Hierarchical Consensus: Scalability Through Optimism and Weak Liveness

Pedro Antonino
The Blockhouse Technology Ltd., Oxford, UK

Antoine Durand
The Blockhouse Technology Ltd., Oxford, UK

A. W. Roscoe
The Blockhouse Technology Ltd., Oxford, UK
University College Oxford Blockchain Research Centre, UK

Abstract

Scalability is a central concern of Byzantine Fault Tolerant (BFT) distributed protocols. The ubiquitous approach to work around the well-known Dolev-Reischuk $\Omega(n^2)$ communication complexity lower bound is to use a random selection process to draw a hopefully small committee from a population of agents to run the communication-heavy protocol. We propose a notion of *hierarchical consensus* that combines two sub-protocols: an optimistic primary sub-protocol that can tolerate less than 1/2 failures and a fallback secondary protocol that can tolerate less than 1/3 failures; we achieve the higher failure threshold by requiring a weaker notion of liveness for the primary. This distinction between the level of fault tolerance between primary and secondary is reflected in the size of committees implementing these protocols. For a population of agents with close to 2/3 of honest agents, we need to select a committee with hundreds of agents to reach the level of tolerance expected for the primary, whereas we need thousands to reach the level expected for the secondary with a very small probability of error ϵ. Our hierarchical construct is such that if the primary comes to a decision, it can simply propagate it to the secondary protocol, so it does not need to properly engage in an agreement protocol independently. Our architecture is flexible and allows us to use our technique for most protocols that are based on random sampling. By studying hierarchical protocols, we discovered new theoretical results of independent interest. Specifically, the ability to handover from a primary protocol requires a new *Justifiability* property that allows agents to pre-decide on a value, such that if the protocol decides, it must be on that pre-decided value.

2012 ACM Subject Classification Networks → Network protocol design; Theory of computation → Distributed algorithms

Keywords and phrases Hierarchical, Handover, Justifiability, Consensus, Distributed Systems, Blockchain

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.6

1 Introduction

The scalability of Byzantine fault-tolerant (BFT) protocols [20, 6, 14] has been the subject of renewed attention since the advent of large-scale distributed applications such as blockchains [23, 27, 19]. Permissionless protocols allow in principle an unbounded number participants, so their communication cost (i.e. the number of bits sent by honest agents) must not overwhelm the network.

Deterministic BFT consensus protocols are subject to the well-known Dolev-Reishuk $\Omega(n^2)$ lower bound on communication cost [18, 8], where n is the number of agents participating in the protocol. Scalable agreement protocols try to work around this lower bound by randomly selecting committees: members are picked randomly and independently from an underlying population of agents with a defined proportion of honest-to-Byzantine agents. The goal of this process is to minimise the size n of the committee selected: find the smallest n such

that the committee has the expected ratio of honest-to-Byzantine agents. Of course, by minimising n, the communication cost $\Omega(n^2)$ is reduced. However, selecting a committee size that ensures *with high probability* that more than 2/3 agents are honest – the lower bound for partially synchronous protocols – from a population of agents with a similar ratio of honest agents leads to committee sizes in the thousands, hindering the end goal of scalability.

Our work stems from the observation that it is possible to have much smaller committees if we select, instead, committees with more than 1/2 honest agents in a population with roughly a 2/3 probability of an agent being honest. We rely on the observation from Momose and Ren [22] that BFT protocols can set different fault thresholds t_{safe} and t_{live} for their safety and liveness properties, respectively; protocols that do not use this multi-threshold approach implicitly make $t_{safe} = t_{live}$. The committee size can be set to guarantee fewer than t_{safe} faults, while keeping a reasonable probability that there are fewer than t_{live} faults as well. Since t_{safe} can be set to a value that is higher than t_{live}, the committee can be much smaller. For instance, we can have that $t_{safe} < n/2$ whereas for traditional (single-threshold) protocols $t_{safe} = t_{live} < n/3$.

We propose the notion of *hierarchical consensus* as a framework to create efficient BFT consensus protocols by exploring this threshold flexibility. Although the concept of a hierarchical architecture itself is not novel, it presents an extension of the traditional notion of agreement protocols in the sense that it allows some protocol agents to output values that are *not decisions*. Intuitively speaking, this concept combines two sub-protocols: an optimistic primary protocol that is safe and only weakly live, and a fallback secondary protocol that is safe and live. By only requiring weak liveness in non-optimistic executions, the secondary can be guaranteed to execute when necessary. Thus, in optimistic executions of this protocol, the primary sub-protocol comes to a decision that is simply propagated to the secondary protocol; the latter does not have to be active at all. It is important to note here that t_{safe} also ensures weak liveness for the primary; this notion allows the fallback to be started when the primary agents do not reach a decision.

We introduce the notion of a *Justifiable reliable broadcast* (JRBC) protocol as a way to characterise protocols that can be used as a primary. It captures the safety and weak liveness requirements that enable a safe handover to the secondary sub-protocol in the case that the primary is unable to decide. We show that for a JRBC protocol, it must be that $2t_{safe} + t_{live} < n$, whereas non-Justifiable protocols are only bound by the strictly weaker relation $t_{safe} + 2t_{live} < n$. This impossibility shows for the first time that there are extra constraints when implementing such primary/secondary optimistic architecture. We also propose an implementation *psync JRBC* of a partially synchronous JRBC that demonstrates that this bound is tight. Notably, a similar multi-threshold approach to consensus has been tried [11], but that attempt falls short of reaching the same performance gains, precisely because the Justifiability requirements that we identified were not met. In [25, 26], the authors propose handover protocols that are similar in nature to ours but different in design: they rely on a different handover mechanism and communication model.

We propose the *handover construction* $\mathcal{P}^H$ that creates a hierarchical consensus protocol by combining a Justifiable RBC protocol $\mathcal{P}_o$ (with participant agents $\mathcal{A}_o$) and a consensus protocol $\mathcal{P}_l$ (with participant agents $\mathcal{A}_l$). Our *psync JRBC* implementation can be combined with a consensus implementation of choice using this construction to create a hierarchical consensus implementation. Our *psync JRBC* implementation shows that JRBC protocols can tolerate up to $\lfloor \frac{n-1}{2} \rfloor$ faults while remaining safe and weakly live as opposed to the $\lfloor \frac{n-1}{3} \rfloor$ failure tolerance level for live consensus protocols; this difference in their ability to tolerate failures is crucial to achieve efficiency. In an optimistic run, the agents implementing $\mathcal{P}_o$ can

come to a decision independently of the agents in $\mathcal{P}_l$. Thus, by minimising the size of $\mathcal{A}_o$, we can improve $\mathcal{P}_o$'s communication complexity. As an additional bonus, although optimistic executions can happen with up to t_{live} faults, the probability of having an optimistic execution only depends on *the actual* number of faults. Of course, non-optimistic runs can be inefficient as both sub-protocols attempt to reach agreements in succession.

We employ a notion of *stochastic reasoning* to demonstrate the impact that these levels of tolerance have on the size of committees – that is, $\mathcal{A}_o$ and $\mathcal{A}_l$ – necessary to implement $\mathcal{P}_o$ and $\mathcal{P}_l$. For an error parameter ϵ, a probability p of drawing an honest agent in a population of agents, and a threshold t of Byzantine agents, we can calculate the smallest size n such that for a collection of n agents randomly and independently drawn from this population of agents the probability that this collection has more than t is at most ϵ. Thus, by choosing ϵ exponentially small in the security parameter, the probability that this collection does not respect the tolerance level t is negligible. As a concrete example, for $p = 0.68$ and $\epsilon = 10^{-18}$, $t = \lfloor \frac{n-1}{2} \rfloor$ can be achieved with committee size $n = 543$ whereas $t = \lfloor \frac{n-1}{3} \rfloor$ is achieved with $n = 94366$. This demonstrates the significant efficiency gains that can be achieved by our framework. In the context of blockchains, we can implement (financial) incentive structures to encourage even Byzantine agents to participate in the decision of $\mathcal{P}_o$. Note that Byzantine agents in $\mathcal{A}_o$ can prevent $\mathcal{P}_o$ from reaching a decision, but they cannot prevent $\mathcal{P}^H$ from reaching one. Thus, the incentive structure must be designed to make delaying a decision (from $\mathcal{P}_o$ to $\mathcal{P}_l$) not worthwhile.

We summarise our contributions below:

- A *hierarchical consensus* framework that can be used to design efficient agreement protocols by combining a primary (tolerating $< 1/2$ failures) and a secondary (tolerating $< 1/3$ failures) protocol, the former of which does not need to be strictly live.
- A characterisation of primary protocols via the notion of *Justifiable* protocols and an impossibility result that demonstrates some notion of Justifiability is necessary for hierarchical consensus. We additionally show that partially synchronous JRBC protocols must respect $2t_{safe} + t_{live} < n$, and we build the implementation *psync JRBC* that also demonstrates the tightness of this bound.
- A *handover construction* that demonstrates how a JRBC protocol and consensus protocol implementations can be combined to implement a hierarchical consensus protocol.
- A *stochastic analysis* of (randomly sampled) committee sizes for typical primary and secondary protocols that demonstrates the level of efficiency gains (in terms of communication complexity) that can be realised by selecting smaller committees for the primary.

Paper organisation. Section 2 reviews the research literature that is closely related to our paper. In Section 3, we present the necessary background to make the paper self-contained. Section 4 presents our framework for efficient BFT agreement using hierarchical consensus in detail. In Section 5 we show how to instantiate that framework. Finally, we present our concluding remarks in Section 6.

2 Related Work

De la Rocha et al. [12] have proposed a hierarchical architecture for consensus protocols. They focus on running independent, arbitrary consensus instances where the hierarchical relation concerns the instances' trust assumptions. So, their technical contributions are vastly different from ours.

Similar techniques. The work from Bernardo et al. [11] is closest to ours. They also split the tolerance thresholds into t_{safe} and t_{live} to sample committees (called shards) of smaller size, with the same resulting probability of being live. They have a control shard which is large enough to always guarantee liveness and safety, like our secondary protocol. The main difference with our work is that they do not require Justifiability (or anything equivalent), and enjoy the weaker, common restriction of $t_{safe} + 2t_{live} < n$ instead of $2t_{safe} + t_{live} < n$. However, their solution is thereby bound by the impossibility in Theorem 9, and they cannot build a handover-like mechanism. Instead, the output of the smaller shards must always be fed to the control shard, meaning that the large committee runs an agreement protocol *for every finality decision*, even when the optimistic protocol is live. This related work illustrates well how our Justifiability notion is critical to achieving the efficiency benefits of multi-threshold BFT protocols.

In [25, 26], the authors propose handover protocols that are similar in intent to ours but different in design. For instance, while we have the primary triggering the handover, in those papers, the secondary does that, usually via a timeout. Even though the secondary triggers the handover, their system still enjoys the same optimistic behavioural pattern whereby the primary can come to a decision at which point the secondary protocol does not need to run. Also, the communication model employed there is different to ours: while they make use of signals (i.e. single-writer multiple-readers shared memory locations for storing signals), we use point-to-point communication.

Optimistic efficiency. More generally, our work aims to offer a lower communication cost under favourable executions, which falls into the realm of *optimistically efficient* protocols, also known as protocols with adaptive communication cost. Although such protocols have received less interest than optimistically low-latency (e.g., responsive) protocols, there are notable contributions.

The protocol from Gelles and Komargodski [17] has an overall communication cost that is constant in fully honest executions, and a (balanced) $O(n^{\frac{3}{2}})$ otherwise. On the other hand, their protocol is in the unauthenticated, strongly synchronous setting, whereas ours is authenticated, partially synchronous. Cohen et al. [10] proposed a $O(nf)$ algorithm where f is the *actual* number of faults, using threshold signatures. This was subsequently improved upon by Civit et al. [7] and Elsheimy et al. [15] to different variants of consensus and to extend the efficiency gains to the input size. Their protocols are synchronous and tolerate up to a $\frac{1}{2}$ fraction of Byzantine agents. Interestingly, except for our work, there is no optimistically efficient partially synchronous protocol that we know of. In addition, one appeal of our technique is that it can be applied to any committee-sampling protocol, which is orthogonal to the technique employed by other optimistically efficient protocols.

Justifiability concepts. Interestingly, although our Justifiability notion is new, similar concepts can be found in somewhat unrelated parts of the literature. We expect that fully relating those concepts could lead to deeper insights. Abraham et al. [1] proposed a *Binding* property which is almost the same as the existence of a pre-decision, except that it is in the context of Crusader Agreement. Deligios et al. [13] proposed a similar primary/fallback architecture, but dually to our work, the fault thresholds are split depending on network synchronicity instead of liveness/safety properties. They show analogous feasibility and impossibility results for their synchronous/asynchronous fallback architecture. More generally, there are a few other works in other contexts that explore the trade-offs dealing with liveness and Byzantine agreement[16, 5, 21].

3 Background

A distributed protocol $\mathcal{P}$ is an algorithm that makes computational steps and has an interface to receive and send external data, i.e. messages, input/output to the protocol, and time-related information. We analyse distributed protocols involving a set of agents $\mathcal{A}$ that interact by exchanging messages. In the following, we define the network that is used to exchange these messages, the types of faults that we expect from these agents, and some cryptographic assumptions.

Network model. We assume that agents have synchronised clocks, and they communicate through a point-to-point complete partially synchronous network. A partially synchronous network behaves asynchronously – that is, messages can be delayed arbitrarily – until an unknown point in time, called the Global Stabilisation Time, denoted by GST. From this point onwards, the network behaves synchronously, i.e., there is a known bound Δ on the message delay. Messages are always eventually delivered, and never duplicated.

Fault model. Non-faulty agents, namely, agents that follow the protocol, are called honest. Some other agents may present Byzantine faults and behave arbitrarily as a consequence. Our protocols tolerate an adversary that can corrupt agents statically, i.e., at the protocol onset.

Cryptographic assumptions. We rely on an asymmetric digital signature scheme to authenticate messages sent between agents. Moreover, we assume the existence of a public-key infrastructure (PKI) that identifies the cryptographic keys of agents in the protocol. Hence, we may use the identity of an agent in the place of a cryptographic key, and we implicitly assume in our algorithms that all messages are signed and checked for validity. We assume ideal cryptography, namely, that Byzantine agents cannot forge or tamper with digital signatures.

3.1 Agreement protocols

In this paper, we propose and analyse a number of *agreement protocols*. We use the term agreement protocol to broadly describe a distributed protocol where participants interact to select a common output value; this property is also sometimes called *consistency* in the literature. In the following, we precisely define a number of traditional agreement protocols, together with the properties that they must follow. They are later used in our exposition of hierarchical consensus protocols. For all our protocols, we require that both $n = |\mathcal{A}| \geq 2$ and $|\mathcal{V}| \geq 2$, where $|S|$ gives the cardinality of set S, we use $\mathcal{V}$ as the set of decision values of the protocol.

Reliable broadcast protocol (RBC). A reliable broadcast protocol is an agreement protocol where a special participant, called the *leader*, disseminates a value amongst the participating agents. Formally, a protocol $\mathcal{P}$ with participating agents $\mathcal{A}$, a known set of *valid* decision values $\mathcal{V}$, a leader agent $a_L \in \mathcal{A}$, and a leader input value $v_L \in \mathcal{V}$ is *a reliable broadcast protocol* if and only if the following properties hold:

- Safety:
  - Consistency: If two honest agents a and a' output values v and v', respectively, then $v = v'$.
  - Integrity: If the leader is honest, an honest agent cannot output v such that $v \neq v_L$.

- Liveness:
  - Validity: If the leader is honest, an honest agent eventually outputs a value.
  - Totality: If an honest agent outputs a value, all honest agents eventually output a value.

Note that this protocol is consistent, as any agreement protocol must be.

Consensus protocol. We will use consensus as a sub-protocol in our constructions. A consensus protocol is an agreement protocol where agents must eventually output a single agreed-upon value. Formally, a protocol $\mathcal{P}$ with participating agents $\mathcal{A}$, a chosen set of *valid* decision values $\mathcal{V}$, and where each agent $a_i \in \mathcal{A}$ has an input value $v_i \in \mathcal{V}$ is a *consensus protocol* if and only if the following properties hold:

- Safety:
  - Consistency: If two honest agents a and a' output values v and v', respectively, then $v = v'$.
  - Weak Validity: If all agents are honest and have the same input value v, then they can only output v.
- Liveness:
  - Termination: All honest agents eventually output a value.

In this work, we demonstrate how to use a hierarchical architecture to efficiently achieve consensus. Our techniques should be easily generalisable to a wide range of agreement protocols, but for simplicity and presentation purposes, we chose to focus on this weak version of consensus, as it is a versatile building block for other distributed applications and particularly state-machine replication.

Multi-threshold protocols. A protocol $\mathcal{P}$ is said to be t-resilient if it can tolerate up to t Byzantine faults, that is, in the presence of up to t Byzantine agents, the protocol can still satisfy its prescribed properties; t can also be seen as a *fault threshold* for the protocol. In this paper, we make use of the notion of multi-threshold protocols to take advantage of our hierarchical consensus architecture. Multi-threshold protocols have been first investigated by Blum *et al.* [2, 3, 4] and more comprehensively studied by Momose and Ren [22]. A *multi-threshold protocol* $\mathcal{P}$ has two fault thresholds t_{safe} and t_{live} such that, given a categorisation of its properties into Safety and Liveness, it must be the case that:

- $\mathcal{P}$ satisfies *at least* its Safety properties in the presence of at most t_{safe} faults;
- $\mathcal{P}$ satisfies *both* its Safety and Liveness properties in the presence of at most t_{live} faults.

Note that this definition implies that $t_{safe} \geq t_{live}$.

Momose and Ren [22] consider the resilience of the same protocol when executed in a synchronous network (with thresholds t^s_{safe} and t^s_{live}) or a partially synchronous network (with thresholds t^a_{safe} and t^a_{live}). They try to establish relations involving these four thresholds. In this paper, however, we only analyse (and use thresholds related to) partially synchronous protocols.

The usual theorems for Byzantine fault tolerance can be extended to multi-threshold protocols, with the main result being that *partially synchronous* multi-threshold agreement protocols require $t_{safe} + 2t_{live} < n$ [22], even when assuming the existence of digital signatures and a PKI.

Publicly verifiable protocols. The outputs of *publicly verifiable* protocols are endowed with *verifiability*, namely, outputs are intrinsically supported by a publicly verifiable proof. Given a predefined and protocol-specific publicly verifiable predicate $\mathit{Verify}(\pi, v)$ that checks whether π is a valid *proof* for output value v, output verifiability means that an honest agent

outputs v if and only if it has a proof π such that $\mathit{Verify}(\pi, v)$ holds. A consequence of this definition is that the reception of a proof by an honest agent is equivalent to outputting the associated supported value. So, verifiability can be used to achieve a sort of totality by relying on the propagation of a proof.

For instance, a *publicly verifiable reliable broadcast protocol* is defined as being an RBC protocol which has a publicly verifiable function $\mathit{Verify}(\pi, v)$, and, in addition to its Safety and Liveness properties, the following *Verifiability* property:

- Verifiability: An honest agent $a \in \mathcal{A}$ outputs $v \in \mathcal{V}$ if and only if it has a proof π such that $\mathit{Verify}(\pi, v)$.

The equivalence of outputting and obtaining a proof for a value arising from output *verifiability* in combination with the consistency property prevents honest agents from obtaining proofs supporting two different values for the same execution. These proofs allow third parties who do not take part in the protocol to validate its output: inconsistent proofs cannot be produced even by Byzantine agents. Verifiability is more of a *structural property* of the protocol as opposed to a safety property, but we place it in the Safety category for convenience as we expect it to be required for both failure thresholds t_{safe} and t_{live}. A publicly verifiable reliable broadcast protocol must respect the following inequality $t_{safe} + t_{live} < n$ [22], which also implies $t_{live} < \frac{n}{2}$ in our case.

4 Hierarchical Consensus: a framework for scalable BFT agreement

In this section, we present the notion of *hierarchical consensus* (HC). Intuitively speaking, this consensus protocol combines two agreement protocols: one optimistic that does not necessarily reach a decision, and a fallback one that is triggered if the optimistic fails to decide. The optimistic protocol is designed to come to a decision more efficiently than the fallback one. By giving up liveness in a strict sense (i.e., not requiring a decision to be eventually made), we can increase the resilience of our optimistic protocol – i.e., its ability to tolerate faulty agents – and, as a consequence, we can have a protocol that operates with a smaller number of agents if compared to a live consensus protocol as it is the case with the fallback.

4.1 Hierarchical consensus

We start off with a general definition of protocols that follows an hierarchical architecture, and we will refine this definition as needed for our results next. This definition uses Weak Validity, but it can be restated to use other validity notions, if preferred.

▶ **Definition 1.** *A protocol $\mathcal{P}$ with participating agents $\mathcal{A}$, a set $\mathcal{A}_l \subseteq \mathcal{A}$ of* live *agents, a known set of* valid *decision values $\mathcal{V}$, where each agent $a_i \in \mathcal{A}$ has an input value $v_i \in \mathcal{V}$, and a publicly verifiable function $\mathit{Verify}(\pi, v)$ is a* hierarchical consensus protocol *if and only if the following properties hold:*

- *Safety:*
  - *Consistency: If two honest agents $a, a' \in \mathcal{A}$ output values v and v', respectively, then $v = v'$.*
  - *Weak Validity: If all agents in $\mathcal{A}$ are honest and have the same input value v, an honest agent cannot output v' such that $v' \neq v$.*
  - *Verifiability: An honest agent $a \in \mathcal{A}$ outputs v if and only if it has a proof π such that $\mathit{Verify}(\pi, v)$.*
- *Liveness:*
  - *Termination: All honest agents in $\mathcal{A}_l$ eventually output a value.*

The hierarchical nature of this protocol appears in the separation of its agents. One can understand this definition as abstractly relying on two sub-protocols: one implemented by agents $\mathcal{A}_l$ and another implemented by agents in $\mathcal{A}_o = \mathcal{A} \setminus \mathcal{A}_l$, where $X \setminus Y$ denotes set difference between sets X and Y. The agents in $\mathcal{A}_o$ implement a not-necessarily-live (optimistic) sub-protocol, whereas agents in $\mathcal{A}_l$ implement a live (fallback) one, and the overarching definition enforces their safety, both locally (within these sub-protocols) and globally (across these sub-protocols).

If we consider the publicly verifiable version of HC and consensus, we can see that they can both be used to solve the other: agents in $\mathcal{A}_o$ can be guaranteed to terminate by simply waiting for $\mathcal{A}_l$'s output. This (feasibility) equivalence between HC and consensus implies, for instance, that bounds on failure thresholds for one definition must apply to the other. However, this equivalence does not imply that these two definitions must lead to protocols with the same executions, especially if we are interested in optimistic (i.e. best-case) executions. In this paper, we are interested in *optimistic* cases where HC can reach an agreement with less overhead compared to consensus. We are interested in the case where agents $\mathcal{A}_o$ finish of their own accord, as opposed to waiting for $\mathcal{A}_l$'s decision.

4.2 Handover

In this section, we devise a variant of hierarchical consensus where $\mathcal{A}_o$ is expected to output first.

▶ **Definition 2.** *A* handover protocol $\mathcal{P}$ *is a hierarchical consensus protocol with agents* $\mathcal{A}$*, live agents* $\mathcal{A}_l$*, and* $\mathcal{A}_o = \mathcal{A} \setminus \mathcal{A}_l$*, where communication between* $\mathcal{A}_o$ *and* $\mathcal{A}_l$ *is unidirectional from* $\mathcal{A}_o$ *to* $\mathcal{A}_l$*, namely, agents in* $\mathcal{A}_o$ *cannot receive messages from* $\mathcal{A}_l$*.*

We call it a handover protocol because, with the abstract view that $\mathcal{A}_o$ and $\mathcal{A}_l$ implement two sub-protocols, this protocol has to account for the safe handing over of control from $\mathcal{A}_o$ to $\mathcal{A}_l$. This definition allows agents in $\mathcal{A}_o$ to agree on a value and pass this decision on to agents in $\mathcal{A}_l$, at which point they can just propagate the agreed-upon value. However, agents in $\mathcal{A}_l$ must also be able to detect when agents in $\mathcal{A}_o$ are unable to come to a decision, at which point agents in $\mathcal{A}_l$ must take over and decide on a value. The intuitive idea behind this notion is that by lifting the (strict) *liveness* requirement for agents in $\mathcal{A}_o$, they can implement a (sub)protocol that is more efficient than a live one, while relying on a fallback (sub)protocol implemented by agents in $\mathcal{A}_l$ to rescue them when they are unable to come to a decision. One intuitive way to think about the hierarchical consensus and its handover counterpart is: the former is a flexible composition of two sub-protocols, whereas the latter is a type of chaining operator where the outputs of $\mathcal{A}_o$ are used by the agents in $\mathcal{A}_l$ to set their inputs.

4.3 Justifiable protocol

What types of protocols can agents in $\mathcal{A}_o$ implement in order to create a handover protocol? A safe *Justifiable Reliable Broadcast* (JRBC) protocol meets the requirements that we are after for the behaviour of the agents in $\mathcal{A}_o$. The (agents in this) protocol can output a decision, an indecision (i.e. the inability to reach a decision), and even a pre-decision; a guarantee that if the protocol decides, it must be on that pre-decided value. For a set of decision values $\mathcal{V}$, the outputs of agents are captured by the corresponding set of *output values* $\mathcal{O}$ that is a (disjoint) union of decision outputs on $\mathcal{V}$ (i.e. $\{dec.v \mid v \in \mathcal{V}\}$), the

pre-decision outputs on $\mathcal{V}$ (i.e. $\{pre.v \mid v \in \mathcal{V}\}$), and the indecision output $\{\bot\}$. We use a decision (and pre-decision) on v to denote $dec.v$ (and $pre.v$, respectively). We also require public verifiability. The addition of pre-decisions means that, in contrast to usual agreement protocols, Justifiable protocols can make *multiple* outputs. Of course, these outputs must still respect the properties of the protocol, e.g., consistency.

▶ **Definition 3.** *A protocol $\mathcal{P}$ with participating agents $\mathcal{A}$, a known set of* valid *decision values $\mathcal{V}$ and output values $\mathcal{O}$, a special designated* leader *agent $a_L \in \mathcal{A}$ with input value v_L, a publicly verifiable predicate Verify(π, o) for outputs, is a* Justifiable reliable broadcast *if and only if the following properties hold:*

- *Safety:*
  - *Consistency: If honest agents a and a' output decisions on v and v', respectively, then $v = v'$.*
  - *Pre-decision consistency: If honest agent a outputs a pre-decision on v and honest agent a' outputs a pre-decision or decision on v', then $v = v'$.*
  - *Indecision consistency: If honest agent a outputs $\bot$ and honest agent a' outputs o, then either $o = \bot$ or o is a pre-decision.*
  - *Verifiability: An honest agent a makes an output o if and only if it has a proof π such that Verify(π, o). o can be a decision, a pre-decision, or an indecision.*
  - *Integrity: If the leader is honest, an honest agent cannot output a decision on v such that $v \neq v_L$.*
  - *Pre-decision Integrity: If the leader is honest, an honest agent cannot output a pre-decision on v such that $v \neq v_L$.*
  - *Weak Termination: All honest agents eventually output (at least) a value. It can be a pre-decision, a decision, or an indecision.*
- *Liveness:*
  - *Validity: If the leader is honest, an honest agent eventually outputs a decision.*
  - *Totality: If an honest agent outputs a decision, all honest agents eventually output a decision.*

Note that *Indecision consistency and verifiability* implies that an indecision output by any agent is a proof of indecision for the protocol. That is, if an agent outputs an indecision value, it must be that no agent has outputted or will output a (verifiable) decision. In this definition, we have added *weak termination* as part of the Safety category of properties, even though this is, strictly speaking, a liveness property. This is to emphasise that *weak termination* is required to hold up to t_{safe} faults.

The ability to output a pre-decision releases agents from the obligation of reaching a strictly coordinated decision. The addition of pre-decision outputs allows the (safe version of the) protocol to separate executions reaching a potential decision from those that do not, to enable a safe handover, while being compatible with a weaker notion of liveness.

Impossibility theorems. We present two impossibility results that together show that Justifiability is a fundamental concept for handover protocols. First, we present a claim (that is more formally stated as Theorem 9 in Appendix A) that states that for any protocol P that can be used as a primary in a handover, P must satisfy some form of Justifiability.

▷ Claim 4. A handover protocol can only be constructed if the agents in $\mathcal{A}_o$ implement a justifiable protocol.

Furthermore, Justifiability adds a weak liveness requirement *even when* there are up to t_{safe} faults if compared to RBC protocols. Thus, an argument similar to the one demonstrating that a RBC protocol must satisfy $t_{safe} + 2t_{live} < n$ [22] can be applied to show that a JRBC protocol must satisfy $2t_{safe} + t_{live} < n$. As $t_{safe} \geq t_{live}$, the latter relation is strictly more restrictive than the former.

▶ **Theorem 5.** *A partially synchronous multi-threshold JRBC protocol must satisfy* $2t_{safe} + t_{live} < n$.

The proof is essentially the same as in [22], except that one of the executions that is required to be live with t_{live} faults is replaced with an execution that is weakly live with t_{safe} faults.

Proof. We proceed by contradiction. Let us assume that there is a partially synchronous multi-threshold Justifiable RBC protocol $\mathcal{P}$ such that $2t_{safe} + t_{live} = n$; the following proof can be trivially extended to the case $2t_{safe} + t_{live} > n$.

Let S, L_0 and L_1 be sets partitioning the set of agents $\mathcal{A}$, such that $|S| = |L_0| = t_{safe}$, $|L_1| = t_{live}$ and the sender a_L is in S. Let E, E_0 and E_1 be three executions as follows.

In E_0, agents in L_0 crashed and do not send any message, the remaining agents behave honestly, and a_L has input v_0. By weak liveness and t_{safe} faults, honest agents in E_0, including agents in L_1, eventually output either an indecision, a pre-decision on v_0, or a decision on v_0; let T_0 be the time when the last honest agent outputs in E_0. E_1 is the same as E_0 but with L_1 crashed instead of L_0 and a_L input being v_1 such that $v_1 \neq v_0$. By liveness and t_{live} faults, honest agents in E_1, including agents in L_0, eventually output a decision for v_1; let T_1 be the time when the last honest agent outputs in E_1.

In E, agents in S are Byzantine while the remaining agents behave correctly. Messages between L_0 and L_1 are not delivered until $GST = max(T_0, T_1)$, and agents in S behave towards each L_i for $i \in \{0, 1\}$ as in E_{1-i}, with the same respective network delays. Thus, agents in L_i cannot distinguish between E_{1-i} and S until GST, and so they must output as per E_{1-i}. So, agents in L_0 must output a decision on v_1 as per E_1, whereas agents in L_1 must output either an indecision, a pre-decision on v_0, or a decision on v_0 as per E_0. If an agent in L_1 outputs an indecision, we have a violation of Indecision consistency. Similarly, the output of a pre-decision (decision) on v_0 by an agent in L_1, would violate Pre-decision (Decision, respectively) consistency. Hence, $\mathcal{P}$ cannot be a JRBC protocol. ◀

Our proposed implementation for a partially synchronous JRBC, presented later, shows that this bound is tight. Given this bound, we have that when t_{safe} goes from $\frac{n}{3}$ to $\frac{n}{2}$, our JRBC protocol is limited to t_{live} going from $\frac{n}{3}$ to 0 whereas non-justifiable ones may have t_{live} going from $\frac{n}{3}$ to $\frac{n}{2}$ (and t_{safe} go above $\frac{n}{2}$ as well).

The notion of justifiability can also be applied to consensus protocols. Similarly to JRBC, Justifiable consensus protocols would have the three types of outputs (decisions, pre-decision, and indecision), the associated consistency, validity, verifiability, and weak termination notions. Unlike JRBC, it does not have an agent with the role of a leader and the properties associated with this role (e.g., integrity, totality, and termination based on leader honesty).

A Justifiable reliable broadcast can be constructed from a Justifiable consensus protocol in the typical way: the leader value is used to set up the initial values of the agents in the Justifiable consensus, and their consensus outputs become outputs for the Justifiable broadcast protocol construction. So, any impossibility for Justifiable reliable protocols must apply to Justifiable consensus protocols too.

5 Instantiating a hierarchical consensus protocol

5.1 Handover construction

In the following, we demonstrate a construction that creates a handover protocol by combining a JRBC protocol $\mathcal{P}_o$ with a consensus one $\mathcal{P}_l$. Intuitively speaking, the agents implementing $\mathcal{P}_l$ wait for the agents $\mathcal{P}_o$ to come to a decision. If they are unable to do so, the agents in $\mathcal{P}_l$ take over and decide. The intricate element in implementing this behaviour is ensuring that the handing over from agents in $\mathcal{P}_o$ to agents in $\mathcal{P}_l$ is safe.

▶ **Definition 6.** *Given a multi-threshold JRBC protocol $\mathcal{P}_o$ and a single-threshold consensus protocol $\mathcal{P}_l$, one can construct the handover protocol $\mathcal{P}^H(\mathcal{P}_l, \mathcal{P}_o)$ with agents $\mathcal{A}$, live agents $\mathcal{A}_l$ and remaining agents $\mathcal{A}_o = \mathcal{A} \setminus \mathcal{A}_l$ as follows:*

- *Agent $A_o \in \mathcal{A}_o$ runs $\mathcal{P}_o$ to obtain output o_o and proof π_o and then:*
  - *if o_o is a decision, it outputs o_o and sends o_o and associated proof π_o to all agents in $\mathcal{A}_l$;*
  - *if o_o is a pre-decision or indecision, it sends o_o and associated proof π_o to all agents in $\mathcal{A}_l$.*
- *The decision set for $\mathcal{P}_l$ is given by $\mathcal{V}_l = \{v \in \mathcal{V}_o \mid \mathit{Verify}(\pi_o, \mathit{pre}.v)\} \cup \{v \in \mathcal{V}_o \mid \mathit{Verify}(\pi_o, \bot)\}$, where $\mathcal{V}_o$ is the decision set of $\mathcal{P}_o$.*
- *An agent in $a_l \in \mathcal{A}_l$ behaves as follows:*
  - *Upon receiving a verifiable decision, it outputs this value and propagates it to agents in $\mathcal{A}_l$;*
  - *Upon receiving a verifiable pre-decision on v, it starts to behave as per $\mathcal{P}_l$ with input value v.*
  - *Upon receiving an indecision, it starts to behave as per $\mathcal{P}_l$ with input value v_i.*
  - *Upon $\mathcal{P}_l$ outputting some value v, output v.*

Note that the decision set $\mathcal{V}_l$ for $\mathcal{P}_l$ depends on the verifiable outputs of $\mathcal{P}_o$, namely, $\mathcal{P}_l$ can only decide on values that have been supported by the outputs of $\mathcal{P}_o$ as per $\mathcal{V}_l$. For instance, a proof of indecision allows agents in $\mathcal{P}_l$ to decide on any decision value in $\mathcal{V}_o$, whereas a pre-decision proof for v supports only the decision value v. To enforce that a decision value v for $\mathcal{P}_l$ conforms to $\mathcal{V}_l$, we can require that it be accompanied by a supporting proof π_o from $\mathcal{P}_o$. We omit this step from our construction for conciseness. Also, note that if $\mathcal{P}_l$ is publicly verifiable, we can make $\mathcal{P}^H$ publicly verifiable as well by outputting the proof π generated by $\mathcal{P}_l$.

The protocol $\mathcal{P}^H$ allows the agents $\mathcal{A}_o$ to decide and simply propagate their decision to the agents in $\mathcal{A}_l$; this sort of *optimal* behaviour is the main motivation for proposing handover protocols. Of course, in the worst-case scenario, if the agents in $\mathcal{A}_o$ are unable to reach a decision, the consensus protocol $\mathcal{P}_l$ has to run in addition to $\mathcal{P}_o$. Note that the protocol $\mathcal{P}^H$ does not output the indecision or pre-decisions output by $\mathcal{P}_o$; it merely uses them internally as part of the handover mechanism we devise. Thus, only decisions are required to be publicly verifiable. Also, for convenience, we have that the execution of $\mathcal{P}_o$ by an agent returns an output and a proof.

We show that $\mathcal{P}^H$ is indeed a handover protocol next.

▶ **Theorem 7.** *For a* safe *Justifiable reliable broadcast protocol $\mathcal{P}_o$ with safety threshold t^w_{safe} and a consensus protocol $\mathcal{P}_l$ with liveness threshold t^l_{live}, the protocol $\mathcal{P}^H(\mathcal{P}_l, \mathcal{P}_o)$ is a handover protocol.*

Proof. The messages sent between $\mathcal{A}_o$ and $\mathcal{A}_l$ flow only from the former to the latter. So, we only need to demonstrate that $\mathcal{P}^H(\mathcal{P}_l, \mathcal{P}_o)$ satisfies all the properties of a hierarchical consensus protocol.

- Consistency: $\mathcal{P}_o$ and $\mathcal{P}_l$ enforce local consistency, namely, no two agents within these protocols can decide on different values. The definition of $\mathcal{V}_l$ predicated on the outputs of $\mathcal{P}_o$ ensures inter-protocol consistency. Let us assume that an agent $a_o \in \mathcal{A}_o$ outputs a decision on v'. By the consistency properties of $\mathcal{P}_o$, it must be that all agents in $\mathcal{A}_o$ output either a pre-decision or a decision on that value. Hence, the decision set $\mathcal{V}_l$ for agents in $\mathcal{A}_l$ can contain only the value v'. Therefore, $\mathcal{P}_l$ can only output v'.
- Weak Validity: If all the agents in $\mathcal{A}_o$ are honest and have the same input value v', if they output a pre-decision or decision, it must be on v', by their integrity properties. So, the agents in $\mathcal{A}_l$ receive from agents in $\mathcal{A}_l$ enough messages to support v' as their initial value or enough indecision so they use their own initial value, which is also v'. By $\mathcal{P}_l$ own notion of validity, we have that agents in $\mathcal{A}_l$ can only output v' too.
- Verifiability: Decisions of $\mathcal{P}^H$ are verifiable by relying on the publicly verifiable functions of the underlying protocols $\mathcal{P}_o$ and $\mathcal{P}_l$.
- Termination: If an agent in $\mathcal{A}_o$ outputs a decision, it will eventually be propagated to all agents in $\mathcal{A}$. Otherwise, by weak termination of $\mathcal{P}_o$, each honest agent in $\mathcal{A}_l$ is guaranteed to eventually receive either a pre-decision or an indecision (with accompanying proof) from agents in $\mathcal{A}_o$. Therefore, all honest agents in $\mathcal{A}_l$ will eventually run $\mathcal{P}_l$ with some input value. By termination of $\mathcal{P}_l$, they all eventually output the value from $\mathcal{P}_l$. ◀

Note that we do not use the (strong) liveness property of JRBC to show Termination for $\mathcal{P}^H$. This property is proven using only JRBC's weak liveness. On the other hand, JRBC's Liveness guarantees that *when* t_{live} is met, the optimistic protocol will succeed, provided that the timeout is large enough.

To complete the construction of a handover protocol, we propose a partially synchronous JRBC protocol implementation; many well-known consensus protocol implementations exist, of course.

5.2 Partially synchronous JRBC (psync-JRBC)

Algorithm 1 depicts the behaviour of an honest agent for a JRBC protocol, which we call *psync-JRBC*. This protocol behaves similarly to a traditional RBC protocol, but it relies on a timeout to ensure agents output a value even if they are yet unaware of a decision being reached. The timeout should be tuned to the network delay as best as possible, but note that its value only impacts performance.

Psync-JRBC is implemented in two rounds of voting to ratify the proposal of the leader; the message $(\textsc{Proposal}, v)$ denotes the proposal of value v. In the first round, agents use message $(\textsc{Prepare}, v)$ to ratify the value v. When an agent receives a quorum of containing more than half of distinct honest agents Prepare-ratifying the same value v, it has obtained a *pre-decision* for v; these Prepare messages form a proof for this pre-decision. Reaching this quorum (before a timeout) triggers the second round of voting, which relies on message $(\textsc{Commit}, v)$ to ratify v the pre-decided value v. A quorum including $2t_{safe}+1$ honest agents Commit-ratifying the same value v means that a decision has been reached; these Commit messages represent a proof for a decision. If the timeout happens before a pre-decision is obtained, the agent aborts by sending an Abort message. A quorum of $n - 2t_{safe}$ abort messages from distinct honest agents represents a proof of indecision. If the timeout happens

Algorithm 1 Implementation of protocol *psync JRBC*.

local variables:
$\mathcal{A}$. The protocol agents.
v_i. The agent input.
a_i. The agent identity.
a_L. The leader identity.
timer. An object that expires after some timeout duration.
procedure *start()*
  start *timer*;
  if $a_i = a_L$ **then**
    send (PROPOSAL, v_i) to $\mathcal{A}$;
end
upon *receiving* (PROPOSAL, v) *from* a_L **do**
  if (PREPARE, _) *not sent* **then**
    send (PREPARE, v) to $\mathcal{A}$;
end
upon *receiving* (PREPARE, v) *from* $T_p = \left\lceil \frac{t_{safe}+n+1}{2} \right\rceil$ *distinct senders* **do**
  if *timer not expired* **then**
    send (COMMIT, v) to $\mathcal{A}$;
end
upon *receiving* (COMMIT, v) *from* $T_d = 2t_{safe}+1$ *distinct senders* **do**
  send ($dec.v, \pi_d$) to $\mathcal{A}$;
  output $dec.v$;
end
upon *receiving* (ABORT) *from* $T_a = n - t_{safe}$ *distinct senders* **do**
  send ($\perp, \pi_i$) to $\mathcal{A}$;
  output $\perp$;
end
upon *timer expires* **do**
  if (COMMIT, _) *not sent* **then**
    send ABORT to $\mathcal{A}$;
  if (COMMIT, v) *was sent for some* v **then**
    send ($pre.v, \pi_p$) to $\mathcal{A}$;
    output $pre.v$;
  end
end
upon *receiving* (o, π) *where Verify*(π, o) **do**
  send (o, π) to $\mathcal{A}$;
  output o;
end

after a pre-decision, the agent outputs the pre-decision. The agent outputs a decision or an indecision as soon as it has obtained a proof for it. We assume that signature creation and verification are being carried out as messages are sent and received.

▶ **Theorem 8.** *The protocol psync-JRBC is a JRBC protocol with thresholds* $t_{safe} \in \left[0, \left\lfloor \frac{n-1}{2} \right\rfloor\right]$ *and* $t_{live} = \min(\{n - T_p, n - T_d, t_{safe}\})$*, where* min *returns the minimal element of the set.*

Proof. We show that psync-JRBC satisfies all Safety properties of a JRBC protocol for $t_{safe} \in \left[0, \left\lfloor \frac{n-1}{2} \right\rfloor\right]$ and both Safety and Liveness properties for $t_{live} = \min(\{n - T_p, n - T_d, t_{safe}\})$.

- Consistency: We prove this by contradiction. So, let us assume that two honest agents output decisions on v and v' such that $v \neq v'$. For each of these decisions, at least one honest agent must have COMMIT for each of these values. Let us call them a and a' for values v and v', respectively. Thus, a and a' must have obtained pre-decisions for v and v', respectively. However, a pre-decision requires a quorum of $T_p = \left\lceil \frac{t_{safe}+n+1}{2} \right\rceil$ agents, namely, it requires the participation of more than half of the honest agents. Hence, by a quorum intersection argument, it must be that an honest agent has sent a PREPARE message for both values v and v', which contradicts our algorithm.
- Pre-decision consistency: We can use the pre-decision quorum intersection argument again to show here that two pre-decisions cannot be obtained for two distinct values. We can prove the other case, i.e., that a decision and a pre-decision output by honest agents cannot support different values, using a similar argument. Let us assume that agent a has obtained a pre-decision for v and that agent a' has obtained a decision for v'. The quorum required for a decision must include at least one honest agent; let us call this agent a^*. This agent must have, then, obtained a pre-decision for v'. That contradicts the fact that two pre-decisions for distinct values cannot be obtained for the same run of the protocol.
- Indecision consistency: We prove this by contradiction. Let us assume that agent a outputs $\perp$ and agent a' outputs a decision on v. By our algorithm, for a to output $\perp$, it must be that $n - 2t_{safe}$ distinct honest agents have aborted. For the decision on v, by

our algorithm, there must have been $t_{safe}+1$ distinct honest agents who have obtained a pre-decision for v. There are $T_d + T_a = (n - t_{safe} + 2t_{safe} + 1 - n) = t_{safe} + 1$ overlapping agents in the intersection of these two quorums, but an honest agent cannot both abort and possess a pre-decision, a contradiction.

- Decision, Pre-decision, and indecision verifiability: A proof of a decision is a collection of $T_d = 2t_{safe}+1$ COMMIT messages from distinct agents; a proof of a pre-decision is a set of $T_p = \left\lceil \frac{t_{safe}+n+1}{2} \right\rceil$ PREPARE messages from distinct agents; and a proof of indecision is a set of $T_a = n - t_{safe}$ ABORT messages from distinct agents. Given that $t_{live} \leq t_{safe}$, to satisfy the public verifiability requirement of $t_{safe} + t_{live} < n$, it must be that $t_{safe} \in \left[0, \left\lfloor \frac{n-1}{2} \right\rfloor\right]$.
- Integrity: If the leader a_L is honest, honest agents can only send PREPARE messages for v_L. Given that a pre-decision quorum requires the participation of honest agents, a pre-decision can only be for v_L and, as a consequence, a decision can only be for v_L.
- Pre-decision integrity: As per the previous property.
- Weak termination: The timer of honest agents will eventually expire. When that happens, two cases can arise. If an honest agent has obtained a pre-decision, it will propagate this pre-decision, eventually compelling all honest agents to output this pre-decision. If no honest agent has obtained a pre-decision, they will all send abort messages. Given that there are $n - t_{safe}$ honest agents, they can eventually obtain an abort proof and output $\perp$.
- Termination: Our requirement $t_{live} = \min(\{n - T_p, n - T_d, t_{safe}\})$ means that we have enough honest agents to meet the quorum for a pre-decision and a decision. Hence, if the leader is honest and the timer lasts for enough time, the algorithm will converge after GST to a decision on the leader's input value. ◀

5.3 Putting it all together

With our handover construction $\mathcal{P}^H$ on Def. 6 and our psync-JRBC protocol in Alg. 1 used as $\mathcal{P}_o$, we have a protocol that relies on the creation of committees for $\mathcal{A}_o$ and $\mathcal{A}_l$ with the appropriate failure thresholds. The benefit of our approach is apparent when these collections of agents are obtained via *random sampling with repetition*, i.e., when they are selected randomly and independently from an underlying (larger) population of agents. The precise mechanism by which agents are sampled is irrelevant to our construction, as long as each agent has a probability $p > \frac{2}{3}$ to be honest.

Stochastic reasoning. The higher fault tolerance for JRBC (i.e. primary) protocols allows them to rely on small committees if compared to typical consensus protocols. In this section, we dive into a probabilistic approach to choose committees to implement (i.e. run) our *deterministic* protocols. That does not make our protocols stochastic, but that means that our correctness guarantees become probabilistic. We rely on a notion of *stochastic reasoning* to bound the probability with which correctness may be violated. We use the error parameter ϵ as the upper bound on correctness violation, which should be negligible in the security parameter. In our context, a violation arises if we select a committee with the wrong proportion of honest agents. This error parameter can be understood as the probability with which we will select a committee with the wrong proportion of honest agents.

We select agents for our committees from a population of agents with a probability p of being honest. We rely on a process that selects randomly and independently (with repetition) n agents from this population. The probability that this set has k honest agents is governed by a binomial distribution with parameters (n, p). The probability that this collection has up to k

honest agents is equal to the cumulative distribution $F(k;n,p) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$. This probability is equivalent to the probability of the collection having at least $n-k$ Byzantine agents, and so the probability that it has at most $n-k-1$ Byzantine agents is $1-F(k;n,p)$. So, by minimising $F(k;n,p)$, we increase the probability that $n-k-1$ bounds the number of Byzantine agents in this collection, and $F(k;n,p)$ can be seen as the probability that $n-k-1$ is not a correct bound. We use $K(n,p,\epsilon) = max(\{k \in \{0 \dots n\} \mid F(k;n,p) \leq \epsilon\})$ to find the k, if it exists, that represents the tightest $k+1$ lower bound to the number of honest agents in the committee, respecting the error probability ϵ. For a given probability p of agents to be honest, a target maximum number of Byzantine fault t and error threshold ϵ, we compute the committee size $n^\epsilon(p,t)$ as the smallest number, if it exists, that guarantees less than t faults, i.e., $n^\epsilon(p,t) = \min(\{n \in \mathbb{N} \mid k = K(n,p,\epsilon) \wedge n-k-1 \leq t\})$.

In Table 1, we illustrate how the ability to tolerate more Byzantine agents can lead to smaller committee sizes. We present the values of $n^\epsilon(p,t)$ for some values of p and ϵ and for failure thresholds $t = \lfloor \frac{n-1}{2} \rfloor$ and $t = \lfloor \frac{n-1}{3} \rfloor$; we choose 0.68 as a starting value for p because of the need for a 2/3 ratio of honest agents to implement consensus. These values demonstrate the substantial impact that tolerating more failures can have in the size of the committees being randomly selected. For instance, for $p = 0.68$ and $\epsilon = 10^{-14}$, a primary protocol would require a committee of size 411, whereas the secondary protocol would require the participation of 72061 agents; the secondary committee size would be about 17500% that of the primary. As expected, the discrepancy between committee sizes reduces as the probability of an agent being honest increases.

Communication cost. We examine the communication complexity of handover protocols, i.e. the number of bits sent by honest agents. We note $\mathcal{C}^P(n)$ the communication cost of protocol P as a function of the number of participating agents. It is straightforward to see that $\mathcal{C}^{\mathcal{P}^H}(|\mathcal{A}|) = \mathcal{C}^{\mathcal{P}_o}(|\mathcal{A}_o|) + \mathcal{C}^{\mathcal{P}_l}(|\mathcal{A}_l|) + O(|\mathcal{A}_o||\mathcal{A}_l|)$ in general, and for optimistic executions $\mathcal{C}^{\mathcal{P}^H}(|\mathcal{A}|) = \mathcal{C}^{\mathcal{P}_o}(|\mathcal{A}_o|) + O(|\mathcal{A}_o||\mathcal{A}_l|)$. This shows that there are gains to be expected from the optimistic executions, although the handover can, in some situations, incur a cost higher than just running the fallback. More generally, since our technique yields a constant factor improvement in the committee size, we cannot analyse them using asymptotic notation. Depending on the protocols at hand, tail bounds on the binomial distribution (e.g., the Chernoff bound) may be used instead. Increasing t_{safe} for $\mathcal{P}_o$ makes $|\mathcal{A}_o|$ smaller, at the expense of decreasing t_{live} and therefore the probability of making $\mathcal{P}_o$ (strongly) live.

We intended to use our hierarchical protocol as a consensus engine for blockchains. In this context, we could set up incentive structures that can be naturally implemented in these distributed systems to encourage optimistic executions. For instance, they could financially reward more favourably these executions against fallback ones. These incentive structures

Table 1 Tables depicting the committee size $n^\epsilon(p,t)$ for different values of p and ϵ contrasting the size requirements for committees tolerating $\lfloor \frac{n-1}{2} \rfloor$ and those tolerating $\lfloor \frac{n-1}{3} \rfloor$.

$\epsilon = 10^{-10}$		
p	$\lfloor \frac{n-1}{2} \rfloor$	$\lfloor \frac{n-1}{3} \rfloor$
0.68	281	49795
0.72	179	2944
0.76	123	901
0.8	87	403
0.84	61	214
0.88	45	124
0.92	31	70

$\epsilon = 10^{-14}$		
p	$\lfloor \frac{n-1}{2} \rfloor$	$\lfloor \frac{n-1}{3} \rfloor$
0.68	411	72061
0.72	265	4276
0.76	179	1303
0.8	125	592
0.84	89	313
0.88	65	181
0.92	45	106

$\epsilon = 10^{-18}$		
p	$\lfloor \frac{n-1}{2} \rfloor$	$\lfloor \frac{n-1}{3} \rfloor$
0.68	543	94366
0.72	351	5608
0.76	237	1720
0.8	167	769
0.84	121	415
0.88	87	238
0.92	59	142

can be used to optimise another aspect of our construction. Note that when $t_{safe} > t_{live}$ for $\mathcal{P}_o$, it only takes a small number of Byzantine agents ($n - 2t_{safe}$) to prevent the primary from reaching a decision. For instance, for $n = 2t_{safe} + 1$, it only takes one Byzantine agent (being silent, for example) to prevent $\mathcal{P}_o$ from being live. Of course, weak liveness still holds for the primary, so the Byzantine agents in $\mathcal{P}_o$ can delay a $\mathcal{P}^H$ decision, but they cannot prevent it. Thus, once more, incentive structures can be set up to ensure that delaying a decision is not worthwhile, encouraging, then, the Byzantine agents in $\mathcal{A}_o$ to cooperate in reaching a $\mathcal{P}_o$ decision.

6 Conclusion

In this paper, we introduce the notion of *hierarchical consensus* as a framework to create efficient BFT consensus protocols by combining an optimistic efficient primary sub-protocol that is safe and only weakly live with a fallback (not as efficient) secondary protocol that is safe and live. Giving away (strict) liveness is the price paid for the efficiency of the primary protocol.

We propose the notion of a *Justifiable RBC* protocol, and *justifiability* more generally, as a way to characterise primary sub-protocols: it captures the safety and weak liveness requirements necessary for a safe handover to the secondary sub-protocol. We show that for a Justifiable RBC protocol, it must be that $2t_{safe} + t_{live} < n$; this inequality should be a useful guide for designers of such protocols. Moreover, we devise a JRBC implementation that shows that this is a tight bound.

We also introduce the *handover construction* $\mathcal{P}^H$ that creates a hierarchical consensus protocol by combining a Justifiable RBC protocol $\mathcal{P}_o$ and a consensus protocol $\mathcal{P}_l$. This construction can combine our *psync JRBC* implementation with a consensus implementation of choice to create a hierarchical consensus implementation. We show that JRBC protocols can tolerate up to $\lfloor \frac{n-1}{2} \rfloor$ faults while remaining safe and weakly live as opposed to the $\lfloor \frac{n-1}{3} \rfloor$ failure tolerance level for live consensus protocols; this difference is crucial to achieve efficiency.

We employ a notion of *stochastic reasoning* to demonstrate the impact that these levels of tolerance have on the size of committees implementing $\mathcal{P}_o$ and $\mathcal{P}_l$. Our calculation show that, for a very small ϵ and p close to 0.68 (i.e. the probability of drawing an honest agent is close for 2/3), the size for a committee where $t = \lfloor \frac{n-1}{2} \rfloor$ is in the hundreds of agents, whereas for $t = \lfloor \frac{n-1}{3} \rfloor$ it is usually in the thousands. This demonstrates the significant efficiency gains that can be achieved by our framework.

Future work. The area of hierarchical consensus seems very rich and not yet well explored. We intend to analyse the handover mechanisms in [25, 26, 24] within our framework. Additionally, while we provide a single instantiation of our framework, we believe there are many more that could be of interest. One could consider a handover to another optimistic protocol instead of a secondary. This could be retried some number of times or indefinitely, depending on the efficiency of the various protocols involved. Similarly, so far we have only instantiated a Justifiable Reliable Broadcast, whereas Justifiability could be added to consensus and its variants, e.g., to avoid the reliance on leaders. Going further, we plan to enhance the modularity of our framework by making the notion of Justifiability generically applicable to any agreement protocol, as well as to remove any specific notion of Validity for consensus and hierarchical consensus, instead relying on the insights from Civit et al. [9]. Adding to the theoretical exploration of Justifiability, we expect to complete its characterisation with

tighter feasibility results, and to show to possibility of compiling any agreement protocol into a Justifiable one.

Other avenues of research include taking full advantage of multi-threshold protocols for more resilient primary and/or secondary, as well as better analysis to determine the best trade-off between t_{safe} and the probability to make the protocol live, depending on the protocols at hand. In the context of using our hierarchical protocol as a consensus engine for blockchains, we plan to more formally analyse the incentive structures that can be put in place to encourage optimistic executions. Our architecture is particularly suited to this because we can readily detect optimistic executions simply by observing the provenance of the output.

References

1 Ittai Abraham, Naama Ben-David, and Sravya Yandamuri. Efficient and adaptively secure asynchronous binary agreement via binding crusader agreement. In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 381–391. ACM, 2022. doi:10.1145/3519270.3538426.

2 Erica Blum, Jonathan Katz, and Julian Loss. Synchronous consensus with optimal asynchronous fallback guarantees. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 131–150. Springer, 2019. doi:10.1007/978-3-030-36030-6_6.

3 Erica Blum, Jonathan Katz, and Julian Loss. Tardigrade: An atomic broadcast protocol for arbitrary network conditions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 547–572. Springer, 2021. doi:10.1007/978-3-030-92075-3_19.

4 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 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 707–731. Springer, 2020. doi:10.1007/978-3-030-56880-1_25.

5 Manuel Bravo, Gregory V. Chockler, and Alexey Gotsman. Making byzantine consensus live. *Distributed Comput.*, 35(6):503–532, 2022. doi:10.1007/S00446-022-00432-Y.

6 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002. doi:10.1145/571637.571640.

7 Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. DARE to agree: Byzantine agreement with optimal resilience and adaptive communication. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*, pages 145–156. ACM, 2024. doi:10.1145/3662158.3662792.

8 Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, Anton Paramonov, and Manuel Vidigueira. All byzantine agreement problems are expensive. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*, pages 157–169. ACM, 2024. doi:10.1145/3662158.3662780.

9 Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. On the validity of consensus. In Rotem Oshman, Alexandre Nolin, Magnús M. Halldórsson, and Alkida Balliu, editors, *Proceedings of the 2023 ACM Symposium on Principles of Distributed*

Computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023, pages 332–343. ACM, 2023. doi:10.1145/3583668.3594567.

10 Shir Cohen, Idit Keidar, and Alexander Spiegelman. Make every word count: Adaptive byzantine agreement with fewer words. In Eshcar Hillel, Roberto Palmieri, and Etienne Rivière, editors, *26th International Conference on Principles of Distributed Systems, OPODIS 2022, December 13-15, 2022, Brussels, Belgium*, volume 253 of *LIPIcs*, pages 18:1–18:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.OPODIS.2022.18.

11 Bernardo David, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 683–696. ACM, 2022. doi:10.1145/3548606.3559375.

12 Alfonso de la Rocha, Lefteris Kokoris-Kogias, Jorge M. Soares, and Marko Vukolic. Hierarchical consensus: A horizontal scaling framework for blockchains. In *42nd IEEE International Conference on Distributed Computing Systems, ICDCS Workshops, Bologna, Italy, July 10, 2022*, pages 45–52. IEEE, 2022. doi:10.1109/ICDCSW56584.2022.00018.

13 Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 623–653. Springer, 2021. doi:10.1007/978-3-030-90459-3_21.

14 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988. doi:10.1145/42282.42283.

15 Fatima Elsheimy, Giorgos Tsimos, and Charalampos Papamanthou. Deterministic byzantine agreement with adaptive $O(n \cdot f)$ communication. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1120–1146. SIAM, 2024. doi:10.1137/1.9781611977912.43.

16 Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. *ArXiv*, abs/2106.10362, 2021. arXiv:2106.10362.

17 Yuval Gelles and Ilan Komargodski. Scalable agreement protocols with optimal optimistic efficiency. In *Security and Cryptography for Networks - 14th International Conference, SCN 2024, Amalfi, Italy, September 11-13, 2024, Proceedings, Part I*, volume 14973 of *Lecture Notes in Computer Science*, pages 297–319. Springer, 2024. doi:10.1007/978-3-031-71070-4_14.

18 Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement. *Math. Syst. Theory*, 26(1):41–102, 1993. doi:10.1007/BF01187074.

19 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889, 2016. URL: https://eprint.iacr.org/2016/889.

20 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

21 Andrew Lewis-Pye, Joachim Neu, Tim Roughgarden, and Luca Zanolini. Accountable liveness. *arXiv preprint arXiv:2504.12218*, 2025.

22 Atsuki Momose and Ling Ren. Multi-threshold byzantine fault tolerance. In *CCS '21, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1686–1699. ACM, 2021. doi:10.1145/3460120.3484554.

23 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

24 A. W. Roscoe. *Understanding Decentralised Systems: Fundamentals and Blockchain*. Springer, 2026 (to appear).

25 A. W. Roscoe, Pedro Antonino, and Jonathan Lawrence. *The Consensus Machine: Formalising Consensus in the Presence of Malign Agents*, pages 136–162. Springer Nature Switzerland, Cham, 2023. doi:10.1007/978-3-031-40436-8_6.

26 A. W. Roscoe, Pedro Antonino, and Jonathan Lawrence. *Abstracting and Verifying Decentralised Systems in CSP*, pages 172–202. Springer Nature Switzerland, Cham, 2024. doi:10.1007/978-3-031-67114-2_8.

27 G. Wood. Ethereum: A secure decentralised generalised transaction ledger. http://gavwood.com/Paper.pdf.

A Impossibility Theorem for Justifiability

Before stating our proof, we add some slight precisions to our execution model. An execution E of the protocol P is represented by a potentially infinite *list* of *events*. Events in E are accessed by an index $i \in \mathbb{N}$, that we also call *point* or *instant*. An event may be one of the following: sending/reception of a message by some agent (Byzantine or not), local input/output to P by some honest agent, and the emission of a tick event measuring the passage of time. We say that E can be *completed* into another execution E' at a point $i \in \mathbb{N}$ if E and E' are both executions of P and for $k \leq i$, their kth events are equal. We call an execution *terminating* if some honest agent makes an output. We say that a pre-decision exists in an execution E, if there is a point i in that execution where all completions of E from i that terminate decide the same value v_E.

▶ **Theorem 9.** *Let H^P be a protocol that takes as a parameter an input protocol P, and the nodes in $\mathcal{A}_w$ must only execute P. P is also allowed to send external messages, e.g., to agents in A_l, and those messages are implicitly sent to A_w as well. If H^P is a handover protocol, then we can build a protocol P' that is exactly P with the addition of local output, such that P' must eventually output a proof that there exists a pre-decision value.*

Proof. Without loss of generality, we will consider H^P to be a protocol with a single agent in $\mathcal{A}_l$ that is always honest. This is possible because, if there cannot be a handover from P using a trusted party, then no handover protocol such as H^P can exist. We name the agent $a_l \in \mathcal{A}_l$.

Assume that there is an execution X of P, where:

1. No honest agent ever makes an output in X.
2. At any point i in X, X can be completed into two other executions X_0^i and X_1^i, such that in X_0^i and X_1^i, all honest agents outputs v_0 (respectively, v_1), and $v_0 \neq v_1$.

Then, in an execution of H^P where P is running X, H^P must eventually output a value v_h at some point i in X. At that point, X can be completed into X_j^i or X_j^i, with $j \in \{0, 1\}$ such that $v_h \neq v_j$. This is a violation of Consistency for H^P, therefore such X does not exist.

The non-existence of X can be precisely stated as follows: For all executions E of P, either

1. Some honest agent a eventually makes an output in E.
2. There is a point i in E, such that for all pairs of completions of E E_0^i and E_1^i, either one of those do not make an output, or they output the same value v.

This can be simplified to say that the following statement holds: for all of P's executions E, if E does not terminate, then there is a point i in E such that all completions of E from i that do terminate must all output the same value noted v_E, i.e. all non terminating executions have a pre-decision. Because all terminating executions have a pre-decision, we conclude that all executions have a pre-decision.

We have shown that P must reach a state where a pre-decision exists, but we have not shown that honest nodes must know when that has happened. We do so below.

Consider an execution E of P, and let i be the earliest instant (i.e. the smallest index) in E when there is a pre-decision noted v_E. Because of Liveness, a_l have to eventually output a value v, let j be the instant when that happens. If $j < i$, then there is either no pre-decision at that instant, or there is a completion from j with a different decision. Both statements imply that there is an additional completion of E from j that output a different values $v' \neq v_E$. Either $v' \neq v$ or $v_E \neq v$, hence one of those executions creates a contradiction with consistency. Therefore, $j \geq i$.

In other words, for all executions E, a_l will eventually know that a pre-decision exists. Building P' is therefore trivial: Run a_l's code along with P. Whenever a_l outputs a value, we are guaranteed that a pre-decision exists. The messages fed to a_l's code constitute the proof of this fact, ◀