Quorum Systems in Permissionless Networks

Christian Cachin ✉ 🆔
University of Bern, Switzerland

Giuliano Losa ✉ 🆔
Stellar Development Foundation, San Francisco, CA, USA

Luca Zanolini ✉ 🆔
University of Bern, Switzerland

──── **Abstract** ────

Fail-prone systems, and their quorum systems, are useful tools for the design of distributed algorithms. However, fail-prone systems as studied so far require every process to know the full system membership in order to guarantee safety through globally intersecting quorums. Thus, they are of little help in an open, permissionless setting, where such knowledge may not be available. We propose to generalize the theory of fail-prone systems to make it applicable to permissionless systems. We do so by enabling processes not only to make assumptions about failures, but also to make assumptions about the assumptions of other processes. Thus, by transitivity, processes that do not even know of any common process may nevertheless have intersecting quorums and solve, for example, reliable broadcast. Our model generalizes existing models such as the classic fail-prone system model [Malkhi and Reiter, 1998] and the asymmetric fail-prone system model [Cachin and Tackmann, OPODIS 2019]. Moreover, it gives a characterization with standard formalism of the model used by the Stellar blockchain.

1 Introduction

A common problem in distributed computing is to implement synchronization abstractions such as reliable broadcast, shared memory, or consensus, given some assumptions about the possible Byzantine failures that may occur in an execution.

A *fail-prone system* $\mathcal{F}$ [14] is a set of sets of processes, called *fail-prone sets*, where no fail-prone set is a subset of another. A fail-prone system $\mathcal{F}$ denotes the assumption that the set of processes A that may suffer Byzantine failures is contained in one of the fail-prone sets. For example, in a system of n processes, it is common to assume that less than a third will fail, i.e., the fail-prone sets are the sets of cardinality exactly $\lfloor (n-1)/3 \rfloor$.

Fail-prone systems are useful because of their relationship to quorum systems [14]. A Byzantine quorum system $\mathcal{Q}$ for $\mathcal{F}$ is a collection of subsets of processes, called *quorums*, such that for every two quorums Q_1 and Q_2 in $\mathcal{Q}$ and for every fail-prone set $F \in \mathcal{F}$ it holds that Q_1 and Q_2 have a common member outside F (Consistency) and for every fail-prone set $F \in \mathcal{F}$, there exists a quorum disjoint from F (Availability).

26th International Conference on Principles of Distributed Systems (OPODIS 2022).
Editors: Eshcar Hillel, Roberto Palmieri, and Etienne Rivière; Article No. 17; pp. 17:1–17:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Many distributed algorithms (implementing, e.g., Byzantine reliable broadcast or consensus) are parameterized by a quorum system $\mathcal{Q}$ and their guarantees hold under the assumptions of a fail-prone system $\mathcal{F}$ if and only if Consistency and Availability of $\mathcal{Q}$ hold. This allows the designers of a distributed system to make assumptions about failures, pick a corresponding quorum system, and then choose among existing algorithms to solve the desired synchronization problem.

Traditionally, such fail-prone systems have been used in closed systems with assumptions of the form "less than on third of the processes are faulty". This can work even in permissionless systems using Proof-of-Stake, e.g., assuming that less than one third of the stake-holders are faulty, or Proof-of-Work, e.g., assuming that the faulty set of processes holds less than one third of the mining power [17]. Proof-of-Stake and Proof-of-Work however have their disadvantages, e.g., long-range attacks in Proof-of-Stake, or excessive energy consumption in Proof-of-Work. Both may also suffer from the "rich getting richer" problem, leading to very few entities eventually controlling the system.

Instead of Proof-of-Stake and Proof-of-Work, one can envision letting every participant make its own, subjective failure assumptions. Distributed computing models based on this idea was first investigated by Damgård *et al.* [8], followed by Sheff et al. [18], and more recently by Cachin and Tackmann [5] with the *asymmetric-trust model*, Malkhi et al. [13] with *flexible quorums*, and Sheff et al. [19] with Heterogeneous Paxos. On the practical side, Ripple (`https://ripple.com`) deployed a permissionless consensus protocol based on subjective quorums in 2012, in every process declares a *Unique Node List* (UNL) of processes that it trusts. Stellar (`https://stellar.org`, [16]) later followed suite with a different permissioness model based on subjective trust.

One problem is that, in all the models cited above, even though participants are free to make their own failure assumptions or choose their own quorums, maintaining consistency requires compatible assumptions (in the sense that the resulting quorums will sufficiently intersect) and thus prior common knowledge or prior synchronization, which is not desirable in a permissionless system.

For example, Cachin and Tackmann [5] assume that for every two participants p_i and p_j, for every two quorums Q_i of p_i and Q_j of p_j, if F is a set that can fail according to the assumptions of both p_i and p_j, then $(Q_i \cap Q_j) \setminus F \neq \emptyset$; this is called the *consistency* property. Together with an *availability* property, this defines an *asymmetric quorum system*. Consistency ensures that if both p_i and p_j make correct assumptions, then they can avoid disagreeing in, e.g., a reliable broadcast protocol. Achieving liveness additionally requires the availability property, i.e., that a group of participants, called a *guild*, whose assumptions are correct and which do not fail, satisfy the consistency property (pairwise) and additionally that every member of the guild has a quorum in the guild.

Maintaining safety in such a model requires a strong form of a-priori coordination. Indeed, two participants cannot be prevented from disagreeing unless every two of their quorums have at least one non-faulty participant in common. Thus participants must coordinate beforehand in order to pick sufficiently overlapping survivor sets, and corresponding quorums. In the model of Ripple, for instance, every two processes must have UNLs that overlap by some sufficient fraction [1].

Global assumptions implying the intersection of survivor sets and quorums, as in the two preceding examples, are problematic in a permissionless setting because they postulate some form of pre-agreement or common knowledge, which might be hard to achieve in practice.

Interestingly, the Stellar network (`https://stellar.org`, [16]), a deployed blockchain system based on quorums, is able to maintain safety and liveness without requiring that participants choose intersecting quorums. Instead, participants choose *quorum slices* that

need not intersect, and the quorums of a participant are defined in terms of the slices of other participants. Consensus can then be solved within an *intact set* $\mathcal{I}$, a set of correct processes such that every two processes p_i and p_j in $\mathcal{I}$ have all their own quorums that intersect in at least a member of $\mathcal{I}$ and such that $\mathcal{I}$ is itself a quorum for every of its members.

We observe that quorum slices can be interpreted as a new kind of failure assumptions: a participant assumes that at least one of its quorum slices is made exclusively of participants that do not fail and make correct assumptions. In other words, a participant's assumption are not only about failures, but also about whether other participants make correct assumptions. In practice, the Stellar model makes it easier for participants to achieve quorum intersection by relying on the failure assumptions of other participants that might have more knowledge about the system than they have.

The main contribution of this paper is to show that this new kind of failure assumptions yield a generalization of the theory of fail-prone systems (i.e., classic fail-prone systems are a special case) which allows to obtain intersecting quorums even when participants do not know any common third party.

Moreover, based on this, we introduce the notion of *permissionless fail-prone system* from which it is possible to derive a *permissionless quorum system.*

This paper is structured as follows. In Section 2 we formally define the assumptions of a process; each process assumes that one of its slices S will not fail and, *additionally*, that the assumptions of every process in S will be satisfied too. Then, we introduce the notion of permissionless fail-prone system. This extends and generalizes the *asymmetric fail-prone system* [5,8]. Crucially, we note that the new meaning of the assumptions of the processes allows processes to transitively rely on the assumptions of other processes. However, this in turn enables malicious processes to lie about their assumptions.

In Section 3, we propose a computation model, based on the notion of *view*, that takes this phenomenon into account. Moreover, always in Section 3, we derive from the permissionless fail-prone system the notion of *permissionless Byzantine quorum system.*

In Section 4 we introduce the notion of *league*, which is a set of processes L that enjoys Consistency (quorums intersection) and Availability (existence of a quorum in L consisting of correct processes) among the correct members of L even when faulty processes lie about their configuration.

We compare our permissionless model with the classic model based on fail-prone systems [14], with the asymmetric model [5,8], with the federated Byzantine agreement system [16] and with the personal Byzantine quorum system model [12] in Section 5. Interestingly we show that classic fail-prone systems can be understood as a special case of our model.

In Section 6 we show how a traditional synchronization protocol, i.e., Bracha broadcast [2], can be adapted to work in our model, thereby offering a new toolbox for the design of permissionless distributed systems.

In the full version of this paper [4] we present another application of permissionless quorum systems through the emulation of shared memory, represented by a register. In particular we show how to implement a single-writer multi-reader register with permissionless quorum systems.

Related work and conclusions are presented in Section 7 and Section 8, respectively. Finally, our model also leads to a characterization of the Stellar model with standard formalism [5,8,14].

2 Model

We consider an unbounded set of *processes* $\Pi = \{p_1, p_2, ...\}$ that communicate asynchronously with each other by sending messages. We assume that processes do not necessarily know which other processes are in the system (i.e., each process only knows a subset of Π).

Processes are assigned a protocol to follow. Protocols are presented in a modular way using the event-based notation of Cachin et al. [3]. A process that follows its algorithm during an execution is called *correct*. Initially, all processes are correct, but a process may later fail, in which case it is called *faulty*. We assume Byzantine failures, where a process that fails thereafter behaves arbitrarily.

We assume that point-to-point communication between any two processes (that know each other) is available, as well as a best-effort gossip primitive that will reach all processes. In a protocol, this primitive is accessed through the events "sending a message through gossip" and "receiving a gossiped message." We assume that messages from correct processes to correct process are eventually received and cannot be forged. The system itself is asynchronous, i.e., the delivery of messages among processes may be delayed arbitrarily and the processes have no synchronized clocks.

Processes make failure assumptions about other processes. However, since a process does not know exactly who is part of the system, it cannot make failure assumptions about the whole system. Instead, each process p_i makes assumptions about a set $P_i \subseteq \Pi$, called p_i's *trusted set*, using a *fail-prone system* $\mathcal{F}_i$ over P_i (Section 1). Here, $\mathcal{F}_i$ is a collection of subsets of P_i and p_i believes that up to any set $F \in \mathcal{F}_i$ may jointly fail. We say that P_i and $\mathcal{F}_i$ constitute p_i's *assumptions* and we assume that they remain fixed during an execution.

Note that assuming that the assumptions of the processes are fixed is a simplification. In practice, the system may experience churn, i.e., processes frequently entering and departing the system. Processes that remain in the system can adjust their assumptions in response to churn: for example, if a process stops responding (maybe because it has left the system), then other processes can remove it from their assumptions. Conversely, if a new process joins the system, existing processes may adjust their assumptions to include that process. However, analyzing the system under churn is out of the scope of this paper.

▶ **Definition 1** (Fail-prone system). *A* fail-prone system $\mathcal{F}$ *over* $P \subseteq \Pi$ *is a set of subsets of* Π *called* fail-prone sets, *none of which contain the other (i.e., if* $F \in \mathcal{F}$ *and* $F' \subset F$, *then* $F' \notin \mathcal{F}$).

A *permissionless fail-prone system* (abbreviated PFPS) describes the assumptions of all the processes:

▶ **Definition 2** (Permissionless fail-prone system). *A permissionless fail-prone system is an array* $\mathbb{F} = [(P_1, \mathcal{F}_1), (P_2, \mathcal{F}_2), \ldots]$ *that associates each process* p_i *to a trusted set* $P_i \subseteq \Pi$ *and a fail-prone system* $\mathcal{F}_i$ *over* P_i. *We refer to* $(P_i, \mathcal{F}_i)$ *as the* configuration *of process* p_i.

We now consider a fixed PFPS $\mathbb{F}$.

▶ **Definition 3** (Tolerated execution and tolerated set). *We say that* the assumptions of a process p_i are satisfied *in an execution if the set* A *of processes that actually fail is such that there exists a fail-prone set* $F \in \mathcal{F}_i$ *and:*
1. $A \cap P_i \subseteq F$; *and*
2. *the assumptions of every member of* $P_i \setminus F$ *are satisfied.*

If $p_i \in \Pi$ *has its assumptions satisfied in an execution* e, *we say that* p_i tolerates *the execution* e.

Finally, a set of processes L tolerates *a set* of processes A *if and only if every process* $p_i \in L \setminus A$ *tolerates an execution* e *with set of faulty processes* A.

▶ **Example 4.** Consider a set of processes $\Pi = \{p_1, p_2, p_3, p_4\}$ and a permissionless fail-prone system $\mathbb{F} = [(\Pi, \mathcal{F}_1), (\Pi, \mathcal{F}_2), (\Pi, \mathcal{F}_3), (\Pi, \mathcal{F}_4)]$ with $\mathcal{F}_1 = \{\{p_3, p_4\}\}$, $\mathcal{F}_2 = \{\{p_1, p_4\}\}$, $\mathcal{F}_3 = \{\{p_1, p_4\}\}$, and $\mathcal{F}_4 = \{\{p_1, p_2\}\}$. Then, Π tolerates the sets $\emptyset$, $\{p_1\}$, $\{p_4\}$ and $\{p_1, p_4\}$. To see this, let us assume an execution e with set of faulty processes $A = \{p_1, p_4\}$. Then, for every $p_i \in \Pi \setminus A$, there exists a fail-prone set $F \in \mathcal{F}_i$ such that $A \cap \Pi \subseteq F$. In particular, $\Pi \setminus A = \{p_2, p_3\}$ and $\{p_1, p_4\} \in \mathcal{F}_2$ and $\{p_1, p_4\} \in \mathcal{F}_3$. The same reasoning can be applied for the other sets.

Note that here we depart significantly from the traditional notion of fail-prone systems [5, 14]: in a PFPS, a process not only makes assumptions about failures, but also makes assumptions about the assumptions of other processes.

Next we define *survivor sets* analogously to Junqueira and Marzullo [11]. In the traditional literature, a survivor set of p_i is the complement, within Π, of some fail-prone set. However, defining them as the complement of fail-prone sets within P_i does not work because of Item 2 in Definition 3. To obtain this definition, we first define a *slice*.

▶ **Definition 5** (Slice). *A set $\overline{F} \subseteq \Pi$ is a* slice *of p_i if and only if p_i has a fail-prone set $F \in \mathcal{F}_i$ such that $\overline{F} = P_i \setminus F$.*

For any $S \subseteq \Pi$ we often say p_i *has a slice in S* when a slice of p_i is contained in S or when S contains a superset of a slice of p_i.

▶ **Definition 6** (Survivor-set system). *A survivor-set system $\mathcal{S}_i$ of p_i is the minimal set of subsets S of Π such that:*
1. *p_i has a slice in S; and*
2. *every member of S has a slice in S.*
Each $S \in \mathcal{S}_i$ is called a survivor set *of p_i.*

▶ **Example 7.** Continuing from Example 4, process p_1 has only one slice consisting of $\{p_1, p_2\}$, processes p_2 and p_3 have the set $\{p_2, p_3\}$ as slice, and process p_4 has the set $\{p_3, p_4\}$ as slice. Moreover, the survivor-set systems are $\mathcal{S}_1 = \{\{p_1, p_2, p_3\}, \{p_1, p_2, p_3, p_4\}\}$ for process p_1, $\mathcal{S}_2 = \{\{p_2, p_3\}, \{p_1, p_2, p_3, p_4\}\}$ for process p_2, $\mathcal{S}_3 = \{\{p_2, p_3\}, \{p_1, p_2, p_3, p_4\}\}$ for process $p3$, and $\mathcal{S}_4 = \{\{p_2, p_3, p_4\}, \{p_1, p_2, p_3, p_4\}\}$ for process p_4. This follows from Definition 6: given a survivor set $S \in \mathcal{S}_i$ for p_i, process p_i must have a slice in S and every member of S must have a slice in S. So, for example, given the survivor set $\{p_1, p_2, p_3\}$ in the survivor set system $\mathcal{S}_1$ for p_1, process p_1 has a slice in $\{p_1, p_2, p_3\}$, i.e., $\{p_1, p_2\}$, and every process $p_i \in \{p_1, p_2, p_3\}$ has a slice in $\{p_1, p_2, p_3\}$, i.e., $\{p_2, p_3\}$.

▶ **Lemma 8.** *The assumptions of a process $p_i \in \Pi$ are satisfied in an execution e with set of faulty processes A if and only if there exists a survivor set $S \in \mathcal{S}_i$ of p_i such that S does not fail.*

Proof. Let p_i be a process such that, given an execution e with set of faulty processes A, the assumptions of p_i are satisfied in e. This implies that, by Definition 3, there exists a set of processes such that each of these processes has its assumptions satisfied. Moreover, by Definition 5, each of these processes has a slice $\overline{F}_j$ such that $\overline{F}_j \cap A = \emptyset$. This leads to have a set S obtained as union of all of these slices such that $S \cap A = \emptyset$ and such that S is minimal with respect to this union, in the sense that is the minimal set of processes such that every process in S has its assumptions satisfied. The set S is a survivor set of p_i.

Conversely, we show that given a survivor set S of p_i, given a process $p_i \in S$ and given an execution e with set of faulty processes A, if $S \cap A = \emptyset$, then the assumptions of p_i are satisfied in e. Observe that, from the assumptions, we have that every process in S has a slice $\overline{F}$ in S such that $\overline{F} \cap A = \emptyset$. This means that for every process p_i in S, there exists a fail-prone set $F \in \mathcal{F}_i$ such that $P_i \cap A \subseteq F$. This implies that every process in S has its assumptions satisfied and, in particular, that $p_i \in S$ has its assumptions satisfied in e. ◀

3   Permissionless Quorum Systems

A classic (or symmetric) fail-prone system [14] determines a canonical quorum system known to all processes through the Q^3-condition. Specifically, given a fail-prone system $\mathcal{F}$, the Q^3-condition requires that no three fail-prone sets of $\mathcal{F}$ cover the complete set of processes and this condition holds if and only if there exists a quorum system for $\mathcal{F}$ [10, 14]. Such a quorum system could be, for example, the complement of every fail-prone set of $\mathcal{F}$, which we call the *canonical* quorum system. Traditional algorithms such as read-write register emulations [14], Byzantine reliable broadcasts [2, 20] or the PBFT algorithm [7] make use of quorums.

In the model of asymmetric trust [8] the assumptions of the processes may differ, and asymmetric quorum systems [5, 6] allow to implement the above-mentioned algorithms in a more flexible way. However, they still require a system that is known to every process.

In a permissionless system, processes do not know the membership and have different, partial, and potentially changing views of its composition.

Given a PFPS, we would therefore like to obtain a quorum system to implement algorithms for register emulation, broadcast, consensus and more, while allowing the processes to have different assumptions in an open network.

We are therefore interested in defining a notion of quorums for open systems where:

1. each process has its own quorum system; and
2. the quorums of a process p_i depend on the assumptions of other processes, which p_i learns by communicating with them.

In other words, we consider scenarios in which each process p_i communicates with other processes, continuously discovers new processes and learns their assumptions. During this execution, p_i determines its current set of quorums as a function of what it has learned so far. Importantly, this means that the quorums of a process evolve as the process learns new assumptions, and that faulty processes can influence p_i's quorums by lying about their assumption.

We now formalize this model using the notions of a *view* and a *quorum function*.

▶ **Definition 9** (View). *A view $\mathbb{V} = [\mathcal{V}_1, \mathcal{V}_2, \ldots]$ is an array with one entry $\mathbb{V}[j] = \mathcal{V}_j$ for each process p_j such that:*

1. *either $\mathcal{V}_j$ is the special value $\perp$; or*
2. *$\mathcal{V}_j = (P_j, \mathcal{F}_j)$ consists of a set of processes P_j and a fail-prone system $\mathcal{F}_j$ over P_j.*

Observe that every process p_i has its *local* view $\mathbb{V}$, whose non-$\perp$ entries represent the assumptions that p_i has learned at some point in an execution. Every other process p_j such that $\mathbb{V}[j] = \perp$ is a process that p_i has not heard from. We denote with Υ the set of all the possible views.

We assume that, for every process p_j, a process p_i's view contains the assumption that p_i has most recently received from p_j. Finally, note that $\mathbb{F}$ is a view in which no process is mapped to $\perp$. In particular, $\mathbb{F}$ represents the global view if the system could be entirely observed. Since processes cannot observe the complete system, they normally only have partial knowledge of $\mathbb{F}$. Moreover, this knowledge evolves over time.

▶ **Definition 10** (Domain of a view). *For a view $\mathbb{V}$, the set of processes p_i such that $\mathbb{V}[i] \neq \perp$ is the* domain *of $\mathbb{V}$.*

Next, we assume that every process determines its quorums according to its view using a function $\mathcal{Q}$ called a *quorum function*. We assume that all correct processes use the same $\mathcal{Q}$ and that they do not change it during an execution. We then have the following definition.

▶ **Definition 11** (Quorum function). *The* quorum function $\mathcal{Q} : \Pi \times \Upsilon \to 2^{\Pi}$ *maps a process p_i and a view $\mathbb{V}$ to a set of sets of processes such that $Q \in \mathcal{Q}(p_i, \mathbb{V})$ if and only if:*
1. *a slice of p_i is contained in Q; and*
2. *for every process $p_j \in Q$ with $\mathbb{V}[j] \neq \perp$ and $\mathbb{V}[j] = (P_j, \mathcal{F}_j)$, there exists $F \in \mathcal{F}_j$ such that $P_j \setminus F \subseteq Q$.*

Every element of $\mathcal{Q}(p_i, \mathbb{V})$ is called a quorum for p_i *(in $\mathbb{V}$).*

Notice that in the first condition, the quorum Q may itself be a slice of p_i. Moreover, Q is a quorum for every one of its members and it is defined by slices of every $p_i \in Q$. As shown in the following lemma, a quorum for p_i in view $\mathbb{V}$ for p_i is a survivor set of p_i.

▶ **Lemma 12.** *For every view $\mathbb{V}$ for $p_i \in \Pi$, every quorum $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ is a survivor set of p_i. Moreover, given S a survivor set of p_i, there exists a view $\mathbb{V}$ for p_i such that $S \in \mathcal{Q}(p_i, \mathbb{V})$.*

Proof. Let $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ be a quorum for p_i with $\mathbb{V}$ a view for p_i. By Definition 11, all processes in Q including p_i have a slice in Q. From Definition 6, this implies that Q is a survivor set of p_i.

Moreover, given a survivor set S of p_i, the set S consists of slices of every member of S. This means that there exists a view $\mathbb{V}$ for p_i in which S satisfies Definition 11 and it is a quorum for p_i. This is the view $\mathbb{V}$ defined as follows:
1. for every $p_j \in S$, $\mathbb{V}'[j] = \mathbb{F}[j]$, and
2. for every $p_j \notin S$, $\mathbb{V}'[j] = (\emptyset, \{\emptyset\})$. ◀

▶ **Example 13.** Let us consider Example 4 with survivor-set systems as shown in Example 7. Since all the processes already know all the configurations of every other process, we have that $\mathcal{S}_i = \mathcal{Q}(p_i, \mathbb{F})$, with $\mathbb{F}$ the permissionless fail-prone system.

Combining the quorum sets of all processes, we now obtain a *permissionless quorum system* for $\mathbb{F}$.

▶ **Definition 14** (Permissionless quorum system). *A* permissionless quorum system *for Π and $\mathbb{F}$ is an array of collections of sets $\mathbb{Q}_{perm} = [\mathcal{Q}(p_1, \mathbb{F}), \mathcal{Q}(p_2, \mathbb{F}), \ldots]$, where $\mathcal{Q}(p_i, \mathbb{F})$ is called the* quorum system *for p_i and is determined by the quorum function $\mathcal{Q}$.*

Observe that our notion of a quorum system differs from that in the existing literature [5, 14, 15]. In particular, standard Byzantine quorum systems are defined through a pair-wise intersection among quorums. This is possible in scenarios where the full system membership is known to every process. However, in permissionless settings, this requirement cannot as clearly be achieved globally.

▶ **Definition 15** (Current quorum system). *Let $\mathbb{V}$ be the view representing the assumptions that a process p_i has learned so far. Then the* current quorum system of p_i *is the set $\mathcal{Q}(p_i, \mathbb{V})$. Moreover, a set of processes Q is a* current quorum *of p_i if and only if $Q \in \mathcal{Q}(p_i, \mathbb{V})$; we also say that p_i has a quorum Q.*

Note that, in this model, each process has its own set of quorums and the set of quorums of a process changes throughout an execution as the process learns the assumptions of more processes. Importantly, note that faulty processes may lie about their configuration and influence the quorums of correct processes. In an execution e with faulty set A, a correct process p_i might have a view in which the assumptions of processes in A are arbitrary because processes in A lied about their assumptions. However, processes outside A do not lie about their assumptions. We capture this with the following definition.

▶ **Definition 16** (T-resilient view). *Given a set of processes T, we say that a view $\mathbb{V}$ is T-resilient if and only if for every process $p_i \notin T$, either $\mathbb{V}[i] = \bot$ or $\mathbb{V}[i] = \mathbb{F}[i]$.*

Intuitively, a correct process p_i will either not have heard from $p_j \notin A$ or it will have the correct assumption for p_j. Thus, p_i's view is A-resilient at all times in execution e.

As we said, processes in A may lie about their assumptions causing quorums to contain unreliable slices. Moreover, processes in A may aim at preventing intersection among quorums of correct processes. In the following definition we characterize the notion of worst-case view, i.e., when faulty processes gossip only empty configurations. By doing so, quorums of correct processes will contain fewer members, increasing the chances of an empty intersection among them.

▶ **Definition 17** (Worst-case view). *Given a set of processes T, the* worst-case view *with respect to T is the view $\mathbb{V}_T^*$ such that:*
1. *for every $p_i \in \Pi \setminus T$, $\mathbb{V}_T^*[i] = \mathbb{F}[i]$, and*
2. *for every $p_i \in T$, $\mathbb{V}_T^*[i] = (\emptyset, \{\emptyset\})$.*

Finally, every quorum for a process $p_i \notin A$ in a A-resilient view contains a quorum for p_i in a worst-case view with respect to A. This is shown in the following lemma.

▶ **Lemma 18.** *Consider a set of processes T, a T-resilient view $\mathbb{V}$, and a process $p_i \notin T$. Moreover, let us assume that processes in T may lie about their assumptions. For every quorum $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$, there exists a quorum $Q_i' \in \mathcal{Q}(p_i, \mathbb{V}_T^*)$ such that $Q_i' \subseteq Q_i$.*

Proof. Let T be a set of processes, $\mathbb{V}$ be a T-resilient view, p_i a process not in T. Since $\mathbb{V}$ is a T-resilient view, for every $p_j \notin T$ it holds either $\mathbb{V}[j] = \bot$ or $\mathbb{V}[j] = \mathbb{F}[j]$. However, processes in T may lie about their assumptions and, because of that, the view of process $p_i \notin T$ may contain arbitrary configurations received from processes in T.

If $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ is a quorum for p_i in $\mathbb{V}$, then Q_i might contain slices of processes in T which are derived from false assumptions. One can easily show that by starting from a T-compatible view and by removing the configurations received by processes in T, it is possible to obtain the corresponding worst-case view. By removing configurations from $\mathbb{V}$, also Q_i becomes smaller, i.e., with less members, obtaining a quorum $Q_i' \subseteq Q_i$. In fact, by removing from Q_i a slice $\overline{F}_j$ of a process $p_j \in T$, also slices of other processes in $\overline{F}_j$ might get removed in order for Definition 11 to be satisfied on Q_i'. This proves the lemma.   ◀

4   Leagues

We now define the notion of a *league*. In Section 6 we show how a league allows to implement Bracha broadcast.

▶ **Definition 19** (League). *set of processes L is a* league *for the quorum function $\mathcal{Q}$ if and only if the following property holds:*

Consistency: *For every set $T \subseteq \Pi$ tolerated by L, for every two T-resilient views $\mathbb{V}$ and $\mathbb{V}'$, for every two processes $p_i, p_j \in L \setminus T$, and for every two quorums $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ and $Q_j \in \mathcal{Q}(p_j, \mathbb{V}')$ it holds $(Q_i \cap Q_j) \setminus T \neq \emptyset$.*

Availability: *For every set $T \subseteq \Pi$ tolerated by L and for every $p_i \in L \setminus T$, there exists a quorum $Q_i \in \mathcal{Q}(p_i, \mathbb{F})$ for p_i such that $Q_i \subseteq L \setminus T$.*

If we consider an execution e tolerated by a league L, where A is the set of faulty processes, the consistency property of L implies that, at any time, any two quorums of correct processes in L have some correct process in common. This is similar to the consistency property of symmetric and asymmetric Byzantine quorum systems [5, 14].

Moreover, since the set of faulty processes A is tolerated by L, by the availability property of L, every correct process in L has a quorum in $\mathbb{F}$ consisting of only correct processes.

▶ **Example 20.** Observe that the set Π as introduced in Example 4 is a league. In fact, for every set T tolerated by Π, i.e., $\emptyset$, $\{p_1\}$, $\{p_4\}$ and $\{p_1, p_4\}$, for every two processes $p_i, p_j \in \Pi \setminus T$ and for every two quorums $Q_i \in \mathcal{S}_i$ and $Q_j \in \mathcal{S}_j$ as in Example 13, it holds $(Q_i \cap Q_j) \setminus T \neq \emptyset$, and for every $p_i \in \Pi \setminus T$, there exists a quorum $Q_i \in \mathcal{S}_i$ such that $Q_i \subseteq \Pi \setminus T$.

The following lemma shows that the union of two intersecting leagues L_1 and L_2 is again a league, assuming that for every set T tolerated by both the leagues, L_1 and L_2 have a common process not in T.

▶ **Lemma 21.** *If L_1 and L_2 are two leagues such that $L_1 \cap L_2 \neq \emptyset$ and such that for every set T tolerated by $L_1 \cup L_2$, there exists a process $p_k \in (L_1 \cap L_2) \setminus T$, then $L_1 \cup L_2$ is a league.*

Proof. Let L_1 and L_2 be two leagues such that $L_1 \cap L_2 \neq \emptyset$. For every T tolerated by $L_1 \cup L_2$ (and so, tolerated by L_1 and L_2, independently), for every $p_i \in L_1 \setminus T$ and $p_j \in L_2 \setminus T$, for every two T-resilient views $\mathbb{V}$ and $\mathbb{V}'$ for p_i and p_j, respectively, let $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ and $Q_j \in \mathcal{Q}(p_j, \mathbb{V}')$ be two quorums for p_i and p_j, respectively. Observe that, by assumption, for every tolerated set T by $L_1 \cup L_2$ there exists a process $p_k \in (L_1 \cap L_2) \setminus T$. Let $p_k \in L_1 \cap L_2$ and let $Q_k \in \mathcal{Q}(p_k, \mathbb{V})$ be a quorum for p_k such that $Q_k \subseteq L_1$, according to availability property of L_1. From consistency property of L_2, $(Q_k \cap Q_j) \setminus T \neq \emptyset$ and every process in this intersection belongs to L_1. Observe that, Q_j is a quorum for every of its member. This implies that Q_j is a quorum for every process in $(Q_k \cap Q_j) \setminus T$ and every process in $(Q_k \cap Q_j) \setminus T$ has quorum in L_1. Moreover, $(Q_k \cap Q_i) \setminus T \neq \emptyset$. It follows that $(Q_i \cap Q_j) \setminus T \neq \emptyset$.

Finally, by availability property of L_1 and L_2, for every tolerated set T by L_1 and L_2 and for every process $p_i \in L_1 \setminus T$ and $p_j \in L_2 \setminus T$, eventually there exists a quorum $Q_i \in (p_i, \mathbb{F})$ for p_i and a quorum $Q_j \in \mathcal{Q}(p_j, \mathbb{F})$ for p_j such that $Q_i \subseteq L_1 \setminus T$ and $Q_j \subseteq L_2 \setminus T$, respectively. If $p_i = p_j \in L_1 \cap L_2$, then there exists a quorum Q_i for p_i such that $Q_i \subseteq (L_1 \cup L_2) \setminus T$. ◀

In the following lemma we show that we can characterize the consistency property of a league just by considering worst-case views. Intuitively, this result relies on the observation that every T-resilient view can be seen as extensions of worst-case views with respect to $T \subseteq \Pi$, in the sense that a T-resilient view can be obtained by starting from a worst-case view with respect to T and by considering the non-empty configurations received by processes in T.

▶ **Lemma 22.** *The consistency property a league L holds if and only if for every set $T \subseteq \Pi$ tolerated by L, for every two worst-case views $\mathbb{V}_T^*$ and $\mathbb{V}_T'^*$ with respect to T, for every two processes $p_i, p_j \in L \setminus T$, and for every two quorums $Q_i \in \mathcal{Q}(p_i, \mathbb{V}_T^*)$ and $Q_j \in \mathcal{Q}(p_j, \mathbb{V}_T^*)$ it holds $(Q_i \cap Q_j) \setminus T \neq \emptyset$.*

Proof. Let us assume that the consistency property of a league L holds. Since the property holds for every pair of views, it must hold also for worst-case views. The implication easily follows.

Let us now assume that for every set $T \subseteq \Pi$ tolerated by L, for every two worst-case views $\mathbb{V}_T^*$ and $\mathbb{V}_T'^*$ with respect to T, for every two processes $p_i, p_j \in L \setminus T$, and for every two quorums $Q_i \in \mathcal{Q}(p_i, \mathbb{V}_T^*)$ and $Q_j \in \mathcal{Q}(p_j, \mathbb{V}_T'^*)$ it holds $(Q_i \cap Q_j) \setminus T \neq \emptyset$. Observe that, given a quorum $Q_i \in \mathcal{Q}(p_i, \mathbb{V}_T^*)$ for p_i in a worst-case view $\mathbb{V}_T^*$, all the quorums obtained by also considering all the possible configurations received from processes in T that are not in

$\mathbb{V}_T^*$ do contain Q_i. Moreover, there cannot exist a T-resilient view that does not consist of configurations of a worst-case view with respect to T. If this was the case, then by removing configurations received from processes in T one would obtain a worst-case view with respect to T, reaching a contradiction. So, all the quorums obtained from T-resilient views will also intersect in processes that are not contained in T. ◀

Now we show how a league can be abstracted and defined without considering views. This will be useful in Section 5 when we compare our model with other permissionless models. First, we introduce the following definitions.

▶ **Definition 23** (Inclusive up to). *A set $I \subseteq \Pi$ is* inclusive up to *a set $T \subseteq \Pi$ if and only if for every $p_i \in I \setminus T$, process p_i has a slice in I.*

If we consider an execution e with set of faulty processes A then a set of processes I is inclusive up to A if and only if every correct process in I has a slice contained in I.

▶ **Definition 24** (Rooted at). *A set $R \subseteq Pi$ is* rooted at *a process p_i if and only if p_i has a slice in R. A set $R \subseteq \Pi$ is* rooted *in a set $T' \subseteq \Pi$ whenever R is rooted at a member of T'.*

▶ **Lemma 25.** *If $\mathbb{V}$ is a T-resilient view and $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ for some process p_i, then Q_i is inclusive up to T and rooted at p_i.*

Proof. If $\mathbb{V}$ is a T-resilient view then, by Definition 16, processes outside T do not lie about their assumptions. By definition of a quorum Q_i for a process p_i in a view $\mathbb{V}$, every process in Q_i, and so in $Q_i \setminus T$, has a slice in Q_i. This implies that Q_i is inclusive up to T and rooted at p_i. ◀

In the following lemma we show that given a set of processes T tolerated by $L \subseteq \Pi$, for every set of processes I inclusive up to T and rooted at $p_i \in L \setminus T$ it is possible to find a T-resilient view in which I is a quorum for p_i. This view is a worst-case view with respect to T.

▶ **Lemma 26.** *Let L be a set of processes. For every set $T \subseteq \Pi$ tolerated by L, if $I \subseteq \Pi$ is a set inclusive up to T and rooted at $p_i \in L \setminus T$, then there is a T-resilient view in which I is a quorum for p_i.*

Proof. Let $T \subseteq \Pi$ be a tolerated set by a set of processes L and let $I \subseteq \Pi$ be a set inclusive up to T and rooted at $p_i \in L \setminus T$. This implies that p_i and every other process $p_j \in I \setminus T$ have a slice in I. Let us consider the worst-case view $\mathbb{V}_T^*$ with respect to T. Clearly, $\mathbb{V}_T^*$ is T-resilient. This implies that, in $\mathbb{V}_T^*$, the set I is a quorum for p_i, i.e., $I \in \mathcal{Q}(p_i, \mathbb{V}_T^*)$. ◀

▶ Remark 27. Observe that given a set of processes L and a worst-case view $\mathbb{V}_T^*$ with respect to a set $T \subseteq \Pi$ tolerated by L, every quorum $Q_i \in \mathcal{Q}(p_i, \mathbb{V}_T^*)$ for $p_i \in L \setminus T$ is inclusive up to T and rooted at p_i.

Moreover, given the set $\mathcal{I}$ of all the sets $I \subseteq \Pi$ inclusive up to T and rooted at $p_i \in L \setminus T$, the set $\mathcal{I}$ contains all the quorums $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$ for every T-resilient view $\mathbb{V}$. In fact, by Definition 23, given a set of processes I inclusive up to a set of processes T, the requirement of having a slice in I is only for processes in $I \setminus T$, leaving processes in $T \cap I$ with no requirements on the choice of their slices.

However, given a T-resilient view $\mathbb{V}$, by Definition 11, a quorum Q_i for p_i requires instead every process in Q_i to have a slice contained in Q_i. This means that given a T-resilient view $\mathbb{V}$, quorum Q_i for p_i is contained in $\mathcal{I}$, being a special case of inclusive set up to T.

▶ **Lemma 28.** *The consistency property of a league L holds if and only if for every set $T \subseteq \Pi$ tolerated by L, and for every two sets $I \subseteq \Pi$ and $I' \subseteq \Pi$ that are rooted at $L \setminus T$ and inclusive up to T it holds $(I \cap I') \setminus T \neq \emptyset$.*

Proof. Let us assume that the consistency property of a league L holds. Suppose by contradiction that there is a set $T \subseteq \Pi$ tolerated by L and two sets $I \subseteq \Pi$ and $I' \subseteq \Pi$ that are inclusive up to T and rooted at $L \setminus T$ in p_i and p_j, respectively, such that $(I \cap I') \setminus T = \emptyset$.

By Lemma 26, there are a T-resilient view $\mathbb{V}$ in which I is a quorum for p_i and a T-resilient view $\mathbb{V}'$ in which I' is a quorum for p_j and we reached a contradiction.

Let us now assume that for every set $T \subseteq \Pi$ tolerated by L, and for every two sets $I \subseteq \Pi$ and $I' \subseteq \Pi$ that are inclusive up to T and rooted at $L \setminus T$ it holds $(I \cap I') \setminus T \neq \emptyset$.

Let $\mathcal{I}$ and $\mathcal{I}'$ be the sets of all the sets $I \subseteq \Pi$ and $I' \subseteq \Pi$ inclusive up to T and rooted at $p_i \in L \setminus T$ and $p_j \in L \setminus T$, respectively. The proof follows from the reasoning in Remark 27: for every two T-resilient views $\mathbb{V}$ and $\mathbb{V}'$, every quorum $Q_i \in \mathfrak{Q}(p_i, \mathbb{V})$ for p_i is contained in $\mathcal{I}$ and every quorum $Q_j \in \mathfrak{Q}(p_j, \mathbb{V})$ for p_j is contained in $\mathcal{I}'$. ◀

▶ **Lemma 29.** *The availability property of a league L holds if and only if for every set $T \subseteq \Pi$ tolerated by L, every member of $L \setminus T$ has a survivor set in $L \setminus T$.*

Proof. Let us assume that the availability property of a league L holds, i.e., for every set of processes T tolerated by L and for every $p_i \in L \setminus T$, there exists a quorum $Q_i \in \mathfrak{Q}(p_i, \mathbb{F})$ for p_i such that $Q_i \subseteq L \setminus T$. This means that, by Definition 6, every process in $L \setminus S$ has a survivor set in $L \setminus S$.

Let us now assume that for every set $T \subseteq \Pi$ tolerated by L, every member of $L \setminus T$ has a survivor set S in $L \setminus T$. Let p_i be a process in L, by Definition 11 we have that $S \in \mathfrak{Q}(p_i, \mathbb{F})$ for p_i. The proof follows. ◀

5 Comparison with Other Models

In this section we compare our model with the classic model based on fail-prone systems [14], with the asymmetric model [5,8], with the federated Byzantine agreement system model [16], and with the personal Byzantine quorum system model [12].

5.1 Comparison with Fail-Prone Systems

We show that classic fail-prone systems and quorums can be understood as a special case of our model, when every process knows the entire system and assumes the same, global fail-prone system.

Let $\Pi = \{p_1, \dots, p_n\}$ be a set of processes. A Byzantine quorum system for a fail-prone system $\mathcal{F}$ (Definition 1) satisfies (Consistency) $\forall Q_1, Q_2 \in \mathcal{Q}, \forall F \in \mathcal{F} : Q_1 \cap Q_2 \not\subseteq F$; and (Availability) $\forall F \in \mathcal{F} : \exists\, Q \in \mathcal{Q} : F \cap Q = \emptyset$. Moreover, a Byzantine quorum system $\mathcal{Q}$ for $\mathcal{F}$ exists if and only if the Q^3-condition holds [10,14], which means that for every $F_1, F_2, F_3 \in \mathcal{F} : \Pi \not\subseteq F_1 \cup F_2 \cup F_3$. In particular, if $Q^3(\mathcal{F})$ holds, then $\overline{\mathcal{F}}$, the bijective complement of $\mathcal{F}$, is a Byzantine quorum system; this is the Byzantine quorum system consisting of survivor sets [11]. Notice that, in a classic system, the failures that are tolerated by the processes are all possible subsets of fail-prone sets in $\mathcal{F}$ and we have $P_i = \Pi$ for every $p_i \in \Pi$. Every process also knows the global quorum system.

We define a bijective function f between the set of fail-prone systems and a subset of PFPS such that $f(\mathcal{F}) = [(\Pi, \mathcal{F}), \dots, (\Pi, \mathcal{F})]$ with n repetitions and we notice that in classic fail-prone systems there is only one view, namely $\mathbb{V} = f(\mathcal{F})$.

We define the quorum function $\mathcal{Q} : \Pi \times \Upsilon \to 2^{\Pi}$ such that for every process $p_i \in \Pi$, $\mathcal{Q}(p_i, f(\mathcal{F})) = \overline{\mathcal{F}}$. Observe that any set in $\overline{\mathcal{F}}$ is a slice of every process $p_i \in \Pi$ according to Definition 5. In the next theorem we consider this quorum function and show that, given some assumptions on $\mathcal{F}$, any set in $\overline{\mathcal{F}}$ is also a quorum for every process $p_i \in \Pi$ according to Definition 11.

▶ **Theorem 30.** *Let Π be the set of all processes and $\mathcal{F}$ the fail-prone system for Π. Then $Q^3(\mathcal{F})$ holds if and only if Π is a league for the quorum function $\mathcal{Q}$ in $f(\mathcal{F})$.*

Proof. Let us first assume that $Q^3(\mathcal{F})$ holds. This means that for every $F_1, F_2, F_3 \in \mathcal{F}$, $\Pi \not\subseteq F_1 \cup F_2 \cup F_3$. It follows that $\overline{\mathcal{F}}$ is a quorum system for $\mathcal{F}$. Consistency property of $\overline{\mathcal{F}}$ implies that for every tolerated set F, for every two processes p_i and p_j in $\Pi \setminus F$ and for every two quorums $Q_i \in \mathcal{Q}(p_i, f(\mathcal{F}))$ and $Q_j \in \mathcal{Q}(p_j, f(\mathcal{F}))$ for p_i and p_j, respectively, it holds that $(Q_i \cap Q_j) \setminus T \neq \emptyset$.

The availability property of $\overline{\mathcal{F}}$ implies that for every set $F \in \mathcal{F}$ tolerated by Π, every process $p_i \in \Pi \setminus F$ has a quorum in $\Pi \setminus F$: given F, there exists a quorum $Q \in \mathcal{Q}(p_i, f(\mathcal{F}))$ such that $Q \cap F = \emptyset$ and $Q \subseteq \Pi \setminus F$. It follows that Π is a league for the quorum function $\mathcal{Q}$ in $f(\mathcal{F})$.

Let us now assume that Π is a league for the quorum function $\mathcal{Q}$ in $f(\mathcal{F})$. The consistency property of Π implies that for every T tolerated by Π (which are all the sets in $\mathcal{F}$), for every two processes p_i and p_j in $\Pi \setminus T$, for every two quorums $Q_i \in \overline{\mathcal{F}}$ and $Q_j \in \overline{\mathcal{F}}$ for p_i and p_j, respectively, it holds $(Q_i \cap Q_j) \setminus T \neq \emptyset$. Moreover, by availability property of Π there exists a quorum in $\Pi \setminus T$ (which is the same for every process $p_i \notin T$). This implies that, for every fail-prone set $F \in \mathcal{F}$, there is a quorum Q_i such that $Q_i \cap F = \emptyset$.

These two facts imply that $\overline{\mathcal{F}}$ is a classic Byzantine quorum system for $\mathcal{F}$ and so $Q^3(\mathcal{F})$ holds. ◀

5.2   Comparison with Asymmetric Fail-Prone Systems

In the asymmetric model [8], every process is free to express its own trust assumption about the processes in one common globally known system through a subjective fail-prone system.

An asymmetric fail-prone system $\mathbb{F}' = [\mathcal{F}'_1, \ldots, \mathcal{F}'_n]$ consists of an array of fail-prone systems, where $\mathcal{F}'_i \subseteq 2^{\Pi}$ denotes the trust assumption of p_i.

An *asymmetric Byzantine quorum system* for $\mathbb{F}'$ [5], is an array of collections of sets $\mathbb{Q}' = [\mathcal{Q}'_1, \ldots, \mathcal{Q}'_n]$, where $\mathcal{Q}'_i \subseteq 2^{\Pi}$ for $i \in [1, n]$. The set $\mathcal{Q}'_i \subseteq 2^{\Pi}$ is called the *quorum system of p_i* and any set $Q_i \in \mathcal{Q}'_i$ is called a *quorum (set) for p_i*. Moreover, defining $\mathcal{F}'^* = \{F' | F' \subseteq F, F \in \mathcal{F}'\}$, the following conditions hold: (Consistency) $\forall i, j \in [1, n]$, $\forall Q_i \in \mathcal{Q}'_i, \forall Q_j \in \mathcal{Q}'_j, \forall F_{ij} \in \mathcal{F}'^*_i \cap \mathcal{F}'^*_j : Q_i \cap Q_j \not\subseteq F_{ij}$; and (Availability) $\forall i \in [1, n], \forall F_i \in \mathcal{F}'_i : \exists Q_i \in \mathcal{Q}'_i : F_i \cap Q_i = \emptyset$.

The Q^3-condition in the classic model can be generalized as follows: we say that $\mathbb{F}'$ satisfies the B^3-*condition* [8], abbreviated as $B^3(\mathbb{F}')$, whenever it holds for all $i, j \in [1, n]$ that $\forall F_i \in \mathcal{F}'_i, \forall F_j \in \mathcal{F}'_j, \forall F_{ij} \in \mathcal{F}'^*_i \cap \mathcal{F}'^*_j : \Pi \not\subseteq F_i \cup F_j \cup F_{ij}$.

An asymmetric fail-prone system $\mathbb{F}'$ satisfying the B^3-condition is sufficient and necessary [5] for the existence of a corresponding asymmetric quorum system $\mathbb{Q}' = [\mathcal{Q}'_1, \ldots, \mathcal{Q}'_n]$, with $\mathcal{Q}'_i = \overline{\mathcal{F}'}_i$. Processes in this model are classified in three different types, given an execution e with faulty set A: a process $p_i \in A$ is *faulty*, a correct process p_i for which $A \notin \mathcal{F}'^*_i$ is called *naive*, while a correct process p_i for which $A \in \mathcal{F}'^*_i$ is called *wise*.

Finally, a *guild* $\mathcal{G}$ for A is a set of wise processes that contains at least one quorum for each member.

Let Π be a set of processes in the asymmetric model and $\mathbb{F}' = [\mathcal{F}'_1, \ldots, \mathcal{F}'_n]$ be an asymmetric fail-prone system. Define the function g from asymmetric fail-prone systems to PFPS such that $g(\mathbb{F}') = [(\Pi, \mathcal{F}'_1), \ldots, (\Pi, \mathcal{F}'_n)]$. Observe that, in an asymmetric system,

the failures that may be tolerated by the processes are possible subsets of fail-prone sets in the fail-prone systems of $\mathbb{F}'$ and $P_i = \Pi$ for every $p_i \in \Pi$. Moreover, as in the classic model, there is only one view, which is $\mathbb{V} = g(\mathbb{F}')$.

We define the quorum function $\mathcal{Q} : \Pi \times \Upsilon \to 2^\Pi$ such that for every guild $\mathcal{G} \subseteq 2^\Pi$, if $p_i \in \mathcal{G}$ then $\mathcal{Q}(p_i, g(\mathbb{F}')) = \{\mathcal{G}, \Pi\}$, otherwise $\mathcal{Q}(p_i, g(\mathbb{F}')) = \{\Pi\}$.

Observe that a quorum in the asymmetric model is a slice according to Definition 5 and, for every process $p_i \in \Pi$, every set in $\mathcal{Q}(p_i, g(\mathbb{F}'))$ is a quorum according to Definition 11.

Through the following theorem we establish the relationship between the asymmetric model and the permissionless model.

▶ **Theorem 31.** *Let us consider an asymmetric model among a set Π of processes with asymmetric fail-prone system $\mathbb{F}'$. If $B^3(\mathbb{F}')$ holds and Π tolerates some sets $T \subseteq \Pi$, then there exists a quorum function $\mathcal{Q}$ such that Π is a league in $g(\mathbb{F}')$.*

Proof. Let us assume that Π tolerates some sets $T \subseteq \Pi$ and let us consider the quorum function $\mathcal{Q}$ define in this section in the context of the asymmetric model. This means that, for every set T tolerated by Π, every process $p_i \in \Pi \setminus T$ has a slice contained in $\Pi \setminus T$. This implies that in every execution in which T is the set of faulty processes, every process in $\Pi \setminus T$ is wise and $\Pi \setminus T$ is a guild.

Moreover, let us also assume that $B^3(\mathbb{F}')$ holds. This implies the existence of an asymmetric Byzantine quorum system $\mathbb{Q}'$ such that for every set T tolerated by Π, for every two processes p_i and p_j in $\Pi \setminus T$ and for every two quorums $Q_i \in \mathcal{Q}'_i$ and $Q_j \in \mathcal{Q}'_j$ for p_i and p_j, respectively, it holds that $(Q_i \cap Q_j) \setminus T \neq \emptyset$. Observe that the set $\Pi \setminus T \in \mathcal{Q}(p_i, g(\mathbb{F}'))$ is a quorum in the permissionless model for every $p_i \in \Pi \setminus T$ according to Definition 11. This implies that Π satisfies availability property of a league.

Finally, for the consistency property observe that for every process $p_i \in \Pi$, the set system $\mathcal{Q}(p_i, g(\mathbb{F}'))$ satisfies Definition 11; by construction we have at most only two quorums for every p_i which are Π and $\Pi \setminus T$ both satisfying Definition 11. Consistency of $\mathbb{Q}'$ implies intersection among the quorums in $\mathcal{Q}(p_i, g(\mathbb{F}'))$, for every process in $\Pi \setminus T$.

It follows that Π is a league for the quorum function $\mathcal{Q}$ in $g(\mathbb{F}')$. ◀

Theorem 31 shows a relation between the asymmetric model and the permissionless model. In particular, if $B^3(\mathbb{F}')$ holds and Π tolerates some sets T, then the quorum function $\mathcal{Q}$ makes Π a league. However, we could have scenarios in which only a subset of Π tolerates some sets T. In particular, we have the following result.

▶ **Lemma 32.** *Let $\Pi = \{p_1, \ldots, p_n\}$ be a set of processes, $\mathbb{F}'$ be an asymmetric fail-prone system over Π and $g(\mathbb{F}')$ the corresponding PFPS as described in the text. Moreover, let us consider an execution e with set of faulty processes A with guild $\mathcal{G}$. Then, $\mathcal{G}$ is the only set that tolerates e.*

Proof. By definition of guild, every process in $\mathcal{G}$ is wise and has a quorum contained in $\mathcal{G}$. Observe that, given a wise process p_i, there exists a fail-prone set $F \in \mathcal{F}'_i$ in $\mathbb{F}'$ such that $A \subseteq F$. Moreover, a quorum Q_i for p_i in the asymmetric model satisfies Definition 5 and it is then a slice of p_i. This implies that every process in $\mathcal{G}$ has its assumptions satisfied according to Definition 3. Moreover, every process in $\mathcal{G}$ has a slice contained in $\mathcal{G}$. ◀

In the following lemma we characterize a link between the notion of a guild, in a given execution, and a league.

▶ **Lemma 33.** *Let us consider an asymmetric Byzantine quorum system $\mathbb{Q}'$ and a guild $\mathcal{G}$ in any execution with set of faulty processes A. Then, $\mathcal{G}$ is a league for the quorum function $\mathbb{Q}$ in $g(\mathbb{F}')$.*

Proof. The result follows from Theorem 31 by applying the same reasoning with $\mathcal{G}$ instead of $\Pi \setminus T$ as a guild.   ◀

In the following lemma we show a scenario where no asymmetric Byzantine quorum systems exist but it is possible to find a league for $\mathbb{Q}$ in $g(\mathbb{F}')$.

▶ **Lemma 34.** *There exists an asymmetric fail-prone system $\mathbb{F}'$ such that:*

━ *there is no asymmetric Byzantine quorum system for $\mathbb{F}'$, but*

━ *there exists a quorum function $\mathbb{Q}$ that make Π a league in $g(\mathbb{F}')$.*

Proof. We prove this lemma through an example with four processes. Consider an asymmetric fail-prone system $\mathbb{F}'_4$ over four processes p_1, p_2, p_3, and p_4 with $\mathcal{F}'_1 = \{\{p_3, p_4\}\}$, $\mathcal{F}'_2 = \{\{p_1, p_4\}\}$, $\mathcal{F}'_3 = \{\{p_1, p_4\}\}$, and $\mathcal{F}'_4 = \{\{p_1, p_2\}\}$, as in Example 4.

Observe that, by the availability property of an asymmetric Byzantine quorum system, p_1 must have a quorum in $\{p_1, p_2\}$ and p_4 must have a quorum in $\{p_3, p_4\}$. Since $\{p_1, p_2\}$ and $\{p_3, p_4\}$ are disjoint, it is impossible to satisfy the consistency property. Thus, there does not exist any asymmetric Byzantine quorum system for $\mathbb{F}'_4$. Another way to see this is by observing that $B^3(\mathbb{F}'_4)$ does not hold: $\{p_3, p_4\} \cup \{p_1, p_2\} = \Pi$.

However, as shown in Example 20, Π is a league in $g(\mathbb{F}'_4)$. So, there is no asymmetric Byzantine quorum system for $\mathbb{F}'_4$ but $\mathbb{Q}$ makes Π a league in $g(\mathbb{F}'_4)$.   ◀

5.3   Comparison with Federated Byzantine Agreement Systems

The federated Byantine agreement system (FBAS) model has been introduced by Mazières [16] in the context of the Stellar white paper. Differently from the models presented before in this section, the FBAS model is a permissionless model, where processes, each with an initial set of known processes, continuously discover new processes. In a FBAS, every process p_i chooses a set of slices, which are sets of processes sufficient to convince p_i of agreement and a set of processes Q_i is a quorum for p_i whenever p_i has at least one slice inside Q_i and every member of Q_i has a slice that is a subset of Q_i. In particular, a quorum Q_i is a quorum for every of its members. However, despite the permissionless nature of a FBAS, a global intersection property among quorums is required for the analysis of the Stellar Consensus Protocol (SCP), and the scenario with disjoint quorums is not considered by Mazières.

A central notion in FBAS is that of *intact* set; given a set of processes Π, an execution with set of faulty processes A and a set of correct processes $\mathcal{W} = \Pi \setminus A$, a set of processes $\mathcal{I} \subseteq \mathcal{W}$ is an *intact set* [9,12] when the following conditions hold: (Consistency) for every two processes p_i and p_j in $\mathcal{I}$ and for every two quorums Q_i and Q_j for p_i and p_j, respectively, $Q_i \cap Q_j \cap \mathcal{I} \neq \emptyset$; and (Availability) $\mathcal{I}$ is a quorum for every of its members.

Every process in $\mathcal{I}$ is called *intact*, while every process in $\Pi \setminus \mathcal{I}$ (correct or faulty) is called *befouled* and some properties of the Stellar Consensus Protocol are guaranteed only for intact processes. Moreover, the union of two intersecting intact sets is an intact set. Finally, by requiring a system-wide intersection among quorums (as in the case of the SCP) one obtains an unique intact set (Lemma 34, [9]).

We first show that our model generalizes the FBAS model by showing that a quorum in FBAS satisfies Definition 11.

In FBAS a notion of fail-prone system is missing and definitions are given with respect to an execution with a fixed set of faulty processes A. However, because processes define slices, an implicit fail-prone system for every process can be derived.

In particular, given a set of processes $\Pi = \{p_1, p_2, \ldots\}$, every process in Π defines its slices based on a known subset $P_i \subseteq \Pi$ by p_i and S_i is a slice for $p_i \in \Pi$ if and only if $p_i \in S_i$ and $S_i \subseteq P_i$ [9]. Let $\mathcal{S}_i \subseteq 2^{\Pi}$ be the set of slices of p_i, we can derive the following definition.

▶ **Definition 35** (Federated fail-prone system). *A set $F \subseteq \Pi$ is a* fail-prone set *of p_i if and only if there exists a slice $S_i \in \mathcal{S}_i$ of p_i such that $F = P_i \setminus S_i$. The set $\mathcal{F}_i'' \subseteq 2^{\Pi}$ of all the fail-prone sets of p_i is called* fail-prone system *of p_i. Finally, we call the set $\mathbb{F}'' = [(P_1, \mathcal{F}_1''), (P_2, \mathcal{F}_2''), \ldots]$ the* federated fail-prone system.

In a FBAS, processes discover other processes' slices during an execution and so p_i implicitly learns other processes' federated fail-prone sets. Moreover, correct processes do not lie about their slices [9, 16].

It easy to observe that given different sets of slices received from different processes, Definition 35 implies Definition 9, obtaining a notion of view $\mathbb{V}$ in the FBAS model, and, because correct processes do not lie about their slices, Definition 16. We define the set Υ' to be the set of all the possible views in the FBAS model.

Given the notion of view in the FBAS model, we define the quorum function $\mathfrak{Q} : \Pi \times \Upsilon' \to 2^{\Pi}$ such that $\mathfrak{Q}(p_i, \mathbb{V})$ contains all the sets Q_i, called quorums, with $p_i \in Q_i$ and such that every process $p_j \in Q_i$ has a slice in Q_i. So, a quorum as defined by Mazières [16] satisfies Definition 11. Finally, in the FBAS model we introduce the notion of survivor set as defined in Definition 6.

In the following theorem we show that, by assuming a stronger consistency property for a league L, i.e., that the intersection among any two quorums of any two correct processes in the league contains some correct member of the league, then L is an intact set in every execution tolerated by L.

▶ **Theorem 36.** *Let L be a league for the quorum function $\mathfrak{Q}$ and let us assume that for every set $T \subseteq \Pi$ tolerated by L, for every two T-resilient views $\mathbb{V}$ and $\mathbb{V}'$, for every two processes $p_i, p_j \in L \setminus T$, and for every two quorums $Q_i \in \mathfrak{Q}(p_i, \mathbb{V})$ and $Q_j \in \mathfrak{Q}(p_j, \mathbb{V}')$ it holds $(Q_i \cap Q_j \cap L) \setminus T \neq \emptyset$, then L is an intact set for every every set $T \subseteq \Pi$ tolerated by L.*

Proof. Let L be a league for the quorum function $\mathfrak{Q}$ and let T be a set of processes tolerated by L. If for every two T-resilient views $\mathbb{V}$ and $\mathbb{V}'$, for every two processes $p_i, p_j \in L \setminus T$, and for every two quorums $Q_i \in \mathfrak{Q}(p_i, \mathbb{V})$ and $Q_j \in \mathfrak{Q}(p_j, \mathbb{V}')$ it holds $(Q_i \cap Q_j \cap L) \setminus T \neq \emptyset$, then the consistency property of intact sets follows. The availability property of an intact set follows by observing that, in $\mathbb{F}$, the set $L \setminus T$ is a quorum for every of its members. ◀

Observe that without the stronger consistency property assumed in Theorem 36, since quorums of correct processes in L may intersect in correct processes (not necessarily in L), it may be the case that L is not an intact set.

5.4 Comparison with Personal Byzantine Quorum Systems

The personal Byzantine quorum system (PBQS) model has been introduced by Losa et al. [12] in the context of Stellar consensus aiming at removing the system-wide intersection property among quorums required by Mazières [16] for the SCP.

In the PBQS model a quorum for p_i is a non-empty set of processes Q_i such that if Q_i is a quorum for p_i and $p_j \in Q_i$, then there exists a quorum Q_j for p_j such that $Q_j \subseteq Q_i$. In other terms, a quorum Q_i for some process p_i must contain a quorum for every one of its

members. Losa et al. point out that a global consensus among processes may be impossible since the full system membership is not known by the processes, and define the notion of *consensus cluster* as a set of processes that can instead solve a *local* consensus, i.e., consensus among the processes in a consensus cluster can be solved. In particular, given an execution with set of faulty processes A, a set of correct processes $\mathcal{C}$ is a consensus cluster when the following conditions hold: (Consistency) for every two processes p_i and p_j in $\mathcal{C}$ and for every two quorums Q_i and Q_j for p_i and p_j, respectively, $Q_i \cap Q_j \not\subseteq A$; and (Availability) for every $p_i \in \mathcal{C}$ there exists a quorum Q_i for p_i such that $Q_i \subseteq \mathcal{C}$. Losa et al. prove that the union of two intersecting consensus clusters is a consensus cluster and that maximal consensus clusters are disjoint. The latter implies that maximal consensus clusters might diverge from each other.

In the following we show a relationship between the notions of league and consensus cluster. To do so, we first show that a quorum Q_i for p_i as defined in Definition 11 is also a quorum for p_i in the PBQS model.

▶ **Lemma 37.** *Let $Q_i \in \mathfrak{Q}(p_i, \mathbb{V})$ be a quorum for a process p_i in a view $\mathbb{V}$ according to Definition 11. Then Q_i is a quorum for p_i in the PBQS model.*

Proof. Definition 11 implies that Q_i is a quorum for every of its members. This means that for every process $p_j \in Q_i$, the set Q_i is a quorum for p_j such that $Q_i \subseteq Q_i$. The result follows. ◀

In the following result we show that, given a league L, for every set $T \subseteq \Pi$ tolerated by L, the set $L \setminus T$ is a consensus cluster.

▶ **Theorem 38.** *Let L be a league for the quorum function $\mathfrak{Q}$. Then, for every set $T \subseteq \Pi$ tolerated by L, the set $L \setminus T$ is a consensus cluster.*

Proof. Let L be a league for the quorum function $\mathfrak{Q}$ and let T be a set of processes tolerated by L. Lemma 37 implies that for every process p_i and for every view $\mathbb{V}$, all the quorums in $\mathfrak{Q}(p_i, \mathbb{V})$ for p_i are quorums in the PBQS model. So, the consistency and availability properties of a league imply that $L \setminus T$ satisfies the consistency and availability properties of a consensus cluster, making $L \setminus T$ a consensus cluster. ◀

6   Permissionless Reliable Broadcast

In this section we show how the Bracha broadcast [2], protocol that implements Byzantine reliable broadcast, can be adapted to work in our model. First, we introduce the following definitions and results.

▶ **Definition 39** (Blocking set). *A set $B \subseteq \Pi$ is said to* block *a process p_i if B intersects every slice of p_i.*

▶ **Definition 40** (Inductively blocked). *Given a set of processes B, the set of processes inductively blocked by B, denoted by B^+, is the smallest set closed under the following rules:*
1. $B \subseteq B^+$; *and*
2. *if a process p_i is blocked by B^+, then $p_i \in B^+$.*

As a consequence of Definition 40, given an execution, the set B^+ can be obtained by repeatedly adding to it all the processes that are blocked by $B^+ \cup B$. Eventually no more processes will be added to B^+.

Moreover, given an execution e with set of faulty processes A, if a league L tolerates A, then processes in $L \setminus A$ cannot be inductively blocked by A. This is shown in the following lemma.

▶ **Lemma 41.** *Let L be a league and T be a set tolerated by L. Then, no process in $L \setminus T$ is inductively blocked by T, i.e., $T^+ \cap (L \setminus T) = \emptyset$.*

Proof. Let us assume that $T^+ \cap (L \setminus T) \neq \emptyset$. This means that there exists a process $p_i \in L \setminus T$ that is blocked by T^+, i.e., T^+ intersects every slice of p_i, including the slice contained in the quorum $Q_i \subseteq L \setminus T$ for p_i. Clearly, $(L \setminus T) \cap T = \emptyset$, and this means that there exists a set T' with $T' \subseteq T^+ \setminus T$ such that T' intersects every slice of p_i, including the slice contained in the quorum for p_i consisting only of correct processes in L. This means that we can find a process $p_j \in T'$ with $p_j \in L \setminus T$ and p_j blocked by T. Since L is a league, process p_j must have a slice in $L \setminus T$. However, T cannot intersect every slice of p_j because $L \setminus T$ is disjoint from T. We reached a contradiction.   ◀

Intuitively, starting from $A^+ = \emptyset$, we first consider the processes that are blocked by A. Trivially, every process in A is blocked by A, and so $A^+ = A$. Moreover, no process in $L \setminus A$ can be blocked by A. If this was the case, then there would exist a process $p_i \in L \setminus A$ such that A intersected all of its slices, including the slice contained in the quorum $Q_i \subseteq L \setminus A$, which we know to exist due to the availability property of L. So, only processes p_j not in $L \setminus A$ can be blocked by A. Let p_j be such process. This means that $A \cup \{p_j\} \subseteq A^+$. Now, we can repeat the same reasoning, by considering all the processes blocked by $A \cup \{p_j\}$. Again, no processes in $L \setminus A$ can be blocked by $A \cup \{p_j\}$. In fact, if $A \cup \{p_j\}$ blocked a process $p_k \in L \setminus A$, then every slice of p_k would contain p_j, including the slice contained in $L \setminus A$. However, this would imply that $p_j \in L \setminus A$ which would contradict the fact that p_j is a process not in $L \setminus A$.

In the following theorem we show that if a correct process p_i in a league L is blocked by a set B, then $B = B \cup \{p_i\}$ blocks another process $p_j \notin B \cup A$. Then, $B' = B \cup \{p_j\}$ blocks another process $p_k \notin B' \cup A$ and so on, until, eventually, every correct process in the league is blocked.

▶ **Theorem 42** (Cascade theorem). *Consider the quorum function $\mathcal{Q}$, a league L, and a set $T \subseteq \Pi$ tolerated by L. Moreover, let us consider a process $p_i \in L \setminus T$, a T-resilient view $\mathbb{V}$ for p_i, a quorum $Q_i \in \mathcal{Q}(p_i, \mathbb{V})$, and a set $B \subseteq \Pi$ disjoint from T such that $Q_i \setminus T \subseteq B$. Then, either $L \setminus T \subseteq B$ or there exists a process $p_j \notin B \cup T$ that is blocked by B.*

Proof. It suffices to assume by contradiction that $L \setminus (B \cup T) \neq \emptyset$ and that, for every $p_j \notin B \cup T$, process p_j has a slice disjoint from B. This implies that $S = \overline{B \cup T}$ is a survivor set of every process $p_j \in S$; since $L \setminus (B \cup T) \neq \emptyset$, this includes also at least one process $p_j \in L \setminus (B \cup T)$.

Let us consider such a process $p_j \in L \setminus (B \cup T)$ and consider the view $\mathbb{V}'$ for p_j such that: (1) for every $p_k \notin T$, $\mathbb{V}'[k] = \mathbb{F}[k]$; and (2) for every $p_k \in T$, $\mathbb{V}'[k] = (\emptyset, \{\emptyset\})$. Observe that $\mathbb{V}'$ is a T-resilient view for p_j. By Lemma 12, we have that $S \in \mathcal{Q}(p_j, \mathbb{V}')$. This implies that $S \cap Q_i \subseteq T$. But combined with the fact that $p_j \in L \setminus (B \cup T)$, this contradicts the consistency property of L.   ◀

We will see how this theorem has a direct effect on the liveness of permissionless Byzantine reliable broadcast.

In a Byzantine reliable broadcast, the sender process may *broadcast* a value v by invoking *r-braodcast(v)*. The broadcast primitive outputs a value v through an *r-deliver(v)* event. Moreover, the broadcast primitive presented in this section delivers only one value per instance. Every instance has an implicit label and a fixed, well-known sender p_s.

▶ **Definition 43** (Permissionless Byzantine reliable broadcast). *A protocol for* permissionless Byzantine reliable broadcast *satisfies the following properties. For every league L and every execution tolerated by L:*

Validity: *If a correct process p_s r-broadcasts a value v, then all correct processes in L eventually r-deliver v.*

Integrity: *For any value v, every correct process r-delivers v at most once. Moreover, if the sender p_s is correct and the receiver is correct and in L, then v was previously r-broadcast by p_s.*

Consistency: *If a correct process in L r-delivers some value v and another correct process in L r-delivers some value v', then $v = v'$.*

Totality: *If a correct process in L r-delivers some value v, then all correct processes in L eventually r-deliver some value.*

We implement this primitive in Algorithm 1, which is derived from Bracha broadcast [2] but differs in some aspects.

In principle, the protocol follows the original one, but does not use one global quorum system known to all processes. Instead, the correct processes implicitly use the same quorum function $\mathcal{Q}$ (Definition 11), of which they initially only know their own entry in $\mathcal{Q}$. They discover the quorums of other processes during the execution.

Because of the permissionless nature of our model, we consider a best-effort gossip primitive to disseminate messages among processes instead of point-to-point messages.

A crucial element of Bracha's protocol is the "amplification" step, when a process receives $f + 1$ READY messages with some value v, with f the number of faulty processes in an execution, but has not sent a READY message yet. Then it also sends a READY message with v. This generalizes to receiving the same READY message with value v from a *blocking set* for p_i and is crucial for the *totality* property.

Finally, we introduce the ANY message as a message sent by a process p_i that is blocked by two sets carrying two different values v and v'. The reason for this new message lies in the consistency property of L: given an execution e with set of faulty processes A tolerated by L, the consistency property of L implies that any two quorums of any two correct processes in L have some correct process in common. Quorum intersection is then guaranteed only for correct processes in L and nothing is assured for correct processes outside L, which might gossip different values received by non-intersecting quorums. In particular, if a correct process p_i is blocked by a set containing a value v and later is blocked by a set containing a value $v' \neq v$, then p_i gossips an ANY message containing $*$. ANY messages are then ignored by correct processes in L. As we show in the Theorem 44, correct process in L cannot be blocked by sets containing different values.

▶ **Theorem 44.** *Algorithm 1 implements permissionless Byzantine reliable broadcast.*

Proof. Observe that all the properties assume the existence of a league L and an execution e with set of faulty processes A tolerated by L.

Let us start with the *validity* property. Since the sender p_s is correct and from the availability property of L, every correct process p_i in L eventually receives a quorum Q_i for itself of ECHO messages containing the value v sent from p_s and updates its view $\mathbb{V}$ according to the views received from every process in Q_i.

Algorithm 1 **Algorithm 1** Permissionless Byzantine reliable broadcast protocol for process p_i, with sender p_s.

State

    $sent\text{-}echo \leftarrow$ FALSE: indicates whether p_i has gossiped ECHO

    $echos[j] \leftarrow [\bot]$: collects the received ECHO messages from other processes

    $sent\text{-}ready \leftarrow$ FALSE: indicates whether p_i has gossiped READY

    $readys[j] \leftarrow [\bot]$: collects the received READY messages from other processes

    $sent\text{-}any \leftarrow$ FALSE: indicates weather p_i has gossiped $[\text{ANY}, *, \mathbb{F}[i]]$

    $delivered \leftarrow$ FALSE: indicates whether p_i has delivered a value

    $\mathbb{V}[j] \leftarrow$ if $i = j$ then $\mathbb{F}[i]$ else $\bot$: the current view of p_i

upon invocation $r\text{-}broadcast(v)$ **do**

    send message $[\text{SEND}, v, \mathbb{F}[s]]$ through gossip   // only sender p_s

upon receiving a gossiped message $[\text{SEND}, v, (P_s, \mathcal{F}_s)]$ from p_s **and** $\neg sent\text{-}echo$ **do**

    $sent\text{-}echo \leftarrow$ TRUE

    $\mathbb{V}[s] \leftarrow (P_s, \mathcal{F}_s)$

    send message $[\text{ECHO}, v, \mathbb{F}[i]]$ through gossip

upon receiving a gossiped message $[\text{ECHO}, v, (P_j, \mathcal{F}_j)$ from p_j **do**

    if $echos[j] = \bot$ **then**

        $\mathbb{V}[j] \leftarrow (P_j, \mathcal{F}_j)$

        $echos[j] \leftarrow v$

upon exists $v \neq \bot$ such that $\{p_j \in \Pi | echos[j] = v\} \in \mathcal{Q}(p_i, \mathbb{V})$ **and** $\neg sent\text{-}ready$ **do**

    $sent\text{-}ready \leftarrow$ TRUE

    send message $[\text{READY}, v, \mathbb{F}[i]]$ through gossip

upon receiving a gossiped message $[\text{READY}, v, (P_j, \mathcal{F}_j)]$ from p_j **do**

    if $readys[j] = \bot$ **then**

        $\mathbb{V}[j] \leftarrow (P_j, \mathcal{F}_j)$

        $readys[j] \leftarrow v$

upon exists $v \neq \bot$ such that $\{p_j \in \Pi | readys[j] = v\}$ blocks p_i **and** $\neg sent\text{-}ready$ **do**

    $sent\text{-}ready \leftarrow$ TRUE

    send message $[\text{READY}, v, \mathbb{F}[i]]$ through gossip

upon exists $v' \neq \bot$ such that $\{p_j \in \Pi | readys[j] = v'\}$ blocks p_i **and** $readys[i] = v$ **and**

        $v \neq v'$ **and** $sent\text{-}ready$ **and** $\neg sent\text{-}any$ **do**

    $sent\text{-}any \leftarrow$ TRUE

    send message $[\text{ANY}, *, \mathbb{F}[i]]$ through gossip

upon receiving a gossiped message $[\text{ANY}, *, (P_j, \mathcal{F}_j)]$ from p_j **do**

    $\mathbb{V}[j] \leftarrow (P_j, \mathcal{F}_j)$

    $readys[j] \leftarrow *$

upon exists $v \neq \bot$ such that $\{p_j \in \Pi | readys[j] = v\} \in \mathcal{Q}(p_i, \mathbb{V})$ **and** $\neg delivered$ **do**

    $delivered \leftarrow$ TRUE

    output $r\text{-}deliver(v)$

Then, p_i gossips $[\text{READY}, v, \mathbb{F}[i]]$ containing the value v and its current view $\mathbb{F}[i]$ unless $sent\text{-}ready = \text{TRUE}$. If $sent\text{-}ready = \text{TRUE}$ then p_i already gossiped $[\text{READY}, v, \mathbb{F}[i]]$.

Observe that there exists a unique value v such that if a correct process in L sends a READY message, this message contains v. In fact, if a process $p_i \in L \setminus A$ sends a READY message, either it does so after receiving a quorum Q_i for itself of ECHO messages containing v or after being blocked by a set of processes that received READY messages containing v.

In the first case, if a correct process p_i in L receives a quorum Q_i for itself of ECHO messages containing v and another correct process p_j in L receives a quorum Q_j for itself of ECHO messages containing v', by the consistency property of L, $v = v'$ and both send a READY message containing the same v.

In the second case, first observe that by Lemma 41 we know that $p_i \in L \setminus A$ cannot be inductively blocked by processes in A. Moreover, correct processes in L cannot be blocked by sets containing different values. If this was the case, then there would exist two correct processes p_i and p_j in L and two slices of p_i and p_j, respectively, in $L \setminus A$ containing two correct processes in L that received two different values v after ECHO. Again, by the consistency property of L, this is not possible. Hence, every correct process p_j in L gossips [READY, v, $\mathbb{F}[j]$]. Eventually, every correct process p_i in L receives a quorum for itself containing [READY, v, $(P_j, \mathcal{F}_j)$] messages and *r-delivers* v.

The first part of the *integrity* property is ensured by the *delivered* flag. For the second part observe that, by assumption, the receiver p_i is correct and in L. This implies that the quorum for p_i used to reach a decision contains some correct processes that have gossiped ECHO containing a value v they received from p_s.

For the *totality* property, let us assume that a correct process $p_i \in L$ *r-delivered* some value v. If $p_i \in L \setminus A$ *r-delivered* some value v, then it has received READY messages containing v from a quorum Q_i for itself. From Theorem 42 we know that exists a set B such that $Q_i \setminus A \subseteq B$ and either $L \setminus A \subseteq B$ or B blocks at least a process $p_j \in L \setminus (B \cup A)$ in an A-resilient view $\mathbb{V}'$ for p_j. In the latter case, p_j gossips a READY message containing v and B becomes $B \cup \{p_j\}$. Observe that, by assumption, if a correct process receives a gossiped message, then eventually every other correct process receives it too. Eventually, $L \setminus A$ is covered by B and this means that every correct process in L is blocked with the same value v.

Moreover, observe that given two correct processes not in L, they may become ready for different values received from non-intersecting quorums of ECHO messages. Because of this, if a correct process $p_j \notin L$ observes a blocking set B containing a value v' different from a value v that has previously gossiped in a READY message and such that *sent-any* = FALSE, process p_j gossips an ANY message containing the value $*$. Eventually every correct process p_i in L receives a quorum Q_i for itself of [READY, v, $(P_j, \mathcal{F}_j)$] messages and it *r-delivers* v.

Finally, for the *consistency* property notice that by the consistency property of L, every two quorums Q_i and Q_j of any two correct processes p_i and p_j in L intersect in some correct process p_k. Process p_k could then be outside L. If $p_k \notin L$ then, as seen for the totality property, it can be blocked by sets containing different values. If this is the case then p_k gossips an ANY message. Correct processes in L then ignore the values received from p_k and wait until receiving a quorum unanimously containing the same value v. Observe that, because L tolerates A, by availability property of L every correct process in L eventually receives a quorum made by correct processes in L. The consistency property then follows.   ◄

7   Related Work

A *fail-prone system* [14], also called adversary structure [10], is a well-adopted way to describe the failure assumptions in a distributed system. This is a collection of subsets of participants in the system that may fail together, and that are tolerated to fail, in a given execution. Fail-prone systems implicitly define *Byzantine quorum systems* [14] which are used to ensure consistency and availability to distributed fault-tolerant protocols in the presence of arbitrary failures. Originally, fail-prone systems have been expressed globally, shared by every participant in the system. Damgård et al. [8] introduce the notion of *asymmetric* fail-prone system in which every participant in the system subjectively selects its own fail-prone system, allowing for a more flexible model and where the guarantees of the system are derived from personal assumptions. This is the *asymmetric-trust model* [5]. Processes in this model are classified in three different types, *faulty*, *naive*, and *wise* and this characterization is done

with respect to an execution. In particular, given an execution e with faulty set F, a process p_i is faulty if it belongs to F, p_i is naive if it does not have F in its subjective fail-prone system and p_i is wise if F is contained in its subjective fail-prone system. Properties of protocols are then guaranteed for wise processes and, in some cases [5, 6], for a subset of the wise processes called *guild*.

Cachin and Tackmann [5] introduce *asymmetric Byzantine quorum systems*, a generalization of the original Byzantine quorum in the asymmetric-trust model. They show how to implement register abstraction and broadcast primitives using asymmetric Byzantine quorum systems. An asynchronous consensus protocol has subsequently been devised by Cachin and Zanolini [6]. Moreover, they extended the knowledge about the guild and about the relation between naive and wise processes in protocols with asymmetric trust.

As a basis for the *Stellar consensus protocol*, Mazières [16] introduces a new model called *federated Byzantine agreement* (FBA) in which participants may also lie. Here, every participant declares *quorum slices* – a collection of trusted sets of processes sufficient to convince the particular participant of agreement. These slices make a *quorum*, a set of participants that contains one slice for each member and sufficient to reach agreement. All quorums constitute a *federated Byzantine quorum system* (FBQS). In this model, even if the processes do not a priori choose intersecting quorums as in the classic model [14] or as in the asymmetric one [5, 8], an intersection property among quorums is later required for the analysis of the Stellar consensus protocol.

García-Pérez and Gotsman [9] study the theoretical foundations of a FBQS, build a link between FBQS and the classical Byzantine quorum systems and show the correctness of broadcast abstractions over federated quorum systems. Moreover, they investigate decentralized quorum constructions by means of FBQS. Finally, they propose the notion of subjective dissemination quorum system, where different participants may have different Byzantine quorum systems and where there is a system-wide intersection property. FBQS are a way towards an extension of quorum systems in a permissionless setting.

Losa et al. [12] introduce *personal Byzantine quorum systems* (PBQS) by removing from FBQS the requirement of a system-wide intersection among quorums. This might lead to disjoint *consensus clusters* in which safety and liveness are guaranteed in each of them, separately, in a given execution. Moreover, they abstract the Stellar Network as an instance of PBQS and use a PBQS to solve consensus.

8   Conclusions

This work introduces a new way of specifying trust assumptions among processes in a permissionless setting: processes not only make assumptions about failures, but also make assumptions about the assumptions of other processes. This leads to formally define the notions of permissionless fail-prone system and permissionless quorum system and to design protocols to solve known synchronization problems such as Byzantine reliable broadcast.

We introduce the notion of league, a set of processes for which consistency and availability properties hold for a given quorum function. Properties of our protocols are guaranteed assuming the existence of a league.

As a future work we plan to generalize known consensus protocols such as, for example, PBFT [7], to work in our permissionless model. We believe that, by assuming the existence of a league L, properties of consensus protocols can be guaranteed to every correct process in L.

───── **References** ─────

1 Ignacio Amores-Sesar, Christian Cachin, and Jovana Micic. Security analysis of ripple consensus. In *OPODIS*, volume 184 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

2 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987.

3 Christian Cachin, Rachid Guerraoui, and Luís E. T. Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011.

4 Christian Cachin, Giuliano Losa, and Luca Zanolini. Quorum systems in permissionless network, 2022. `doi:10.48550/arXiv.2211.05630`.

5 Christian Cachin and Björn Tackmann. Asymmetric distributed trust. In *OPODIS*, volume 153 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

6 Christian Cachin and Luca Zanolini. Asymmetric asynchronous byzantine consensus. In *DPM/CBT@ESORICS*, volume 13140 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2021.

7 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

8 Ivan Damgård, Yvo Desmedt, Matthias Fitzi, and Jesper Buus Nielsen. Secure protocols with asymmetric trust. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 357–375. Springer, 2007.

9 Álvaro García-Pérez and Alexey Gotsman. Federated byzantine quorum systems. In *OPODIS*, volume 125 of *LIPIcs*, pages 17:1–17:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

10 Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptol.*, 13(1):31–60, 2000.

11 Flavio Paiva Junqueira and Keith Marzullo. Synchronous consensus for dependent process failure. In *ICDCS*, pages 274–283. IEEE Computer Society, 2003.

12 Giuliano Losa, Eli Gafni, and David Mazières. Stellar consensus by instantiation. In *DISC*, volume 146 of *LIPIcs*, pages 27:1–27:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

13 Dahlia Malkhi, Kartik Nayak, and Ling Ren. Flexible byzantine fault tolerance. In *CCS*, pages 1041–1053. ACM, 2019.

14 Dahlia Malkhi and Michael K. Reiter. Byzantine quorum systems. *Distributed Comput.*, 11(4):203–213, 1998.

15 Dahlia Malkhi, Michael K. Reiter, and Avishai Wool. The load and availability of byzantine quorum systems. *SIAM J. Comput.*, 29(6):1889–1906, 2000.

16 David Mazières. The Stellar consensus protocol: A federated model for Internet-level consensus. Stellar, available online, `https://www.stellar.org/papers/stellar-consensus-protocol.pdf`, 2016.

17 Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, volume 91 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

18 Isaac C. Sheff, Robbert van Renesse, and Andrew C. Myers. Distributed protocols and heterogeneous trust: Technical report. *CoRR*, abs/1412.3136, 2014. `arXiv:1412.3136`.

19 Isaac C. Sheff, Xinwen Wang, Robbert van Renesse, and Andrew C. Myers. Heterogeneous paxos. In *OPODIS*, volume 184 of *LIPIcs*, pages 5:1–5:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

20 T. K. Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Comput.*, 2(2):80–94, 1987.