Byzantine Broadcast Under a *Selective Broadcast* Model for Single-hop Wireless Networks*

Lewis Tseng Nitin Vaidya
University of Illinois at Urbana-Champaign
Email: {ltseng3, nhv}@illinois.edu

Abstract

This paper explores an old problem, *Byzantine fault-tolerant Broadcast* (BB), under a new model, *selective broadcast model*. The new model "interpolates" between the two traditional models in the literature. In particular, it allows fault-free nodes to exploit the benefits of a broadcast channel (a feature from reliable broadcast model) and allows faulty nodes to send mismatching messages to different neighbors (a feature from point-to-point model) simultaneously. The *selective broadcast* model is motivated by the potential for *directional* transmissions on a wireless channel.

We provide a collection of results for a single-hop wireless network under the new model. First, we present an algorithm for *Multi-Valued* BB that is order-optimal in bit complexity. Then, we provide an algorithm that is designed to achieve BB efficiently in terms of message complexity. Third, we determine some lower bounds on both bit and message complexities of BB problems in the *selective broadcast model*. Finally, we present a conjecture on an "exact" lower bound on the bit complexity of BB under the *selective broadcast* model.

1 Introduction

In this paper, we address Byzantine fault-tolerant broadcast (or *Byzantine Broadcast* for short) under a new model of a wireless broadcast channel. Byzantine Broadcast (BB) is a fundamental problem in distributed computing [9]. Consider a system of n nodes, namely $p_1, ..., p_n$, of which at most t nodes may be faulty. We assume that node p_1 is the *source* for the Byzantine Broadcast (BB); the remaining $n-1$ nodes will be referred as *peers*. Byzantine Broadcast must satisfy the following three properties, assuming that source p_1's input is x:

- **Termination**: every fault-free peer p_i eventually decides on an output value y_i.

*This research is supported in part by National Science Foundation award CNS 1059540. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

- **Consistency**: the output values of all the fault-free peers are equal, i.e., there exists y such that, for every fault-free peer p_i, $y_i = y$.
- **Validity**: if the source node p_1 is fault-free, then the agreed value must be identical to p_1's input value, i.e., $y = x$.

We assume that the input x at source p_1 is L bits long. The case when $L = 1$ will be referred to as *Binary* BB, with the case when $L > 1$ being referred to as *Multi-Valued* BB.

We are interested in two measures of complexity of BB algorithms.

- **Message complexity**: Message complexity of an algorithm is defined as the maximum (i.e., worst-case) number of messages transmitted by all the nodes following the specification of the algorithm over all permissible executions.
- **Bit complexity**: Bit complexity of an algorithm is defined as the maximum number of bits transmitted by all the nodes following the specification of the algorithm over all permissible executions.

The Byzantine Broadcast problem has been considered under different models of the communication network:

- **Point-to-point model** (e.g., [9, 3, 1, 10]): Nodes are connected via pairwise private channels, with the network being modeled as a directed graph. Thus, if a channel (p_i, p_j) exists then node p_i can transmit information to p_j – the channel is private in the sense that no other node can overhear this transmission.
- **Reliable Broadcast model** (e.g., [8, 2, 4]): There are two models assuming the existence of reliable broadcast channels. In both models, each node has a certain set of *neighbors* such that a message sent by the node on the broadcast channel is received (reliably) by all its neighbors. The first model corresponds to a radio network wherein nodes within a certain distance of each node are considered its neighbors (e.g., [8, 2]). [4, 7] considers a model wherein every subset of three nodes shares a reliable broadcast channel. Importantly, in both models, the broadcast property holds for transmissions from faulty nodes as well. Thus, each transmission of a (faulty or fault-free) node is received by all its neighbors.

In this paper, we consider a new model which "interpolates" between the above two models. The motivation behind the new model is to understand the impact of the network assumption on performance metrics of interest. Observe that:

- In the reliable broadcast model, performance can be improved by exploiting the broadcast channel, since a single transmission can be received by multiple nodes. However, this model is somewhat optimistic in the sense that it does not allow for the possibility that a faulty node may send different messages to different nodes. For instance, in a wireless setting, it is conceivable that a faulty node may use beam-forming (or directional) antennas to send different messages to its neighbors.
- While the point-to-point network model allows the faulty nodes to send different messages to different neighbors, it does not provide the benefits of broadcast to the fault-free nodes.

We now introduce our *selective broadcast* model. To simplify the discussion and analysis, we assume a "single-hop" broadcast channel, as elaborated in the model below. However, the model can be extended to "multi-hop" networks as well.

Selective Broadcast Model: The system is assumed to be synchronous. We assume that all the n nodes share a channel on which broadcasts and unicasts can both be performed. A fault-free node can broadcast its messages to all the nodes. However, a faulty node can simultaneously (in time) send different messages to different nodes to maximize the impact of its misbehavior.

Faulty nodes cannot cause collisions on the channel. All transmissions are assumed to be reliable. While multiple nodes may transmit messages on the shared channel within the same "round", for the purpose of the analysis of bit and message complexity, it is adequate to assume that these transmissions are performed in parallel. (In real networks, the transmissions will be serialized by a medium access mechanism – this detail is ignored in our work, since we focus on bit and message complexity, not time complexity).

It is assumed that each node can correctly identify the transmitter of each message on the *selective broadcast* channel.

Failure Model: The Byzantine adversary has complete knowledge of the algorithm, and the source's input value. The adversary can *compromise* up to $t < n/3$ nodes over the entire execution of the algorithm. These compromised nodes are said to be *faulty*. The faulty node can arbitrarily deviate from the algorithm specification, including sending mismatching messages to other nodes.

2 Upper Bound on Bit Complexity - Multi-Valued BB

In this section, we present a Byzantine Broadcast algorithm for an L-bit input. The algorithm is motivated by past algorithms that utilize "dispute control" or similar structures [6, 10]. For a suitably chosen integer D, the algorithm consists of L/D "generations", with Byzantine Broadcast of D bits of the input being achieved in each generation g ($1 \leq g \leq L/D$). For simplicity, we assume that L/D is an integer. Input x at source node p_1 is viewed as the tuple

$$x(1), x(2), \cdots, x(L/D),$$

where each $x(g)$ consists of D bits. In each generation g, each fault-free node p_i obtains output $y_i(g)$, and its L-bit output for all the generation together is obtained as the tuple

$$y_i(1), y_i(2), \cdots, y_i(L/D)$$

Our *Byzantine Broadcast* algorithm has three phases: *Detectable Broadcast*, *Detection Dissemination* and *Dispute Control*.

Byzantine Broadcast using Dispute Control

Perform the following steps in the g-th generation, $g = 1, \cdots, L/D$.

- **Detectable Broadcast of** $x(g)$: By the end of the *Detectable Broadcast* phase each fault-free node p_i receives a value, denoted as z_i, such that one of the following two conditions is true:

(i) at least one fault-free node detects misbehavior by some faulty node(s) in the network, without necessarily identifying the faulty node(s), or

(ii) no fault-free node detects any misbehavior, and for all fault-free peers p_i and p_j, we have $z_i = z_j$, and additionally, if p_1 is fault-free then $z_i = z_j = x(g)$.

- **Detection Dissemination:** Each node performs a Byzantine broadcast (BB) of a single bit, indicating whether it detected any misbehavior during *Detectable Broadcast* phase or not. The 1-bit BB is performed using any previously proposed BB algorithm (e.g., [9, 3, 1]) for the point-to-point model (note that algorithms designed for the point-to-point model can be used under our *selective broadcast* model as well).

  If no node announces that it detected misbehavior, then each fault-free node p_i sets output $y_i(g)$ equal to z_i received in the *Detectable Broadcast* phase, and the *Dispute Control* phase is not performed in the g-th generation.

- **Dispute Control:** If any node indicates, in the *Detection Dissemination* phase, that misbehavior has been detected, then additional steps are taken to learn new information regarding the potential identity of the faulty nodes. In particular, a new pair of nodes is found "in dispute" such that at least one node in this pair is guaranteed to be faulty. Two fault-free nodes are never found *in dispute* with each other. The *Dispute Control* phase is performed using a BB algorithm previously proposed for the point-to-point model (similar to the *Detection Dissemination* phase), and as a by-product of *Dispute Control*, Byzantine Broadcast of $x(g)$ is also achieved. For brevity, we omit the details of the *Dispute Control* phase; similar dispute control mechanisms have been included in prior work as well [6, 10].

  For future reference, note that, any node that is found *in dispute* with more than t other nodes must necessarily be faulty itself. Such a node is then essentially excluded from the future generations of the algorithm. If the source node p_1 is thus identified as faulty, then the algorithm terminates with the nodes agreeing on a default value for all future generations.

As shown in prior work using *Dispute Control*, the bit complexity of such algorithms is dominated by the first phase – *Detectable Broadcast* in our case – when the input size L is sufficiently large. For brevity, we will omit the proof of this claim; the reader is referred to prior work (e.g., [10]) for examples of similar algorithms. There are two key reasons for this outcome:

- Although the 1-bit BB in the *Detection Dissemination* phase is performed using an expensive (existing) algorithm, the cost of these disseminations is amortized over a large number of bits, specifically D, in the input for each generation.
- Similarly, the *Dispute Control* phase is also expensive. However, it turns out that this phase is performed only a finite number of times (specifically, at most $t(t+1)$ times). Thus, its amortized cost over a large number of generations (i.e., large L/D) is small.

In the rest of this section, we present and analyze a *Detectable Broadcast* algorithm for D bits.

2.1 D-Bit Detectable Broadcast

The *Detectable Broadcast* phase makes use of an error detection code. Therefore, we first describe the code and its parameters.

Error Detection Code: With a suitable choice of parameter c, we will use a $(n, n-2t)$ Reed-Solomon code over Galois Field GF(2^c). In particular, c is chosen large enough such that $n \leq 2^c - 1$. The D-bit value to be agreed on in each generation is viewed as consisting of $n-2t$ *data* symbols from $GF(2^c)$. Thus, each of these symbols can be represented with c bits, and therefore, $D = c(n-2t)$. Given the $(n-2t)$ data symbols corresponding to a certain D-bit value, n "coded" symbols in the corresponding codeword are obtained as linear independent combinations of the $(n-2t)$ data symbols over $GF(2^c)$. The code specification is part of the specification of the *Detectable Broadcast* algorithm. The $(n, n-2t)$ Reed-Solomon code has the following useful property: Any $n-2t$ (coded) symbols in a codeword can be used to compute the corresponding $n-2t$ data symbols, and therefore, the corresponding D-bit value. We summarize the relationships between the code parameters:

- n coded symbols in each codeword, corresponding to $(n-2t)$ data symbols,
- $n \leq 2^c - 1$, and $D = c(n-2t)$

This implies that $n \leq 2^{D/(n-2t)} - 1$, and $D \geq (n-2t)\log_2(n+1)$. Thus, we need $D = \Omega(n \log n)$.

Detectable Broadcast: Algorithm 1 below specifies execution of *Detectable Broadcast* in the g-th generation of the *Byzantine Broadcast* algorithm described above. Recall that $x(g)$ is the input for source p_1 in the g-th generation.

Algorithm 1: Detectable Broadcast for the g-th Generation

1. Source p_1 transmits $x(g)$ on the *selective broadcast* channel. $x(g)$ is viewed as a vector of $n-2t$ *data* symbols, each consisting of $c = D/(n-2t)$ bits each.
2. Peer p_i $(2 \leq i \leq n)$ performs the following steps:
   (a) If p_i is in dispute with the source, stay silent.
   Else, encode the $n-2t$ data symbols received in step 1 into a codeword consisting of n *coded* symbols using the aforementioned coding scheme. Denote the resulting codeword obtained at peer i as $s_i = s_i[1], s_i[2], \cdots, s_i[n]$. Transmit $s_i[i]$ on the *selective broadcast* channel.
   (b) Node p_i ignores symbols received from peers that it has been previously *in dispute* with. For each peer p_j that node p_i is *not* in dispute with: let the symbol received from peer p_j be denoted as $r_i[j]$. Also, if p_i is *not in dispute* with p_1, then let $r_i[1] = s_i[1]$. For all nodes p_k such that p_i is in dispute with p_k $(1 \leq k \leq n)$, define $r_i[k] = \perp$ ($\perp$ denotes a distinguished *null* symbol).

   At the end of step 2b, every peer p_i has one non-null symbol $r_i[j]$ corresponding to each node p_j $(1 \leq j \leq n)$ that has not yet been dispute with either p_i or p_1. Since p_1 and p_i can each be in dispute with at most t nodes (otherwise they would have already been identified as faulty), node p_i has at least $n-2t$ non-null symbols in the r_i vector.
3. Peer p_i $(2 \leq i \leq n)$ performs the following steps:
   (a) Find the solution for each subset of $n-2t$ non-null coded symbols in the r_i vector received above in step 2b – the solution consists of $n-2t$ data symbols that correspond to the $n-2t$ coded symbols.

(b) If the solutions to all these subsets of size $n-2t$ is not unique, then p_i has detected faulty behavior by some peer. In this case, z_i is set equal to some default value.

Else, z_i is set equal to the unique solution corresponding to any of the $n-2t$ non-null symbols received in step 2b. The solution consists of $n-2t$ symbols, which correspond to $c(n-2t) = D$ bits.

The bit complexity of *Detectable Broadcast* algorithm can be further reduced by using a more efficient coding scheme. For lack of space, we omit the discussion here.

Correctness of Detectable Broadcast Algorithm

Lemma 1 *By the end of Detectable Broadcast in g-th generation, the following conditions hold:*

- *either at least one fault-free node detects misbehavior by some faulty nodes,*
- *or for all fault-free peers p_i and p_j, $z_i = z_j$, and additionally, if p_1 is fault-free then $z_i = z_j = x(g)$.*

Proof: In the *Detectable Broadcast* algorithm, the following misbehaviors are possible: Source p_1 may misbehave in step 1 by transmitting different D-bit values (represented as $n-2t$ symbols) to at least two different fault-free peers that are both not in dispute with the source. A peer node may misbehave by transmitting incorrect symbols to some fault-free peer(s) in step 2b.

Consider two cases in the g-th generation:

- *Source does not misbehave:* : Since there are at least $n-t$ fault-free nodes, and fault-free nodes are never in dispute with each other, it is easy to see that, for each pair of fault-free peers p_i and p_j, the vectors r_i and r_j will include at least $n-t$ identical and correct coded symbols corresponding to the data symbols sent by p_1 in step 1. Then it should be easy to see that for each pair of fault-free peers p_i and p_j, either (i) at least one of them will detect misbehavior by some node, or (ii) $z_i = z_j = x(g)$.
- *Source node misbehaves:* In this case, the source node is in dispute with at most t nodes (otherwise, it would have been identified as faulty already). Thus, the source node is not in dispute with at least $(n-1)-t$ peers. Of these, $n-1-t$ peers, at least $(n-1-t)-(t-1) = n-2t$ peers are fault-free. Since these fault-free peers cannot in dispute with each other, they will collectively transmit at least identical $n-2t$ symbols to all the nodes. Therefore, all the fault-free peers will share at least $n-2t$ symbols in common in their r vectors. Then it should be easy to see that for each pair of fault-free peers p_i and p_j, either (i) at least one of them will detect misbehavior by some node, or (ii) $z_i = z_j$.

□

Lemma 1 together with the correctness of *Detection Dissemination* and *Dispute Control* (similar to the proofs in [10]) proves the correctness of the algorithm presented in this section.

2.2 Bit Complexity

Source node p_1 transmits D bits in step 1. In step 2a, at most $n-1$ peers each transmit a coded symbol consisting of $D/(n-2t)$ bits, for a total cost of $(n-1)D/(n-2t)$ bits in step 2a. Thus, the

worst-case cost of a single instance of *Detectable Broadcast* (in bits) is

$$D + \frac{(n-1)D}{n-2t}$$

Thus, the total cost of *Detectable Broadcast* over all the L/D generations required to perform Byzantine Broadcast of the L-bit input at node p_1 is given by

$$L + \frac{(n-1)L}{n-2t} \;=\; L\,\frac{2n-2t-1}{n-2t}$$

As noted previously, with the *dispute control* framework, when L is large, the bit complexity of Byzantine Broadcast of L-bits is dominated by the bit complexity of *Detectable Broadcast*. In particular, it can be shown that if we use the Modular Algorithm [3] to perform 1-bit BB in *Detection Dissemination* and *Dispute Control* phase, then the communication cost (in bits) of the proposed *Byzantine Broadcast* algorithm is

$$L\,\frac{2n-2t-1}{n-2t} \;+\; O(n^4)L^{0.5}$$

When L is large, the first term dominates the above cost, and the bit complexity becomes $O(L)$. By assumption, $n \geq 3t+1$, and therefore,

$$2 < \frac{2n-2t-1}{n-2t} < 4$$

3 Upper Bound on Message Complexity

In this section, we briefly describe a Byzantine Broadcast algorithm, named Algorithm 2, with message complexity $O(t \log t)$ under the *selective broadcast* model. This algorithm is derived from the *Modular Algorithm* proposed by Coan and Welch [3], which achieves optimal message complexity of $O(n^2)$ in the point-to-point model. Algorithm 2 has the same structure as the *Modular Algorithm* except for the following modifications:

- In the *Modular Algorithm*, a node may often send identical messages to multiple neighbors. In our algorithm, such transmissions are replaced by **one** broadcast message over the *selective broadcast* channel.
- The *Modular Algorithm* is performed by all the n nodes. In our case, we use a modified version of that algorithm to achieve Byzantine Broadcast among $3t+1$ nodes. After that, any $2t+1$ of these $3t+1$ nodes transmit the agreed value on the *selective broadcast* channel; the remaining $n-3t-1$ nodes agree on a majority vote of these transmissions.
- The number of levels of recursion in our algorithm is $\log n$, different from that in the original *Modular Algorithm* [3].

Now, we briefly present the framework from [3] and its variation for the *selective broadcast* model.

3.1 Modular Framework [3]

The framework proposed by Coan and Welch [3] consists of recursive applications of two transformations, BC2BCB and BCB2BC: the first transformation, namely BC2BCB, uses a Byzantine Consensus (BC) algorithm to solve a Byzantine Committee Broadcast (BCB) problem, and the latter transformation, namely BCB2BC, uses a BCB algorithm to solve the BC problem.

The base case of the Modular Framework is a previously proposed BC algorithm (e.g., [9, 1]), say A_0. Then, the recursive definition of the Modular Framework is as follows: Given algorithm A_{i-1} ($i \geq 1$), A_i is defined as BCB2BC(BC2BCB(A_{i-1})). For brevity, we omit the details of the framework; the reader is referred to the prior work [3]. The main reason that the Modular Framework achieves low message complexity is as that the number of message transmissions induced by the expensive base algorithm is small. This is achieved by recursively dividing the nodes into many "committees" of small size such that the base algorithm is only executed within each of these small committees.

Algorithm 2: Modular Algorithm [3] is designed for the Byzantine Consensus (BC) problems, wherein each node has an input. To perform Byzantine Broadcast (BB) using a modified Modular algorithm, the source node first broadcasts its input value on the *selective broadcast* channel, and then a modified Modular algorithm is used to reach consensus on the value received by all the peers from the source . Now, we describe two modifications of Modular Algorithm to achieve BC efficiently in the *selective broadcast* model.

In the Modular Algorithm, when fault-free nodes transmit, they always transmit the same message to all the intended receivers. Thus, replacing these transmissions by a **single** transmission in *selective broadcast* channel reduces the message complexity.

The second modification further exploits the reliable broadcast channel. Unlike Modular Algorithm, Algorithm 2 only requires $n' = 3t + 1$ *active* nodes participating in the algorithm, i.e., executing Algorithm 2, due to the existence of *selective broadcast* channel. The remaining $n - 3t - 1$ nodes (which we call *passive* nodes) do not transmit messages at all, but listen to the messages announcing the agreed value, transmitted by any $2t + 1$ *active* nodes. The passive nodes then use majority voting on these $2t + 1$ values to decide on their output.

3.2 Message Complexity

Suppose that $M_*(n')$ is the message complexity of the base algorithm executed by n' nodes. For any fixed integer B such that $t + 1 \geq B \geq 2$, by using analysis similar to that in [3], we can show the upper bound below on message complexity of A_i, denoted as M_i, for all $\log_B t \geq i \geq 0$. In the inequality below, α is a certain constant that depends on the transformation used in the Modular Algorithm.

$$M_i(3t + 1) \leq B^i M_*(3t/B^i + 1) + \alpha B t i \,, \tag{1}$$

Replacing $i = \log_B t$ and $M_*(n') = O((n')^3)$ in equation 1 yields $M_{\log_B t}(3t + 1) \leq O(t \log t)$. [1]

The message complexity of Algorithm 2 is $M_{\log_B t}(3t + 1) + 2t + 1$, and is thus bounded by $O(t \log t)$.

[1] Here, we use the Gradecast-based algorithm [1] as the base algorithm.

4 Lower Bounds

In this section, we state some simple lower bounds on bit and message complexity. In deriving these bounds, we assume that the nodes only communicate explicitly through messages. That is, no implicit communication mechanism is used to convey information, such as the time between two message transmissions.

Lower Bound on Message Complexity:

Theorem 1 *The lower bound on message complexity of Byzantine Broadcast under the* selective broadcast *model is* $\Omega(t)$.

Proof: The proof is by contradiction. Assume that there exists a correct algorithm A under the *selective broadcast* model that has message complexity of $o(t)$. The transmission of a message by a certain node on the *selective broadcast* channel can be simulated by sending a copy of the message to each of the remaining $n-1$ nodes in a fully connected point-to-point network. Thus, algorithm A can solve the Byzantine Broadcast problem using $o(nt)$ messages in a point-to-point model. This contradicts the lower bound in [5].

Alternatively, the theorem can be proved by arguing that more than t messages must be sent in the worst-case: if only t (or fewer) messages are sent, it is possible that the transmitters (at most t) of all these messages are faulty nodes, making it impossible to guarantee agreement among the fault-free nodes. □

The gap between the above lower bound of $\Omega(t)$ and the upper bound of $O(t \log t)$ in Section 3 is small, but the problem of closing this gap remains open.

Lower Bound on Bit Complexity: It should be obvious that the total number of bits transmitted in any BB algorithm is at least L (in the worst case), since there are 2^L possible input values. Thus, $\Omega(L)$ is a trivial lower bound on bit complexity, and our algorithm presented in Section 2 matches this bound when L is large enough.

Lower Bound on Bit Complexity of "Static" Algorithms for Detectable Broadcast: An algorithm A is characterized by a set of *schedules* where each schedule consists of a sequence of transmission slots. In each slot i, a single node is selected as the *transmitter*, denoted as T_i. The transmitter T_i transmits a message via the *selective broadcast* channel to all the other nodes. An algorithm is said to be **static** if it has a fixed schedule, such that the transmitters in all the slots are pre-determined and are independent of the source's input and the behavior of the faulty nodes.

We state the following lower bound on bit complexity of *static* algorithms for *Detectable Broadcast* (DB). For lack of space, we only present the case when no two nodes are in dispute, and the proof is omitted. The proof argues that in a static schedule, the source p_1 must transmit at least L bits, and then argue that the remaining nodes must transmit at least $(n-1)\frac{L}{n-f}$ bits to satisfy the conditions of *Detectable Broadcast*.

Claim 1 *The lower bound on bit complexity of static DB algorithm is* [2]

[2]Note that this lower bound is also a lower bound of static BB algorithms, since any BB algorithm also solves DB problems. Note that algorithms in [9, 3, 1] are static BB algorithms, but the algorithm presented in Section 2 is not static.

$$L + (n-1)\frac{L}{n-f}$$

5 Summary

This paper introduces a new communication model that "interpolates" between two old models in the literature. In particular, we explore the impact of allowing nodes to *select* between broadcast and unicast on bit and message complexity. In the new model, we present a Multi-Valued BB algorithm that is order-optimal in bit complexity, and another BB algorithm that is efficient in message complexity. At last, we briefly discuss about lower bounds on bit and message complexity in the new model.

References

[1] M. Ben-Or, D. Dolev, and E. N. Hoch. Brief announcement: Simple gradecast based algorithms. In *DISC*, pages 194–197, 2010.

[2] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, PODC '05, pages 138–147, New York, NY, USA, 2005. ACM.

[3] B. A. Coan and J. L. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61 – 85, 1992.

[4] J. Considine, M. Fitzi, M. K. Franklin, L. A. Levin, U. M. Maurer, and D. Metcalf. Byzantine agreement given partial broadcast. *J. Cryptology*, 18(3):191–217, 2005.

[5] D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, Jan. 1985.

[6] M. Fitzi and M. Hirt. Optimally efficient multi-valued byzantine agreement. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, PODC '06, pages 163–168, New York, NY, USA, 2006. ACM.

[7] A. Jaffe, T. Moscibroda, and S. Sen. The price of equivocation: Characterizing byzantine agreement via hypergraph coloring. *to appear at PODC 2012*.

[8] C.-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, PODC '04, pages 275–282, New York, NY, USA, 2004. ACM.

[9] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 1982.

[10] G. Liang and N. Vaidya. Error-free multi-valued consensus with byzantine failures. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '11, pages 11–20, New York, NY, USA, 2011. ACM.