Randomized Consensus

Randomization allows us to solve problems that cannot be solved by deterministic algorithms. This tutorial presents a randomized algorithm that solves the consensus problem.

In addition to the non-determinism imposed by the adversary, which consists of the inputs and the schedule, it introduces another source of non-determinism by the local coin-flips.

Since now an adversary (inputs+schedule) does not define one execution but a probability space over executions, we should define the power of the adversary.

- A very weak adversary may have to decide upon the whole schedule in advance.
- A less weaker adversary may decide upon the schedule adaptively, however, it may observe only the shared memory.
- A strong adversary may decide upon the schedule adaptively and may observe all of the local states of the processes.

We will use randomization in order to overcome the impossibility of achieving consensus in a shared-memory asynchronous system. We will have to relax our requirements for consensus. We will not relax the agreement condition because it is a safety condition. We do not want to allow disagreement even in a single execution with low probability. Instead, we relax the termination condition:

**Termination with probability 1** For every non-faulty process, the expected number of steps until it decides is finite (it eventually decides, with probability 1).

This does not mean that there may be no executions that do not terminate, only that their probability is 0.

**shared coin** A *shared coin* with agreement parameter $\rho$ is an algorithm without inputs, which has probability $\rho$ for all the outputs to be 0, and probability $\rho$ for all the outputs to be 1. In the rest of the executions there may not be agreement.

We will talk later about shared coins, but for now consider a naive one, which is simply a local coin-flip. The agreement parameter is $2^{-n}$. 
We use a phase-based mechanism to guarantee safety, in which a process decides on the result of the shared coin only if there is agreement upon it.

The idea is that in each phase $r$, a process:

- writes its preference into $Propose[r][i]$, and the collects the array.
- If all the preferences were the same, it writes 'agree' to $Check[r][i]$, otherwise it writes 'disagree'.
- It then collects this array to decide upon its new preference (perhaps by a shared coin) or to decide.

Lemma 1 For every phase $r$ the following hold:

1. If all the processes that started phase $r$ have the same preference $v$, then every non-faulty process decide $v$ in this phase.

2. All the processes that write 'agree' to $Check[r]$ wrote the same preference to $Propose[r]$.

3. If a process decides $v$ in phase $r$ then every non-faulty process that finishes phase $r$ sees at least one 'agree' in $Check[r]$.

4. All the processes that see at least one 'agree' in $Check[r]$ and finish phase $r$ have the same preference.

5. If a process decides $v$ in phase $r$ then every non-faulty process decides $v$ in phase $r'$, where $r'$ is $r$ or $r+1$.

It is easy to see that validity is obtained by part 1 of Lemma 1, and that agreement is obtained by part 5 of Lemma 1. Notice that they are both satisfied always, and not just with probability 1.
For termination, we prove the following lemma:

**Lemma 2** The expected number of phases until all non-faulty process decide is $1 + 1/\rho$.

**Proof:** There is probability $\rho$ for all processes whose preference at the end of phase $r$ came from the shared coin to prefer at the beginning of phase $r+1$ the same value $v$, and that $v$ will also be the value preferred by the other processes that decided or chose their preference in Line 12 (by Lemma 1 they can have only one value). So for every phase $r$ there is probability $\rho$ for the next phase to start with identical preferences. Therefore:

$$
E[\text{number of phases till all decide}] = \sum_{i=1}^{\infty} (i \cdot Pr[\text{identical preferences at start of phase } i]) + 1
$$

$$
= \sum_{i=1}^{\infty} (i \cdot (1 - \rho)^{i-1} \cdot \rho) + 1 = \frac{\rho}{1 - \rho} \cdot \sum_{i=1}^{\infty} (i \cdot (1 - \rho)^{i}) + 1
$$

$$
= \frac{\rho}{1 - \rho} \cdot \frac{1 - \rho}{\rho^2} + 1 = 1 + \frac{1}{\rho}
$$

This implies that with a naive shared coin we have an expected number of $O(2^n)$ phases. With each phase requiring $O(n)$ steps because each process does two $O(n)$ collects plus a constant number of steps, we have a randomized consensus algorithm with $O(n \cdot 2^n)$ expected number of steps.