Real Life Is Uncertain. Consensus Should Be Too!

Reginald Frank
reginaldfrank77@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Octavio Lomeli
octavio.lomeli@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Neil Giridharan
giridhn@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Soujanya Ponnapalli
soujanya@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Marcos K. Aguilera
mkaguilera@gmail.com
Broadcom
Palo Alto, CA, USA

Natacha Crooks
ncrooks@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Abstract

Modern distributed systems rely on consensus protocols to build a fault-tolerant-core upon which they can build applications. Consensus protocols are correct under a specific failure model, where up to f machines can fail. We argue that this f-threshold failure model oversimplifies the real world and limits potential opportunities to optimize for cost or performance. We argue instead for a probabilistic failure model that captures the complex and nuanced nature of faults observed in practice. Probabilistic consensus protocols can explicitly leverage individual machine *failure curves* and explore side-stepping traditional bottlenecks such as majority quorum intersection, enabling systems that are more reliable, efficient, cost-effective, and sustainable.

CCS Concepts

• **Computer systems organization** → **Reliability**; **Availability**.

Keywords

consensus, distributed systems

ACM Reference Format:
Reginald Frank, Octavio Lomeli, Neil Giridharan, Soujanya Ponnapalli, Marcos K. Aguilera, and Natacha Crooks. 2025. Real Life Is Uncertain. Consensus Should Be Too!. In *Workshop in Hot Topics in Operating Systems (HOTOS 25), May 14–16, 2025, Banff, AB, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3713082.3730374

HOTOS 25, Banff, AB, Canada

ACM ISBN 979-8-4007-1475-7/25/05
https://doi.org/10.1145/3713082.3730374

1 Introduction

Modern distributed systems rely on a *fault tolerant core* that provides the abstraction of a single failure-free component atop which application-logic is implemented. At scale, failures are the norm, not the exception. Most cloud-native databases [23, 72], configuration managers [2, 42], decentralized platforms [6, 40, 55, 67], and the latest AI model training and serving platforms [68] build on fault-tolerant cores with consensus protocols [13, 36, 44, 50, 61] as their key building block. Consensus protocols guarantee data reliability under a specific *fault model*. A fault model captures one's belief about the world; it defines the specific assumptions made by the protocol. In the case of consensus, developers must make assumptions about the network, the type of faults, and the number of faults. The choice of fault model is crucial as it significantly impacts protocol design and performance. Much like car insurance, an overly optimistic fault model will reduce the cost of consensus but may not capture reality, causing the system to fail. At one extreme, one can, for example, consider a fault model where no faults are considered possible! In contrast, an overly pessimistic fault model may unnecessarily complicate the protocol.

Correctly capturing the reality of faults is a complex task. As such, most systems today simplify reality and guarantee correctness under the *f-threshold* model. The system is safe (all nodes agree on the committed data) and live (all new operations are eventually committed) up to f node faults. Nodes can fail either by crashing (requiring *crash fault tolerance* or CFT) or through malicious compromise where nodes can deviate arbitrarily from the protocol (requiring *Byzantine fault tolerance* or BFT). If more than f nodes fail, the system provides no guarantees. Reconfiguration can progressively replace faulty nodes by correct ones to ensure that the fault threshold is never exceeded.

This abstraction is clean but too simplistic: it hides important aspects of faults in modern systems and aligns poorly with how people reason about reliability today.

Faults are complex. The f-threshold model classifies servers as correct or faulty, and it treats faults uniformly:

each fault contributes one unit to the current fault count, without recognizing that some servers are more prone to faults than others. In reality, all servers eventually fail and each server has a unique probability of failing that depends on its type [48] or even its location in the datacenter [66]. This probability changes over time as a function of device age [25, 48], update rollouts [24], or workload shifts [77]. Moreover, server faults are neither uniform nor independent, as faults tend to cluster around software rollouts, unexpected workload shifts, or when new vulnerabilities are discovered [5, 19, 29, 53].

Guarantees are only probabilistic. Current consensus protocols report all-or-nothing guarantees: they claim to be fully safe and live with fewer than f faults, but provide no guarantees otherwise. This is unrealistic. As faults are probabilistic, it is *always* possible for the number of faults to exceed f. Thus no consensus protocol can offer a guarantee stronger than probabilistic safety or liveness. Storage systems recognise this reality already. S3, for instance, describes its guarantees [1] in terms of *nines* of availability (99.9%) and durability (99.999999999%).

In this paper, we argue that consensus protocols should acknowledge reality: in real deployments, all servers may fail, and do so with different likelihood. As such, no consensus protocol is 100% safe or live. In fact, we find that Raft, a popular CFT protocol, is only 99.97% safe and live in three node deployments when nodes suffer a 1% failure rate (§3). In our opinion, a consensus protocol should offer a specific safety (respectively liveness) *probabilistic* guarantee. This guarantee should be computed as a function of the specific protocol, and of servers' specific *fault curves*. Fault curves capture the unique, time-dependent, fault profile of a given server and can be computed using the large amount of telemetry that modern deployments track on a daily basis.

Moreover, adopting a probabilistic approach to reasoning about failure allows us to investigate an intriguing opportunity: that of leveraging the heterogeneous fault curves of different servers to provide the same probabilistic safety/liveness guarantees but at much lower dollar or energy cost! We find, for instance, that one can run Raft on nine, less reliable nodes that suffer a 8% failure rate and obtain the same 99.97% safety and liveness. If these resources are 10× cheaper (*e.g.,* spot instances [17], older hardware), this yields a 3× reduction in cost. This analysis hinges on incorporating fault curves in existing consensus protocols. We believe that one can go a step further and develop *probability-native* consensus protocols, which use fault curves to improve performance and reduce cost, by side-stepping expensive quorum intersection invariants that are essential to traditional consensus.

In the rest of this paper, we first describe the properties of real-world faults (§2). We then conduct a detailed analysis of the probabilistic safety and liveness guarantees (§3) of well-known CFT and BFT protocols [18, 61]. We conclude by asking: what are the challenges and opportunities of designing true probability-native consensus protocols (§4).

2 Faults are probabilistic

The f-threshold model discretizes reality in the name of simplicity: it assumes that at most f nodes are faulty, where correct nodes *never* fail. It further does not recognize that some servers are more likely to fail than others, as faults are treated equally: they each contribute one unit to the current fault count. We argue that this simplification is counterproductive. It hides the true nature of faults and leads to consensus protocols that 1) offer guarantees that align poorly with how people think of reliability, and 2) miss opportunities for performance optimizations. Server faults should instead be modeled as probability distributions to account for their inherent heterogeneity and dynamic nature. We refer to this as a server's *fault curve* p_u.

The extensive research on hardware faults at scale [25, 31, 37, 39, 76] can help us precisely characterize p_u. Large-scale datacenters keep detailed telemetry on the observed fault rates of their servers [25] or GPUs [49], including the number of memory faults, bit flips or block corruptions on disk drives, and how they vary based on manufacturer, write cycles, or time. Similar work exists on quantifying spot instances failure rates [4, 43, 77]. In addition to experimental results, advanced analytical studies also predict fault rates. Advanced predictive models, for instance, can estimate fault rates based on transistor and chip aging [8–10, 12]. Large organizations similarly conduct large-scale threat and risk analysis to capture the likelihood of machine compromise and attacks [70].

Large-scale fault studies draw three primary conclusions: *(1) Nodes do not fail equally.* Most consensus protocols assume all nodes are equally likely to fail. In reality, fault probability is rarely homogeneous and depends on the hardware manufacturer, [39, 48, 64], where the hardware is placed in the datacenter [66], disk capacity or usage [64, 66]. Probability of failure is not limited to machine faults but can additionally be used to capture social concepts like human trust and incentives. In the distributed trust context, a lower fault probability can be assigned to parties with whom a long-standing contractual agreement exists [58], whereas members from an enemy state may be assigned a higher fault probability. Stake in blockchain systems captures a similar idea: nodes with higher stake have more to lose if the system fails, and thus are considered more trustworthy [6, 34].

(2) Fault likelihood evolves over time. Consensus protocols do not currently reason about fault likelihood, and can thus only react to evolving fault rates by changing the threshold

f. Unfortunately, changing f is cumbersome as it requires costly reconfiguration [27]. Yet, fault probabilities evolve over time. At the software level, faults tend to cluster around major software updates as seen with the CrowdStrike debacle [24], or with peak operation hours and sudden workload changes. Hardware reliability also evolves [25, 39]. Disk failures, for instance, follow a bathtub curve: they have a high chance of failure during the *infancy* and *wear-out stage*, but comparatively lower failure rates during the *useful life stage*. Both Google and Facebook report that silent corruption errors (processors that compute data incorrectly) become more frequent as cores age [25, 39]. In the context of distributed trust, fault probability (aka trustworthiness) may evolve as a function of the geopolitical context.

(3) Faults are correlated. Protocols assume a maximum of f faults and treat faults uniformly, implicitly assuming that faults are independent. Unfortunately, faults often are correlated or planned. At the software level, they arise from periodic reboots, software rollouts, or operational updates. At the hardware level, research shows that devices used in a similar way, or placed close to each other exhibit similar fault patterns. For instance, disks that share similar vibration or temperature exhibit similar fault patterns [66]. Malicious attacks also frequently compromise *classes* of machines. Consider for instance the distributed trust setting such as Azure Confidential Ledger or Signal's key recovery service. Both use trusted hardware like Intel SGX or AMD SEV-SNP to strengthen defenses against a machine compromise. When a vulnerability is discovered in those architectures, as is sadly frequent [5, 19, 29, 53, 54, 65], the risk of platform-wide attacks grow.

(4) Faults cannot be simply treated as crashes or Byzantine. Current consensus protocols, with few exceptions [21] force developers into a stark choice: either optimistically assume that malicious faults will *never* arise, or always pay the cost that they will. In reality, most nodes fail by crashing but from time to time exhibit malicious behavior. Consider for example the *corruption execution errors* at Google and Facebook triggered by mercurial cores [25, 39]; these errors amount to Byzantine failures. They are, however, much rarer (approx. 0.01% at Google) than traditional server faults (4% Annual Failure Rate). The same rationale holds in the distributed trust setting: TEEs prevent Byzantine attacks most of the time, but undiagnosed vulnerabilities can lurk.

Reliability research in storage systems, unlike consensus protocols, has designed effective metrics to capture these patterns. Disk reliability is expressed in terms of Annual Failure Rate (AFR), often measured across a large fleet of disks [48]. The storage community relies on Markov models of their system to quantify metrics like Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF), and Mean Time to Data Loss (MTTDL) [46, 63]. In a Markov model, states capture configurations (*i.e.,* number of operational disks) and transitions resulting from disk failures, repair or recovery, with rates governed by failure probabilities (λ) and repair probabilities (μ). With steady-state probabilities, the expected values for MTTDL and MTTF guide the design of effective mechanisms for reliable systems (*e.g.,* the expected MTTDL with a striping scheme of n disks and k parity disks striping in RAID). These systems provide configurable *nines* of probabilistic guarantees to applications, in line with how people reason about reliability today. This rich knowledge of failures has been used to deploy erasure coded data in the cloud [45, 46], obtaining over 40% in disk cost savings.

We believe that a similar approach is possible in consensus. We foresee two approaches: 1) better understand and exploit the probabilistic guarantees offered by *existing* consensus protocols, and 2) revisit whether the core primitives of consensus (quorum intersection, leader election, etc.) can be redesigned to use fault curves and probabilistic guarantees. We describe each in turn.

3 Analysis of Consensus Protocols

We first analyze existing consensus protocols to understand what guarantees they offer when thinking of faults as probabilistic. For simplification, we do not consider reconfiguration (adding or removing nodes) and assume faults are independent. Rather than considering fault curves, we assume that every machine u has a constant probability p_u of failing. In this setting, there are 2^N possible combinations of machine failures (*failure configurations*). Each configuration yields a set of possible system runs, which may differ based on the scheduling of messages. We deem a configuration safe if all of its system runs ensure agreement across non-failed nodes. We consider a configuration live if in all runs, all non-failed nodes eventually commit all operations. By calculating how likely each failure configuration is, we can compute the overall probability that an algorithm guarantees safety and liveness in this specific deployment environment.

3.1 Consensus Primer

Most consensus protocols follow a similar structure. They proceed in a sequence of views, where each view is led by a distinct leader tasked with proposing client operations. Within each view, committing an operation requires progressing through a series of steps, where one or more nodes broadcast a message and wait for a set of replies (a *quorum*) before moving to the next step:

Step 1. Non-Equivocation (for BFT only). This step ensures that, within a view, at most one operation will reach agreement per each slot. This phase is only necessary to defend against Byzantine leaders, as these can send conflicting proposals to nodes in the system. The non-equivocation phase

terminates once participants obtain a non-equivocation quorum (Q_{eq}).

Step 2. Persistence. This next step ensures that any (possibly) committed operation is preserved across view and leader changes. Nodes commit operations once they obtain a persistence quorum (Q_{per}).

Step 3. View Change. When nodes detect that the consensus protocol is no longer making progress, a new leader is elected by receiving a view-change quorum (Q_{vc}). As spurious view changes can hamper liveness, most BFT consensus protocols ensure that correct nodes join new views only after hearing about those views from sufficiently many other correct nodes (a view-change trigger quorum Q_{vc_t}).

The size of each of these quorums depends on the invariants they wish to maintain and the failure models they assume. In BFT, non-equivocation quorums must intersect in at least one correct node to ensure that no two quorums can form for the same operation (a correct node will never vote for both). The view change quorum Q_{vc} and persistence quorum Q_{per} must also intersect in one correct node, thus ensuring that all committed operations will be included in the next view. To avoid spurious view changes, Q_{vc_t} must be guaranteed to include at least one correct node (correct nodes will not fabricate a view-change).

The required invariants are simpler in the CFT setting: Q_{per} need, for instance, only intersect Q_{vc} in one correct node to ensure persistence across views. Violating any of these invariants will trigger a safety violation, while failing to form any of these quorums (for instance, because too many nodes have failed) will violate liveness.

3.2 Analysis and Key Takeaways

Specialising the aforementionned invariants for PBFT and Raft, two popular BFT and CFT protocols, yields the following two theorems, for a specific failure configuration.

Theorem 3.1.
PBFT is safe iff all these conditions hold:

(1) $|Byz| < 2|Q_{eq}| - N$
(2) $|Byz| < |Q_{per}| + |Q_{vc}| - N$

PBFT is live iff all these conditions hold:

(1) $|Byz| \leq |Q_{vc\_t}| - |Q_{vc}|$
(2) $|Correct| \geq |Q_{eq}|, |Q_{per}|, |Q_{vc}|$
(3) $|Byz| < |Q_{vc\_t}|$

Safety conditions (1,2) state that quorum sizes need to be large enough to ensure intersection in at least one correct node for a system of size N. Liveness instead requires that quorums be small enough that there will always be sufficiently many correct nodes to assemble said quorums. The f-threshold model simply counts the size of quorums to determine if they intersect, but a more precise accounting is possible if we know the servers' fault probabilities.

N	$\|Q_{eq}\|$	$\|Q_{per}\|$	$\|Q_{vc}\|$	$\|Q_{vc\_t}\|$	Safe %	Live %	Safe and Live %
4	3	3	3	2	99.94%	99.94%	99.94%
5	4	4	4	2	99.9990%	99.90%	99.90%
7	5	5	5	3	99.997%	99.997%	99.997%
8	6	6	6	3	99.99993%	99.995%	99.995%

Table 1: PBFT reliability, uniform $p_u = 1\%$.

N	$\|Q_{per}\|$	$\|Q_{vc}\|$	S&L $p_u = 1\%$	S&L $p_u = 2\%$	S&L $p_u = 4\%$	S&L $p_u = 8\%$
3	2	2	99.97%	99.88%	99.53%	98.18%
5	3	3	99.9990%	99.992%	99.94%	99.55%
7	4	4	99.99997%	99.9995%	99.992%	99.88%
9	5	5	99.999998%	99.99996%	99.9988%	99.97%

Table 2: Raft reliability for uniform node failure p_u.

Theorem 3.2.
Raft is safe iff all these conditions hold:

(1) $N < |Q_{per}| + |Q_{vc}|$ *and*
(2) $N < 2|Q_{vc}|$

Raft is live iff:

(1) $|Correct| \geq |Q_{per}|, |Q_{vc}|$

Safety conditions (1) and (2) state that quorums must be large enough for any two quorums to intersect in at least one node, ensuring (1) operations persist across views and (2) a unique leader is elected.

The probability of the algorithm being safe and/or live can then simply be calculated by summing the probability of each safe (respectively live) failure configuration. Exploring how fault probabilities impact safety/liveness guarantees across different network and quorum sizes yields several interesting observations (Table 1 and 2).

Consensus is probabilistic, like it or not. f-threshold protocols assert that they are fully safe and live when $N = 3$ and $f = 1$. In reality, our analysis reveals that Raft with $N = 3$ is only 3 nines safe and live ($p_u = 1\%$) (Table 2).

Linear size quorums can be overkill. Quorums in PBFT and Raft (§3.1) grow linearly with the network size. For instance, an Q_{vc_t} quorum is at least $f + 1$ in size ($N = 100, |Q_{vc\_t}| = 34$) to ensure that at least one correct node requests a view change [15]. This is overkill: if $p_u = 1\%$, there are already ten nines of probability that a random quorum of five nodes includes at least one correct node.

Larger networks of less reliable nodes can help. We find that a three-node Raft cluster ($p_u = 1\%$) has equal safety/liveness probability as a nine node cluster with $p_u = 8\%$. If reliability is proportional to pricing (*e.g.,* Spot instances), this could yield 3× lower cost. Hardware operators can thus use this analysis to pick the most sustainable,

affordable, and/or performant hardware with no reliability trade-off. Operators could similary reuse older hardware to reduce carbon emissions while meeting the target quality-of-service (QoS) for reliability [3, 14, 38, 59, 75].

Raft and PBFT underutilize reliable nodes. Raft and PBFT are oblivious to fault curves. Consider a seven node cluster with p_u=8% nodes running Raft. This cluster is 99.88% safe (Table 2). If we replace three nodes with more reliable ones (p_u=1%)—almost half of the nodes—safety improves only to 99.98% (not shown in table). As Raft does not know which nodes are more reliable, it may persist data only on the unreliable nodes. If we required quorums to include at least one reliable node (by leveraging knowledge of fault curves), data durability would increase to 99.994%.

There is a hidden exploitable trade-off between safety and liveness. The f-threshold model hides an inherent trade-off between safety and liveness in consensus protocols. Exposing this trade-off can save resources. Consider f=1 and two PBFT systems, one with $3f$+1=4 nodes and the other with $3f$+2=5 nodes. In the f-threshold model, both systems tolerate 1 fault, so these is no reason to deploy 5 nodes. However, in the probabilistic world, our analysis finds that using 5 nodes improves PBFT safety by 42–60× with a small 1.67× decrease in liveness compared to 4 nodes (Table 1)—in fact, the 5-node system is more safe than a 7-node system, which is 40% more expensive to deploy and operate. The safety gain in the 5-node system over 4 nodes occurs because it has larger quorums. Larger quorums improve safety (better probability of intersection) but degrade liveness (fewer failures can prevent progress)—in this case just a little. The 7-node system, despite tolerating an additional failure (f=2), increases the odds of faults due to its larger size, which partly offsets its gain in safety due to tolerating more faults. The f-threshold model not only hides these insights, but also misleads.

4 A Probabilistic Vision

The takeaways from our early analysis inspire several promising directions for designing probabilistic consensus; we first outline two challenges that must be addressed to realize usable probabilistic consensus: capturing an accurate fault model and reasoning about end-to-end guarantees.

Accurate fault curves. This research hinges on the ability to accurately express, in simple terms, potentially complex fault curves. Pessimistic characterizations will hurt performance while overly simplified or optimistic ones may cause the system to break when deployed. Fault curves can be computed from telemetry, proactive monitoring for failures, studies modeling hardware faults, ... The storage community already relies on such data to model failure rates of disks [46]; they rely on realistic estimates of failure probabilities and repair or disaster recovery probabilities, to design reliability mechanisms [63]. These numbers are then used to derive metrics like MTBF or MTTDL, thus defining reliability as the expected time until "something bad happens". Consensus, in contrast, has always been designed to optimistically prevent "bad things" from ever happening. Future research should address this mismatch when formalizing fault curves. Moreover, modeling correlated failures remains an open challenge; Markov models, for instance, which are typically used to compute MTBF and MTTDL are unable to capture dependent system transitions [37, 74].

End-to-end guarantees. Applications care about end-to-end reliability guarantees, where consensus is a small part of the system. Traditional reliability guarantees [20, 30], expressed in terms of nines of *availability* or *durability*, do not align well with even the probabilistic type of safety and liveness offered by consensus. A consensus protocol that is $> 0\%$ available will ensure the system remains live. A live consensus protocol, however, might not be able to meet the availability requirements if its recovery or reconfiguration is intolerably slow. Outside of availability, an unsafe system may commit different operations at different nodes yet remain durable if both forks are preserved.

Towards Probability-native consensus. Once we have accurate fault curves, the next question becomes, how do we use them? Our preliminary analysis suggests multiple steps. First, we can incorporate fault curves and probabilistic safety/liveness into *existing* consensus protocols. For instance, we can choose quorum sizes dynamically such that they overlap with high probability. Even this seemingly simple step is non-trivial as quorums are not formed independently, but instead must intersect [41]. This dependence makes calculating the probability of this intersection significantly more challenging as traditional tools like Chernoff bounds [13, 57] no longer apply.

Second, probabilistic approaches can choose leaders among the most reliable nodes, avoiding more failure-prone nodes. This is similar in spirit to leader reputation schemes [22, 71] in the f-threshold model. Such a strategy can improve tail latency, reduce reconfiguration delays, and improve safety when nodes fail. Probabilistic approaches can be further used to design new types of failure detectors [28, 73], which are more realistic and accurate. Similarly, predictive models for node reliability enable preemptive reconfiguration, mitigating potential failures from jeopardizing safety or liveness.

Third, in deployments where nodes' reliability exceeds application requirements, probabilistic protocols can sample committees, in particular, to select only the reliable nodes.

Finally, choosing to lean in fully into the probabilistic nature of consensus allows us to explore more radical design decisions. For instance, most consensus protocols have been designed around a few fundamental concepts such as majority-based quorum intersection. Probabilistic abstractions call for re-imagining consensus beyond quorums (*e.g.,* like in Ben-Or [16] or Rabia [62]). The nature of quorum systems is, by definition, pessimistic: they guarantee that any two quorums will always intersect. In practice, however, sampling from much smaller subsets of nodes can guarantee intersection with high enough probability. Similarly, quorum systems that enforce durability are too conservative as they consider worst-case adversarial scenarios. In theory, they no longer guarantee safety if *any* combination of $|Q_{per}|$ nodes fail. But, in reality, the probability that $|Q_{per}|$ failures leads to data loss is vanishingly unlikely. For example, in a 100 node cluster where $|Q_{per}| = 10$ and $p_u = 10\%$ there is a 50% chance that $|Q_{per}|$ faults occur. However, for this situation to incur data loss, the $|Q_{per}|$ failures must perfectly overlap with the most recently formed persistence quorum which has a one in ten billion probability.

5 Related Work

Quorum Systems. Prior works [56, 60] introduce measures of load, capacity, and availability for quorums; however, they assume each node fails with equal probability. Probabilistic quorums [7, 13, 57] relax the traditional quorum intersection requirements with smaller, $O(\sqrt{N})$-sized quorums that overlap with high probability.

Committee sampling. King and Saia [47] propose a consensus mechanism that achieves $O(n^{1.5})$ communication complexity, by selecting subsets of the network contain a fraction of faulty servers representative of the entire cluster. Algorand [34] leverages Verifiable Random Functions (VRFs) for efficient and secure random committee sampling.

Refined failure and trust models. Upright [21] introduces separate thresholds for crash and Byzantine failures. Stake-based consensus protocols [32, 33, 35] assume that servers have unique stake and more than f stake will never fail simultaneously. Stellar [58] generalizes this approach to enable collective agreement among servers with differing views on stake assignment. While these more expressive failure models each address different shortcomings of the f-threshold model, they are unable to take full advantage of the rich knowledge of failures in practice to provide usable, end-to-end guarantees at low overheads.

Analysis of f-threshold systems. Zorfu [11] uses Markov analysis to study mean time to $> f$ failures in f-threshold consensus systems. However, it does not extend this analysis to mean time to data loss (MTTDL) nor does it design its consensus algorithm based on failure rates. [26, 51] analyze round complexity of f-threshold consensus protocols when the number of actual failures is less than f. [52, 69] analyze consistency and accountability when the number of actual failures exceed f but are less than $2f$.

6 Conclusion

This paper argues that the f-threshold model is well-intended but ultimately unhelpful. Instead, it argues for explicitly capturing the probabilistic, evolving nature of hardware faults, much like the storage community already does. With this shift, we envision the emergence of new, more efficient consensus protocols that better align with how people think about reliability today.

References

[1] Data Protection in Amazon S3. https://docs.aws.amazon.com/AmazonS3/latest/userguide/DataDurability.html.
[2] Production-Grade Container Orchestration — kubernetes.io. https://kubernetes.io/. [Accessed 16-01-2025].
[3] Reducing the carbon footprint of data centers — buffalo.edu. https://www.buffalo.edu/news/news-releases.host.html/content/shared/mgt/news/reducing-carbon-footprint-data-centers.detail.html. [Accessed 13-01-2025].
[4] Spot Instance interruptions - Amazon Elastic Compute Cloud — docs.amazonaws.cn. https://docs.amazonaws.cn/en_us/AWSEC2/latest/UserGuide/spot-interruptions.html. [Accessed 16-01-2025].
[5] https://community.intel.com/t5/Blogs/Products-and-Solutions/Security/Intel-Google-TDX-Security-Review/post/1471177.
[6] Ethereum. https://github.com/ethereum/, 2019.
[7] Ittai Abraham and Dahlia Malkhi. Probabilistic quorums for dynamic systems. In *Distributed Computing: 17th International Conference, DISC 2003, Sorrento, Italy, October 1-3, 2003. Proceedings 17*, pages 60–74. Springer, 2003.
[8] Mridul Agarwal, Bipul C Paul, Ming Zhang, and Subhasish Mitra. Circuit failure prediction and its application to transistor aging. In *25th IEEE VLSI Test Symposium (VTS'07)*, pages 277–286. IEEE, 2007.
[9] Hussam Amrouch. *Techniques for aging, soft errors and temperature to increase the reliability of embedded on-chip systems.* PhD thesis, Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2015, 2015.
[10] Hussam Amrouch, Javier Martin-Martinez, Victor M van Santen, Miquel Moras, Rosana Rodriguez, Montserrat Nafria, and Jörg Henkel. Connecting the physical and application level towards grasping aging effects. In *2015 IEEE International Reliability Physics Symposium*, pages 3D–1. IEEE, 2015.
[11] James W. Anderson, Hein Meling, Alexander Rasmussen, Amin Vahdat, and Keith Marzullo. Local recovery for high availability in strongly consistent cloud services. *IEEE Transactions on Dependable and Secure Computing*, 14(2):172–184, 2017.
[12] Md Toufiq Hasan Anik, Sylvain Guilley, Jean-Luc Danger, and Naghmeh Karimi. On the effect of aging on digital sensors. In *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, pages 189–194. IEEE, 2020.
[13] Diogo Avelas, Hasan Heydari, Eduardo Alchieri, Tobias Distler, and Alysson Bessani. Probabilistic byzantine fault tolerance. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, PODC '24, page 170–181, New York, NY, USA, 2024. Association for Computing Machinery.
[14] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *The datacenter as a computer: Designing warehouse-scale machines.* Springer

Nature, 2019.

[15] Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. State machine replication in the libra blockchain. *The Libra Assn., Tech. Rep*, 2019.

[16] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, page 27–30, New York, NY, USA, 1983. Association for Computing Machinery.

[17] brittanyrowe. General Availability of Spot Priority Mix for Azure Virtual Machine Scale Sets Flexible Orchestration Mode — techcommunity.microsoft.com. https://techcommunity.microsoft.com/discussions/compute/announcing-general-availability-of-spot-priority-mix-for-azure-virtual-machine-s/3765160, 2023. [Accessed 16-01-2025].

[18] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[19] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 142–157. IEEE, 2019.

[20] Asaf Cidon, Stephen Rumble, Ryan Stutsman, Sachin Katti, John Ousterhout, and Mendel Rosenblum. Copysets: Reducing the frequency of data loss in cloud storage. In *2013 USENIX Annual Technical Conference (USENIX ATC 13)*, pages 37–48, 2013.

[21] Allen Clement, Manos Kapritsos, Sangmin Lee, Yang Wang, Lorenzo Alvisi, Mike Dahlin, and Taylor Riche. Upright cluster services. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP '09, page 277–290, New York, NY, USA, 2009. Association for Computing Machinery.

[22] Shir Cohen, Rati Gelashvili, Lefteris Kokoris Kogias, Zekun Li, Dahlia Malkhi, Alberto Sonnino, and Alexander Spiegelman. Be aware of your leaders, 2021.

[23] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):1–22, 2013.

[24] CrowdStrike. Technical Details: Falcon Update for Windows Hosts | CrowdStrike — crowdstrike.com. https://www.crowdstrike.com/blog/falcon-update-for-windows-hosts-technical-details/. [Accessed 10-09-2024].

[25] Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, and Sriram Sankar. Silent data corruptions at scale. *arXiv preprint arXiv:2102.11245*, 2021.

[26] Danny Dolev, Ruediger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, October 1990.

[27] Sisi Duan and Haibin Zhang. Foundations of dynamic bft. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1317–1334, 2022.

[28] Dacfey Dzung, Rachid Guerraoui, David Kozhaya, and Yvonne-Anne Pignolet. Never say never – probabilistic and temporal failure detectors. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 679–688, 2016.

[29] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security vulnerabilities of sgx and countermeasures: A survey. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[30] Daniel Ford, François Labelle, Florentina I Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[31] Pietro Frigo, Emanuele Vannacc, Hasan Hassan, Victor Van Der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Trrespass: Exploiting the many sides of target row refresh. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 747–762. IEEE, 2020.

[32] Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 690–719. Springer, 2019.

[33] Peter Gaži, Aggelos Kiayias, and Dionysis Zindros. Proof-of-stake sidechains. Cryptology ePrint Archive, Report 2018/1239, 2018. https://eprint.iacr.org/2018/1239.

[34] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery.

[35] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.

[36] Neil Giridharan, Florian Suri-Payer, Ittai Abraham, Lorenzo Alvisi, and Natacha Crooks. Autobahn: Seamless high speed bft. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 1–23, 2024.

[37] Kevin M Greenan, James S Plank, and Jay J Wylie. Mean time to meaningless:{MTTDL}, markov models, and storage system reliability. In *2nd Workshop on Hot Topics in Storage and File Systems (HotStorage 10)*, 2010.

[38] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S Lee, David Brooks, and Carole-Jean Wu. Act: Designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 784–799, 2022.

[39] Peter H Hochschild, Paul Turner, Jeffrey C Mogul, Rama Govindaraju, Parthasarathy Ranganathan, David E Culler, and Amin Vahdat. Cores that don't count. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 9–16, 2021.

[40] Heidi Howard, Fritz Alder, Edward Ashton, Amaury Chamayou, Sylvan Clebsch, Manuel Costa, Antoine Delignat-Lavaud, Cédric Fournet, Andrew Jeffery, Matthew Kerner, Fotios Kounelis, Markus A. Kuppe, Julien Maffre, Mark Russinovich, and Christoph M. Wintersteiger. Confidential consortium framework: Secure multiparty applications with confidentiality, integrity, and high availability. *Proceedings of the VLDB Endowment*, 17(2):225–240, October 2023.

[41] Heidi Howard, Dahlia Malkhi, and Alexander Spiegelman. Flexible paxos: Quorum intersection revisited. *arXiv preprint arXiv:1608.06696*, 2016.

[42] Patrick Hunt, Mahadev Konar, Flavio P Junqueira, and Benjamin Reed. ZooKeeper: Wait-free coordination for internet-scale systems. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[43] ju shim. Use Azure Spot Virtual Machines - Azure Virtual Machines — learn.microsoft.com. https://learn.microsoft.com/en-us/azure/virtual-machines/spot-vms#eviction-policy. [Accessed 16-01-2025].

[44] Flavio P. Junqueira, Benjamin C. Reed, and Marco Serafini. Zab: High-performance broadcast for primary-backup systems. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN)*, pages 245–256, 2011.

[45] Saurabh Kadekodi. *DISK-ADAPTIVE REDUNDANCY: tailoring data redundancy to disk-reliability heterogeneity in cluster storage systems.*

PhD thesis, PhD thesis, Carnegie Mellon University, 2020.

[46] Saurabh Kadekodi, K. V. Rashmi, and Gregory R. Ganger. Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*, pages 345–358, Boston, MA, February 2019. USENIX Association.

[47] Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with bits. In *International Symposium on Distributed Computing*, pages 464–478. Springer, 2009.

[48] Andy Klein. Backblaze Drive Stats for Q1 2024 — backblaze.com. https://www.backblaze.com/blog/backblaze-drive-stats-for-q1-2024/, 2024. [Accessed 14-09-2024].

[49] Apostolos Kokolis, Michael Kuchnik, John Hoffman, Adithya Kumar, Parth Malani, Faye Ma, Zachary DeVito, Shubho Sengupta, Kalyan Saladi, and Carole-Jean Wu. Revisiting reliability in large-scale machine learning research clusters. *arXiv preprint arXiv:2410.21680*, 2024.

[50] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58, 2001.

[51] Christoph Lenzen and Sahar Sheikholeslami. A recursive early-stopping phase king protocol. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 60–69, New York, NY, USA, 2022. Association for Computing Machinery.

[52] Jinyuan Li and David Mazières. Beyond One-Third faulty replicas in byzantine fault tolerant systems. In *4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 07)*, Cambridge, MA, April 2007. USENIX Association.

[53] Mengyuan Li, Luca Wilke, Jan Wichelmann, Thomas Eisenbarth, Radu Teodorescu, and Yinqian Zhang. A systematic look at ciphertext side channels on amd sev-snp. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 337–351. IEEE, 2022.

[54] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. {CIPHERLEAKS}: Breaking constant-time cryptography on {AMD}{SEV} via the ciphertext side channel. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 717–732, 2021.

[55] Joshua Lund. Technology preview for secure value recovery, 2019.

[56] Dahlia Malkhi and Michael Reiter. Byzantine quorum systems. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 569–578, New York, NY, USA, 1997. Association for Computing Machinery.

[57] Dahlia Malkhi, Michael Reiter, and Rebecca Wright. Probabilistic quorum systems. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 267–273, 1997.

[58] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 32:1–45, 2015.

[59] Justin Meza, Qiang Wu, Sanjev Kumar, and Onur Mutlu. A large-scale study of flash memory failures in the field. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):177–190, 2015.

[60] M. Naor and A. Wool. The load, capacity and availability of quorum systems. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 214–225, 1994.

[61] Diego Ongaro and John Ousterhout. The raft consensus algorithm. *Lecture Notes CS*, 190:2022, 2015.

[62] Haochen Pan, Jesse Tuglu, Neo Zhou, Tianshu Wang, Yicheng Shen, Xiong Zheng, Joseph Tassarotti, Lewis Tseng, and Roberto Palmieri. Rabia: Simplifying state-machine replication through randomization. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, SOSP '21, page 472–487, New York, NY, USA, 2021. Association for Computing Machinery.

[63] David A Patterson, Garth Gibson, and Randy H Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 109–116, 1988.

[64] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, FAST '07, page 2, USA, 2007. USENIX Association.

[65] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. Voltjockey: Breaking sgx by software-controlled voltage-induced hardware faults. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6. IEEE, 2019.

[66] Korakit Seemakhupt, Brent E Stephens, Samira Khan, Sihang Liu, Hassan Wassel, Soheil Hassas Yeganeh, Alex C Snoeren, Arvind Krishnamurthy, David E Culler, and Henry M Levy. A cloud-scale characterization of remote procedure calls. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 498–514, 2023.

[67] Alex Shamis, Peter Pietzuch, Burcu Canakci, Miguel Castro, Cedric Fournet, Edward Ashton, Amaury Chamayou, Sylvan Clebsch, Antoine Delignat-Lavaud, Matthew Kerner, Julien Maffre, Olga Vrousgou, Christoph M. Wintersteiger, Manuel Costa, and Mark Russinovich. IA-CCF: Individual accountability for permissioned ledgers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 467–491, Renton, WA, April 2022. USENIX Association.

[68] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.

[69] Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. Bft protocol forensics. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1722–1743, New York, NY, USA, 2021. Association for Computing Machinery.

[70] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. A probabilistic relational model for security risk analysis. *Comput. Secur.*, 29(6):659–679, September 2010.

[71] Alexander Spiegelman, Balaji Arun, Rati Gelashvili, and Zekun Li. Shoal: Improving dag-bft latency and robustness, 2023.

[72] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, et al. Cockroachdb: The resilient geo-distributed sql database. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pages 1493–1509, 2020.

[73] Jing Tian, Zhi Yang, Wei Chen, Ben Y. Zhao, and Yafei Dai. Probabilistic failure detection for efficient distributed storage maintenance. In *2008 Symposium on Reliable Distributed Systems*, pages 147–156, 2008.

[74] Max Tschaikowski and Mirco Tribastone. Tackling continuous state-space explosion in a markovian process algebra. *Theoretical Computer Science*, 517:1–33, 2014.

[75] Jaylen Wang, Daniel S Berger, Fiodar Kazhamiaka, Celine Irvene, Chaojie Zhang, Esha Choukse, Kali Frost, Rodrigo Fonseca, Brijesh Warrier, Chetan Bansal, et al. Designing cloud servers for lower carbon. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 452–470. IEEE, 2024.

[76] Erci Xu, Mai Zheng, Feng Qin, Yikang Xu, and Jiesheng Wu. Lessons and actions: What we learned from 10k {SSD-Related} storage system failures. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 961–976, 2019.

[77] Fangkai Yang, Bowen Pang, Jue Zhang, Bo Qiao, Lu Wang, Camille Couturier, Chetan Bansal, Soumya Ram, Si Qin, Zhen Ma, et al. Spot virtual machine eviction prediction in microsoft cloud. In *Companion Proceedings of the Web Conference 2022*, pages 152–156, 2022.