $O(n \log n)$ expected steps.
  - Also no lower bounds other than $\Omega(n)$.
  - Can we reduce overhead of the translation?