Brief Announcement: Stabilizing Consensus with the Power of Two Choices

Benjamin Doerr[1], Leslie Ann Goldberg[2], Lorenz Minder[3], Thomas Sauerwald[4], and Christian Scheideler[5]

[1] Max-Planck Institute for Computer Science, Saarbrücken
[2] Department of Computer Science, University of Liverpool
[3] Computer Science Division, University of California, Berkeley
[4] Simon Fraser University, Burnaby, Canada
[5] Department of Computer Science, University of Paderborn

Consensus problems occur in many contexts and have therefore been extensively studied in the past. In the original consensus problem, every process initially proposes a value, and the goal is to decide on a single value from all those proposed. We are studying a slight variant of the consensus problem called the *stabilizing consensus problem* [2]. In this problem, we do not require that each process irrevocably commits to a final value but that eventually they arrive at a common, stable value without necessarily being aware of that. This should work irrespective of the states in which the processes are starting. In other words, we are searching for a *self-stabilizing* algorithm for the consensus problem. Coming up with such an algorithm is easy without adversarial involvement, but we allow some adversary to continuously change the states of some of the nodes at will. Despite these state changes, we would like the processes to arrive quickly at a common value that will be preserved for as many time steps as possible (in a sense that almost all of the processes will store this value during that period of time). Interestingly, we will demonstrate that there is a simple algorithm for this problem that essentially needs logarithmic time and work with high probability to arrive at such a stable value, even if the adversary can perform arbitrary state changes, as long as it can only do so for a limited number of processes at a time.

Our approach. We are focussing on synchronous message-passing systems with adversarial state changes. More precisely, we are given a fixed set of n processes that are numbered from 1 to n. The time proceeds in synchronized *rounds*. In each round, every process can send out one or more requests, receive replies to its requests, and perform some local computation based on these replies. We assume that every process can send a message to any other process (i.e., there are no connectivity constraints) and faithfully follows the stated protocol (given its current state, which might have been changed by the adversary).

Stabilizing consensus problem. We are studying a slight variant of the original consensus problem called the *stabilizing consensus problem* [2]. In this problem, we may start with an arbitrary set of initial output values. As in the usual convention, a *configuration* includes all of the processes' local states. A configuration C is called *output-stable* if in all possible executions starting from C, the

N.A. Lynch and A.A. Shvartsman (Eds.): DISC 2010, LNCS 6343, pp. 528–530, 2010.

output of each process does not change. If every process outputs x in an output-stable configuration C, we say the outputs stabilize to x in C. A *self-stabilizing consensus protocol* must satisfy the following three properties:

- **Stabilization:** the protocol eventually reaches an output-stable configuration.
- **Validity:** if a process outputs a value v, then v must have been output by some process in the previous round.
- **Agreement:** for every reachable output-stable configuration, all processes have the same output.

The adversary. We also assume that there is a T*-bounded adversary* that knows the entire configuration of the system in the current round. Based on that information, it may decide to introduce arbitrary state changes in up to T processes *at the end* of each round except that the protocol code cannot be manipulated. Of course, under a T-bounded adversary we cannot reach an output-stable configuration any more. Therefore, we will only require the system to reach a configuration C so that for any polynomial in n many time steps following C, all but at most $\mathcal{O}(T)$ processes agree on some stable value v (note that these $\mathcal{O}(T)$ processes can be different from round to round). We will call this an *almost output-stable configuration.*

Our contribution. We are focussing on stabilizing consensus problems based on an arbitrary (finite or countably infinite) set S of *legal* values with a total order. Classical examples are $S = \{0, 1\}$ and $S = \mathbb{N}$. All initial output values of the processes must be from S and also the adversary is restricted to choosing only values in S. (In case the adversary chooses a value outside of S in some process p, we may assume that p instantly recognizes that and then switches over to some default value in S.) We propose the following simple protocol called the *median rule*:

> In each round, every process i picks two processes j and k uniformly and independently at random among all processes (including itself) and requests their values. It then updates v_i to the *median* of v_i, v_j and v_k. Any request sent to process i will be answered with the value i had *at the beginning* of the current round.

For example, if $v_i = 10$, $v_j = 12$ and $v_k = 100$, then the new value of v_i is 12. The median rule works surprisingly well. We can prove that we reach an almost stable consensus in time $\mathcal{O}(\log m \log\log n + \log n)$ w.h.p.[1], where $m = |S|$. If no adversary is present, then the runtime decreases to $\mathcal{O}(\log m)$ w.h.p. Our results are listed in Table 1.

The work closest to our work is the paper by Angluin et al. [1] in the context of population protocols. Their consensus protocol can tolerate Byzantine behavior of $o(\sqrt{n})$ nodes. However, they only focus on two values whereas we allow any number of legal values, and in our model any node can experience state changes.

[1] We write w.h.p. to refer to an event that holds with probability at least $1 - n^{-c}$ for any constant $c > 1$.

	with adversary	without adversary
worst-case, $m = 2$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
worst-case, arb. m	$\mathcal{O}(\log m \log\log n + \log n)$	$\mathcal{O}(\log m + \log n)$
average-case, arb. m	$\mathcal{O}(\log m + \log\log n)$ if m is odd $\Theta(\log n)$ if m is even	$\mathcal{O}(\log m + \log\log n)$ if m is odd $\Theta(\log n)$ if m is even

Fig. 1. Our results on the time and work required by each process to reach an almost stable consensus (with adversary) or stable consensus (without adversary). $m = |S|$ is the number of legal values. By average-case we refer to the case where every initial value is chosen independently and uniformly at random among $m \leq n^{1/2-\epsilon}$ legal values for some constant $\epsilon > 0$. All results hold with probability at least $1 - n^{-c}$ for any constant $c > 1$.

Of course, $|S|$ may not be finite. In this case, our results still hold if we define m as the number of legal values between v_ℓ and v_r, where v_ℓ is the $(n/2 - c\sqrt{n \log n})$-smallest and v_r is the $(n/2 + c\sqrt{n \log n})$-smallest value of the initial values for some sufficiently large constant c. Therefore, as a by-product, the median rule computes a good approximation of the median, even under the presence of an adversary.

Our results also hold if instead of state changes at arbitrary processes we have a fixed number of $\sqrt{n}$ Byzantine processes. The bound on T is essentially tight as $T = \Omega(\sqrt{n \log n})$ would not allow the median rule to stabilize any more w.h.p. (even if the adversary can only put T processes to sleep instead of changing their states) because the adversary could keep two groups of processes with equal values in perfect balance for at least a polynomially long time.

Finally, if the T-bounded adversary is *static* in a sense that there is a *fixed* set of *bad* T processes that can experience adversarial state changes throughout the execution, then a simple extension of the median rule, called *careful median rule*, reaches a *stable* consensus for all non-bad processes for any polynomial number of time steps w.h.p.: each process runs the median rule as before but outputs the majority value of the last k rounds of the median rule as its value, for some constant $k \geq 3$. A stable consensus is not possible with the original median rule as with $T = \sqrt{n}$ there is a constant probability that some non-bad process contacts two bad processes and therefore changes its value to a value selected by the adversary. The details can be found in [3].

References

1. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. In: Pelc, A. (ed.) DISC 2007. LNCS, vol. 4731, pp. 20–32. Springer, Heidelberg (2007)
2. Angluin, D., Fischer, M., Jiang, H.: Stabilizing consensus in mobile networks. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 37–50. Springer, Heidelberg (2006)
3. Doerr, B., Goldberg, L.A., Minder, L., Sauerwald, T., Scheideler, C.: Stabilizing consensus with the power of two choices. Technical report, University of Paderborn (2010), `http://wwwcs.upb.de/cs/scheideler`