View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Brief Announcement: Crash-Tolerant Consensus in Directed Graph Revisited

Ashish Choudhury[*1], **Gayathri Garimella**[†2], **Arpita Patra**[‡3], **Divya Ravi**[4], **and Pratik Sarkar**[5]

1 **International Institute of Information Technology Bangalore, India**
ashish.choudhury@iiitb.ac.in
2 **International Institute of Information Technology Bangalore, India**
AnnapurnaGayathri.Garimella@iiitb.org
3 **Computer Science and Automation, Indian Institute of Science, Bangalore, India**
arpita@csa.iisc.ernet.in
4 **Computer Science and Automation, Indian Institute of Science, Bangalore, India**
divya.ravi@csa.iisc.ernet.in
5 **Computer Science and Automation, Indian Institute of Science, Bangalore, India**
pratik.sarkar@csa.iisc.ernet.in

Abstract

We revisit the problem of distributed consensus in directed graphs tolerating crash failures; we improve the round and communication complexity of the existing protocols. Moreover, we prove that our protocol requires the optimal number of communication rounds, required by any protocol belonging to a specific class of crash-tolerant consensus protocols in directed graphs.

1998 ACM Subject Classification B.8.1 Reliability, Testing and Fault-Tolerance, E.1 Distributed data structures

Keywords and phrases Directed graph, Consensus, Crash failure, Round complexity

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.46

1 Introduction

A crash-tolerant reliable consensus protocol [3] allows a set of n mutually distrusting parties, each with some private input, to agree on a common output. This is ensured even in the presence of a *computationally unbounded* centralized adversary, who may crash any f out of the n parties and try to prevent the remaining parties from achieving consensus. While most of the prior work in the literature (see [2, 1, 4] and their references) have considered the undirected graph model, where parties are assumed to be a part of a *complete undirected graph*, in [5], necessary and sufficient condition for crash-tolerant consensus is presented for a more generic *directed* graph model. We revisit the round complexity of crash-tolerant consensus protocols in the latter model.

* The author would like to acknowledge the financial support from Infosys foundation.
† The author would like to acknowledge the financial support from Infosys foundation.
‡ The author would like to acknowledge the financial support by INSPIRE Faculty Fellowship (DST/INSPIRE/04/2014/015727) from Department of Science & Technology, India.

© Ashish Choudhury, Gayathri Garimella, Arpita Patra, Divya Ravi, and Pratik Sarkar;
licensed under Creative Commons License CC-BY
31st International Symposium on Distributed Computing (DISC 2017).
Editor: Andréa W. Richa; Article No. 46; pp. 46:1–46:4

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Informally, in directed graphs the necessary condition for crash-tolerant consensus demands that even if an arbitrary set of f nodes crashes, there should still exist a special node in the graph, called *source*, which should have a directed path to every other node in the remaining graph. The authors in [5] proved the sufficiency of this condition by presenting two consensus protocols, a consensus protocol for the binary domain and a multi-valued consensus protocol for an arbitrary domain $\{0, \ldots, K\}$. These protocols belong to a special class of protocols, based on "flooding". In more detail, the protocols consist of several "phases" of d rounds of "send-receive-update", where d is called the *crash-tolerant diameter* of a directed graph. Informally, d is the maximum distance of any node from a potential source in the graph. Thus any given potential source can propagate its value to all remaining nodes in a single phase within the d rounds of flooding. In a round every node (including the source) broadcasts its value to its neighbours. At the end of the round, each node "updates" its value, by locally applying an update function to the received values. In the subsequent round, nodes broadcast their updated value. The two types of update function applied are a min function for a *min phase* and a max function for a *max phase*. The min (resp. max) function requires nodes to update their value by taking the minimum (resp. maximum) of all the received values (including its own value).

The binary consensus protocol of [5] requires $2f + 2$ alternate min-max phases, each with d rounds. The round complexity of the protocol is $(2f + 2) \cdot d$ rounds and the communication complexity is $\mathcal{O}(nfd)$ bits. In [5] the authors claimed that their binary consensus protocol cannot be extended trivially to the multi-valued case. They present a multi-valued consensus protocol, which requires $(2f + 2) \cdot d \cdot K$ rounds of interaction and communication complexity is $\mathcal{O}(nfdK \log K)$ bits. Clearly the protocol has exponential round and communication complexity, as $K = 2^{\log K}$ (K is the domain size).

Our Results. In this work, we improve the round and communication complexity of the min-max based consensus protocols of [5]. We consider the binary consensus protocol of [5] and observe that if instead of d, we allow $d+1$ rounds of communication in each of the phases, then it is possible to achieve consensus with just $f+2$ alternate min-max phases, thus making the round complexity $(f + 2)(d + 1)$. We then show an optimization of our protocol, where we allow *only* d rounds in the *first* and the *last* phase, thus reducing the round complexity to $(f + 2)(d + 1) - 2$. Interestingly, we show that our protocol works even for the multi-valued case, with *no* modifications what so ever. Thus, unlike [5], the round complexity of our multi-valued consensus protocol is *independent* of K. The communication complexity of our protocol is $\mathcal{O}(nfd \log K)$ bits and for significantly large values of K our protocol improves upon the round and communication complexity of the multi-valued consensus protocol of [5]. Moreover, we improve the number of rounds for the binary consensus, for every $f, d \geq 2$.

We also address the problem of lower bound on the minimum number of rounds required by any crash-tolerant consensus protocol in a directed graph, based on min-max strategy and derive three interesting lower bounds. We first consider the case, where only $f + 1$ min-max phases are allowed in the protocol and with *no restriction* on the number of communication rounds in each phase. We show that it is impossible to achieve crash-tolerant consensus within $f + 1$ phases. Next we consider min-max based consensus protocols with *at least* d rounds in each phase. For such protocols, we show that it is impossible to achieve consensus in general with $(f + 2)(d + 1) - 3$ rounds in total. This implies that our min-max based protocol with $(f + 2)(d + 1) - 2$ rounds is *round optimal*. Finally we consider min-max based consensus protocols with *exactly* d rounds of communication in each phase. Note that the consensus protocols of [5] belong to this class. For several values of f and d, we show that the

minimum number of phases required to achieve consensus in this case is $2f+2$, thus showing that the binary consensus protocol of [5] has the *optimal* number of communication rounds. The lower bounds establish that our protocol is the best in terms of the round complexity if one is interested to design consensus protocols based on min-max strategy.

High Level Description of Our Protocol. Our starting point is the binary consensus protocol of [5] with $2f+2$ phases, each with d rounds. The correctness of their protocol is based on the guaranteed occurrence of two *consecutive* crash-free phases, among the $2f+2$ alternate min-max phases, within which consensus is shown to be achieved. We observe that if instead of d rounds, we allow $d+1$ rounds in each phase then consensus can be achieved if we either have two consecutive crash-free phases or a crashed phase followed by a crash-free phase, provided *only one* node crashes during the crashed phase. The base of our observation is the following: if during the crashed phase the single node to be crashed is a non-source node, then it is equivalent to having two consecutive crash-free phases (with source node(s) being unaltered) and so consensus will be achieved within these two phases. On the other hand, if during the crashed phase the single node to be crashed is a source node, then at least one of new source nodes will be at a distance of *one* from the crashed source (this observation lies at the heart of our protocol). So if at all the crashed source node sends its value to one of the new source node before crashing, there will be still d rounds left for this new source node in the crashed phase to further propagate the crashed source node's value in the remaining graph. So in essence, we still get the effect of two consecutive crash-free phases. We further show that with $f+2$ alternate min-max phases, there always exist either two crash-free phases or a crashed phase with a single crash, followed by a crash-free phase.

We find that the above ideas are applicable even for the multi-valued case. For simplicity, we consider the case when there are two crash-free phases and without loss of generality, let these be a min phase followed by a max phase. Let λ^{min} be the least value among the source nodes at the beginning of crash-free min phase. If the non-source nodes have their value greater than or equal to λ^{min} at the beginning of this phase, then clearly consensus will be achieved at the end of this min phase itself; this is because each node will update their value to λ^{min} at the end of the min phase. On the other hand, if some *non-source* node has a value smaller than λ^{min} at the beginning of the crash-free min phase, then consensus will not be achieved in this phase. However, at the end of this min phase, the modified values of all the nodes (both source as well as non-source) is upper bounded by λ^{min}; moreover all the *source nodes* will have λ^{min} as their modified value. Hence in the next crash-free phase which is a max phase, the value λ^{min} of the source nodes will be the maximum value in the graph and hence consensus will be achieved at the end of the crash-free max phase. The above argument also works for the case when there is a crashed phase followed by a crash-free phase, where it is guaranteed that exactly one node crashes during the crashed phase. The complete formal details of the protocols and the lower bounds will be available in the full version of the article.

References

1 H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulation and Advanced Topics.* Wiley series on Parallel and Distributed Computing, 2004.

2 N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

3 M. Pease, R. E. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *JACM*, 27(2):228–234, 1980.

4 L. Tseng. Recent Results on Fault-Tolerant Consensus in Message-Passing Networks. In *SIROCCO*, volume 9988 of *Lecture Notes in Computer Science*, pages 92–108, 2016.
5 L. Tseng and N. H. Vaidya. Crash-Tolerant Consensus in Directed Graphs. *CoRR*, abs/1412.8532, 2014. Full version appeared in PODC 2015.