Brief Announcement: Weaker Assumptions for Asymmetric Trust

Christian Cachin
University of Bern, Switzerland

Juan Villacis
University of Bern, Switzerland

Abstract

In protocols with asymmetric trust, each participant is free to make its own trust assumptions about others, captured by an asymmetric quorum system. This contrasts with ordinary, symmetric quorum systems and threshold models, where trust assumptions are uniformly shared among participants. Fundamental problems like reliable broadcast and consensus are unsolvable in the asymmetric model if quorum systems satisfy only the classical properties of consistency and availability. As a result, existing solutions introduce stronger assumptions to circumvent this limitation. We show that some requirements used by state-of-the-art approaches are overly restrictive, so much so that they effectively eliminate the benefits of asymmetric trust. To address this, we propose a new approach to characterize asymmetric problems and, building upon it, present an asymmetric asynchronous unauthenticated reliable broadcast algorithm that significantly weakens the assumptions needed to solve the problem. Our techniques are general and can be readily adapted to other core problems in the asymmetric trust setting.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Asymmetric Trust, Quorum Systems, Reliable Broadcast

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.50

Funding This work was supported by the Swiss National Science Foundation (SNSF) under grant agreement Nr. 219403 (Emerging Consensus) and the Initiative for Cryptocurrencies and Contracts (IC3).

1 Introduction

Asymmetric trust models, such as those proposed by Damgård *et al.* [3], Alpos *et al.* [1], and Li, Chan, and Lesani [4] allow the development of distributed protocols in which each participant can operate under its own trust settings. These algorithms are built on top of asymmetric quorum systems, where each process independently defines its own quorums. The traditional consistency and availability properties must be satisfied by these systems, but as was shown by Li *et al.* [4], these are not enough to solve fundamental problems like reliable broadcast and consensus. Therefore, the models introduce additional assumptions. The protocols for reliable broadcast and consensus of Alpos *et al.* [1] require the existence of a *guild*, a set of processes that contains a quorum for each of its members. Li *et al.* [4] and Losa, Gafni, and Mazieres [5] identify similar conditions. These assumptions can be very strong and restrictive, and while they are sufficient to provide solutions to the aforementioned problems, we will show that they are not necessary. This leads to asymmetric algorithms that work under weaker assumptions, making such systems more flexible and usable.

We show that assumptions currently identified for algorithms such as reliable broadcast or consensus are so restrictive that it becomes unnecessary to use asymmetric trust models altogether. That is, given an asymmetric trust assumption that satisfies such requirements, we can build an equivalent symmetric trust assumption such that no process will be worse off by adopting it. This result is surprising, considering that Alpos *et al.* [1] show that asymmetric trust is more expressive than symmetric trust.

To solve this, we introduce the notion of *depth* of a process and propose a new way to characterize asymmetric problems based on it. We use this concept to provide a more general definition of asymmetric asynchronous reliable broadcast and to propose a new algorithm for this problem that foregoes the *guild* requirement in favor of a weaker assumption on *depth* .

2 Asymmetric Trust Model

For a complete presentation of the asymmetric trust model, we refer the reader to the work of Alpos *et al.* [1]. In protocols with asymmetric trust, each participant is free to make its own individual trust assumptions about others, captured by an asymmetric quorum system. This contrasts with ordinary, symmetric and threshold quorum systems, where all participants share the same trust assumptions. Given a set of processes $\mathcal{P}$, an *asymmetric fail-prone system* $\mathbb{F} = [\mathcal{F}_1, \ldots, \mathcal{F}_n]$, where $\mathcal{F}_i$ represents the trust assumptions of process p_i, captures the heterogeneous model. Each $\mathcal{F}_i$ is a collection of subsets of $\mathcal{P}$ (the set of processes) such that some $F \in \mathcal{F}_i$ with $F \subseteq \mathcal{P}$ is called a fail-prone set for p_i and contains all processes that, according to p_i, may at most fail together in some execution [3]. We can in turn proceed to define the asymmetric Byzantine quorum systems. Here and from now on, the notation $\mathcal{A}^*$ for a system $\mathcal{A} \subseteq 2^{\mathcal{P}}$, denotes the collection of all subsets of the sets in $\mathcal{A}$, that is, $\mathcal{A}^* = \{A' | A' \subseteq A, A \in \mathcal{A}\}$.

▶ **Definition 1.** *An* asymmetric Byzantine quorum system $\mathbb{Q}$ *for* $\mathbb{F}$ *is an array of collections of sets* $\mathbb{Q} = [\mathcal{Q}_1, \cdots, \mathcal{Q}_n]$ *where* $\mathcal{Q}_i \subseteq 2^{\mathcal{P}}$ *for* $i \in [1, n]$. *The set* $\mathcal{Q}_i \subseteq 2^{\mathcal{P}}$ *is a symmetric quorum system of* p_i *and any set* $Q_i \in \mathcal{Q}_i$ *is called a quorum for* p_i. *The system* $\mathbb{Q}$ *must satisfy the following two properties.*

Consistency: *The intersection of two quorums for any two processes contains at least one process for which either process assumes that it is not faulty, i.e.,*

$$\forall i, j \in [1, n], \forall Q_i \in \mathcal{Q}_i, \forall Q_j \in \mathcal{Q}_j, \forall F_{ij} \in \mathcal{F}_i^* \cap \mathcal{F}_j^* : Q_i \cap Q_j \nsubseteq F_{ij}.$$

Availability: *For any process* p_i *and any set of processes that may fail together according to* p_i*, there exists a disjoint quorum for* p_i *in* $\mathcal{Q}_i$*, i.e.,*

$$\forall i \in [1, n], \forall F_i \in \mathcal{F}_i : \exists Q_i \in \mathcal{Q}_i : F_i \cap Q_i = \emptyset.$$

Given an asymmetric quorum system $\mathbb{Q}$ for $\mathbb{F}$, an asymmetric kernel system for $\mathbb{Q}$ is defined analogously as the array $\mathbb{K} = [\mathcal{K}_1, \ldots, \mathcal{K}_n]$ that consists of the kernel systems for all processes in $\mathcal{P}$. A set $K_i \in \mathcal{K}_i$ is called a *kernel* for p_i and for each K_i it holds that $\forall Q_i \in \mathcal{Q}_i,\ K_i \cap Q_i \neq \emptyset$, that is, a kernel intersects all quorums of a process.

The existence of a valid $\mathbb{Q}$ is conditioned on $\mathbb{F}$ satisfying the B^3 condition.

▶ **Definition 2** (B^3-condition)**.** *An asymmetric fail-prone system* $\mathbb{F}$ *satisfies the* B^3-condition*, abbreviated as* $B^3(\mathbb{F})$*, whenever it holds that*

$$\forall i, j \in [1, n], \forall F_i \in \mathcal{F}_i, \forall F_j \in \mathcal{F}_j, \forall F_{ij} \in \mathcal{F}_i{}^* \cap \mathcal{F}_j{}^* : \mathcal{P} \nsubseteq F_i \cup F_j \cup F_{ij}$$

Existing work shows that an asymmetric fail-prone system $\mathbb{F}$ satisfies $B^3(\mathbb{F})$ if and only if there exists an asymmetric quorum system for $\mathbb{F}$. If $B^3(\mathbb{F})$ holds, then the canonical quorum system, defined as the complement of the asymmetric fail-prone system, is a valid asymmetric quorum system.

During an execution the set of processes that fail is denoted by F. The members of F are unknown to the processes and can only be identified by an outside observer and the adversary. A process p_i correctly foresees F if $F \in \mathcal{F}_i^*$, that is, F is contained in one of its fail-prone sets. Based on this information, it is possible to classify processes in three categories.

Faulty: a faulty process, i.e., $p_i \in F$;
Naive: a correct process p_i, i.e. $p_i \notin F$, where $F \notin \mathcal{F}_i^*$; or
Wise: a correct process p_i, i.e. $p_i \notin F$, where $F \in \mathcal{F}_i^*$

Alpos *et al.* [1] show that naive processes might affect the safety and liveness guarantees of some protocols. In order to formalize this notion, they introduced the concept of a *guild*. This concept is central for many of the algorithms they propose (e.g., reliable broadcast, binary consensus), as properties can only be ensured for executions where a guild exists.

▶ **Definition 3.** *A guild is a set of wise processes that contains one quorum for each member. Formally, a guild $\mathcal{G}$ for $\mathbb{F}$ and $\mathbb{Q}$, for an execution with faulty processes F, is a set of processes that satisfies the following two properties.*

Wisdom: *$\mathcal{G}$ is a set of wise processes, that is,*

$$\forall p_i \in \mathcal{G} : F \in \mathcal{F}_i^*.$$

Closure: *$\mathcal{G}$ contains a quorum for each of its members, that is,*

$$\forall p_i \in \mathcal{G} : \exists Q_i \in \mathcal{Q}_i : Q_i \subseteq \mathcal{G}.$$

3 Guild Assumptions are not Asymmetric

Li *et al.* [4] show that given a Byzantine asymmetric quorum system, the classical properties of consistency and availability alone do not suffice to solve fundamental problems such as reliable broadcast or consensus. The heterogeneous views of the system make it necessary assume more structure. The corresponding algorithms of Alpos *et al.* [1] require the existence of a guild. In a similar manner, Li *et al.* [4] propose that the *strong availability* property should be satisfied to be able to solve such problems. This requirement closely resembles the notion of a guild; in fact, any execution that satisfies the strong availability property is guaranteed to contain at least one guild. We explore here the implications of requiring this kind of assumptions, focusing our attention on the model of Alpos *et al.* [1].

Guilds are a very strong assumption, essentially requiring all members to have common beliefs, which goes against the spirit of asymmetry. We show that given an asymmetric fail-prone system $\mathbb{F}$, in all executions where there is a guild, it is possible to construct an equally valid symmetric trust assumption $\mathcal{F}$ from $\mathbb{F}$. As a result, asymmetric trust reduces to symmetric trust.

In the crash-fault model, Senn and Cachin [6] already show that given an asymmetric fail-prone system $\mathbb{F}'$, it is possible to construct a symmetric fail-prone system $\mathcal{F}'$ from $\mathbb{F}'$ such that if all processes adopt $\mathcal{F}'$ as their trust assumption, no process will be worse off. This result implies that there is no need for asymmetric trust in that setting. We extend this by showing that a similar scenario occurs in the Byzantine model for all executions with a guild. This effectively invalidates the advantages of asymmetric trust for the algorithms proposed by Alpos *et al.* [1], which rely on the existence of a guild.

▶ **Theorem 4.** *Let $\mathbb{F}$ be an asymmetric fail-prone system. Given an execution with faulty processes F and at least one guild $\mathcal{G}$, it is possible to construct a symmetric fail-prone system $\mathcal{F}$ derived from $\mathbb{F}$ such that $F \in \mathcal{F}^*$.*

Theorem 4 shows that for executions with a guild, every process is at least as well off using the symmetric quorum system $\mathcal{F}$ instead of $\mathbb{F}$. For a wise process p_i the situation will remain the same regardless of the quorum system chosen. With the asymmetric system

$F \in \mathcal{F}_i^*$ will hold, while the symmetric alternative satisfies $F \in \mathcal{F}^*$. For naive processes the situation improves, as now the faulty processes will be considered in their trust assumption. Thus, there is no reason for a process not to adopt the derived symmetric fail-prone system $\mathcal{F}$. One limitation of this result is that it assumes knowledge of $\mathbb{F}$, the trust assumptions for all processes. Although Alpos *et al.* [1] also make this assumption, it is not required in other asymmetric trust models [5]. We also note that knowing the faulty processes F is not required to derive the symmetric fail-prone system presented in Theorem 4.

If a guild is needed to solve a problem in the asymmetric setting, there exists a way to use existing algorithms in the symmetric model and obtain the same guarantees that an asymmetric algorithm would provide. Therefore, if a problem can only be solved with a guild, the interest in using asymmetric algorithms decreases. This motivates to look for other models and algorithms to implement reliable broadcast, consensus, and other problems, where the asymmetric algorithms known so far require a guild.

4 Depth-Characterized Reliable Broadcast

The result of Section 3 motivates the search for new guildless algorithms for problems where the only known protocols require a guild. We explore this within the context of the Byzantine reliable broadcast problem. We present a new approach that significantly weakens the assumptions needed to implement the problem and removes the reliance on guilds.

In order to do this, we introduce the notion of the *depth* of a process. Intuitively, this captures the extent to which a process can depend on others during a multi-round protocol execution. We then show how it can be used to characterize asymmetric problems in a more fine-grained approach.

▶ **Definition 5** (Depth of a process). *For an execution, we recursively define the notion of a correct process having depth d as follows:*

- *Any correct process p_i has depth 0.*
- *Additionally, a correct process p_i has depth $d \geq 1$ if*

$$\exists Q \in \mathcal{Q}_i, \forall p_j \in Q : \; p_j \text{ is correct, has depth } s, \text{ and } s \geq d-1.$$

We will center our attention on the maximal depth of a process. Note that a process with maximal depth d also has depth d' for all $0 \leq d' \leq d$. Following the terminology of Alpos *et al.* [1], naive processes have maximal depth 0, wise processes have maximal depth at least 1, and processes in a guild have maximal depth ∞.

Definition 6 presents a way to characterize reliable broadcast based on the depth of processes. Its properties are specific to processes that have at least a certain maximal depth.

▶ **Definition 6** (Depth-characterized asymmetric asynchronous Byzantine reliable broadcast). *A protocol for* depth-characterized asymmetric (Byzantine) reliable broadcast *with sender p_s and depth d, shortened as RB[d], defined through the events dar-deliver and dar-broadcast, satisfies the following properties:*

- ***Validity:*** *If a correct process p_s dar-broadcasts a message m, then all processes with depth d eventually* dar-deliver *m.*
- ***Consistency:*** *If some process with depth d dar-delivers m and another process with depth d dar-delivers m', then $m = m'$*
- ***Integrity:*** *Every process with depth d dar-delivers m at most once. Moreover, if the sender p_s is correct and the receiver has depth d, m was previously dar-broadcast by p_s*
- ***Totality:*** *If a process with depth d* dar-delivers *some message, then all processes with depth d eventually* dar-deliver *a message.*

RB[∞] gives a solution for all processes with depth ∞, while the protocol by Alpos *et al.* only guarantees a solution for processes with depth ∞ that also belong to the maximal guild. Lemma 7 shows that if an algorithm solves *RB[s]* then it also solves *RB[s′]* for all $s' \geq s$. This simplifies the search for a solution by reducing it to finding an algorithm that works for the minimal value of s. Lemma 8 shows that there are no algorithms that can solve *RB[1]*, therefore we must search for a protocol that solves the problem for processes with depth ≥ 2.

▶ **Lemma 7.** *If an algorithm solves RB[s] then it also solves RB[s′] for all $s' > s$*

▶ **Lemma 8.** *There is no algorithm that solves RB[1]*

Algorithm 1 presents a solution for *RB[3]*. Every process in the algorithm waits to receive a quorum of READYAFTERECHO messages associated to the same message m and round r before delivering m. A process with depth 3 will only receive such a quorum if there exists a set of processes Q (which form a quorum for a wise process) that can attest that the sender indeed sent the value m. Since there will be an attesting quorum for every process that delivers, and since quorums for wise processes intersect in at least one correct process, we can deduce that all processes with depth 3 that deliver a value m will deliver the same value. In addition, we use a technique similar to Bracha's kernel amplification [2] to ensure that if a valid process delivers then all valid processes will deliver. The arrays in lines 4, 5, 6, and 7 are hashmaps, so even though they are depicted as having infinite size they are actually sparsely populated. The algorithm has a latency of 3 asynchronous rounds in the best case and 5 in the worst case scenario.

▶ **Lemma 9.** *Algorithm 1 solves RB[3]*

5 Conclusion

We have shown that if a problem requires a guild assumption to be solved it is of less interest to use asymmetric trust to solve it. This arises from Theorem 4, which shows that for these cases, asymmetric trust reduces to symmetric trust.

To address this we presented a more fine-grained approach to characterize asymmetric problems, using the concept of *depth*, and showed that reliable broadcast can be solved for all processes with depth at least 3. This weakens the requirements needed to solve the problem in an asymmetric setting compared to the solution proposed by Alpos *et al.* [1], which only solves reliable broadcast for a fraction of processes with depth ∞. It is an open question if there exists an algorithm that solves *RB[2]*. These techniques can also be applied to other problems, such as binary consensus and common coin, whose existing solutions in the asymmetric trust setting rely on guilds. This approach could enable the development of new algorithms that operate under significantly weaker assumptions.

Algorithm 1 Depth-based asymmetric reliable broadcast for processes with depth 3 with sender p_s ($RB[3]$) (process p_i).

State
  $sentecho \leftarrow$ FALSE: indicates whether p_i has sent ECHO
  $echos \leftarrow [\perp]^n$: collects the received ECHO messages from other processes
  $sentrae \leftarrow [\text{FALSE}]^\infty$: indicates whether p_i has sent READYAFTERECHO in round n, a hashmap
  $sentrar \leftarrow [\text{FALSE}]^\infty$: indicates whether p_i has sent READYAFTERREADY in round n, a hashmap
  $readysafterecho \leftarrow [\perp, \perp]^{\infty \times n}$: collects READYAFTERECHO messages from other processes, a hashmap
  $readysafterready \leftarrow [\perp, \perp]^{\infty \times n}$: collects READYAFTERREADY messages from other processes, a hashmap
  $delivered \leftarrow$ FALSE: indicates whether p_i has delivered a message

upon invocation *dar-broadcast*(m) **do**
  send message $[\text{SEND}, m]$ to all $p_j \in \mathcal{P}$

upon receiving a message $[\text{SEND}, m]$ from p_s **such that** $\neg sentecho$ **do**
  $sentecho \leftarrow$ TRUE
  send message $[\text{ECHO}, m]$ to all $p_j \in \mathcal{P}$

upon receiving a message $[\text{ECHO}, m]$ from p_j **do**
  if $echos[j] = \perp$ **then**
    $echos[j] \leftarrow m$

upon exists $m \neq \perp$ **such that** $\{p_j \in \mathcal{P} | echos[j] = m\} \in \mathcal{Q}_i$ **and** $\neg sentrae[1]$ **do** // a quorum for p_i
  $sentrae[1] \leftarrow$ TRUE
  send message $[\text{READYAFTERECHO}, 1, m]$ to all $p_j \in \mathcal{P}$

upon exists $m \neq \perp, r > 0$ **such that** $\{p_j \in \mathcal{P} | readysafterecho[r][j] = m\} \in \mathcal{K}_i$ **and** $\neg sentrar[r]$ **do**
  $sentrar[n] \leftarrow$ TRUE
  send message $[\text{READYAFTERREADY}, r, m]$ to all $p_j \in \mathcal{P}$

upon receiving a message $[\text{READYAFTERECHO}, r, m]$ from p_j **do**
  if $readysafterecho[r][j] = \perp$ **then**
    $readysafterecho[r][j] \leftarrow m$

upon receiving a message $[\text{READYAFTERREADY}, r, m]$ from p_j **do**
  if $readysafterready[r][j] = \perp$ **then**
    $readysafterready[r][j] \leftarrow m$

upon exists $m \neq \perp, r > 1$ **such that** $\{p_j \in \mathcal{P} | readyafterready[r][j] = m\} \in \mathcal{Q}_i$ **and** $\neg sentrae[r]$ **do**
  $sentrae[r+1] \leftarrow$ TRUE
  send message $[\text{READYAFTERECHO}, r+1, m]$ to all $p_j \in \mathcal{P}$

upon exists $m \neq \perp, r > 0$ **such that** $\{p_j \in \mathcal{P} | readysafterecho[r][j] = m\} \in \mathcal{Q}_i$ **and** $\neg delivered$ **do**
  $delivered \leftarrow$ TRUE
  output *dar-deliver(m)*

References

1 Orestis Alpos, Christian Cachin, Björn Tackmann, and Luca Zanolini. Asymmetric distributed trust. *Distributed Comput.*, 37(3):247–277, 2024. doi:10.1007/S00446-024-00469-1.

2 Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, 1985. doi:10.1145/4221.214134.

3 Ivan Damgård, Yvo Desmedt, Matthias Fitzi, and Jesper Buus Nielsen. Secure protocols with asymmetric trust. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 357–375. Springer, 2007. doi:10.1007/978-3-540-76900-2_22.

4 Xiao Li, Eric Chan, and Mohsen Lesani. Quorum subsumption for heterogeneous quorum systems. In Rotem Oshman, editor, *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*, volume 281 of *LIPIcs*, pages 28:1–28:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.DISC.2023.28.

5 Giuliano Losa, Eli Gafni, and David Mazières. Stellar consensus by instantiation. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, volume 146 of *LIPIcs*, pages 27:1–27:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.DISC.2019.27.

6 Michael Senn and Christian Cachin. Asymmetric failure assumptions for reliable distributed systems. In Davide Frey and Gowtham Kaki, editors, *Proceedings of the 12th Workshop on Principles and Practice of Consistency for Distributed Data, PaPoC 2025, World Trade Center, Rotterdam, The Netherlands, 30 March 2025- 3 April 2025*, pages 8–14. ACM, 2025. doi:10.1145/3721473.3722143.