Broadcast and Byzantine Agreement

Jonathan Katz[*]

March 7, 2021

This note defines broadcast (and the related problem of Byzantine agreement), and presents the fundamental feasibility and infeasibility results for constructing broadcast protocols among a set of n parties in a synchronous network:

- With no prior setup, broadcast is possible when $t < n/3$ parties are corrupted.
- With no prior setup, broadcast is impossible when $t \geq n/3$ parties are corrupted.
- If the parties have a pre-existing public-key infrastructure (PKI), then broadcast is possible when $t < n$ parties are corrupted.

Corrupted parties are malicious, and may behave arbitrarily. In all cases we assume a fully connected network with authenticated channels between each pair of parties. We do not assume private channels for the positive results; the impossibility result holds even with private channels. The positive results allow the adversary to adaptively corrupt parties (though in this case we caution the reader that one can consider simulation-based definitions of security that are stronger than the ones considered here [4, 3]); the impossibility result holds even for static corruptions.

1 Definitions

Protocols for *broadcast* allow a designated sender to transmit a message to all parties, such that all parties are assured they receive the same message.

Definition 1 (Broadcast.) Let Π be a protocol executed by parties $P_1, \ldots, P_n$, where a sender $P^* \in \{P_1, \ldots, P_n\}$ begins holding input $m \in \{0,1\}^*$ and all parties are guaranteed to terminate. We say Π is t-secure if the following hold:

Validity: if P^* is honest and at most t parties are corrupted, all honest parties output m.

Consistency: if at most t parties are corrupted, all honest parties output the same value. ■

A related task is *Byzantine agreement* (BA). Here, all parties begin holding input, and the goal is for the parties to reach agreement on their output.

Definition 2 (Byzantine agreement.) Let Π be a protocol executed by parties $P_1, \ldots, P_n$, where each party P_i begins holding input $v_i \in \{0,1\}^*$ and all parties are guaranteed to terminate. We say Π is t-secure if the following hold:

[*] `jkatz2@gmail.com`. Department of Computer Science, University of Maryland.

Validity: if at most t of the parties are corrupted and every honest party's input is equal to the same value v, then every honest party outputs v.

Consistency: if at most t parties are corrupted, there is a $v \in \{0,1\}^*$ such that every honest party outputs v. ■

Note that BA only makes sense if $t < n/2$ (why?). On the other hand, broadcast is well-defined for any $t < n$. When $t < n/2$, a protocol for broadcast immediately implies a protocol for Byzantine agreement: each party broadcasts its input value (so the parties run n broadcast protocols overall), and then parties output the majority value. Conversely, a protocol for Byzantine agreement immediately implies a protocol for broadcast: the sender P^* sends its input m to each party, and then the parties run a Byzantine agreement protocol where each party uses as input the value it received from P^*.

2 Feasibility of Broadcast without Setup

Assume $t < n/3$. Berman, Garay, and Perry [1] give a t-secure BA protocol for single-bit inputs with polynomial complexity and optimal resilience. (This implies a t-secure broadcast protocol, as noted earlier.) The protocol involves running the following *phase-king* subroutine $t+1$ times, with parties $P_1, \ldots, P_{t+1}$ successively playing the role of the king.

Round 1 Each party P_i sends its input v_i to all other parties.

P_i sets $C_i^b := 1$ (for $b \in \{0,1\}$) if at least $n-t$ parties sent it the bit b, and $C_i^b := 0$ otherwise.

Round 2 Each party P_i sends C_i^0 and C_i^1 to all other parties. Let $C_{i \to j}^b$ denote the relevant value received by P_j from P_i.

Each party P_i sets $D_i^b := \left|\left\{j : C_{j \to i}^b = 1\right\}\right|$. If $D_i^1 > t$, it sets $v_i := 1$; otherwise, it sets $v_i := 0$.

Round 3 The king P_k sends v_k to all parties. Each party P_i then updates their input as follows: If $D_i^{v_i} < n-t$ then set v_i equal to the value the king sent to P_i; otherwise, leave v_i unchanged.

We begin with two lemmas about the phase king (sub-)protocol.

Lemma 1 *Let $t < n/2$, and assume all $n-t$ honest parties begin the phase-king protocol holding the same input v. Then all honest parties terminate the protocol with the same output v.*

Proof Since all honest parties begin with input v, in the first round each honest party receives v from at least $n-t$ parties, and receives $\bar{v}$ from at most $t < n-t$ parties. So each honest P_i sets $C_i^v := 1$ and $C_i^{\bar{v}} := 0$. It follows that in round 2, each honest P_i has $D_i^v \geq n-t > t$ and $D_i^{\bar{v}} \leq t$, and so $v_i = v$ at the end of that round. Since $D_i^{v_i} = D_i^v \geq n-t$ for an honest P_i, all honest parties ignore the value sent by the king and terminate the phase-king protocol with output v. ■

Lemma 2 *Let $t < n/3$. If the king is honest in an execution of the phase-king protocol, then the outputs of all honest parties agree at the end of that protocol.*

Proof An honest king sends the same value v_k to all parties. So the only way agreement can possibly fail to hold is if some honest party P_i does not set their input to the king's value, i.e., if $D_i^{v_i} \geq n - t$. We claim that if there exists an honest party P_i for which $D_i^{v_i} \geq n - t$, then $v_i = v_k$ and so agreement holds anyway. To see this, consider the two possibilities:

Case 1: $v_i = 1$. Since $D_i^1 \geq n - t$ we have $D_k^1 \geq n - 2t > t$, and so $v_k = 1$ as well.

Case 2: $v_i = 0$. The fact that $D_i^0 \geq n - t$ implies $D_k^0 \geq n - 2t > t$. So at least one honest party P_j sent $C_{j \to k}^0 = 1$ to P_k, implying that at least $n - t$ parties sent '0' to P_j in round 1 and consequently at most t parties sent '1' to P_j in round 1. But then any honest party received a '1' from at most $2t < n - t$ parties in round 1, and so any honest party P_i has $C_i^1 = 0$. It follows that each honest party P_i, and P_k in particular, has $D_i^1 \leq t$; we conclude that $v_k = 0$ as desired. ■

Theorem 1 *The above protocol achieves Byzantine agreement for any $t < n/3$.*

Proof Say all honest parties begin holding the same input. Then Lemma 1 implies that none of the honest parties ever changes its input value in any of the phase-king subroutines, and so in particular all honest parties terminate with the same output.

In any other case, there must be at least one execution of the phase-king subroutine in which the king is honest. Following that execution, Lemma 2 guarantees that all honest parties hold the same input. Lemma 1 ensures that this will not change throughout the rest of the protocol. ■

The above protocol only allows the parties to agree on a single bit, however it can be extended to allow agreement on an ℓ-bit string by simply repeating the protocol (in parallel) ℓ times. In fact, there is a more efficient conversion of binary BA to multi-valued BA that involves running the binary BA protocol only once; we refer to the work of Turpin and Coan [7] for details.

3 Impossibility of Broadcast without Setup

We show the following classical result [6, 5]:

Theorem 2 *If $n \geq 3$ and $t \geq n/3$ and the parties have no initial setup, there is no t-secure protocol for broadcast (or BA).*

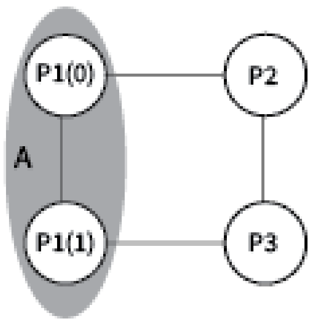

Figure 1: Imaginary execution for the proof.

Proof We focus on the case $n = 3$, $t = 1$; the general case can be proved by reduction to that case. Assume a 3-party broadcast protocol Π that is 1-secure, where P_1 is the sender. Π is defined by three algorithms P_1, P_2, P_3 run by each of the parties, where in each round of the protocol a party decides what messages to send (or whether to produce output and terminate) based on its current state and the messages it received in the previous round.

Imagine connecting these algorithms as shown in Figure 1 and then letting them run the protocol. That is, we run P_2 and P_3 as well as *two* instances of P_1—one with initial input 0 and the other with initial input 1. (For the moment, ignore the shaded oval labeled 'A.') It is not immediately clear what the result will be of connecting the parties in this way; nevertheless, P_2 and P_3 will eventually terminate and output *something*. Let v_2, v_3 denote the outputs of P_2, P_3, respectively, in some execution.

We first claim that $v_2 = v_3$. This is because we can consider a real-world execution of Π among three parties in which an adversary $\mathcal{A}$ corrupts the sender (P_1) and simulates the imaginary execution in Figure 1. In detail: $\mathcal{A}$ runs two instances of P_1 on inputs 0 and 1. When $\mathcal{A}$ receives a message from P_2 it feeds it to $P_1(0)$; when $P_1(0)$ outputs a message to be sent to P_2 then $\mathcal{A}$ sends that message to P_2. Similarly, when $\mathcal{A}$ receives a message from P_3 it feeds it to $P_1(1)$; when $P_1(1)$ outputs a message to be sent to P_3 then $\mathcal{A}$ sends that message to P_3. When $P_1(0)$ outputs a message to be sent to P_3 then $\mathcal{A}$ sends it to $P_1(1)$ as if it came from P_2; similarly, when $P_1(1)$ outputs a message to be sent to P_2 then $\mathcal{A}$ sends it to $P_1(0)$ as if it came from P_3. The honest P_2, P_3 cannot distinguish whether there are running the imaginary execution from Figure 1 or a real execution of Π with the adversary described, and hence there is an execution in which they output v_2, v_3. But consistency of Π implies that $v_2 = v_3$.

We can now repeat the above argument but with an adversary corrupting P_3 in a real-world execution of Π and simulating the two parties on the bottom half of Figure 1. Since the real sender is honest in this real-world execution, validity of Π implies that P_2's output must be equal to P_1's input, and hence $v_2 = 0$. Reasoning analogously (but considering an adversary corrupting P_2 and simulating the two parties on the top half of Figure 1), we must also have $v_3 = 1$. (You should convince yourself that such adversarial behavior could actually be implemented.) But then $v_2 \neq v_3$, a contradiction. ■

4 "Authenticated" Broadcast

We now show a protocol for broadcast that is secure even if all-but-one of the parties are corrupted. This does not contradict the impossibility result from the previous section because here we are going to assume some prior setup among the parties in the form of a public-key infrastructure (PKI). In particular we assume that each party P_i has generated a public/private key pair (pk_i, sk_i) and that all parties hold the (same) vector of public keys $(pk_1, \ldots, pk_n)$. (We stress that *every* party holds the same pk_i for player i—this is crucial.) We will also assume a computationally-bounded adversary who cannot forge a signature (with non-negligible probability) on any previously-unsigned message with respect to the public key of any of the honest parties. We denote signatures by σ, and assume for simplicity that a signature includes the identity of the signer.

It is worthwhile to pause and think about where the impossibility proof from the previous section breaks down in this case. A crucial element in the proof is the ability of corrupted players to simulate the actions of non-corrupted players (for example, we required that a corrupted P_2 could simulate the actions of P_1). But when the players sign messages (and we assume a computationally-bounded adversary) then *it is no longer necessarily possible for corrupted parties to simulate the actions of honest parties* and the proof breaks down.

We now describe the Dolev-Strong protocol [2] for multi-bit messages. Assume P_1 is the sender. We say a message received by a party P in round i is (m,i)-*valid* if it consists of the message m along with valid signatures on m by P_1 and $i-1$ other parties (not including P itself); we say it is m-*valid* if it is (m,i)-valid for some $i \geq 1$. Each party P_j initializes $S_j = \emptyset$, and then the protocol proceeds as follows:

Round 1: P_1 generates a signature σ_1 on its input m, sets $S_1 := \{m\}$, and sends (m, σ_1) to all parties.

Rounds 2 through $n-1$: Each party P_j acts as follows in round i:

- If $|S_j| < 2$: If P_j received an $(m, i-1)$-valid message $(m, \sigma_1, \ldots, \sigma_{i-1})$ in the previous round with $m \notin S_j$, it generates a signature σ on m, sends $(m, \sigma_1, \ldots, \sigma_{i-1}, \sigma)$ to all other parties, and adds m to S_j.

  P_j does the above for at most $2 - |S_j|$ values of m.
- If $|S_j| = 2$ then do nothing.

Conclusion: If party P_j received a $(m, n-1)$-valid message in round $n-1$, it adds m to S_j.

P_j then decides on its output as follows: If $|S_j| \neq 1$, output 0. Otherwise, let $S_j = \{m\}$ and output m.

We claim that the above is an $(n-1)$-secure broadcast protocol. Validity is easy to see: if the sender is honest and has input m, then every party receives an $(m,1)$-authentic message in the first round; by security of the signature scheme, parties will never receive an (m', i)-authentic message with $m' \neq m$ for any i and so all honest parties output m.

The key observation to prove consistency is the following lemma:

Lemma 3 *If an honest P_j ever adds m to S_j, then every party P receives an m-valid message.*

Proof If P_j adds m to S_j in some round $i \leq n-1$ this is immediate. (Note there are two cases depending on whether P's signature is already part of the (m,i)-valid message P_j received.) On the other hand, if P_j adds m to S_j at the conclusion of the protocol because it received an $(m, n-1)$-valid message in the final round, then (by security of the signature scheme) P must have signed m at some point, which means that P must have received an m-valid message. ■

Using the previous lemma it is possible to prove consistency as follows: Fix honest P_i, P_j. If $|S_i| \geq 2$ then $|S_j| \geq 2$, and so P_i, P_j both output 0. A similar argument applies if $|S_i| = 0$. Finally, if $|S_i| = 1$ then $|S_j| = 1$ and $S_i = S_j$, so P_i, P_j agree on their output.

References

[1] P. Berman, J. Garay, and K. Perry. Bit-optimal distributed consensus. In *Computer Science Research*, pp. 313–322, Plenum Publishing Corporation, 1992.

[2] D. Dolev and H.R. Strong. Authenticated algorithms for Byzantine agreement. *SIAM J. Computing* 12(4):656–666, 1983.

[3] J. Garay, J. Katz, R. Kumaresan, and H.-S. Zhou. Adaptively secure broadcast, revisited. PODC 2011.

[4] M. Hirt and V. Zikas. Adaptively secure broadcast. Eurocrypt 2010.

[5] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3): 382–401 (1982).

[6] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2): 228–234 (1980).

[7] R. Turpin and B. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Proc. Lett.* 18(2):73–76, 1984.