Brief Announcement: Communication-Efficient Byzantine Consensus Without a Common Clock

Danny Dolev[1] and Christoph Lenzen[2]

[1] Hebrew University of Jerusalem, 91904 Jerusalem, Israel
[2] Massachusetts Institute of Technology, MA 02139 Cambridge, USA

Abstract. Many consensus protocols assume a synchronous system in which all processes start executing the protocol at once and in the same round. However, such a common start requires to establish consensus among the correct processes in the first place, making this assumption questionable in many circumstances. In this work, we show that it is possible to consistently initiate consensus instances without a common round counter. Every correct node can initiate consistent consensus instances, without interfering with other nodes' instances. Furthermore, by bounding the frequency at which nodes may initiate instances, Byzantine faulty nodes can be prevented from initiating too many instances.

Keywords: communication complexity, simulation framework

Consensus is a fundamental fault-tolerance primitive in distributed systems, which has been introduced several decades ago [3]. Both in asynchronous and synchronous models, it is a very common assumption that all nodes start to execute the algorithm at a given point in time.[3]

We provide a self-stabilizing solution to this problem. In this brief announcement, we assume a synchronous model and a deterministic binary consensus algorithm. Furthermore, output 0 is supposed to mean "take no action", bearing no effect on the system; it can thus be used as a save fallback value. All of these restrictions are dropped in the full paper [2]. Moreover, our results can be used to derive new algorithms for Byzantine-tolerant self-stabilizing pulse synchronization (cf. [1]). This is subject to future publication.

Problem Statement. We assume that executions proceed in rounds, where in each round, each node may perform local computations and send a message to each other node, where all messages sent by correct nodes are received before the end of the round. Up to $f < n/3$ faulty nodes are controlled by an adversary that can disobey the algorithm in any fashion. In the following we assume that a consensus protocol $\mathcal{P}$ is given. The goal of the *initiation problem* is to enable correct nodes to initiate independent executions of $\mathcal{P}$. More precisely:

[3] If in an asynchronous setting nodes unconditionally wake up and join an instance upon receiving the "first" message, this essentially means to *always* run *any* possible instance, concurrently, resulting in unbounded message complexity in case of Byzantine faults.

1. Each instance carries a label $(r, v) \in \mathbb{N} \times V$, where r is a round and v a node. We say that the corresponding instance is *initialized* by node v in round r (note that the nodes do not know r).
2. For each instance, each correct node decides whether it *participates* in the instance at the beginning of round $r + 2$.
3. We assume that for each instance (r, v), each participating node $w \in V$ can compute some input $i_w(r, v) \in \{0, 1\}$.
4. If correct node w participates in instance (r, v), it terminates this instance at the latest in round $r + R + 4$ and outputs some value $o_w(r, v) \in \{0, 1\}$.
5. If correct nodes w, w' participate in instance (r, v), then $o_w(r, v) = o_{w'}(r, v)$.
6. If all correct nodes participate in an instance with input b, all output b.
7. If $o_w(r, v) \neq 0$ for some correct node participating in instance (r, v), then all correct nodes participate in this instance (and output $o_w(r, v)$).
8. If a correct node v initializes instance (r, v), all correct nodes w participate in this instance with input $i_w(r + 2, w)$.

Compared to "classical" consensus, property 4 corresponds to *termination*, property 5 to *agreement*, and property 6 to *validity*. Note that validity is replaced by a safety property in case not all correct nodes participate: property 7 states that non-zero output is feasible only if no correct node is left out. Finally, property 8 makes sure that all nodes participate in case a non-faulty node initializes an instance, therefore ensuring validity for such instances.

Theorem 1. *Given a synchronous, deterministic R-round consensus protocol resilient to f faults, we can construct an algorithm solving the initiation problem that self-stabilizes within $R + 4$ rounds.*

Communication complexity. In case of crash faults, our solution is efficient in the sense that for each initiated instance, there is an overhead of 4 rounds and 4 broadcasts per node. However, Byzantine faults may result in each faulty node initiating an instance every round, even if correct nodes are known to do so very infrequently.

If we decide that correct nodes may initialize an instance at most once within T rounds, we can force faulty nodes to do so as well. The modified algorithm can be implemented such that it self-stabilizes in $\min\{T, R + 4\}$ rounds. Note that the choice of $T = R$ is particularly attractive, enabling nodes to initiate and complete an instance within $2R + 4$ rounds, yet ensuring that never more than one instance per node causes communication.

References

1. Dolev, D., Hoch, E.N.: Byzantine Self-Stabilizing Pulse in a Bounded-Delay Model. In: Proc. 9th Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS). pp. 234–252 (2007)
2. Dolev, D., Lenzen, C.: Communication-Efficient Byzantine Consensus Without a Common Clock. Computing Research Repository abs/1307.7976 (2013)
3. Pease, M., Shostak, R., Lamport, L.: Reaching Agreement in the Presence of Faults. Journal of the ACM 27, 228–234 (1980)