

# Scalable Byzantine Agreement

Clifford Scott Lewis

Jared Saia \*

## Abstract

This paper gives a scalable protocol which solves the synchronized Byzantine agreement problem. For Byzantine agreement over  $n$  processors, our protocol requires each processor to send only  $O(\log n)$  messages in expectation. To the best of our knowledge this is the first result for this problem which requires each processor to send  $o(n)$  messages. Our protocol uses randomness and is correct with high probability<sup>1</sup> It can tolerate any fraction of faulty processors which is strictly less than  $1/8$ . Our motivation for this problem is to provide a building block for secure and scalable computation in peer-to-peer systems.

## 1 Introduction

Peer-to-peer (p2p) networks have emerged for a wide range of applications including data-sharing (e.g. Napster [17], Gnutella [13], Kazaa [15] and Morphus [16]), computation (e.g. SETI@home [18], FOLDING@home [12] DataSynapse [11], NetBatch [9]), collaboration (e.g. Groove Networks [14]), internet infrastructure systems (e.g. I3 [10]), and digital payment systems (e.g. Yaga [19]). Distributed computation is an integral part of many of these new p2p systems. Some of these systems use a trusted third party to initiate or direct the computation. However in many cases, a trusted third party may not be available and so completely distributed algorithms are required.

Unfortunately, distributed computation by a p2p network is vulnerable to attack. Because p2p systems generally lack admission control, there can be large numbers of malicious peers in the system at

any time. There are examples where such malicious peers acting alone or in concert have wreaked havoc on a p2p system [2]. For this reason, p2p systems which perform distributed computation need algorithms which work even in the face of malicious attack. In particular, these algorithms must be both scalable *and* attack-resistant.

Towards this end, in this paper, we consider the problem of designing a *scalable* solution to the Byzantine agreement problem [7, 4, 1, 8]. The Byzantine agreement problem is one of the most well studied problems in distributed computation and is a building block in many secure distributed protocols. Unfortunately, there are no results for solving this problem which require less than a linear number of messages per processor. In a p2p system, where the number of processors can easily be on the order of hundreds of thousands, this is unacceptable. In fact, for a p2p system to be scalable, we generally require all resource costs per peer to be polylogarithmic in the number of peers.

This paper gives a protocol which solves the synchronized Byzantine Agreement problem while requiring each processor to send only  $O(\log n)$  messages in expectation. To the best of our knowledge this is the first result for this problem which requires each processor to send  $o(n)$  messages. Our protocol uses randomness and is correct with high probability. It can tolerate any fraction of faulty processors which is strictly less than  $1/8$ .

The Byzantine agreement problem was introduced by Pease, Shostak and Lamport [7]. Fischer and Lynch [3] showed that in the synchronous communication model, any deterministic protocol requires a linear number of rounds to reach agreement in the worst case. Rabin [8] gave a simple randomized protocol for the synchronous communication model which requires only a constant number of rounds in expectation. However, Rabin's protocol sends  $O(n^2)$  total messages in expectation.

---

\*Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386; email: {cs1, saia}@cs.unm.edu. This research was partially supported by NSF grant CCR-0313160 and Sandia University Research Program, Grant No. 191445.

<sup>1</sup>In particular, the probability of failure is  $\frac{1}{n^c}$ , for some constant  $c > 0$ .

## 2 Scalable Byzantine Agreement

### 2.1 The Byzantine Agreement Problem

We now describe the Byzantine agreement problem. We start with  $n$  processors each having an initial binary input. Some fraction of these processors may be faulty. There are no restrictions on the actions of the faulty processors: they may fail to act, act inconsistently or act in collaboration. The correct processors are those which follow the protocol. The correct processors are trying to ensure that at the end of the protocol the following two properties hold:

1. All correct processors have the same output result.
2. If all correct processors have the same input,  $i$ , then they will all have  $i$  as an output.

### 2.2 The Scalable Byzantine Agreement Protocol

Our protocol, which we call Scalable Byzantine Agreement, is presented in Figure 1. This protocol is very similar to Rabin's randomized protocol [8], as presented in [6]. In our protocol, each processor first takes input from a small random sample of the processors:  $O(\log n)$  processors chosen uniformly at random. Then the value of the global coin flip is established and used to set a threshold for the round. Each processor, based on its sampled inputs, calculates its estimate of the number of correct processors which have the majority vote. The processor sets its vote according to whether or not this estimate exceeds the threshold chosen for that round. The protocol continues until all processors have crossed the highest threshold. The protocol sends  $O(n \log n)$  total messages in expectation and also has a constant expected number of rounds.

### 2.3 Analysis

Theorem 1 presents the key properties of our protocol. In our full technical report [5], we provide the proof of this theorem and give scalable procedures for implementing a random global coin flip and sampling uniformly at random from the set of processors.

**Theorem 1.** *The Scalable Byzantine Agreement protocol has the following properties:*

- *With high probability, the protocol is correct, i.e.:*

- *all correct processors will have the same output result*
- *if all correct processors have the same input,  $i$ , then they will all have  $i$  as an output.*

- *The expected number of rounds is a constant.*
- *The number of messages sent per round is  $O(n \log n)$ . Thus, the total expected number of messages sent is  $O(n \log n)$ .*

Input: A value  $b_i$ .

Output: A decision  $d_i$ .

1.  $\text{vote} \leftarrow b_i$ ;
2. For each round, do:
  - (a) Select  $C \log n$  processors uniformly at random from which to receive input, where  $C$  is a constant dependent on our fault-tolerance parameters;
  - (b) Request input from those processors;
  - (c) Send vote to the processors which request it;
  - (d)  $\text{maj} \leftarrow$  majority value (0 or 1) among votes received, including own vote.  $\text{tally} \leftarrow$  the number of occurrences of  $\text{maj}$  among the votes received
  - (e)  $M_i \leftarrow \text{tally} * \frac{n}{c \log n}$  ( $M_i$  is an estimate of the number of correct processors that have the majority vote);
  - (f) if  $\text{coin} = \text{heads}$  then  $\text{threshold} \leftarrow L$ ; else  $\text{threshold} \leftarrow H$ ;
  - (g) if  $M_i \geq \text{threshold}$  then  $\text{vote} \leftarrow \text{maj}$ ; else  $\text{vote} \leftarrow 0$ ;
  - (h) if  $M_i \geq G$  then set  $d_i$  to  $\text{maj}$  permanently;

Figure 1: Scalable Byzantine Agreement Protocol. The threshold values are  $G = \frac{9}{10}n - \alpha n$ ,  $H = \frac{8}{10}n - 4\alpha n$ ,  $L = \frac{7}{10}n - 7\alpha n$ , where  $\alpha = .01$ . The value  $\text{coin}$  is the result of a random global coin flip.

## References

- [1] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement

- protocols (extended abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 27–30, Montreal, Quebec, Canada, 17–19 August 1983.
- [2] John Borland. Gnutella girds against spam attacks. *CNET News.com*, August 2000. <http://news.cnet.com/news/0-1005-200-2489605.html>.
- [3] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, June 1982.
- [4] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.
- [5] Scott Lewis and Jared Saia. Scalable byzantine agreement. 2003.
- [6] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [7] M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980. This paper is similar to their 1982 publication, but contains a rigorous proof of the impossibility of Byzantine agreement for the case  $n = 3$ ,  $t = 1$ . As usual,  $n$  is the total number of processes and  $t$  is the number of faulty processes.
- [8] M. O. Rabin. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (FOCS '83)*, pages 403–409, Los Alamitos, Ca., USA, November 1982. IEEE Computer Society Press.
- [9] John Spooner and Ken Popovich. Intel: The future is peer. *ZDNet News*, August 2000. <http://zdnet.com.com/2100-11-523296.html?legacy=zdn>.
- [10] I. Stoica, D. Atkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proceedings of the ACM SIGCOMM 2002 Technical Conference*, 2002.
- [11] DataSynapse Website. <http://www.datasynapse.com>.
- [12] FOLDING@home Website. <http://folding.stanford.edu>.
- [13] Gnutella Website. <http://gnutella.wego.com/>.
- [14] Groove Networks Website. <http://www.groove.com>.
- [15] Kazaa Website. <http://www.kazaa.com>.
- [16] Morpheus Website. <http://gnutella.wego.com/>.
- [17] Napster Website. <http://www.napster.com/>.
- [18] SETI@home Website. <http://setiathome.ssl.berkeley.edu>.
- [19] Yaga Website. <http://www.yaga.com>.