

Byzantine Agreement with Faulty Majority using Bounded Broadcast

Jeffrey Considine, Leonid A. Levin*, David Metcalf
Boston University[†]

Abstract

Byzantine Agreement introduced in [Pease, Shostak, Lamport, 80] is a widely used building block of reliable distributed protocols. It simulates broadcast despite the presence of faulty parties within the network, traditionally using only private unicast links. Under such conditions, Byzantine Agreement requires more than $2/3$ of the parties to be compliant. [Fitzi, Maurer, 00], constructed a Byzantine Agreement protocol for any compliant majority based on an additional primitive allowing transmission to any two parties simultaneously. They proposed a problem of generalizing these results to wider channels and fewer compliant parties. We prove that a $\lfloor 2n/h \rfloor$ party channel is necessary and sufficient for implementing n -party broadcast with h compliant parties.

1 Introduction

Broadcast primitives play a special role in multi-player game theory as an integral component in the fault-tolerant implementation of game protocols. Given a compliant majority, broadcast and private channels are sufficient to simulate any multi-party computation [Rabin, Ben-Or, 89], based on [Goldreich, Micali, Wigderson, 87], and [Ben-Or, Goldwasser, Wigderson, 88]. With additional primitives, such as private and oblivious transfer channels, even a majority of faulty parties can be tolerated [Beaver, Goldwasser, 89], [Goldwasser, Levin. 90].

Since reliable hardware solution is a strong assumption, Byzantine Agreement protocols simulate broadcast on networks with faulty parties. Given only private channels, Byzantine Agreement is possible if and only if faulty parties are in a $< 1/3$ minority ([Pease, Shostak, Lamport, 80]). For this reason, protocols tolerant to more faults generally assume broadcast as a primitive.

Various hardware assumptions and communication goals were studied in the literature. For instance, [Angluin 80], [Goldreich, Goldwasser, Linial 91], [Franklin, Yung 95] and other papers studied problems of private communication on an incomplete broadcast network. [Franklin, Wright 98] showed that such a network with p disjoint paths from sender to receiver could tolerate $< p/2$ faulty parties. [Wang, Desmedt 01] showed that $< p$ faulty parties could be handled with probabilistic reliability.

The broadcast primitive is rather special in that, unlike other common primitives, it involves an unlimited number of parties. It is thus natural to explore the power of a limited version of broadcast, with a constant number of recipients. Assuming any compliant majority, [Fitzi, Maurer, 00] used a 3-party broadcast primitive to simulate full broadcast. They asked what fraction of compliant parties would be required given wider broadcast primitives. This is especially interesting in view of results (e.g. [2, 11]) that convert arbitrary protocols into equivalent ones with added tolerance to any faulty majorities, assuming the availability of broadcasts and two-party primitives, such as oblivious transfer and private channels. We show that a $\lfloor 2n/h \rfloor$ party channel is necessary and sufficient for implementing n -party broadcast with h compliant parties.

*Supported by NSF grants CCR 9820934, 0311411

[†]Leonid A. Levin (Lnd at bu.edu), Computer Sci. dept., 111 Cummington St., Boston, MA 02215.

2 Definitions and Results

Definition 1 A size k channel or k -channel is a primitive for authenticated reliable communication among k parties. To use it, (to k -send) one party, the sender, selects $k-1$ recipients, and a message m . Each recipient gets m , as well as the identities of the sender and the other recipients.

Definition 2 A protocol is an algorithm used in rounds by several communicating parties. Each party starts with an input appended with its and other parties' identities. We assume channels for all groups of k parties, who at each round can k -send messages to be used by the recipients as inputs for the next round. Besides the algorithm, the interaction is affected by the Adversary who selects the initial inputs of all parties and assigns, possibly with restrictions, their loyalties, i.e., chooses a subset of faulty parties and replaces their communications (inputs, messages, and outputs) by data of its choice.

Definition 3 Byzantine agreement is a broadcast simulating protocol. The party's value is its output for a recipient or input for the sender. The protocol succeeds if the values of non-faulty (compliant) parties are all identical.

Theorem 1 A $\lfloor 2n/h \rfloor$ party channel is necessary and sufficient for n -party Byzantine agreement with h compliant parties.

2.1 Broadcast and Consensus

In a traditional consensus model, each party starts with an input value. After running a consensus protocol, all compliant parties output values which agree with each other and with an input of at least one compliant party. With compliant majority, consensus is easily shown to be equivalent to broadcast. To achieve consensus, each party broadcasts its value to the others who then output the majority value. To broadcast, the caster sends its input to all parties who then run a consensus protocol on the values received.

This equivalence fails when majority is faulty. The Adversary gives inputs 0 and 1 to equal number of parties. They all run the protocol faithfully. The Adversary defeats the consensus by keeping compliant some parties with different outputs or declaring faulty all parties whose inputs match the uniform output.

One can generalize the consensus model, by assuming each party to have not just one input and output values but rather a *distribution*, i.e., a value for each $(k-1)$ -node set he belongs to. All compliant members of the set get the same input value for it. All output values of compliant parties must agree with each other and with at least one input of a compliant party. This model can simulate broadcast after the sender *distributes* his input, i.e., k -sends it to all $(k-1)$ -node sets.

3 Proof of the Lower Bound

3.1 Big Rings and Chains

The Adversary's ability to defy the agreement will depend on having enough parties to build a big ring. A (k, h) -ring is a set of more than k clusters of parties arranged in a cycle, with at least h parties in any two consecutive clusters. These bounds assure that no message can be sent to all clusters and all compliant parties can fit in any two adjacent clusters. Adding up parties in all pairs of consecutive clusters, we get $(k+1)h$ or more, counting each party twice. So, to build the ring, Adversary needs $n \geq (k+1)h/2$ parties which means $k < \lfloor 2n/h \rfloor$. We split the Sender's cluster S in two, S_0 and S_1 , duplicating all its nodes, thus opening the ring into a (k, h) -chain.

3.2 The Adversary Strategy

For $k < \lfloor 2n/h \rfloor$, the Adversary arranges the parties into a (k, h) -chain, splitting the sender's cluster S as in section 3.1. It picks two adjacent clusters corrupting all parties outside them.

Each transmission from S misses all parties in at least one other cluster. The leftmost such

cluster splits the chain into two parts: C_0 and C_1 , one which has no compliant parties.

All parties faithfully execute the protocol P except that the sender's cluster is split, its S_m copy gets input m , and the parties in C_m receive (or pretend to) messages from S as if sent by S_m .

With these restrictions on the adversary, no messages or outputs depend on its choice of compliant parties. Since the values of S_0 and S_1 copies of the sender differ, there must be parties in adjacent cluster that disagree on their value. The Adversary defeats the protocol by keeping the conflicting clusters compliant. ■

4 Proof of the Upper Bound

4.1 Trust Graphs

As a tool in visualizing and analyzing relationships between parties, we use *trust graphs* linking pairs of parties that report consistently inputs received by both and so may be both compliant.

Besides the list of n players, the protocol has two parameters: the upper bounds f, h on numbers of faulty and of compliant players; $h + f = N \geq n$. The discrepancy $d = N - n$ may be due to the exclusion of some parties from a recursive call of the protocol. The “sender clusters” S_m are added to the graph and connected to all recipients who report uniform inputs m . Each S_m is a clique of $1 + d$ nodes with identical connections and loyalties.

A *pruning* is then conducted as follows. The h compliant parties must form a clique; edges not in cliques can be removed. Since cliques are hard to detect, we remove instead (until none left) edges (a, b) that do not belong to any *bi-star* i.e., an h -node star with two centers a, b (adjacent to all its nodes).

Trust graphs are useful for choosing an agreement value and placing bounds on the broadcast size it requires. All compliant parties must be adjacent in the graph. If they know a unique sender cluster any of them may have paths to in their respective graphs, compliant recipients

may immediately output its value.

Consider a path connecting nodes in S_m with different m , say, $m = 0$ and $m = 1$. It must have at least k recipients. Otherwise there would be one k -send received by them all; since parties connected to S_m claim this k -send was m , there must be some disagreement along this path.

We break S_0, S_1 and the recipients into clusters according to the distance from S_0 , dropping nodes more distant than S_1 . They form a (k, h) -chain, since every two consecutive clusters include an h -node bi-star. So, by section 3.1, a trust graph with a path between S_0 and S_1 implies $k < \lfloor 2N/h \rfloor$.

4.2 The Protocol

We describe a Byzantine protocol $P_{h,f}$ for h compliant and f faulty parties. It can also be run by a subset of $n \leq h + f = N$ parties with the excluded parties sharing the sender's loyalty.

With this loyalty restriction, let n be minimal for which $P_{h,f}$ can fail. The sender s starts by distributing its input. Then each party i distributes the content M_i of all messages it received, and all parties except s attempt an agreement about M_i . They succeed if s shares loyalty with i (since n was assumed minimal for failure) or s is faulty (which only increases the fraction of compliant participants). Only if s is compliant and i faulty, may the results conflict.

Each recipient then forms a trust graph in which compliant parties always form a clique. If s is not among them, the graphs are identical. Then each party outputs m of S_m closest to it, or 0 if it has no path to either. The agreement can fail only if a path connects both S_m . As per Section 4.1, this would imply $k < \lfloor 2N/h \rfloor$. ■

References

- [STOC] *Proceedings of the Annual ACM Symposium on Theory of Computing*.
- [1] D. Angluin. Local and Global Properties in Networks of Processors. [STOC], 1980. pp. 82-93.
 - [2] D. Beaver, S. Goldwasser. Multi Party Fault Tolerant Computation with Faulty Majority Based on Oblivious Transfer. *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, 1989, pp. 468-473.
 - [3] M. Ben-Or, S. Goldwasser, A. Wigderson. Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. [STOC], 1988, pp. 1-10.
 - [4] Jeffrey Considine, Leonid A. Levin, David Metcalf. Byzantine Agreement with Bounded Broadcast. Preprint at <http://arXiv.org/abs/cs.DC/0012024>, 2000.
 - [5] M. Fischer, N. Lynch, M. Merritt. Easy Impossibility Proofs for Distributed Consensus Problems. *ACM Symposium on Distributed Computing*, 1985, pp. 59-70.
 - [6] Matthias Fitzi, Ueli Maurer. From Partial Consistency to Global Broadcast. [STOC], 2000, pp. 494-503.
 - [7] M. Franklin, R. Wright. Secure Communication in Minimal Connectivity Models. *Advances in Cryptology (Eurocrypt)*, 1998, pp. 346-360.
 - [8] M. Franklin, M. Yung. Secure Hypographs: Privacy from Partial Broadcast. [STOC], 1995, pp. 36-44.
 - [9] O. Goldreich, S. Goldwasser, N. Linial. Fault-tolerant Computation in the Full Information Model. *SIAM J. Computing*, 27(2), 1998, pp. 506-544.
 - [10] O. Goldreich, S. Micali, A. Wigderson. How to Play Any Mental Game, or A Completeness Theorem for Protocols with Honest Majority. [STOC], 1987, pp. 218-229.
 - [11] Shafi Goldwasser, Leonid A. Levin. Fair Computation of General Functions in Presence of Immoral Majority. *Annual CRYPTO Conference (IEEE/LACR)*, Santa Barbara, August 1990 (Proceedings 1991).
 - [12] Danny Dolev, Michael J. Fischer, Rob Fowler, Nancy A. Lynch, and H. Raymond Strong. An efficient algorithm for Byzantine agreement without authentication. *Information and Control*, 52(3):257-274, 1982.
 - [13] M. Pease, R. Shostak, L. Lamport. Reaching agreement in the presence of faults. *J. Comput. Mach.* 27:121-169, 1980.
 - [14] T. Rabin, M. Ben-Or. Verifiable Secret Sharing and Multi-party Protocols with Honest Majority. [STOC], 1989, pp. 73-85.
 - [15] Y. Wang, Y. Desmedt. Secure Communication in Multicast Channels: The Answer to Franklin and Wright's Question. *Advances in Cryptology (Eurocrypt)*, 1999, pp. 446-458.