

Validity in Network-Agnostic Byzantine Agreement

Andrei Constantinescu, Marc Dufay, Diana Ghinea, and Roger Wattenhofer

ETH Zürich

{aconstantine, mdufay, ghinead, wattenhofer}@ethz.ch

Abstract

In Byzantine Agreement (BA), there is a set of n parties, from which up to t can act byzantine. All honest parties must eventually decide on a common value (agreement), which must belong to a set determined by the inputs (validity). Depending on the use case, this set can grow or shrink, leading to various possible desiderata collectively known as validity conditions. Varying the validity property requirement can affect the regime under which BA is solvable. We study how the selected validity property impacts BA solvability in the network-agnostic model, where the network can either be synchronous with up to t_s byzantine parties or asynchronous with up to $t_a \leq t_s$ byzantine parties.

We show that for any non-trivial validity property the condition $2t_s + t_a < n$ is necessary for BA to be solvable, even with cryptographic setup. Noteworthy, specializing this claim to $t_a = 0$ gives that $t < n/2$ is required when one expects a purely synchronous protocol to also work in asynchrony when there are no corruptions. This is especially surprising given that for some validity properties $t < n$ are known to be achievable without the last stipulation. Thereafter, we give necessary and sufficient conditions for a validity property to render BA solvable, both for the case with cryptographic setup and for the one without. This traces the precise boundary of solvability in the network-agnostic model for every validity property. Our proof of sufficiency provides a universal protocol, that achieves BA for a given validity property whenever the provided conditions are satisfied.

1 Introduction

Achieving agreement among the parties involved in a distributed system is crucial for maintaining consistent views. This becomes particularly challenging due to the potential for parties' failures, which can range from benign crashes to malicious (byzantine) behavior. Byzantine Agreement (BA) is an extensively studied problem in distributed computing that tackles this challenge. It seeks to establish a common value amongst a set of n parties even when up to t parties exhibit byzantine behavior. A crucial aspect of BA lies in its validity condition, which requires that the value agreed upon reflects the honest parties' proposals rather than being a default or arbitrary value. The traditional definition of BA considers the so-called *strong validity*: if all honest parties propose the same value v , the agreed-upon output must be v . While strong validity is a powerful guarantee in applications concerning binary decisions, as in that case, it coincides with the even stronger *honest-input validity*, which requires agreement on an honest input, it might fail to provide meaningful outputs when working with larger input spaces. For instance, consider a set of parties running a BA protocol to agree on a room's temperature. Minor measurement errors are inherent, and hence strong validity allows the honest parties to agree on a temperature proposed by the corrupted parties. Similar challenges

arise in scenarios such as deciding on a meeting location using GPS coordinates [7] or when the map modeled as a graph [11, 27].

While achieving honest-input validity in scenarios where the input space is large (size $\omega(n)$) is impossible (due to corrupted parties possibly following the protocol correctly but with inputs of their choice), the literature offers a plethora of weaker alternatives, that are stronger, and hence more meaningful, than strong validity. An immediate option to avoid a corrupted output is to require standard strong validity with an additional condition called *intrusion tolerance* [13, 22]. This added condition requires the honest parties to either agree on an honest input or on a special symbol \perp . In the aforementioned scenarios of deciding on a measurement or a meeting point, a highly suitable alternative is the so-called *convex-hull validity*: the output agreed upon must be in the honest parties' inputs convex-hull, i.e., within the range of honest inputs if the input space is a subset of \mathbb{R} [11, 22, 25, 27, 29]. Stronger alternatives for real values focus on obtaining an output that is close to the honest inputs' median [10, 28], or to the k -th lowest honest input [24]. However, the honest-range approach is not a universal solution. For instance, this would carry no meaning in a voting problem if we represented candidates with integers. Instead, approaches based on social choice theory (*Pareto validity*) lead to more appropriate validity definitions [23].

The multitude of validity definitions leads to a natural question: what are the necessary and sufficient conditions for achieving BA with a given validity property, i.e., *to solve a given validity property?* This question can be concerned with multiple aspects, such as resilience thresholds t , round complexity, or message complexity. In addition, the conditions may depend on whether a cryptographic setup and randomization are available and on the guarantees of the communication network. The literature has considered various ways of modeling the network. One extreme is the synchronous model, where all parties have synchronized clocks, and all messages get delivered within a publicly known amount of time Δ . This model enables elegant protocols that operate in rounds, but that might fail in a real-life network where sporadic issues are possible. The other extreme is the asynchronous model, which makes no assumption except for messages getting delivered eventually. Indeed, the asynchronous model comes with highly robust protocols, but also with important drawbacks: (a) lower resilience threshold: i.e., $t < n/3$ as opposed to $t < n/2$ (assuming a public-key infrastructure) for strong validity, or, e.g., $t < n/4$ as opposed to $t < n/3$ (with or without cryptographic setup) for convex-hull validity in \mathbb{R}^2 ; (b) randomization is a requirement: BA cannot even be achieved deterministically in the asynchronous model even when at most one party might crash [19]. The middle ground between the two models has also been considered. The partially synchronous model [17] bridges the gap between the two extremes by assuming that the network is initially asynchronous and, after an unknown 'Global Stabilization Time' passes, the network becomes synchronous. In the synchronous model and the partially synchronous model, the solvability of validity conditions using deterministic protocols is completely understood [8, 9].

In this paper, we will be concerned with a different paradigm for bridging the gap between synchrony and asynchrony, namely the *network-agnostic* model, or the *best-of-both worlds* model, which has attracted increased attention in recent years [2, 5, 6, 11, 12, 20, 21, 26]. In this model, parties are initially unaware of whether the network is synchronous or not: if it is synchronous, then at most t_s of the parties involved may be corrupted, and otherwise only $t_a \leq t_s$ of the parties may be corrupted. Network-agnostic protocols are designed to provide guarantees in both cases. Hence, we ask the following question:

*What are the necessary and sufficient conditions for achieving
network-agnostic BA with a given validity property?*

1.1 Our Contribution

We provide necessary and sufficient conditions for a validity property to be solvable in the network-agnostic model.

To do this, we first show that, even if cryptographic setup is available, the condition $n > 2 \cdot t_s + t_a$ is a requirement for any non-trivial validity condition to be solvable (i.e., a condition for which simply outputting a default value does not suffice). In contrast, if cryptographic setup is unavailable, we show the stronger requirement of $n > 3 \cdot t_s$. Our proof for the latter, in fact, works in the purely synchronous model and, therefore, strengthens the characterization provided by [8] for the synchronous model. In particular, [8] only focuses on deterministic protocols, while our proofs rely on different techniques that also apply to randomized protocols.

Then, in addition to these lower bounds on n , regardless of whether cryptographic setup is available or not, we prove one more necessary condition for a validity property to be solvable, along the lines of the *similarity condition* of [9] and the *containment condition* of [8]. Roughly, this condition requires that for every input configuration of the honest parties, there is a common valid output for all potential input configurations that are similar to the real one. Here, the term *similar* captures that some of the proposed inputs may come from corrupted parties, and that, if the network is asynchronous, some of the honest parties' proposals might appear to be missing due to high network delays.

Finally, we show that, together, the aforementioned conditions are also sufficient by providing a universal protocol, that achieves network-agnostic BA for a given validity property whenever these conditions are satisfied. This is a more general variant of the protocol of [11]. In particular, the requirement for solvability is precisely $n > 2 \cdot t_s + t_a$ together with the similarity condition assuming cryptographic setup, and $n > 3 \cdot t_s$ together with the similarity condition assuming no cryptographic setup. Our protocol is randomized, which is a requirement when the network may be asynchronous and $t_a > 0$ [19].

1.2 Related Work

General validity conditions. The foundational investigation into general validity properties was initiated by Civit et al. [9] for the partially synchronous model. Subsequently, Civit et al. [8] embarked on a follow-up study, extending their analysis to the synchronous model. Both works provide a complete characterization, identifying the necessary and sufficient conditions for solving a validity property deterministically. We also note that the contributions of [8, 9] extend beyond this characterization, an important side of these works lying in the exploration of lower bounds on message complexity. This effectively generalizes the well-established Dolev-Reischuk bound on message complexity for BA with strong validity [15] to encompass the broader landscape of non-trivial validity properties. Considerations of message complexity are outside our scope.

Network-Agnostic BA and particular validity conditions. Designing protocols that achieve security guarantees in both synchronous and asynchronous networks has been the subject of an extensive line of work. The network-agnostic paradigm where the number of allowed corruptions depends on the network conditions was introduced by Blum, Katz and Loss

[5]. The work of [5] shows that, if a public key infrastructure is provided, BA with strong validity can be achieved if and only if the condition $n > 2 \cdot t_s + t_a$ holds. Further works on network-agnostic BA with strong validity have focused on improving the efficiency guarantees. Deligios, Hirt, and Liu-Zhang [12] have improved the expected round complexity from linear to constant. The efficiency guarantees were further refined by Deligios and Mizrahi Erbes [13] by providing a novel technique to compile any synchronous and any asynchronous BA protocols into a network-agnostic one. This compiler preserves the guarantees of the underlying protocols, both in terms of round complexity and message complexity. We add that the work of [13], in fact, considers a stronger validity guarantee, namely strong validity with the additional requirement of intrusion tolerance. The necessary and sufficient condition here stays $n > 2 \cdot t_s + t_a$.

Due to its broad applicability, convex-hull validity within the network-agnostic communication paradigm has attracted increased attention. Ghinea, Liu-Zhang and Wattenhofer [20,21] have investigated the feasibility of achieving convex-hull validity for a weaker, but inherently deterministic, variant of BA, known as Approximate Agreement [1,14]. In particular, [20] shows that if the parties' inputs are real numbers, Approximate Agreement is solvable under the same necessary and sufficient condition $n > 2 \cdot t_s + t_a$ assuming a public key infrastructure. Building on the previous, [21] gives sufficient conditions for the multidimensional variant of the problem that match the known requirements in the pure synchronous and asynchronous models [25,29]. Returning to the non-approximate version, Constantinescu et al. [11] have provided the necessary and sufficient conditions for achieving network-agnostic BA with convex-hull validity for abstract convex spaces. In this case, the conditions include $n > 2 \cdot t_s + t_a$ or, if no cryptographic setup is available, $n > 3 \cdot t_s$, along with a few additional conditions that depend on the Helly number of the convexity space.

Other takes on network-agnostic BA have been considered, such as *scaling* the validity guarantees with the network conditions. As an example, for the case of real number inputs, [10] proposes a protocol for BA with median validity guaranteeing that the output is closer to the honest inputs' median when the network is synchronous than when it is asynchronous. This protocol simultaneously matches the optimal closeness guarantees for purely synchronous [24,28] and purely asynchronous [10] networks. Such validity properties that scale with the network conditions are outside our scope.

Comparison to previous works. As outlined above, the conditions $n > 2 \cdot t_s + t_a$ and, if no cryptographic setup is available, $n > 3 \cdot t_s$, have been proven to be necessary for strong validity properties, i.e., stronger than strong validity [5]. Our work shows that this is a requirement for weaker validity properties as well, i.e., for any non-trivial property. We find this result surprising especially for *weak validity*: if *all parties are honest* and hold input v , then the output agreed upon must be v . Assuming a public key infrastructure, this property is solvable in the synchronous model for $t_s < n$, as a straightforward application of the Dolev-Strong broadcast protocol [16]. On the other hand, our result implies that if we expect a BA protocol with weak validity to remain secure in the asynchronous model even for no corruptions (i.e., $t_a = 0$), then the synchronous resilience threshold steps down from $t_s < n$ to $t_s < n/2$.

In comparison to the results of [11] regarding network-agnostic BA with convex-hull validity, our results move the difficulty of proving such feasibility results as a whole to only verifying whether a validity condition satisfies our similarity condition. That is, one can (non-trivially) show that convex-hull validity satisfies this similarity condition if and only if the necessary Helly number-based conditions of [11] hold. Our impossibility arguments diverge as we investigate these under any non-trivial validity property, while

the work of [11] considers a fixed validity property but also shows impossibility under weaker agreement requirements. On the other hand, our protocol matching our lower bounds is a more general variant of the protocol of [11].

Our paper provides a characterization that is similar to those of [8,9], which tackle the synchronous and partially synchronous settings, respectively. The key difference is that these two models allow for deterministic protocols, while the network-agnostic model inherently requires randomization for achieving BA when $t_a > 0$ [19]. Consequently, the focus shifts towards randomized protocols, requiring our proofs to employ different techniques. While the arguments behind the *containment* condition of [8] can be easily adapted to hold for randomized protocols as well, this is not immediate for their proof that $n > 3 \cdot t_s$ is necessary when no cryptographic setup is available. Our proof for this lower bound, in fact, assumes the synchronous setting and, therefore, strengthens the characterization of [8]. Summing up, their necessary conditions now hold even for randomized protocols, and, as shown in their paper, they can be matched by deterministic protocols. On the other hand, we note that a randomized variant of the message-complexity results of [8,9] is still an open problem in the (partially) synchronous model, and also in the network-agnostic model.

2 Preliminaries

We consider a setting with n parties $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ running a protocol in a fully-connected network, where links model authenticated channels. We will be working in the network-agnostic model: the network may be synchronous, or asynchronous, and the parties are not aware a priori of the type of network. If the network is synchronous, the parties hold perfectly synchronized clocks and each message is delivered within a publicly known amount of time Δ . Otherwise, if the network is asynchronous, and the only guarantee is that messages get delivered eventually.

Adversary. We assume a central adversary that may corrupt up to t_s of the n parties if the network is synchronous, and up to $t_a \leq t_s$ parties if the network is asynchronous. Corrupted parties permanently become byzantine, and hence may deviate arbitrarily (maliciously) from the protocol. The adversary additionally controls the message delivery schedule, subject to the conditions of the network type. Our protocols provide security against an adaptive adversary (i.e., may choose which parties to corrupt at any point in the protocol’s execution), while our impossibilities hold even for a static adversary (i.e., chooses which parties to corrupt at the beginning of the protocol’s execution).

Cryptographic primitives. We will be considering both settings with no cryptographic assumptions (and computationally unbounded adversaries) and with cryptographic setup (with computationally bounded adversaries). Protocols assuming a cryptographic setup will make use of a public key infrastructure (PKI) and a secure signature scheme, and only hold against a computationally bounded adversary. For simplicity of presentation, we assume that the signatures are perfectly unforgeable. In the same spirit, we also assume that in the absence of cryptographic setup an adversary can simulate any number of parties. This means that a byzantine party can internally simulate multiple parties, schedule the messages they send, and decide which messages they receive.

Byzantine Agreement and Validity. A BA protocol assumes that each (honest) party holds a value $v_{\text{IN}} \in V_{\text{IN}}$ as input, and enables the parties to agree on a common output $v_{\text{OUT}} \in V_{\text{OUT}}$ satisfying a given validity condition. In this paper, we assume that V_{IN} is at most countably infinite. The reason is that we need to take an intersection of events that

happen almost surely over the set of possible input configurations. If V_{IN} is uncountably infinite, the resulting event does not necessarily happen almost surely, making some of our proofs no longer work. We return to this issue in Section 7.

In the following, we formally present each property that a BA protocol needs to satisfy. The first property BA requires is *termination*, which may be deterministic (for synchronous protocols) or probabilistic:

- **(Termination)** Every honest party decides on an output v_{OUT} .
- **(Probabilistic Termination)** As time goes to infinity, the probability that an honest party has not yet decided on an output value v_{OUT} tends to 0.

The second property that BA requires is *agreement*, as defined below.

- **(Agreement)** If two honest parties output v_{OUT} and $v_{\text{OUT}'}$, then $v_{\text{OUT}} = v_{\text{OUT}'}$.

Before describing the validity property, we need to define *input configurations*. An input configuration is a set $I \subseteq V_{\text{IN}} \times \mathcal{P}$ containing pairs of honest parties and their inputs. Hence, if $(v, P) \in I$, then P is an honest party with input v . We use the notation $\text{PARTIES}(I)$ to refer to the set of (honest) parties in the input configuration I . Note that, if $P \notin \text{PARTIES}(I)$, then P is corrupted. Let $\mathcal{I} = \{I \subseteq V_{\text{IN}} \times \mathcal{P} : |I| \geq n - t_s\}$ denote the set of all possible input configurations, consisting of pairs of honest parties and their input values. We also define the inclusion relation for input configurations: for $I, J \in \mathcal{I}$, we say that $J \subseteq I$ if and only if $\text{PARTIES}(J) \subseteq \text{PARTIES}(I)$ and the parties in $\text{PARTIES}(J)$ have the same input value in both I and J . We say that an input configuration I is maximal if $\text{PARTIES}(I) = \mathcal{P}$. Moreover, because we assumed that V_{IN} is at most countably infinite, the size of \mathcal{I} is at most countably infinite. The validity property is then defined by a mapping $\text{VAL} : \mathcal{I} \rightarrow 2^{V_{\text{OUT}}}$ from honest parties' inputs to *valid* outputs:

- **(Validity)** If $I \in \mathcal{I}$ is the input configuration defined by the honest parties and their inputs, then no honest party outputs $v_{\text{OUT}} \notin \text{VAL}(I)$.

A validity property VAL is *trivial* if $\bigcap_{I \in \mathcal{I}} \text{VAL}(I) \neq \emptyset$. Note that, if this condition holds, then we can achieve validity, agreement, and termination with no communication: parties may simply output a value in $\bigcap_{I \in \mathcal{I}} \text{VAL}(I)$.

We say that a validity property VAL is *solvable* if there is a BA protocol *solving* VAL , as defined below for the network-agnostic model, and then for the purely synchronous model.

Definition 1. A protocol Π is a (t_s, t_a) -secure BA protocol solving a validity property VAL if it achieves probabilistic termination, agreement, and validity for the given property VAL even when up to t_s parties are corrupted if it runs in a synchronous network, and even when up to t_a parties are corrupted if it runs in an asynchronous network.

Definition 2. A protocol Π is a t_s -secure BA protocol solving a validity property VAL if, when running in a synchronous network, it achieves (probabilistic) termination, agreement, and validity for the given property VAL even when up to t_s parties are corrupted.

One might be tempted to believe that a $(t_s, 0)$ -secure protocol is simply a t_s -secure protocol, but the difference is rather subtle. Namely, a $(t_s, 0)$ -secure (network-agnostic) BA protocol also provides guarantees in an asynchronous network if all parties are honest. On the other hand, a t_s -secure (synchronous) BA protocol is not required to provide any guarantees if the synchrony assumptions fail. We add that this subtle difference does not apply to a purely asynchronous variant of the definition, which is equivalent to (t_a, t_a) -secure network-agnostic BA.

Executions. For a protocol Π , we define an execution ε by its input configuration $I \in \mathcal{I}$, the behavior of the byzantine parties, the randomness (both public and private) that the parties hold, and the behavior of the scheduler. Even for a randomized protocol, because the randomness and scheduler are fixed, an execution is deterministic. We say that an execution decides if all correct parties of the execution eventually decide an output. We note that, given a protocol satisfying probabilistic termination, with everything but the randomness fixed, including the scheduler and strategy of the adversary, an execution for this protocol decides almost surely over the randomness. We will also be using the term *canonical* to refer to executions happening in a synchronous network where all messages are delivered exactly Δ units of time after being sent and where all corrupted parties crash right at the beginning of the protocol (i.e., they do not send any messages). If the randomness is fixed, there is a unique canonical execution for a given input configuration.

For a given protocol Π , we say that two executions ε_1 and ε_2 cannot be distinguished by a party P that is honest in both executions if it has the same initial state in ε_1 and ε_2 (i.e., input and randomness), and receives in both ε_1 and ε_2 the same messages at the same times. As a consequence, if P cannot distinguish between ε_1 and ε_2 , then P is in exactly the same state at any time t in ε_1 and ε_2 . Hence, if P decides a value v_{OUT} at time t in one of the two executions, then it also decides v_{OUT} at time t in the other.

Validity in indistinguishable executions. Indistinguishability is a powerful tool, especially when considering scenarios where byzantine parties follow the protocol correctly, but with inputs of their own choice. This leads us to the following lemma, which will be often used in our proofs.

Lemma 3. *Let VAL be a validity property and Π be (t_s, t_a) -resilient BA protocol solving VAL . Consider two input configurations I, J such that $J \subseteq I$. Then, the value agreed upon in any execution of Π which decides and the input configuration is I must be in $\text{VAL}(J)$.*

Proof. Consider an executions ε_I of Π for I which decides. Construct the execution ε_J , which is identical to ε_I except that its input configuration is J instead of I . Observe that ε_J is an execution of Π for J . Indeed, parties in $\text{PARTIES}(I) \setminus \text{PARTIES}(J)$ (which are byzantine) can pretend that they are honest and behave exactly as they do in ε_I . The honest parties in J cannot distinguish between ε_I and ε_J , so the agreed-upon value in both executions must be the same. Since ε_J is a valid execution of Π for J , this value must be in $\text{VAL}(J)$. \square

For any validity property VAL , we can define a new validity property VAL' such that $\text{VAL}'(I) = \bigcap_{J \subseteq I} \text{VAL}(J)$ for all $I \in \mathcal{I}$. Property VAL' is simultaneously a stronger version of VAL , in that for all $I \in \mathcal{I}$ we have $\text{VAL}'(I) \subseteq \text{VAL}(I)$, but it also has the property of being monotonically decreasing, in that for $J \subseteq I$ we have $\text{VAL}'(I) \subseteq \text{VAL}'(J)$. Armed as such, the previous lemma has the following immediate corollaries:

Corollary 4. *A protocol solves VAL if and only if it solves VAL' . Hence, VAL is solvable if and only if VAL' is solvable.*

Corollary 5. *A solvable validity property VAL is trivial if and only if it permits deciding the same value for all maximal input configurations, i.e., $\bigcap_{I \in \mathcal{I}, \text{PARTIES}(I)=\mathcal{P}} \text{VAL}(I) \neq \emptyset$.*

We end with a quite technical lemma that will be used multiple times to finish a proof:

Lemma 6. *Let VAL be a solvable validity property and Π a protocol solving VAL . Let $I_1 \in \mathcal{I}$ be a maximal input configuration. If for every maximal input configuration $I_2 \in \mathcal{I}$, the*

canonical executions of Π for I_1 and I_2 given the same randomness R almost surely over R decide the same value, then VAL is trivial.

Proof. Note that a countable intersection of events that happen almost surely happens almost surely. We assume the lemma's condition and want to prove that VAL is trivial. Because V_{IN} is at most countably infinite, the number of maximal input configurations $I_2 \in \mathcal{I}$ is at most countably infinite. By taking an intersection of events, one for each maximal $I_2 \in \mathcal{I}$, that happen almost surely by the lemma condition, we get that almost surely over the randomness R , for all $I_2 \in \mathcal{I}$ the canonical executions of Π for I_1 and I_2 given R decide the same value. Hence, pick an R for which for all maximal $I_2 \in \mathcal{I}$ the canonical executions of Π for I_1 and I_2 given R decide the same value. This implies that, for all maximal $I \in \mathcal{I}$, the canonical execution of Π on I given R decides the same value $v \in V_{\text{OUT}}$. Therefore, by Corollary 5, VAL is trivial. \square

3 Lower Bound on n

In this section, we prove that for any non-trivial validity property, the condition $n > 2 \cdot t_s + t_a$ is necessary for it to be solvable in the network-agnostic model even if cryptographic setup is available. Our proof will be organized as follows:

- We start with a preliminary lemma in Section 3.1 which focuses on a simplified setting where $n := 2$. In this setting, at most one party can crash if the network is synchronous, and both parties are honest if the network is asynchronous. Our preliminary lemma will show a somewhat counter-intuitive result: roughly, the value decided upon in canonical executions is independent of the honest parties' inputs.
- In Section 3.2, we move from the simplified setting with two parties to a warm-up variant of our main proof. We will be working with n parties, but we will only consider the particular case $t_a := 0$. By reducing this case to our preliminary lemma, we show that the condition $n > 2 \cdot t_s$ is necessary in this setting.
- Finally, Section 3.3 focuses on the general case, showing that $n > 2 \cdot t_s + t_a$ is necessary by reducing to our preliminary lemma. The main proof will be a more general version of the warm-up proof.

We note that the proofs by Civit et al. in [8] rely on a reduction from any non-trivial validity property to weak validity. This has the advantage of only having to focus on weak validity for lower bounds. However, this reduction is invalid for randomized protocols and, therefore, cannot be used in our setting.

3.1 Preliminary Lemma

As aforementioned, our preliminary lemma only focuses on a simplified setting with $n = 2$ parties in the network-agnostic model. When the network is synchronous, we allow the adversary to corrupt up to one party ($t_s := 1$). We restrict the capabilities of the adversary by only allowing the corrupted party to crash. When the network is asynchronous, on the other hand, both parties are honest ($t_a := 0$). Then, we assume a protocol achieving (probabilistic) termination and agreement in this setting. We show that, almost surely, once randomness is fixed, all canonical executions decide the same value. Note that there may be a set of possible randomnesses such that some canonical executions do not even decide, but the probability of picking a randomness in this set is zero.

Lemma 7. *Assume $n := 2, t_s := 1$ and $t_a := 0$, and that corrupted parties are only allowed to crash. Consider a protocol A that achieves probabilistic termination and agreement in*

this setting. Then, almost surely, for fixed randomness R , there is a value v such that all canonical executions of A decide v .

Proof. Given a fixed randomness R (potentially including both public and private randomness), we will define a few canonical executions of A . For now, we assume that these executions are deciding, and we show at the end of the proof that this assumption holds almost surely.

Let P_1 and P_2 be the two parties, and $v_1, v_2 \in V_{\text{IN}}$ be two arbitrary input values. Consider a canonical execution ε_1 of A where P_1 is honest and has input v_1 , while P_2 crashes at the beginning of the execution. We assume ε_1 is a deciding execution, and we denote the value agreed upon by v . We show that v is the value agreed upon in every deciding canonical execution with randomness R .

Canonical executions where one of the parties crashes. First, we focus on the canonical executions where P_1 crashes. Hence, let ε_2 be a canonical execution where P_1 crashes, and P_2 has input v_2 . In addition, consider an execution $\varepsilon_{1,2}$ of A in the asynchronous model. In $\varepsilon_{1,2}$, both parties P_1 and P_2 are honest, with inputs v_1 and v_2 respectively. The messages between P_1 and P_2 are delayed until after the two parties decide on their outputs. The randomness in both executions is R , and we assume both ε_2 and $\varepsilon_{1,2}$ are deciding executions. We remark that P_1 cannot distinguish between executions ε_1 and $\varepsilon_{1,2}$ (in both cases it receives no messages at all). The agreement property of A ensures that P_1 and P_2 output the same value in $\varepsilon_{1,2}$, hence P_2 outputs v . Moreover, P_2 cannot distinguish between $\varepsilon_{1,2}$ and ε_2 , so P_2 must output v in execution ε_2 , which proves our claim for the case where P_1 crashes. By symmetry, considering execution ε_2 , where the value agreed upon is v , we get that the value agreed upon in any deciding canonical execution where P_2 crashes is v .

Canonical executions where none of the parties crashes. We still need to consider canonical executions where neither P_1 nor P_2 crash, and show that the two parties output the same value v in these executions. Hence, we define one more canonical execution ε_M , where both P_1 and P_2 are honest with inputs v_1 and v_2 respectively. We assume that ε_M is a deciding execution, and therefore the number of messages exchanged in ε_M is finite. Let ℓ be the number of messages exchanged during ε_M before a value is decided.

Roughly, to prove that v is decided in execution ε_M as well, we build a chain of scenarios between execution $\varepsilon_{1,2}$, where no messages are exchanged between P_1 and P_2 , and execution ε_M . In scenario m with $0 \leq m \leq \ell$, the first m messages get delivered exactly Δ units of time after being sent, and the others are delayed until after P_1 and P_2 decide their outputs. Scenario $m = 0$ corresponds to execution $\varepsilon_{1,2}$, while scenario $m = \ell$ corresponds to execution ε_M . Consecutive scenarios will be indistinguishable to the party that sent the last message, which ensures that executions $\varepsilon_{1,2}$ and ε_M decide the same value v .

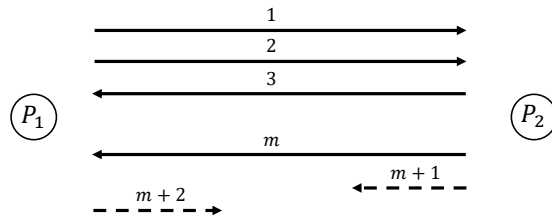


Figure 1: Example of execution $\varepsilon_{M,m}$: Any messages sent after the first m messages get delayed until after a value has been decided.

Then, for $0 \leq m \leq \ell$, we consider an execution $\varepsilon_{M,m}$ of A in the asynchronous model: the randomness is R , both P_1 and P_2 are honest, and they have inputs v_1 and v_2 respectively. The first m messages sent are received exactly Δ units of time later, while the following messages get delayed until after P_1 decides its value. We remark that $\varepsilon_{M,0}$ is $\varepsilon_{1,2}$ while ε_M is $\varepsilon_{M,\ell}$. We make the assumption that each of these $\ell + 1$ executions is deciding.

We prove that every execution $\varepsilon_{M,m}$ decides v using induction on m . The base case $m = 0$ holds immediately, as $\varepsilon_{M,0}$ is $\varepsilon_{1,2}$. For the induction step, let $m \geq 0$ and assume that execution $\varepsilon_{M,m}$ outputs v . The only difference between $\varepsilon_{M,m}$ and $\varepsilon_{M,m+1}$ is that one additional message is sent. However, as the sender cannot get any kind of acknowledgment that its message was received before deciding its output, it cannot differentiate between these two executions. To be more precise, if the $(m + 1)$ -th message was sent by P_1 , then P_1 cannot distinguish between $\varepsilon_{M,m}$ and $\varepsilon_{M,m+1}$, so it outputs the same value. The same holds for P_2 if the $(m + 1)$ -th message was sent by P_2 . As a consequence, because of agreement, the value agreed upon in $\varepsilon_{M,m}$ and $\varepsilon_{M,m+1}$ is the same. Therefore, ε_M will decide v .

Assumption on the fixed randomness. Before concluding the proof, we need to discuss the assumption we made on the described executions being deciding. For every possible input configuration, the proof assumes that a finite amount of fixed executions are deciding. Using the probabilistic termination property, an execution is almost surely deciding. Moreover, the set of input configurations is countable because we only consider countable input sets V_{IN} . Therefore, by taking a countable intersection of events that happen almost surely, we obtain that the assumptions we made in the proof happen almost surely. \square

3.2 Warm-up: $t_a := 0$

As a warm-up towards the main result, we focus on the particular case $t_a := 0$, and show that the condition $n > 2 \cdot t_s$ is necessary. The intuition behind this requirement is that, in an honest-minority setting, one cannot distinguish between a scenario where half of the parties crash in a synchronous network, and a scenario where half of the parties are honest but delayed in an asynchronous one. This way, the presence of the asynchronous case, even with no corruptions, allows two disjoint sets of honest parties to only run the protocol within their own set. Since the two sets run the protocol independently, the honest parties agree on a valid value only if the given validity property is trivial. Note that this is the case even for weak validity, which can be solved in the purely synchronous model even for up to $t_s < n$ corruptions. Our result implies that even expecting a synchronous protocol to provide guarantees when it runs in an asynchronous network with no corrupted parties impacts the feasibility conditions.

Theorem 8. *Assume $t_s > 0$ and consider a validity property VAL . If $n = 2 \cdot t_s$ and there is a $(t_s, 0)$ -secure BA protocol solving VAL , then VAL is trivial.*

Proof. Consider a (possibly randomized) $(t_s, 0)$ -secure BA protocol Π that solves VAL when $n = 2 \cdot t_s$. Let I_1 and I_2 be two arbitrary maximal input configurations. We show that, given the same randomness, the canonical executions of I_1 and I_2 almost surely decide on the same output value. Then, Lemma 6 ensures that VAL is trivial.

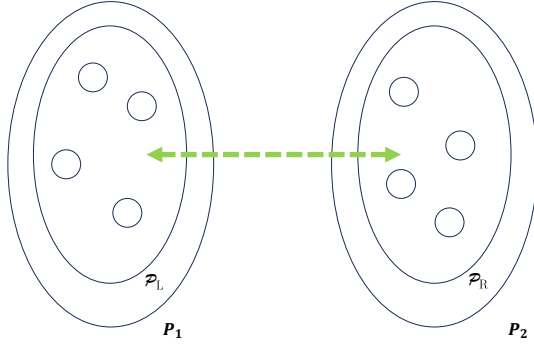


Figure 2: P_1 simulates all parties of \mathcal{P}_L while P_2 simulates all parties of \mathcal{P}_R . All the parties of \mathcal{P} communicate between one another not knowing they are being simulated.

We use Π to construct an protocol A for two parties that matches the setting described in Lemma 7. For protocol A , we consider the input set $\{0, 1\}$ and output set V_{OUT} , and we denote the two parties running A by P_1 and P_2 . Since $n = 2 \cdot t_s$, we may partition the set of n parties \mathcal{P} into two sets of t_s parties each, denoted by \mathcal{P}_L and \mathcal{P}_R . Then, as shown in Figure 2, P_1 will simulate the parties in \mathcal{P}_L , while P_2 will simulate the parties in \mathcal{P}_R . Concretely, in protocol A , P_1 proceeds as follows:

- P_1 simulates all t_s parties of \mathcal{P}_L running Π .
- If P_1 has input 0, then the simulated parties in \mathcal{P}_L use their inputs from I_1 . Otherwise, they use their inputs from I_2 .
- Messages between the simulated parties in \mathcal{P}_L are received after exactly Δ units of time, and P_1 forwards to P_2 every message sent by a simulated party in \mathcal{P}_L to a party in \mathcal{P}_R . Once P_2 receives this message, it immediately forwards it to the simulated receiver in \mathcal{P}_R .
- As soon as a party in \mathcal{P}_L decides a value, P_1 decides this value. P_1 continues forwarding messages as described above.

The behavior of P_2 is the same as the behavior of P_1 , switching \mathcal{P}_L and \mathcal{P}_R .

Note that running A in an asynchronous network where both P_1 and P_2 are honest corresponds to running Π in an asynchronous network where all n parties are honest. In addition, running A in a synchronous network where at most one party may crash corresponds to running Π in a synchronous network where at most t_s parties may crash. Then, since Π is a $(t_s, 0)$ -secure BA protocol, A achieves probabilistic termination and agreement in the setting described in Lemma 7.

We may then apply Lemma 7: almost surely, given the same randomness R , there is a value v such that all canonical executions of A decide on the same value v . Moreover, the canonical execution of A with input values $(0, 0)$ matches the canonical execution of I_1 for Π . Similarly, the canonical execution of A with input $(1, 1)$ matches the canonical execution of I_2 . As a consequence, canonical executions of I_1 and I_2 with the same randomness decide the same value almost surely. Using Lemma 6, we may therefore conclude that VAL is trivial. \square

3.3 General Result

We are now ready to prove the general version of our statement, as follows:

Theorem 9. *Consider a validity property VAL, and t_s, t_a such that $t_s > 0$ and $t_s \geq t_a$. If there is a (t_s, t_a) -secure BA protocol solving VAL when $n \leq 2 \cdot t_s + t_a$, then VAL is trivial.*

Proof. Assume $n = 2 \cdot t_s + t_a$ and that there is a (t_s, t_a) -secure BA protocol Π solving VAL. We consider two arbitrary input configurations I_1 and I_2 such that $\text{PARTIES}(I_1) = \text{PARTIES}(I_2) = \mathcal{P}$ (i.e., all parties are honest). We show that, given the same randomness R , the canonical executions of I_1 and I_2 almost surely decide on the same output value. Then, Lemma 6 ensures that VAL is trivial. Similarly to the proof of Theorem 8, we use Π to build a protocol A for two parties that matches the setting of Lemma 7. For protocol A , we consider the input space $\{0, 1\}$ and the output space V_{OUT} .

This time, we partition \mathcal{P} into three sets \mathcal{P}_L , \mathcal{P}_M and \mathcal{P}_R such that $|\mathcal{P}_L| = |\mathcal{P}_R| = t_s$ while $|\mathcal{P}_M| = t_a$. A crucial difference from the proof of Theorem 8 is that, as shown in Figure 3, each party in \mathcal{P} simulates its own copy of the parties in \mathcal{P}_M . Our construction will ensure that, when A runs in the synchronous setting, the two simulated copies of each party in \mathcal{P}_M will be in the exact same state at any point, hence maintaining the guarantees of Π when at most t_s of the parties are corrupted. Meanwhile, in the asynchronous setting, the copies of the t_a parties in \mathcal{P}_M may be in different states, but Π is able to tolerate t_a byzantine parties in this case.

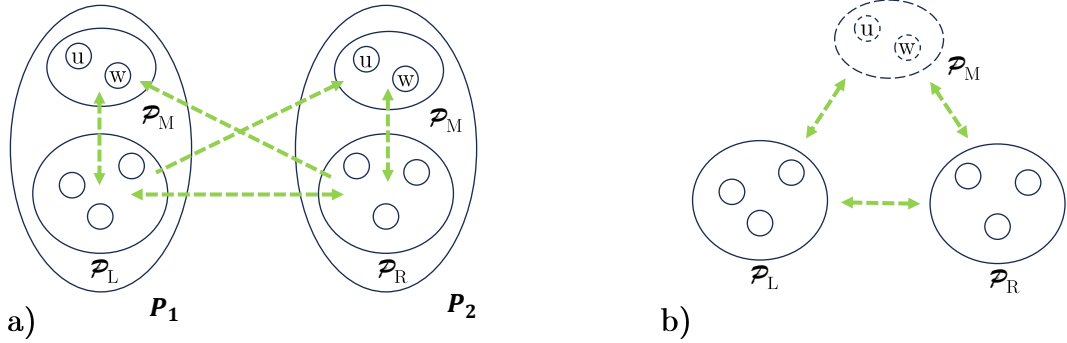


Figure 3: a) P_1 simulates all parties of \mathcal{P}_L and its own copies of the parties in \mathcal{P}_M . P_2 simulates all parties of \mathcal{P}_R , and its own copy of the parties \mathcal{P}_M . The arrows show which simulated group of parties can send messages to which other group. b) This is how the network looks like from the point of view of a party in \mathcal{P}_R or \mathcal{P}_M : they are unaware of the second set \mathcal{P}_M being simulated in parallel.

Concretely, in protocol A , party P_1 proceeds as follows:

- P_1 simulates all $t_s + t_a$ parties of $\mathcal{P}_L \cup \mathcal{P}_M$ running Π .
- If P_1 has input 0, then parties in $\mathcal{P}_L \cup \mathcal{P}_M$ take their input from I_1 . Otherwise, they take their input from I_2 .
- As depicted in Figure 4, the messages sent between the parties simulated by P_1 are exchanged as if all $t_s + t_a$ parties are running independently: each such message is delivered after exactly Δ units of time. Additionally, once a simulated party in \mathcal{P}_L sends a message to a simulated party in \mathcal{P}_M , P_1 immediately forwards this message to P_2 as well. Once P_2 receives this message, it immediately forwards it to its own simulated receiver in \mathcal{P}_M . The message delay here will depend on the type of network that A is running in.
- Messages from a party in \mathcal{P}_L to \mathcal{P}_R are sent from P_1 to P_2 . P_2 then forwards each such message to its simulated receiver in \mathcal{P}_R .
- P_1 discards any messages sent by the simulated parties in \mathcal{P}_M to parties in \mathcal{P}_R .
- As soon as a party in \mathcal{P}_L (but not \mathcal{P}_M) decides a value, P_1 decides this value. P_1 continues forwarding messages as described above until all its simulated parties

terminate.

The behavior of P_2 is the same as the behavior of P_1 , switching \mathcal{P}_L and \mathcal{P}_R .

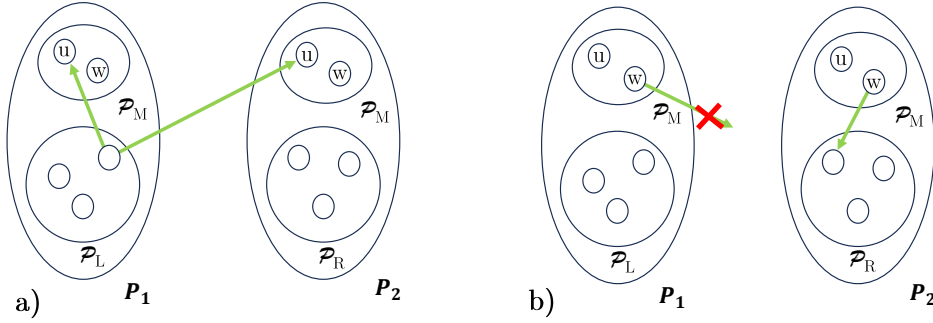


Figure 4: Examples of how different messages are handled. If a simulated party in \mathcal{P}_L wants to send a message to a party in \mathcal{P}_M (a), then two identical messages are actually sent: the first one to the copy of the receiver simulated by P_1 , and the second to the copy simulated by in P_2 . If a party in \mathcal{P}_M wants to send a message to a party in \mathcal{P}_R , if the two parties are simulated by the same entity (here P_2), then the message gets received as expected. Otherwise, the message is discarded and the party in \mathcal{P}_R never receives it.

We now need to analyze A in the setting described by Lemma 7. Running A in an asynchronous network where both parties are honest corresponds to running Π in a network where the communication between parties in \mathcal{P}_L and \mathcal{P}_R is asynchronous. While the simulated copies in \mathcal{P}_M are not necessarily consistent, Π is resilient against $t_a := |\mathcal{P}_M|$ byzantine corruptions, and therefore maintains its guarantees. Hence, A achieves agreement and probabilistic termination in this setting. It remains to show that A achieves these guarantees when running in a synchronous network where one of the two parties may crash. Settings where both P_1 and P_2 are honest correspond to running Π in a synchronous network where all n parties are honest. We note that, in this case, for each party in \mathcal{P}_M , the copy simulated by P_1 and the copy simulated by P_2 maintain the same state at all times. Settings where at most one of P_1 and P_2 may crash correspond to running Π in a synchronous setting where the t_s parties meant to be simulated only by the party that crashes in A are corrupted. Since Π is resilient to t_s corruptions in this case, it maintains its guarantees. Therefore, A achieves agreement and probabilistic termination in this setting as well.

We conclude the proof in an identical manner to the proof of Theorem 8. Applying Lemma 7, we obtain that, almost surely, given the same randomness R , all canonical executions of A decide on the same value. Moreover, for fixed randomness R , the canonical execution of A with input values $(0,0)$ matches the canonical execution of I_1 for Π . Similarly, the canonical execution of A with input $(1,1)$ matches the canonical execution of I_2 . As a consequence, canonical executions of I_1 and I_2 with the same randomness decide the same value almost surely. Using Lemma 6, we can therefore conclude that VAL is trivial. \square

4 Lower Bound on n Without Cryptographic Setup

In the previous section, we have proven that the condition $n > 2 \cdot t_s + t_a$ is necessary regardless of whether a cryptographic setup is available or not. We now focus on settings without a cryptographic setup, and we prove an even stronger lower bound on n . Namely,

we show that, when no cryptographic setup is available, the condition $n > 3 \cdot t_s$ is necessary even in the synchronous model. Since a protocol achieving (t_s, t_a) -secure BA in the network-agnostic model also achieves t_s -secure BA in the synchronous model, this bound immediately applies to the network-agnostic model. We add that our result extends the requirement of $n > 3 \cdot t_s$ provided by Civit et al. [8] for synchronous deterministic protocols to also cover randomized protocols as well.

We note that the following proof is an improvement over the FLM result for weak validity [18]. The FLM result only applies for weak validity and protocols must always decide in a finite amount of time, which is a lot stronger than probabilistic termination. Our proof uses the main core idea but improves it to lift these restrictions.

4.1 Preliminary Lemma

Similarly to the arguments in Section 3, we first consider a setting with three parties, out of which at most one may be byzantine. Afterwards, we focus on the general case.

Lemma 10. *Consider $n := 3$ parties in a synchronous network, and assume a protocol A that achieves (probabilistic) termination and agreement in this setting even when up to one party is byzantine. Then, almost surely, for a fixed randomness R , all canonical executions of A using randomness R decide the same value.*

To prove our statement, we will be running A in a larger ring containing multiple copies of each party, as depicted in Figure 5. The ring is constructed from two canonical executions with different input configurations. We will show that parties that are adjacent in this ring are unable to distinguish between the ring and the original setting of three parties, as the third party may be byzantine and simulate the rest of the ring. This will force parties adjacent in the ring to output the same value almost surely, which, in turn, guarantees that the two original executions lead to the same output almost surely.

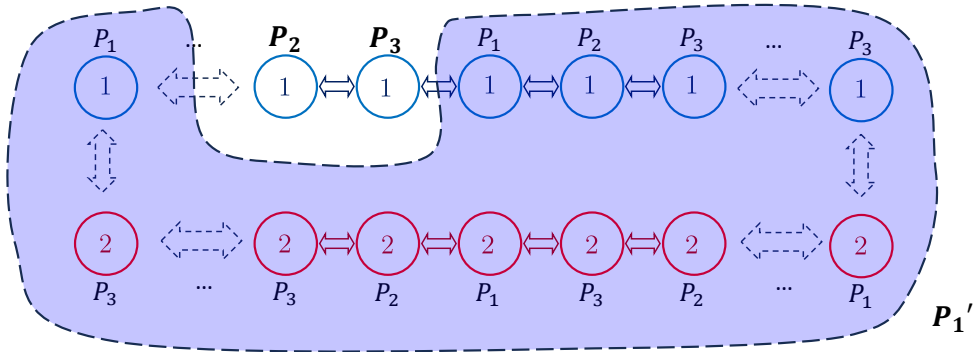


Figure 5: Defining the behavior of a byzantine party (here P_1') to guarantee that two consecutive parties decide the same value.

Constructing the ring. In order to formally describe the construction of the ring, we have to fix randomness (both public and private) R , and two input configurations and their canonical executions with randomness R in the original setting with three parties. Denote the three parties by $\mathcal{P} = \{P_1, P_2, P_3\}$, and let I_1, I_2 be two arbitrary maximal input configurations. We assume a deciding canonical execution ε_1 where the input configuration is I_1 , and a deciding canonical execution ε_2 where the input configuration is I_2 .

As ε_1 is a deciding execution, there exists a number of rounds $r_1 > 0$ such that, in ε_1 , all parties have decided the same value within r_1 rounds. Similarly, there is an r_2 such that, in the canonical execution ε_2 of I_2 , all parties have decided the same value within r_2 rounds. Let $r := \max\{r_1, r_2\}$.

To construct the ring depicted in Figure 5, we make $4(r+1)$ copies of each party P_i . Out of the $4(r+1)$ copies of party P_i , $2(r+1)$ will be the copies of P_i having its input value from I_1 and $2(r+1)$ will be the copies of P_i with input from I_2 . The copies of P_i are then denoted by $P_{k,i,j}$, where $k \in \{1, 2\}$, $i \in \{1, 2, 3\}$, and $j \in \{0, 1, \dots, 2r\}$. The copies indexed by $k := 1$ are the ones on the top row of Figure 5, while the copies indexed by $k := 2$ are the ones on the bottom row. We now connect these copies via bidirectional communication channels:

- For $k \in \{1, 2\}$ and $j \in \{0, 1, \dots, 2r\}$, we add a channel between $P_{k,1,j}$ and $P_{k,2,j}$, and one between $P_{k,2,j}$ and $P_{k,3,j}$.
- Then, to complete the path on the row indicated by index $k := 1$, for each $j \in \{0, 1, \dots, 2r-1\}$, we add a channel between $P_{1,3,j}$ and $P_{1,1,j+1}$.
- Similarly, to complete the path on the row indicated by $k := 2$, for each $j \in \{0, 1, \dots, 2r-1\}$, we add a channel between $P_{2,1,j}$ and $P_{2,3,j+1}$.
- We now connect the two rows and hence complete the ring: we add a channel between $P_{1,1,0}$ and $P_{2,3,0}$, and one between $P_{1,3,2r}$ and $P_{2,1,2r}$.

Outputs of adjacent copies. We consider a synchronous execution ε on the ring with fixed randomness R , where each copy $P_{k,i,j}$ runs A as party P_i , using the input value assigned to it in input configuration I_k . We assume that messages are received exactly Δ units of time after being sent. In the lemma below, we show that adjacent copies in the ring obtain the same output.

Lemma 11. *Consider two parties Q_1 and Q_2 that are adjacent in the ring. Then, in execution ε , Q_1 and Q_2 almost surely obtain outputs, and decide on the same value.*

Proof. Without loss of generality, assume that Q_1 and Q_2 are copies of P_1 and P_2 . We consider an execution of A with 3 parties with the same fixed randomness R as ε , denoted by ε' . In execution ε' , P_1 and P_2 have the same input values as Q_1 and Q_2 . P_3 is byzantine and simulates the additional parties in the ring, so they have the same behavior as in execution ε . All messages are delivered in exactly Δ time, similarly to execution ε .

We remark that execution ε' and execution ε are identical. Hence, our execution over the ring is equivalent to running A in a synchronous setting with 3 parties out of which one is corrupted; therefore, A maintains its properties, namely probabilistic termination and agreement, on the ring as well. We obtain that Q_1 and Q_2 both decide a value almost surely, and, when they both do, they decide on the same value. \square

Outputs on the entire ring. Lemma 11 establishes that adjacent copies in the ring output the same value in execution ε almost surely. Using induction, we can prove that all parties decide the same value in ε almost surely. As a consequence, the copies $P_{1,i,r}$ and $P_{2,i,r}$ of each party P_i decide the same value almost surely. This will imply that, in the two executions ε_1 and ε_2 that we considered when constructing the ring, all honest parties decide on the same output almost surely.

Lemma 12. *In execution ε , almost surely all parties obtain outputs, and they decide on the same value.*

Proof. We prove by induction that, in execution ε , after $r' \leq r$ rounds, parties $P_{1,i,j}$ for $i \in \{1, 2, 3\}$ and $j \in \{r - (r - r'), \dots, r + (r - r')\}$ are in the same state as party P_i in the canonical execution ε_1 .

The base case $r' := 0$ considers the very beginning of the executions ε and ε_1 , where the parties have not yet received any messages. Their state then is only defined by their input value and the (fixed) randomness R . Each party $P_{1,i,0}$ takes its input from I_1 , which is the case for P_i in execution ε_1 as well. Since the randomness R is fixed, we obtain that parties $P_{1,i,0}$ in the ring have the same input state as party P_i .

For the induction step, assume that our claim holds for $r' < r$, and we prove that it also holds for $r' + 1$. Let $\mathcal{P}_{r'}$ denote the set of parties for which we proved they were in the same state as ε_1 after r' rounds, and let $\mathcal{P}_{r'+1}$ be the set of parties for which we want to prove that the claim holds after $r' + 1$ rounds. The set of direct neighbors of $\mathcal{P}_{r'+1}$ (including themselves) in the ring is $\mathcal{P}_{r'}$. Moreover, within one round, parties are only able to receive messages from their direct neighbors. Using the induction hypothesis, we get that parties in $\mathcal{P}_{r'+1}$ receive exactly the same messages at the $(r' + 1)$ -th round in execution ε_1 and ε , therefore they will stay in the same state after round $r' + 1$.

As a consequence, P_i and $P_{1,i,r}$ are in the same state after the r -th round during executions ε_1 and ε respectively. However, we have assumed that all parties in \mathcal{P} decide their value v_{OUT} by round $r_1 \leq r$ in execution ε_1 . Therefore, in execution ε , $P_{1,i,r}$ decides the same value v_1 as the value P_i decides in ε_1 . With a symmetric argument, one can show that parties $P_{2,i,r}$ decide the same output v_2 . This, in turn, will allow us to prove that ε_1 and ε_2 decide the same value $v_1 = v_2$ almost surely using Lemma 11. \square

Assumption on deciding execution. Before concluding the proof using Lemma 12, we need to discuss the assumptions over deciding executions. For every possible input configuration of size three, we need to assume for a finite amount of fixed executions that they decide. Using the probabilistic termination property, an execution decides almost surely. Moreover, we take into account that there is only at most a countably infinite amount of input configurations (because we assumed V_{IN} was at most countably finite). By taking at most a countably finite intersection of events that occur almost surely, we get that the assumptions we made happen almost surely, which concludes the proof of Lemma 10.

4.2 General Result

We may now present our main result in this setting. To prove the statement below, we use a strategy similar to the proofs of Theorem 8 and Theorem 9.

Theorem 13. *Assume $t_s > 0$ and consider a validity property VAL. If there is a t_s -secure BA protocol solving VAL in the synchronous model when no cryptographic setup is available and $n \leq 3 \cdot t_s$, then VAL is trivial.*

Proof. Assume $t_s \geq \lfloor n/3 \rfloor$, and that there is a t_s -secure BA protocol Π solving VAL in the synchronous model.

We consider two arbitrary maximal input configurations I_1 and I_2 . We show that, given the same randomness R , all canonical executions of I_1 and I_2 with randomness R almost surely decide on the same output value. Then, Lemma 6 ensures that VAL is trivial. Similarly to the proof of Theorem 8 and Theorem 9, we use Π to build a protocol A for three parties, denoted by P_1, P_2, P_3 , that matches the setting of Lemma 10. For protocol A , we consider the input space $\{0, 1\}$ and the output space V_{OUT} .

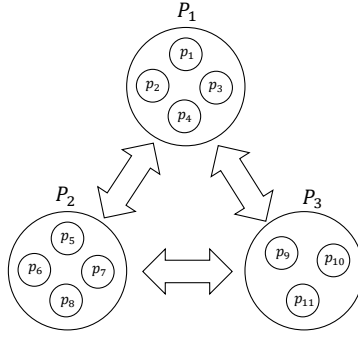


Figure 6: Partitioning P into 3 sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ with $n = 11$

To do so, we partition \mathcal{P} into three sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ of size at most t_s each. As shown in Figure 6, in protocol A , P_i simulates all the parties in set \mathcal{P}_i . P_i ensures that all messages between the simulated parties in \mathcal{P}_i get delivered within Δ time. In addition, P_i forwards any message sent from a party it simulates to a party in set \mathcal{P}_j ($j \neq i$) to P_j . When P_j receives this message, it immediately forwards it to the simulated receiver. If P_i has input 0, the simulated parties in \mathcal{P}_i take their input from I_1 . Otherwise, they take their input from I_2 . When a party in \mathcal{P}_i outputs a value v , P_i outputs v .

Since Π is a t_s -resilient BA protocol when $n \leq 3 \cdot t_s$, we obtain that A achieves probabilistic termination and agreement even when one of the three parties is byzantine. Then, Lemma 10 ensures that, almost surely, for a fixed randomness, all deciding canonical executions of A lead to the same output value.

Moreover, we remark that by construction, for a fixed randomness, the canonical execution of A with input values $(0, 0, 0)$ matches the canonical execution of I_1 for Π . Similarly, the canonical execution of A with input $(1, 1, 1)$ matches the canonical execution of I_2 . As a consequence, canonical executions of I_1 and I_2 with the same randomness decide the same value almost surely. Applying Lemma 6, we conclude that VAL is trivial. \square

Then, as any lower bound in the synchronous model also holds in the network-agnostic model, Theorem 13 immediately implies the following corollary.

Corollary 14. *Consider a validity property VAL, and let t_s, t_a such that $t_s > 0$ and $t_s \geq t_a$. If there is a (t_s, t_a) -secure BA protocol solving VAL when no cryptographic setup is available and $n \leq 3 \cdot t_s$, then VAL is trivial.*

5 Similarity Condition

The lower bounds on n presented in the previous sections are indeed necessary, but not yet sufficient. We may instantiate, for instance, the input space as the (finite) set of vertices of a public graph with maximum clique size $\omega \geq 2$. We consider convex-hull validity, under the so-called *monophonic convexity*. In this particular example, network-agnostic BA requires the stronger lower bound $n > \max(\omega \cdot t_s, \omega \cdot t_a + t_s, 2 \cdot t_s + t_a)$ [11]. Roughly, if any of the additional conditions $n > \omega \cdot t_s$ and $n > \omega \cdot t_a + t_s$ fails to hold, one can find scenarios where the simple presence of byzantine parties (following the protocol correctly, with inputs of their choice) prevents the honest parties from finding a valid output to agree upon. In this section, we prove the need of one more condition that captures these validity-dependent requirements, and ensures that the honest parties are able to find a

valid output even if their view over the honest inputs is not accurate. Our additional condition matches the *similarity* condition of [9] for the partially synchronous model, and the *containment* condition of [8] in the synchronous model.

We need to establish a few definitions. The first is the notion of *neighbors* of an input configuration. The neighbors of I , denoted by $\text{NEIGHBORS}(I)$, are the input configurations J such that the parties in $\text{PARTIES}(I) \cap \text{PARTIES}(J)$ hold the same input values in I and J . Formally, $\text{NEIGHBORS}(I) := \{J \in \mathcal{I} : \forall P \in \mathcal{P}, \text{ if } (v_1, P) \in I \text{ and } (v_2, P) \in J \text{ then } v_1 = v_2\}$. We note that the definition of neighbors is symmetric (if $J \in \text{NEIGHBORS}(I)$ then $I \in \text{NEIGHBORS}(J)$).

The second notion is that of *similar* configurations of an input configuration I , denoted by $\text{SIMILAR}(I)$. These are the input configurations that a protocol may not be able to differentiate from I . In the synchronous model, these will be input configurations $J \in \text{NEIGHBORS}(I)$ such that $J \subseteq I$. Roughly, this models scenarios where some of the parties might be corrupted, but might follow the protocol correctly with inputs of their own choice. In the asynchronous model, these will be input configurations $J \in \text{NEIGHBORS}(I)$ containing $n - t_a$ parties: which also models scenarios where some of the parties might be corrupted but follow the protocol correctly with inputs of their choice, but also takes into account that at most t_a honest parties may be isolated due to network delays. Hence, we define $\text{SIMILAR}(I)$ as: $\text{SIMILAR}(I) = \{J \in \text{NEIGHBORS}(I) : J \subseteq I\} \cup \{J \in \text{NEIGHBORS}(I) : |J| \geq n - t_a\}$.

We may now define the similarity condition, which Lemma 16 proves to be necessary for the network-agnostic model, using standard indistinguishability arguments.

Definition 15 (Similarity condition). *We say that a validity property VAL satisfies the similarity condition if there is a turing-computable function $\sigma : \mathcal{I} \mapsto V_O$ such that, for any input configuration $I \in \mathcal{I}$, $\sigma(I) \in \bigcap_{J \in \text{SIMILAR}(I)} \text{VAL}(J)$.*

Note that this also implies that $\bigcap_{J \in \text{SIMILAR}(I)} \text{VAL}(J) \neq \emptyset$.

Lemma 16. *If a validity property VAL is solvable in the network-agnostic model, then VAL satisfies the similarity condition.*

Proof. Assume that Π is a network-agnostic BA protocol solving VAL . Consider an input configuration I and the following execution:

- ε_1 : Π runs in the synchronous setting, with input configuration I . The parties in $\mathcal{P} \setminus \text{PARTIES}(I)$ are corrupted and crash at the very beginning of the execution.

As Π achieves network-agnostic BA, all honest parties output the same value v in execution ε_1 . We want to prove that $v \in \bigcap_{J \in \text{SIMILAR}(I)} \text{VAL}(J)$, hence we show that $v \in \text{VAL}(J)$ for every $J \in \text{SIMILAR}(I)$. Using the definition of $\text{SIMILAR}(I)$, we split the analysis as follows:

- (i) $J \in \text{NEIGHBORS}(I)$ such that $J \subseteq I$. We then consider an execution ε_2 where the input configuration is J and the network is synchronous. The parties in $\text{PARTIES}(I) \setminus \text{PARTIES}(J)$ are byzantine, but follow the protocol correctly using the values assigned to them in I as inputs. The parties in $\mathcal{P} \setminus \text{PARTIES}(I)$ crash at the very beginning of the execution.

Parties in $\text{PARTIES}(J)$ cannot distinguish between ε_2 and ε_1 , so the output value v must also satisfy $v \in \text{VAL}(J)$

- (ii) $J \in \text{NEIGHBORS}(I)$ such that $|J| \geq n - t_a$. First, note that $I \cap J \neq \emptyset$: since $|I| \geq n - t_s$ and $|J| \geq n - t_a$, we obtain that $|I \cap J| \geq n - t_s - t_a$. As VAL is solvable, Theorem 9 ensures that $n > 2 \cdot t_s + t_a$, and allows us to conclude that $|I \cap J| > 0$.

Let $P \in I \cap J$. We consider an execution ε_3 where the input configuration is J and the network is asynchronous. Similarly to execution ε_2 , parties in $\text{PARTIES}(I) \setminus \text{PARTIES}(J)$ are byzantine, but follow the protocol correctly using the values assigned to them in I as inputs, and parties in $\mathcal{P} \setminus (\text{PARTIES}(I) \cup \text{PARTIES}(J))$ crash at the very beginning of the execution. All messages are delivered in a synchronous way, except for the messages sent from parties in $\text{PARTIES}(J) \setminus \text{PARTIES}(I)$. These are delayed until after party P outputs a value: this is possible as P cannot distinguish between ε_1 and ε_3 and therefore it has to obtain an output without receiving these messages. In addition, the fact that P cannot distinguish between ε_1 and ε_3 ensures that the output v agreed upon in ε_1 satisfies $v \in \text{VAL}(J)$.

Therefore, the output v in execution ε_1 satisfies $v \in \bigcap_{J \in \text{SIMILAR}(I)} \text{VAL}(J)$. We may compute $\sigma(I)$ by simulating ε_1 , which can be done by a turing machine. \square

6 Sufficiency and Main Result

We now show that the conditions presented in the previous sections are not only necessary, but also sufficient, hence proving our main result, stated below.

Theorem 17. *Assume a non-trivial validity condition VAL . Then, there is a (t_s, t_a) -secure protocol solving VAL if and only if the following conditions hold:*

- VAL satisfies the similarity condition.
- $n > 3 \cdot t_s$ or, if a public key infrastructure is available, $n > 2 \cdot t_s + t_a$.

Our sufficiency proof is constructive. In the remainder of the section, we focus on presenting a protocol that matches the necessary conditions, as stated in the lemma below. Theorem 17 then follows from combining this result with the results showing the required lower bounds on n , namely Theorem 9 and Corollary 14, and with Lemma 16, that ensures the similarity condition is needed.

Lemma 18. *Assume a validity condition VAL that satisfies the similarity condition. Then, if $n > 3 \cdot t_s$, or, assuming that a public key infrastructure is available, if $n > 2 \cdot t_s + t_a$, there is a (t_s, t_a) -secure BA protocol solving VAL .*

Our construction generalizes the network-agnostic BA protocol of [11] that achieves convex-hull validity. The parties distribute their input values using a protocol achieving *Agreement on a Core-Set* (ACS), which enables them to obtain an identical view over the inputs, i.e. to agree on a potential input configuration. This is a (randomized) communication primitive introduced for enabling identical views in the pure asynchronous model [3,4]. We make use of the ACS definition of [11], included below, which differs from the standard definition by providing stronger properties in the synchronous model. These stronger guarantees are essential for matching the higher resilience threshold t_s that we expect if the network is synchronous. This way, the parties can apply a deterministic decision over the view agreed upon in ACS and obtain an output.

Definition 19. *Let Π be a protocol where every party P_i holds an input v_i and outputs a set of value-sender pairs $I_i \in \mathcal{I}$. We consider the following properties in addition to those presented in Section 2:*

- **Integrity:** *If P_i and P_j are both honest and $P_i \in \text{PARTIES}(I_j)$, then $(P_i, v_i) \in I_j$.*
- **Honest Core:** *If an honest party outputs I , then $\text{PARTIES}(I)$ contains all honest parties.*

Then, we say that Π is a (t_s, t_a) -secure ACS protocol if it achieves the following:

- Probabilistic termination, agreement, integrity, and honest core when running in a synchronous network where up to t_s parties are corrupted;
- Probabilistic termination, agreement, and integrity when running in an asynchronous network where up to t_a parties are corrupted.

We make use of the ACS construction of [11], described by the result below.

Theorem 20 ([11]). *Consider n, t_s, t_a such that $t_a \leq t_s$ and $3 \cdot t_s < n$, or, if a public key infrastructure is available, $2 \cdot t_s + t_a < n$. Then, there is a protocol Π_{ACS} that achieves (t_s, t_a) -secure ACS.*

We may now present the proof of Lemma 18, describing our protocol.

Proof of Lemma 18. Our BA protocol solving VAL proceeds as follows: the parties distribute their input values via using the protocol Π_{ACS} described in Theorem 20. Π_{ACS} provides the parties with the same output set $I \in \mathcal{I}$ representing a potential input configuration. Once the parties obtain this set, they output $\sigma(I)$, where σ is the turing-computable function provided by VAL satisfying the similarity condition.

Regardless of whether the network is synchronous or asynchronous, since Π_{ACS} achieves probabilistic termination, our BA protocol achieves probabilistic termination as well. In addition, since Π_{ACS} achieves agreement, the parties compute their output identically, and therefore our BA protocol achieves agreement.

It remains to prove that the honest parties' output is in $\text{VAL}(H)$, where H denotes the (actual) input configuration (containing only the honest parties and their inputs). Since $\sigma(I) \in \bigcap_{J \in \text{SIMILAR}(I)} \text{VAL}(J)$, it will be sufficient to show that $H \in \text{SIMILAR}(I)$. Regardless of the type of network, the integrity property of Π_{ACS} ensures that $H \in \text{NEIGHBORS}(I)$. If the network is asynchronous, $|H| \geq n - t_a$ and therefore $H \in \text{SIMILAR}(I)$. Otherwise, if the network is synchronous, the honest core property of Π_{ACS} ensures that honest parties and their inputs are included in I . Then, $H \subseteq I$ and therefore $H \in \text{SIMILAR}(I)$ in this case as well. Thus, as $\sigma(I) \in \bigcap_{H \in \text{SIMILAR}(I)} \text{VAL}(H)$, we get that the value agreed upon is in $\text{VAL}(H)$, which proves the validity of the protocol and concludes the proof. \square

7 On Uncountable Input and Output Sets

In this section, we make a remark on the importance of assuming that either the input set V_{IN} or the output set V_{OUT} is countable. For our proofs, we assumed that this is the case for V_{IN} , and it is possible to adapt all arguments for the other case. In contrast, perhaps surprisingly, not only can our arguments not be adapted for the case where both V_{IN} and V_{OUT} are uncountably infinite, but in the following, we describe a brief counterexample to the fact that $n > 2 \cdot t_s + t_a$ is a requirement for a non-trivial validity property to be solvable.

For the uncountably infinite sets $V_{\text{IN}} = V_{\text{OUT}} := [0, 1]$, we define a validity property VAL^* as follows: if all honest parties hold the same input v , they may output any value in $[0, 1]$ except for v ; otherwise, they may output any value in $[0, 1]$. Hence, $\text{VAL}^*(I) = [0, 1] \setminus \{v : \text{all parties in } I \text{ have input } v\}$. Note that this validity property is non-trivial as $\bigcap_{v \in [0, 1]} ([0, 1] \setminus \{v\}) = \emptyset$. This property escapes our requirement of $n > 2 \cdot t_s + t_a$ if a very strong shared randomness setup is provided. Under the assumption that the parties are provided with a shared uniform random value $s \in [0, 1]$, we can design a $(n - 1, n - 1)$ -secure BA protocol where parties do not even communicate. We present the protocol below.

$(n - 1, n - 1)$ -secure BA solving VAL*

Code for party P_i with input v_i

- 1: Let $s :=$ the shared uniform random value in $[0, 1]$.
- 2: If $v_i = s$, never terminate. Otherwise, output s and terminate.

The non-terminating executions of this protocol are those where s matches some honest party's input, hence they occur with probability 0. That is, all honest parties decide on an output almost surely, which satisfies our probabilistic termination definition requirement. If two honest parties obtain outputs, then they both output the same value s . Hence, the agreement property holds. Finally, since no honest party outputs its own input, our protocol solves VAL*. If either V_{IN} or V_{OUT} is countable, our counterexample fails. We note that restrictions on the protocol's setup may also be promising for extending our results to settings where both V_{IN} and V_{OUT} uncountable: if s comes from a finite domain, then the probability of obtaining a non-terminating execution would not be 0.

8 Conclusions and Future Work

We investigated the conditions that a validity property needs to satisfy in order to be solvable in the network-agnostic model and established the necessary and sufficient conditions. Our results demonstrate that solving a non-trivial validity property VAL requires (i) that VAL satisfies the *similarity condition*, and (ii) that $n > 2 \cdot t_s + t_a$ assuming a public key infrastructure, or $n > 3 \cdot t_s$ otherwise. Further, we provided a universal protocol that solves a given validity property whenever these established conditions are met. Our characterization follows the line of works of [8,9] focusing on the partially synchronous model and on the synchronous model. At the same time, it generalizes prior results on when network-agnostic BA can be achieved [5, 11] from fixed validity properties to arbitrary validity properties.

While our work provides a complete answer for solvability, we leave a number of exciting problems open. Future works could extend our characterization to cover settings where both V_{IN} and V_{OUT} are uncountable sets and consider proving message complexity lower bounds. Other promising directions would aim to improve the efficiency of our universal protocol, or to generalize our results further to network-dependent validity properties (that allow weaker guarantees if the network is asynchronous) [10], or to weaker agreement definitions [11].

References

- [1] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In Teruo Higashino, editor, *Principles of Distributed Systems*, pages 229–239, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [2] Ananya Appan, Anirudh Chandramouli, and Ashish Choudhury. Perfectly-secure synchronous mpc with asynchronous fallback guarantees. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 92–102, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538417.

- [3] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 52–61, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/167088.167109.
- [4] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '94, page 183–192, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/197917.198088.
- [5] Erica Blum, Jonathan Katz, and Julian Loss. Synchronous consensus with optimal asynchronous fallback guarantees. In *Theory of Cryptography Conference*, pages 131–150. Springer, 2019.
- [6] Erica Blum, Chen-Da Liu-Zhang, and Julian Loss. Always have a backup plan: Fully secure synchronous mpc with asynchronous fallback. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 707–731, Cham, 2020. Springer International Publishing.
- [7] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeul. Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science*, 411(34):3154–3168, 2010. doi:10.1016/j.tcs.2010.05.006.
- [8] Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, Anton Paramonov, and Manuel Vidigueira. All byzantine agreement problems are expensive, 2023. arXiv:2311.08060.
- [9] Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. On the validity of consensus. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 332–343, 2023.
- [10] Andrei Constantinescu, Diana Ghinea, Lioba Heimbach, Zilin Wang, and Roger Wattenhofer. A Fair and Resilient Decentralized Clock Network for Transaction Ordering. In Alysson Bessani, Xavier Défago, Junya Nakamura, Koichi Wada, and Yukiko Yamauchi, editors, *27th International Conference on Principles of Distributed Systems (OPODIS 2023)*, volume 286 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.OPODIS.2023.8.
- [11] Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. Convex Consensus with Asynchronous Fallback. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing (DISC 2024)*, volume 319 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:23, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2024.15>, doi:10.4230/LIPIcs.DISC.2024.15.
- [12] Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In *Theory of Cryptography Conference*, pages 623–653. Springer, 2021.

- [13] Giovanni Deligios and Mose Mizrahi Erbes. Closing the efficiency gap between synchronous and network-agnostic consensus. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 432–461, Cham, 2024. Springer Nature Switzerland.
- [14] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, May 1986. doi:10.1145/5925.5931.
- [15] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, jan 1985. doi:10.1145/2455.214112.
- [16] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
- [17] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, apr 1988. doi:10.1145/42282.42283.
- [18] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In Michael A. Malcolm and H. Raymond Strong, editors, *4th ACM PODC*, pages 59–70. ACM, August 1985. doi:10.1145/323596.323602.
- [19] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- [20] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Optimal synchronous approximate agreement with asynchronous fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC’22, page 70–80, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538442.
- [21] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Multidimensional approximate agreement with asynchronous fallback. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA ’23, page 141–151, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3558481.3591105.
- [22] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Brief Announcement: Communication-Optimal Convex Agreement. In *The 43rd ACM Symposium on Principles of Distributed Computing (PODC), Nantes, France*, June 2024.
- [23] Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. Byzantine Preferential Voting. In *14th Conference on Web and Internet Economics (WINE), Oxford, United Kingdom*, December 2018.
- [24] Darya Melnyk and Roger Wattenhofer. Byzantine agreement with interval validity. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 251–260, 2018. doi:10.1109/SRDS.2018.00036.
- [25] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K Garg. Multidimensional agreement in byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.

- [26] Atsuki Momose and Ling Ren. Multi-threshold byzantine fault tolerance. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21*, page 1686–1699, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3460120.3484554.
- [27] Thomas Nowak and Joel Rybicki. Byzantine Approximate Agreement on Graphs. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.DISC.2019.29.
- [28] David Stolz and Roger Wattenhofer. Byzantine Agreement with Median Validity. In Emmanuelle Anceaume, Christian Cachin, and Maria Potop-Butucaru, editors, *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, volume 46 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.OPODIS.2015.22.
- [29] Nitin H. Vaidya and Vijay K. Garg. Byzantine vector consensus in complete graphs. In Panagiota Fatourou and Gadi Taubenfeld, editors, *32nd ACM PODC*, pages 65–73. ACM, July 2013. doi:10.1145/2484239.2484256.