A modular approach to shared-memory consensus, with applications to the probabilistic-write model

James Aspnes
Yale University

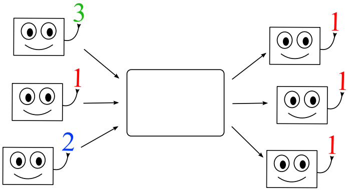
July 28th, 2010

Randomized consensus

Want *n* processes to agree on one of *m* values.

- **Validity**: each output equals some input.
- **Termination**: all non-faulty processes finish with probability 1.
- **Agreement**: all non-faulty processes get the same output.

Model: **Wait-free** asynchronous shared-memory with **multi-writer registers**.



Bounds on consensus

- Tight bounds for extreme cases:
  - **Adaptive adversary**, processes only have **local coins**: $\Theta(n^2)$ expected total operations (Attiya and Censor, 2008), $\Theta(n)$ expected operations per process (Aspnes and Censor, 2009).
  - **Oblivious adversary**, **global coin**, 2 values: $\Omega(1)$ expected operations per process with geometric distribution (Attiya and Censor, 2008), matching upper bound (Aumann, 1997).
- We want to know what happens in the middle: local coins but weak adversary.

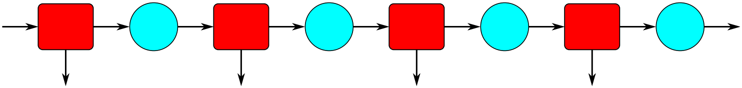
Probabilistic-write model

In the **probabilistic-write model**, after the adversary schedules a process to do a write, it can flip a coin to decide whether to do so or not.

- This is the **strong model** of (Abrahamson, 1988).
- Used by (Cheung, 2005) to get $O(n \log \log n)$ total and individual work for 2-valued consensus.
- We'll get $O(n \log m)$ total and $O(\log n)$ individual work for m-valued consensus.
- $O(\log n)$ individual work is similar to bounds for other weak-adversary models (Chandra, 1996; Aumann, 1997; Aumann and Bender, 2005).
- No lower bounds better than $\Omega(1)$.

(All bounds are in expectation.)

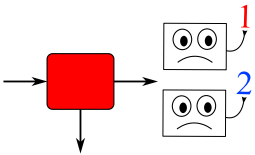
Decomposing consensus



- Most known consensus protocols alternate between detecting agreement and producing agreement.
- We will make this explicit by decomposing consensus into:
  1. **Ratifier** objects, which detect agreement, and
  2. **Conciliator** objects, which produce it with some probability.
- Essentially just **refactoring** existing code.

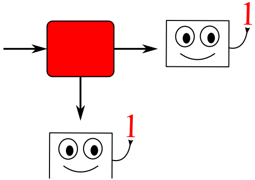
Ratifiers

- Like ordinary consensus objects, except:
  - Output is supplemented with a **decision bit** that says whether to decide on the output (1) or adopt it for later stages of the protocol (0).
  - Agreement is replaced by two new conditions:
    1. **Coherence**: If one process decides on x, every other process gets x as output (but might not decide).
    2. **Acceptance**: If all inputs are equal, all processes decide.
- These are just Gafni's **adopt-commit protocols** (Gafni, 1998) expressed as shared-memory objects.



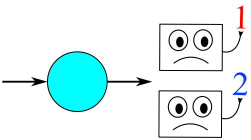
Ratifiers

- Like ordinary consensus objects, except:
  - Output is supplemented with a **decision bit** that says whether to decide on the output (1) or adopt it for later stages of the protocol (0).
  - Agreement is replaced by two new conditions:
    1. **Coherence**: If one process decides on x, every other process gets x as output (but might not decide).
    2. **Acceptance**: If all inputs are equal, all processes decide.
- These are just Gafni's **adopt-commit protocols** (Gafni, 1998) expressed as shared-memory objects.



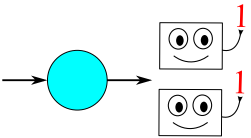
Conciliators

- Like ordinary consensus objects, except agreement is replaced by:
  - **Probabilistic agreement**: All outputs are equal with probability at least δ, for some fixed $\delta > 0$.
- Conciliator objects have the same role as **weak shared coins** of (Aspnes and Herlihy, 1990) (and can be built from weak shared coins).
- But can also be built other ways, e.g. using the first-mover mechanism of (Chor, Israeli, and Li, 1994).

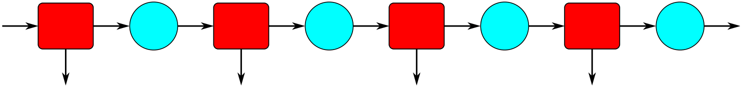


Conciliators

- Like ordinary consensus objects, except agreement is replaced by:
  - **Probabilistic agreement**: All outputs are equal with probability at least δ, for some fixed $\delta > 0$.
- Conciliator objects have the same role as **weak shared coins** of (Aspnes and Herlihy, 1990) (and can be built from weak shared coins).
- But can also be built other ways, e.g. using the first-mover mechanism of (Chor, Israeli, and Li, 1994).



Recomposing consensus



Given infinite alternating sequence of ratifiers and conciliators:

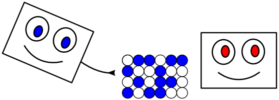
1. Validity follows from validity of components.
2. Agreement follows from coherence + validity.
3. For termination, we go through at most $(1/\delta)$ conciliators on average before one of them produces agreement (probabilistic agreement); then following ratifier makes all processes decide (acceptance).

Building a ratifier

- Basic idea:
  1. **Announce** my input v (using mechanism to be provided later).
  2. If proposal $= \bot$, proposal $\leftarrow v$; else $v \leftarrow$ proposal.
  3. Decide v if no $v' \neq v$ has been announced, else output v without deciding.
- Why it works:
  - If some value v is in proposal before any other v' is announced, any process with v' sees and adopts v.
- Announce-propose-check structure same as in Gafni's adopt-commit protocol (Gafni, 1998), but we'll exploit multi-writer registers to reduce cost.

How to announce a value

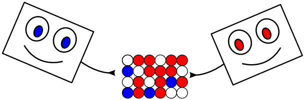
- Assign unique write quorum W_v of k out of $2k$ registers to each value v, where $k = \Theta(\log m)$ satisfies $\binom{2k}{k} \geq m$.
- Announce v by writing all registers in W_v.
- Detect $v' \neq v$ by reading all registers in $\overline{W_v}$.
- I always see you if you finish writing $W_{v'}$.



Cost of ratifier: $O(\log m)$ individual work and $O(\log m)$ space.

How to announce a value

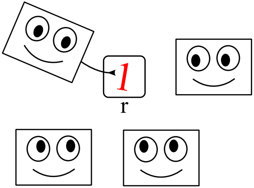
- Assign unique write quorum W_v of k out of $2k$ registers to each value v, where $k = \Theta(\log m)$ satisfies $\binom{2k}{k} \geq m$.
- Announce v by writing all registers in W_v.
- Detect $v' \neq v$ by reading all registers in $\overline{W_v}$.
- I always see you if you finish writing $W_{v'}$.



Cost of ratifier: $O(\log m)$ individual work and $O(\log m)$ space.

Building a conciliator

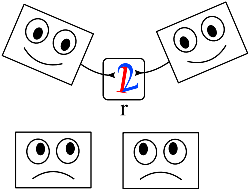
$k \leftarrow 0$
while $r = \bot$ **do**
  write v to r with probability $\frac{2^k}{2n}$
  $k \leftarrow k + 1$
end
return r



- Uses Chor-Israeli-Li technique: First value written wins unless overwritten.
- Increasing probabilities means a lone process finishes quickly.
- But other processes will still have low total probability of overwriting before reading again (or they would have finished already).
- Cost: $O(\log n)$ individual work, $O(n)$ total work, and 1 register.

Building a conciliator

$k \leftarrow 0$
while $r = \bot$ **do**
    write v to r with probability $\frac{2^k}{2n}$
    $k \leftarrow k + 1$
end
return r



- Uses Chor-Israeli-Li technique: First value written wins unless overwritten.
- Increasing probabilities means a lone process finishes quickly.
- But other processes will still have low total probability of overwriting before reading again (or they would have finished already).
- Cost: $O(\log n)$ individual work, $O(n)$ total work, and 1 register.

Conclusions

- Ratifier + conciliator = n-process, m-valued consensus in the probabilistic-write model with
  - $O(\log n + \log m)$ expected individual work.
  - $O(n \log m)$ expected total work.
  - $O(\log m)$ expected space used.
- This just says

$$T_{\text{consensus}} = O\left(T_{\text{ratifier}} + T_{\text{conciliator}}\right).$$

- But: consensus objects are both ratifiers and conciliators. So we also have

$$T_{\text{consensus}} = \Omega\left(T_{\text{ratifier}} + T_{\text{conciliator}}\right).$$

- These bounds hold for *any* additive cost measure in in *any* model.
- Moral: If you want upper *or* lower bounds for consensus, look for bounds on ratifiers and conciliators.