Probabilistic Opaque Quorum Systems

Michael G. Merideth and Michael K. Reiter

March 2007
CMU-CS-07-117

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Also appears as Institute for Software Research
Technical Report CMU-ISRI-07-117

Abstract

Byzantine-fault-tolerant service protocols like Q/U and FaB Paxos that optimistically order requests can provide increased efficiency and fault scalability. However, these protocols require $n \geq 5b + 1$ servers (where b is the maximum number of faults tolerated), owing to their use of *opaque Byzantine quorum systems*; this is $2b$ more servers than required by some non-optimistic protocols. In this paper, we present a family of *probabilistic* opaque Byzantine quorum systems that require substantially fewer servers. Our analysis is novel in that it assumes Byzantine clients, anticipating that a faulty client may seek quorums that maximize the probability of error. Using this as motivation, we present an optional, novel protocol that allows probabilistic quorum systems to tolerate Byzantine clients. The protocol requires only one additional round of interaction between the client and the servers, and this round may be amortized over multiple operations. We consider actual error probabilities introduced by the probabilistic approach for concrete configurations of opaque quorum systems, and prove that the probability of error vanishes with as few as $n > 3.15b$ servers as n and b grow.

This work was partially supported by NSF grant CCF-0424422.

Keywords: Distributed systems, Byzantine fault tolerance, probabilistic quorum systems

1 Introduction

For distributed systems consisting of a large number of servers, a Byzantine-fault-tolerant replication algorithm that requires all servers to communicate with each other for every client request can be prohibitively expensive. Therefore, for large systems, it is critical that the protocol have good *fault scalability* [1]—the property that performance does not (substantially) degrade as the system size is increased—by avoiding this communication.

Byzantine-fault-tolerant service protocols must assign a total order to requests to provide replicated state machine semantics [23]. To minimize the amount of communication between servers, protocols like Q/U [1] and FaB Paxos [19] use opaque quorum systems [17] to order requests *optimistically*. That is, servers independently choose an ordering, without steps that would be required to reach agreement with other servers; the steps are performed only if servers choose different orderings. Under the assumption that servers independently typically choose the same ordering, the optimistic approach can provide better fault scalability in the common case than protocols like BFT [9], which require that servers perform steps to agree upon an ordering *before* choosing it [1]. However, optimistic protocols have the disadvantage of requiring at least $5b + 1$ servers to tolerate b server faults, instead of as few as $3b + 1$ servers, and so they cannot tolerate as many faults for a given number of servers.

In this paper, we present *probabilistic opaque quorum systems* (POQS), a new type of probabilistic quorum system [18], in order to increase the fraction of faults that can be tolerated by an optimistic approach from fewer than $n/5$ to as many as $n/3.15$. A POQS provides the same properties as the strict opaque quorum systems used by, e.g., Q/U and FaB Paxos, but is probabilistic in the sense that quorums are not guaranteed to overlap in the number of servers required to ensure safety. However, we prove that this error probability is negligible for large system sizes (for a given ratio of b to n). Application domains that could give rise to systems of such scale include sensor networks and edge services.

Byzantine clients are problematic for all probabilistic quorum systems because the combination of high fault tolerance and low probability of error that can be achieved is based on the assumption that clients choose quorums uniformly at random (and independently of other quorums and the state of the system, e.g., the values held by each server, and the identities of faulty servers). This can be seen in our results that show: (i) that probabilistic opaque quorum systems can tolerate up to $n/3.15$ faults (compared with less than $n/5$ faults for strict opaque quorum systems) assuming that all quorums are selected uniformly at random, but that the maximum fault tolerance drops to $n/4.56$ faults if Byzantine clients are allowed to choose quorums according to their own goals; and (ii) that to achieve a specified error probability for a given degree of fault tolerance, substantially more servers are required if quorums are not selected uniformly at random.

Therefore, we present a protocol with which we constrain clients to using pseudo-randomly selected access sets (sets of servers contacted in order to find quorums, c.f., [6]) of a prescribed size. In the limit, we can set the sizes of access sets to be the sizes of quorums, thereby dictating that all clients use pseudo-randomly selected quorums, and providing a mechanism that guarantees, in practice, the behavior of clients that is assumed by probabilistic quorum systems. However, as shown in Section 4.4, the notion of restricted access sets allows us a range of options in trading off the low error probability and high fault tolerance of completely random quorum selection, for

the guaranteed single-round access provided when there is an available quorum (one in which all servers respond) in every access set.

Our contributions are as follows:

- We present the first family of probabilistic opaque quorum system constructions. For each construction, we: (i) show that we are able to reduce the number of servers below the $5b + 1$ required by protocols that use strict opaque quorums, (ii) prove that it works with vanishing error probability as the system size grows, and (iii) evaluate the characteristics of its error probability over a variety of specific system sizes and configurations.

- We present the first analysis of a probabilistic quorum system that accounts for the behavior of Byzantine clients. We anticipate that a faulty client may choose quorums with the goal of maximizing the error probability, and show the effects that this may have.

- We present an access-restriction protocol that allows probabilistic quorum systems to tolerate faulty clients with the same degree of fault tolerance as if all clients were non-faulty. One aspect of the protocol is that servers work to propagate the values of established writes to each other in the background. Therefore, we provide analysis, unique to opaque quorum systems, of the number of servers that must propagate a value for it to be accepted by another server.

The remainder of this paper is organized as follows. In Section 2, we frame our contributions in the context of related work. Section 3 presents our assumptions and system model. In Section 4, we introduce probabilistic opaque quorum systems, and compare and contrast them with strict opaque quorums systems, highlighting the properties that make opaque quorums useful for optimistic protocols. We also compute upper bounds on the threshold b for each construction, and prove that the error probability goes to zero as the number of servers (and b) is increased. We present our access-restriction protocol in Section 5. In Section 6, we evaluate the error probability for actual system sizes. Finally, Section 7 concludes the paper.

2 Related Work

Strict Opaque Quorum Systems. Opaque Byzantine quorum systems were introduced by Malkhi and Reiter [17], in two variants: one in which the number of non-faulty servers in a quorum is at least half of the quorum, and the other in which the number of non-faulty servers represents a strict majority of the quorum. The first construction makes it unnecessary for the client to know the sets of servers of which the system can tolerate failure (hence the term 'opaque'), while the second construction additionally makes it possible to create a protocol that does not use timestamps. The paper also proves that $5b$ is the lower bound on the number of servers for the first version; simply changing the inequality to a strict inequality proves $5b + 1$ is the lower bound for the second. In this paper, when we refer to strict (non-probabilistic) opaque quorum systems, we are concerned with the second variant.

The constraints on strict opaque quorums have also been described in the context of consensus and state-machine-replication protocols, e.g., the Q/U [1] and FaB Paxos [19] protocols, though

not explicitly as opaque quorums. Abd-El-Malek et al. [1] provide generic (not just threshold) opaque quorum system constraints that they prove sufficient for providing state-machine replication semantics where both writes and reads complete in a single (pipelined) phase when there is no write–write contention. Martin and Alvisi [19] use an opaque quorum system of acceptors in FaB Paxos, a two-phase consensus protocol (with a designated proposer) and three-phase state-machine-replication protocol requiring at least $5b + 1$ servers.

Probabilistic Quorum Systems. A Probabilistic Quorum System (PQS), as presented by Malkhi et al. [18], can provide better availability and fault tolerance than strict quorum systems can provided; Table 1 compares probabilistic quorums with their strict quorum counterparts.[1] Malkhi et al. provide constructions for dissemination and masking quorums, and prove properties of load and availability for these constructions. They do not address opaque quorum systems, or the effects of concurrent or Byzantine writers; we address each of these. In addition, in Section 4, we borrow analysis techniques from [18], but our analysis is more general in the sense that clients are not all assumed to communicate only with quorums of servers. We also use a McDiarmid inequality [20] for bounding the error probability; this provides a simpler bounding technique for our purposes than do the Chernoff bounds used there. The technique that we present in Section 5 for restricting access to limited numbers of servers should be applicable to the constructions of Malkhi et al. equally well.

Signed Quorum Systems. Signed Quorum Systems [25] are another attempt to weaken the requirements of strict quorum systems. A quorum in a signed quorum system (SQS) can include both servers that respond and servers that are polled but do not respond (and are, therefore, believed by the client to have failed); if a server responds in one quorum but is marked as failed in a different quorum, the quorums are said to *mismatch* for that server. A SQS is constructed such that if any two quorums do not overlap in a server that responds to both quorum accesses, the quorums must have at least 2α mismatches (this is known as the *dual-overlap* property). Then, given the assumptions that it is rare for any two clients to see a mismatch for a given server, i.e., that a mismatch occurs with probability at most ϵ, and that the probability of a mismatch for a given server is independent of that for any other server, the probability of two quorums not overlapping (and

[1] The $2.62b$ lower bound for masking quorums is not shown in [18], but can be quickly derived using our results from Section 4.

Table 1: Minimum servers needed for probabilistic and strict quorum variants.

	prob.	strict	presented
Opaque	$3.15b + 1$	$5b + 1$	Here
Masking	$2.62b + 1$	$4b + 1$	[18]
Dissemination	$b + 1$	$3b + 1$	[18]

hence mismatching in at least 2α servers) is at most $1 - \epsilon^{2\alpha}$. While signed quorums are related to probabilistic quorums, they have not been studied in the context of Byzantine faults. Here, in our analysis of probabilistic quorums, we find that tolerance of Byzantine faults substantially alters both the analysis techniques needed and the outcomes that result.

k-quorums. k-quorums [2] also weaken the requirements of strict quorum systems in an effort to provide greater availability, but focus on offering a property called *bounded staleness* that ensures (with certainty, as opposed to with high probability) that a read will receive one of the last k writes, even if messages may be delivered according to the choices of an unconstrained adversary. This is achieved by requiring that the union of the last k writes intersects any read quorum. k-quorums have recently been extended to support Byzantine failures of servers, and multi-writer protocols [3]. However, as we are not concerned with the bounded staleness property that is central to k-quorums, our results are orthogonal and different from those. Moreover, our results include treatment of Byzantine failures of clients.

Tolerating Byzantine Clients. No prior work on any of the three types of non-strict quorum systems listed above considers Byzantine clients. There has been work on strict quorum systems that can tolerate Byzantine clients (e.g., [15, 8]) but this is fundamentally unconcerned with the way in which quorums are chosen because such choices cannot impact the correctness of strict quorum systems.

3 System Model and Definitions

We assume a system with a set U of servers, $|U| = n$, and an arbitrary but bounded number of clients. Clients and servers can fail arbitrarily (i.e., Byzantine [14] faults). We assume that up to b servers can fail, and denote the set of faulty servers by B, where $B \subseteq U$. Any number of clients can fail. Failures are permanent. Clients and servers that do not fail are said to be *non-faulty*. We allow that faulty clients and servers may collude, and so we assume that faulty clients and servers all know the membership of B (although non-faulty clients and servers do not). We make the standard assumption that nodes are computationally bound such that they cannot subvert the effectiveness of cryptographic primitives.

Throughout the paper, we use San Serif font to denote random variables, uppercase *ITALICS* for set-valued constants, and lowercase *italics* for integer-valued constants. In the literature, *random variables* are sometimes restricted only to functions that output real numbers; for clarity in distinguishing between either fixed sets or permutations and those that are sampled from a probability distribution, we use the term random variable to refer also to a function on a sample space that outputs either sets or permutations.

The remainder of this section is concerned with (i) our assumptions on the behavior of clients, which lead to a specification of the error probability; and (ii), our assumptions on the delivery of messages.

3.1 Behavior of Clients

We abstractly describe client operations as either *writes* that alter the state of the service or *reads* that do not. Informally, a non-faulty client performs a write to update the state of the service such that its value (or a later one) will be observed with high probability by any subsequent operation; a write thus successfully performed is called "established" (we define established more precisely below). A non-faulty client performs a read to obtain the value of the latest established write, where "latest" refers to the value of the most recent write preceding this read in a linearization [11] of the execution. Therefore, we define the *correct* value for the read to return to be the value of this latest established write; other values are called *incorrect*. We assume that the read and write operations by non-faulty clients take the following forms:

- **Writes:** To perform a write, a non-faulty client selects a *write access set* $A_{\mathrm{wt}} \subseteq U$ of size a_{wt} uniformly at random and attempts to inform all servers in A_{wt} of the write value. Formally, the write is *established* once all non-faulty servers in some set $Q_{\mathrm{wt}} \subseteq A_{\mathrm{wt}}$ of size $q_{\mathrm{wt}} \leq a_{\mathrm{wt}}$ servers have *accepted* this write. (Intuitively, an access set is a set of servers contacted in order to find a live quorum, c.f., [6].) We refer to q_{wt} as the *write quorum size*; to any $Q_{\mathrm{wt}} \subseteq U$ of that size as a *write quorum*; and to $\mathcal{Q}_{\mathrm{wt}} = \{Q_{\mathrm{wt}} \subseteq U : |Q_{\mathrm{wt}}| = q_{\mathrm{wt}}\}$ as the *write quorum system*.

- **Reads:** To perform a read, a non-faulty client selects a *read access set* A_{rd} of size a_{rd} uniformly at random and attempts to contact each server in A_{rd} to learn the value that the server last accepted. We denote the minimum number of servers from which a non-faulty client must receive a response to complete the read successfully by $q_{\mathrm{rd}} \leq a_{\mathrm{rd}}$. We refer to q_{rd} as the *read quorum size*; to any $Q_{\mathrm{rd}} \subseteq U$ of that size as a *read quorum*; and to $\mathcal{Q}_{\mathrm{rd}} = \{Q_{\mathrm{rd}} \subseteq U : |Q_{\mathrm{rd}}| = q_{\mathrm{rd}}\}$ as the *read quorum system*.

In a read operation, we refer to each response received from a server in A_{rd} as a *vote* for a read value. We assume that votes for two read values that result from any two distinct write operations are distinguishable from each other, even if the corresponding write values are the same (this is discussed in Section 5). The read operation discerns the correct value from these votes in a protocol-specific way. It is possible in an optimistic protocol such as Q/U [1], for example, that the (at least q_{rd}) votes may reflect a write operation but not provide enough evidence to determine whether that write is established. In this case, the reader may itself establish, or *repair*, the write value before returning it, to ensure that a subsequent reader returns that value, as well (which is necessary to achieve linearizability). In such a protocol, the reader does so by copying its votes for that value to servers, in order to convince them to accept that write.

For this reason, the correctness requirements for POQS discussed in Section 4 treat not only the number of votes that a non-faulty reader observes for the correct value, but also the number of votes that a faulty client can gather for a *conflicting* value. A conflicting value is a specific type of incorrect value characterized by the property that a non-faulty server would accept either it or the correct value, but not both. Two values may conflict because, e.g., they both bear the same timestamp, or are "conditioned on" the same established write in the sense used in Q/U. We assume that this timestamp or similar information can be used to distinguish older (stale) values

from newer values. Enabling a faulty client to obtain sufficiently many votes for a conflicting value would, e.g., enable it to convince other non-faulty servers to accept the conflicting value via the repair protocol, a possibility that must be avoided for correctness.

Consequently, an *error* is said to occur when a non-faulty client fails to return the correct value or a faulty client obtains sufficiently many votes for a conflicting value. This definition (or specifically "sufficiently many") will be made more precise in Section 4.5. The *error probability* then refers to the probability of an error when the client (non-faulty or faulty) reads from a read access set A_{rd} chosen uniformly at random. While we cannot force a faulty client to choose A_{rd} uniformly at random, in Section 5 we demonstrate an access protocol that enables a faulty client to assemble votes for a value that can be verified by servers (and hence, e.g., to perform a repair in Q/U) only if A_{rd} was selected uniformly at random, which is good enough for our purposes. So, from here forward, we restrict our attention to read access sets chosen in this way.

3.2 Communication

The communication assumptions we adopt are common to prior works in probabilistic [18] and signed [25] quorum systems: we assume that each non-faulty client can successfully communicate with each non-faulty server with high probability, and hence with all non-faulty servers with roughly equal probability. This assumption is in place to ensure that the network does not significantly bias a non-faulty client's interactions with servers either toward faulty servers or toward different non-faulty servers than those with which another non-faulty client can interact. Put another way, we treat a server that can be reliably reached by none or only some non-faulty clients as a member of B.

This assumption enables us to refine the read protocol of Section 3.1 in a straightforward way so that non-faulty clients choose read quorums from an access set uniformly at random. (More precisely, a faulty server can bias quorum selection away from quorums containing it by not responding, but this decreases the error probability, and so we conservatively assume that non-faulty clients select read quorums at random from their access sets.) However, because a write is, by definition, established once all of the non-faulty servers in any write quorum within A_{wt} have accepted it, the write quorum at which a write is established contains all servers in $A_{\mathrm{wt}} \cap B$; i.e., only the the non-faulty servers within the write quorum are selected uniformly at random by a non-faulty client.

The access-restriction protocol of Section 5 requires no communication assumptions beyond those of the probabilistic quorums it supports.

4 Probabilistic Opaque Quorum Systems

In this section, we present a family of probabilistic opaque quorum systems. We begin by reviewing the properties of strict opaque quorum systems (Section 4.1) and modeling the worst-case behavior of faulty clients (Section 4.2). Using this, we derive a constraint (PO-Consistency, Section 4.3) that determines the maximum fraction of faulty servers that can be tolerated (Section 4.4). We

prove that the error probability goes to zero as n (and b) is increased if this constraint is satisfied (Section 4.5).

4.1 Properties of Opaque Quorums

As an introduction to probabilistic opaque quorum systems, we begin by reviewing the concepts of strict opaque quorum systems [17]. Define the following functions:

$$\textbf{correct}(Q_{\mathrm{rd}}, Q_{\mathrm{wt}}) \quad : \quad |(Q_{\mathrm{rd}} \cap Q_{\mathrm{wt}}) \setminus B| \tag{1}$$

$$\textbf{conflicting}(Q_{\mathrm{rd}}, Q_{\mathrm{wt}}) \quad : \quad |(Q_{\mathrm{rd}} \cap B) \cup (Q_{\mathrm{rd}} \setminus Q_{\mathrm{wt}})| \tag{2}$$

correct$(Q_{\mathrm{rd}}, Q_{\mathrm{wt}})$ returns the number of non-faulty servers in the intersection of a pair of read and write quorums, while **conflicting**$(Q_{\mathrm{rd}}, Q_{\mathrm{wt}})$ returns the other servers in the read quorum, all of which may return a conflicting value in some protocol execution. Let a read operation return a value that receives at least r votes. Then, the consistency property for strict opaque quorum systems is as follows:

$$\textbf{O-Consistency} : \forall Q_{\mathrm{rd}} \in \mathcal{Q}_{\mathrm{rd}}, \forall Q_{\mathrm{wt}} \in \mathcal{Q}_{\mathrm{wt}} : \textbf{correct}(Q_{\mathrm{rd}}, Q_{\mathrm{wt}}) \geq r > \textbf{conflicting}(Q_{\mathrm{rd}}, Q_{\mathrm{wt}}). \tag{3}$$

The property states that the number of non-faulty servers in the intersection of any read quorum and write quorum must represent a majority of the read quorum. Because of this and the fact that newer values can be distinguished from older values, the correct value—which, by definition, is established by being written to all of the non-faulty servers in a write quorum—can be distinguished from other values, even if some non-faulty servers (and all faulty servers) present conflicting or stale values. At a high level, O-Consistency guarantees:

P1 No two conflicting writes are both established.

P2 Every read observes sufficiently many votes for the correct value to identify it as such.

P3 No (non-faulty or faulty) reader obtains votes for a conflicting value sufficient to repair it successfully.

P1 ensures that there is a single correct value to return. P2 ensures that a read by a non-faulty client always returns the correct value. Finally, P3 ensures that no faulty client is able to gather a majority of votes for a conflicting value as a result of a read. Note that the original statement of O-Consistency [17] considers intersection of any two quorums, not just a read quorum and a write quorum, as the original formulation did not consider multiple quorum types. Our revised statement also implies constraints on the intersection of write quorums sufficient to guarantee P1 (as shown in Appendix A), as well as P2 and P3.

The minimum number of servers required by strict opaque quorum systems is bounded by a worst case scenario (Figure 1(a)), in which b servers are faulty, and all are in the intersection of a given read quorum and write quorum. In this case, the minimum number of non-faulty servers in the intersection is $q_{\mathrm{rd}} + q_{\mathrm{wt}} - n - b$. This means that, if all quorums are of size $n - b$ (the maximum

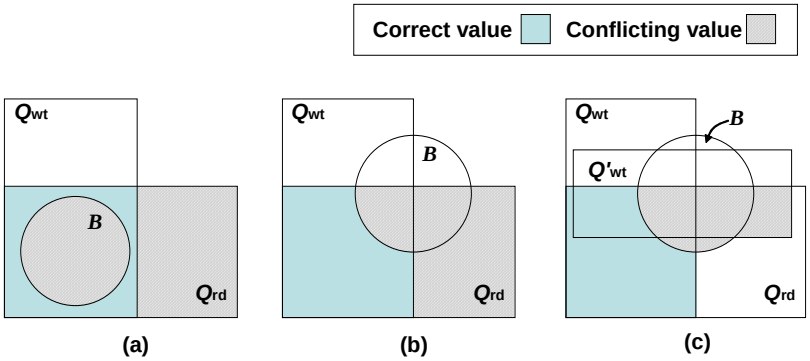


Figure 1: Servers that return a given conflicting value in Q_{rd}: (a) Worst case; (b) Typical case; (c) Restricted writers.

size that guarantees liveness in an asynchronous system), then (3) requires: $2(n-b)-n-b > (n-b)/2$, or $n > 5b$.

Given that the stated assumptions of a strict opaque quorum system hold, the system behaves correctly. In contrast to this, probabilistic opaque quorum systems (POQS) allow for a (small) possibility of error. Informally, this can be thought of as relaxing O-Consistency so that a variant of it holds for most—but not all—quorums. To ensure that the probability of an error happening is small, POQS are designed so that P1, P2, and P3 hold with high probability.

Figure 1(b) illustrates a more typical scenario, in which the faulty servers are partly in the intersection, partly in each quorum, and partly in neither quorum. This is one feature upon which we rely to reduce the total number of servers required by the worst-case scenario and O-Consistency. In addition, because the focus of POQS is the expected case instead of the worst case, we further reduce the expected number of servers that return a conflicting value by assuming that each write request can be sent to only a limited number of servers (this can be enforced by the protocol of Section 5). Figure 1(c) shows an example of this, in which the conflicting value is written to Q'_{wt}, and therefore accepted only by the servers in $Q'_{\mathrm{wt}} \setminus (Q_{\mathrm{wt}} \setminus B)$.

4.2 Behavior of Faulty Clients

Because a faulty client can behave arbitrarily, we examine the way that a faulty client should choose quorums to maximize the chance of error. During a write, a faulty client seeks a write quorum that violates P1 or that maximizes the probability that P2 is violated on a subsequent read by a non-faulty client or that P3 is violated on a subsequent read by any client. During a read, such a client seeks a quorum that violates P3 to use for repair.

Throughout this section, let A_{wt} denote a write access set from which Q_{wt} (a quorum used for an established write) is selected by a faulty client, let A'_{wt} be a write access set used for a conflicting write by a faulty client, and let A_{rd} be a read access set from which Q_{rd}, a read quorum, is selected by a faulty client. Again, we assume that A_{wt}, A'_{wt}, and A_{rd} are selected uniformly at random, an assumption that can be enforced using the protocol of Section 5.

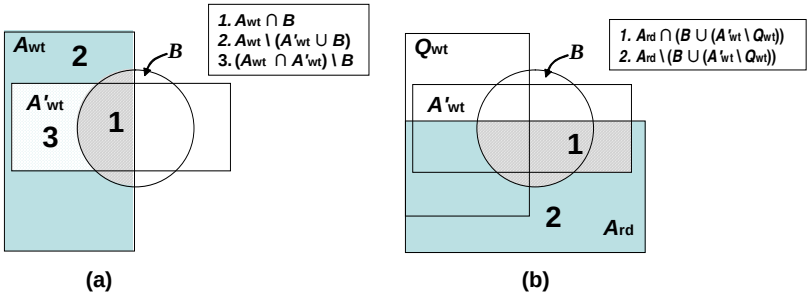


Figure 2: The preference (1st, 2nd, 3rd) a faulty client gives to a server when choosing (a) $Q_{\rm wt}$, or (b) $Q_{\rm rd}$.

Writes by a faulty client. A faulty client can increase the error probability with a write in one of two ways: (i) by establishing a write at a write quorum that contains as many faulty servers as possible, or (ii) by performing the write of a conflicting value in a way that maximizes the number of non-faulty servers that accept it, i.e., by writing to all of $A'_{\rm wt} \setminus Q_{\rm wt}$. Since a faulty client may perform *both* such writes, we assume that this client has knowledge of $A_{\rm wt}$ and $A'_{\rm wt}$ simultaneously. However, it is important to note that a faulty client does not have knowledge of the read access set $A'_{\rm rd}$ used by a non-faulty client—or specifically the non-faulty servers within it, i.e., $A'_{\rm rd} \setminus B$—and so $Q_{\rm wt}$ is chosen independently of $A'_{\rm rd} \setminus B$.[2]

Figure 2(a) shows the preferences that a faulty client gives to servers when choosing $Q_{\rm wt}$ to do both (i) and (ii). Goal (i) requires maximizing $|Q_{\rm wt} \cap B|$ to maximize the probability that P1 or P2 is violated; hence, first preference is given to the servers in $A_{\rm wt} \cap B$ in a write. Goal (ii) requires minimizing $|(Q_{\rm wt} \cap A'_{\rm wt}) \setminus B|$ to maximize the probability that P1 or P3 is violated; hence, the servers in $(A_{\rm wt} \cap A'_{\rm wt}) \setminus B$ are avoided to the extent possible.

Reads by a faulty clients. A faulty client can increase the probability that P3 is violated by choosing a read quorum with the most faulty servers and non-faulty servers that share the same conflicting value. Figure 2(b) shows the preferences that a faulty client gives to servers to do so. Because a faulty client can collude with the servers in B, it can obtain replies from all servers in B that are also in $A_{\rm rd}$, i.e., the servers in $A_{\rm rd} \cap B$. It can also wait for responses from all of the non-faulty servers in $A_{\rm rd}$ with the conflicting value, i.e., those in $A_{\rm rd} \cap (A'_{\rm wt} \setminus Q_{\rm wt})$. Only after receiving all such responses, and only if these responses number fewer than $q_{\rm rd}$, must it choose responses from servers with other values.

4.3 Probabilistic Constraint

In this section, we present PO-Consistency, a constraint akin to O-Consistency specified in terms of expected values for POQS. As detailed below, let MinCorrect be a random variable for the

[2]More precisely, with the access protocol in Section 5, $A'_{\rm rd}$ can be hidden unless, and until, that read access set is used for repair, at which point it is too late for faulty clients to choose $Q_{\rm wt}$ so as to induce an error in that read operation.

minimum number of non-faulty servers that report the correct value in a randomly chosen read quorum taken by a non-faulty client. (Recall that an error is caused by MinCorrect being too small only for reads performed by a non-faulty client.) Also, let MaxConflicting be a random variable for the maximum number of servers that report a conflicting value in a read quorum taken from a randomly chosen read access set by a faulty client that seeks to maximize MaxConflicting. (Recall that an error is caused by MaxConflicting being too large even if the client is faulty.) Then the consistency property for POQS is:

$$\textbf{PO-Consistency} : \mathbb{E}\left[\mathsf{MinCorrect}\right] > \mathbb{E}\left[\mathsf{MaxConflicting}\right]. \tag{4}$$

As shown in Section 4.5, PO-Consistency allows us to choose a threshold, r, for the number of votes used to determine the result of a read operation, while ensuring that the error probability vanishes as we increase n (and b). While this does not guarantee O-Consistency, i.e., it may be possible that P1, P2, or P3 does not hold, the probability that O-Consistency is violated goes to zero as n (and b) is increased.

We now derive expressions for MinCorrect and MaxConflicting. Recall that B is the set of up to b faulty servers. Let A_{wt} be a randomly chosen write access set, and let A_{rd} be a randomly chosen read access set. As stated in the system model, a write to A_{wt} is established once it has been accepted by all of the non-faulty servers in any Q_{wt}, a write quorum within A_{wt}. Therefore, we conservatively assume that the number of faulty servers in Q_{wt} is:

$$\mathsf{MalWrite} = |\mathsf{A}_{\mathrm{wt}} \cap B|. \tag{5}$$

Here, A_{wt} is a random variable taking on a write access set chosen uniformly at random from $\mathcal{A}_{\mathrm{wt}}$.

Q_{wt} also contains $q_{\mathrm{wt}} - \mathsf{MalWrite}$ non-faulty servers, not necessarily chosen at random, in addition to the MalWrite faulty servers. Let C_{wt} represent these non-faulty servers:

$$\mathsf{C}_{\mathrm{wt}} = \mathsf{Q}_{\mathrm{wt}} \setminus B, \tag{6}$$

$$|\mathsf{C}_{\mathrm{wt}}| = q_{\mathrm{wt}} - \mathsf{MalWrite}, \tag{7}$$

where Q_{wt} is a random variable taking on the write quorum at which the write is established, and C_{wt} is a random variable taking on the set of non-faulty servers within this write quorum. Then, the number of non-faulty servers that return the correct value in a read quorum selected by a non-faulty client is,

$$\mathsf{MinCorrect} = |\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{C}_{\mathrm{wt}}|, \tag{8}$$

where Q_{rd} is a random variable taking on a read quorum chosen uniformly at random from A_{rd}, itself chosen uniformly at random from $\mathcal{A}_{\mathrm{rd}}$.

A faulty client may select its read quorum, Q_{rd}, to maximize the number of votes for a single conflicting value in an attempt to invalidate P3. Therefore, as described in Section 4.2, the client first chooses all faulty servers in A_{rd}. The number of such servers is,

$$\mathsf{Malevolent} = |\mathsf{A}_{\mathrm{rd}} \cap B|. \tag{9}$$

The faulty client also chooses the non-faulty servers that vote for the conflicting value that is most represented in A_{rd}; these servers are a subset of $(A_{\mathrm{rd}} \setminus (C_{\mathrm{wt}} \cup B))$. This conflicting value has an associated write access set A'_{wt} chosen uniformly at random from $\mathcal{A}_{\mathrm{wt}}$, and no vote from a non-faulty server not in A'_{wt} will be counted among those for this conflicting value (because votes for any two write operations are distinguishable from each other as discussed in Section 3.1). Let $\mathsf{A}'_{\mathrm{wt}}$ be a random variable taking on A'_{wt}. Then, the number of non-faulty servers in A_{rd} that vote for this conflicting value is,

$$\mathsf{Conflicting} = |\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \setminus (\mathsf{C}_{\mathrm{wt}} \cup B))|. \tag{10}$$

A faulty client can choose all of these servers for Q_{rd}. Therefore, since the sets of servers measured by Malevolent and Conflicting are disjoint (the former consists solely of faulty servers; the latter solely of non-faulty servers), the maximum number of instances of the same conflicting value that a faulty client will select for Q_{rd} is,

$$\mathsf{MaxConflicting} = \mathsf{Malevolent} + \mathsf{Conflicting}. \tag{11}$$

4.4 Minimum System Sizes

In this section, we consider PO-Consistency under various assumptions concerning the sizes of access sets and quorums in order to derive the maximum fraction of faults that can be tolerated with decreasing error probability as a function of n (and b). Our primary result is Theorem 4.4 which provides an upper bound on b for which PO-Consistency holds. It is derived using the expectations of MinCorrect and MaxConflicting that are computed using the worst-case behavior of faulty clients presented in Section 4.2; these expectations are given in Lemmas 4.1, 4.2, and 4.3.

Lemma 4.1.

$$\mathbb{E}\left[\mathsf{MinCorrect}\right] = \frac{q_{\mathrm{rd}}(nq_{\mathrm{wt}} - a_{\mathrm{wt}}b)}{n^2}. \tag{12}$$

Proof. We compute $\mathbb{E}\left[\mathsf{MinCorrect}\right]$ taking into consideration the potential behavior of faulty clients described in Section 4.2. To compute $\mathbb{E}\left[\mathsf{MinCorrect}\right]$, we begin with $\mathbb{E}\left[\mathsf{MalWrite}\right]$. From the definition of MalWrite (5), we see that MalWrite is a hypergeometric random variable, characterized by a_{wt} draws from a population of n elements containing b successes. Therefore,

$$\mathbb{E}\left[\mathsf{MalWrite}\right] = \frac{a_{\mathrm{wt}}b}{n}. \tag{13}$$

Then, considering the definition of MinCorrect (8), we see that Q_{rd} is selected independently of C_{wt}; therefore, $\mathsf{MinCorrect} \mid \mathsf{MalWrite} = m$ is a conditional hypergeometric random variable characterized by q_{rd} draws from a population of n elements containing $q_{\mathrm{wt}} - m$ successes. Therefore,

by the rules of conditional expectation (e.g., see [21, Theorem 2.7]) and linearity of expectation, we have that,

$$\begin{aligned}
&\mathbb{E}\left[\mathsf{MinCorrect}\right] \\
&= \sum_{m} \mathbb{E}\left[\mathsf{MinCorrect} \mid \mathsf{MalWrite} = m\right] \Pr[\mathsf{MalWrite} = m] \\
&= \mathbb{E}\left[\ \mathbb{E}\left[\mathsf{MinCorrect} \mid \mathsf{MalWrite}\right]\ \right] \\
&= \mathbb{E}\left[\frac{q_{\mathrm{rd}}}{n}\left(q_{\mathrm{wt}} - \mathsf{MalWrite}\right)\right] \\
&= \frac{q_{\mathrm{rd}} q_{\mathrm{wt}}}{n} - \frac{q_{\mathrm{rd}}}{n}\mathbb{E}\left[\mathsf{MalWrite}\right] \\
&= \frac{q_{\mathrm{rd}}(n q_{\mathrm{wt}} - a_{\mathrm{wt}} b)}{n^2}.
\end{aligned}$$

□

Lemma 4.2.

$$\mathbb{E}\left[\mathsf{MaxConflicting}\right] \leq \frac{a_{\mathrm{rd}}}{n^3}\left(n^2 b + 2n^2 a_{\mathrm{wt}} - n a_{\mathrm{wt}} b - n^2 q_{\mathrm{wt}} - a_{\mathrm{wt}}^2 n + a_{\mathrm{wt}}^2 b\right). \tag{14}$$

Proof. We compute $\mathbb{E}\left[\mathsf{MaxConflicting}\right]$ taking into consideration the potential behavior of faulty clients described in Section 4.2. By applying linearity of expectation to (11), we have that,

$$\mathbb{E}\left[\mathsf{MaxConflicting}\right] = \mathbb{E}\left[\mathsf{Malevolent}\right] + \mathbb{E}\left[\mathsf{Conflicting}\right]. \tag{15}$$

A_{rd} is selected independently of B. As such, by (9), we have that Malevolent is a hypergeometric random variable characterized by a_{rd} draws from a population of n elements containing b successes. Therefore,

$$\mathbb{E}\left[\mathsf{Malevolent}\right] = \frac{a_{\mathrm{rd}} b}{n}. \tag{16}$$

To calculate $\mathbb{E}\left[\mathsf{Conflicting}\right]$, first note that:

$$\begin{aligned}
&\mathbb{E}\left[\mathsf{Conflicting}\right] \\
&= \mathbb{E}\left[|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \setminus (\mathsf{C}_{\mathrm{wt}} \cup B))|\right] \\
&= \mathbb{E}\left[|((\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B) \setminus \mathsf{C}_{\mathrm{wt}}|\right] \\
&= \mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B| - |(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap (\mathsf{C}_{\mathrm{wt}} \setminus B)|\right] \\
&= \mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B|\right] - \mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}|\right]
\end{aligned} \tag{17}$$

Where the first line is due to (10), and the final line holds due to linearity of expectation and because $C_{\mathrm{wt}} \subseteq U \setminus B$ by definition (6). We calculate $\mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B|\right]$ directly as follows. Consider an indicator random variable Ind_u, such that $\mathsf{Ind}_u = 1$ if $u \in (\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B$, and $\mathsf{Ind}_u = 0$ otherwise. For each $u \in U \setminus B$, we have $\Pr[\mathsf{Ind}_u = 1] = \frac{a_{\mathrm{rd}} a_{\mathrm{wt}}}{n^2}$, since A_{rd} and A'_{wt} are chosen independently. By linearity of expectation:

$$\mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B|\right] = \sum_{u \in U \setminus B} \Pr(\mathsf{Ind}_u = 1) = (n-b)\left(\frac{a_{\mathrm{rd}} a_{\mathrm{wt}}}{n^2}\right) = a_{\mathrm{rd}}\left(\frac{a_{\mathrm{wt}}}{n} - \left(\frac{a_{\mathrm{wt}}}{n}\right)\left(\frac{b}{n}\right)\right). \tag{18}$$

Because a faulty client may perform *both* a write that becomes established and another write that conflicts with the first write, we cannot assume that C_{wt} is selected independently of A'_{wt}. Therefore, we calculate $\mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}|\right]$ as $\mathbb{E}\left[|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}})|\right]$. Let $\mathsf{CI} = |\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}}|$. Since A_{rd} is selected independently of $(A'_{\mathrm{wt}} \cap C_{\mathrm{wt}})$, we see that $(|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}})| \mid \mathsf{CI} = c)$ is a hypergeometric random variable characterized by a_{rd} draws from a population of n elements containing c successes. Therefore, by the rules of conditional expectation and linearity of expectation we have that,

$$\begin{aligned}
&\mathbb{E}\left[|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}})|\right] \\
&= \sum_{c} \mathbb{E}\left[|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}})| \mid \mathsf{CI} = c\right] \Pr[\mathsf{CI} = c] \\
&= \mathbb{E}\left[\, \mathbb{E}\left[|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}})| \mid \mathsf{CI}\right] \,\right] \\
&= \mathbb{E}\left[\frac{a_{\mathrm{rd}}}{n}\mathsf{CI}\right] \\
&= \frac{a_{\mathrm{rd}}}{n}\mathbb{E}\left[\mathsf{CI}\right] \qquad (19)
\end{aligned}$$

As discussed in Section 4.2, to improve the chance that a conflicting write is selected (incorrectly), the client may minimize CI by choosing the servers for C_{wt} from $(A_{\mathrm{wt}} \setminus (A'_{\mathrm{wt}} \cup B))$ first. Thus, we conservatively calculate $\mathbb{E}\left[\mathsf{CI}\right]$ as follows:

$$\begin{aligned}
\mathbb{E}\left[\mathsf{CI}\right] &= \mathbf{max}(\ 0\ ,\ \mathbb{E}\left[|\mathsf{C}_{\mathrm{wt}}|\right] - \mathbb{E}\left[|\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B)|\right]) \\
&\geq \mathbb{E}\left[|\mathsf{C}_{\mathrm{wt}}|\right] - \mathbb{E}\left[|\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B)|\right] \qquad (20)
\end{aligned}$$

By the rules of conditional expectation and (7) we have that,

$$\mathbb{E}\left[|\mathsf{C}_{\mathrm{wt}}|\right] = \mathbb{E}\left[\mathbb{E}\left[|\mathsf{C}_{\mathrm{wt}}| \mid \mathsf{MalWrite}\right]\right] = \mathbb{E}\left[q_{\mathrm{wt}} - \mathsf{MalWrite}\right] = q_{\mathrm{wt}} - \mathbb{E}\left[\mathsf{MalWrite}\right] = q_{\mathrm{wt}} - \frac{a_{\mathrm{wt}}b}{n}. \qquad (21)$$

We calculate $\mathbb{E}\left[|\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B)|\right]$ directly as follows. First, note that $\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B) = (\mathsf{A}_{\mathrm{wt}} \setminus \mathsf{A}'_{\mathrm{wt}}) \setminus B$. Next, consider an indicator random variable Ind_u, such that $\mathsf{Ind}_u = 1$ if $u \in (\mathsf{A}_{\mathrm{wt}} \setminus \mathsf{A}'_{\mathrm{wt}}) \setminus B$, and $\mathsf{Ind}_u = 0$ otherwise. For each $u \in U \setminus B$, we have $\Pr[\mathsf{Ind}_u = 1] = \frac{a_{\mathrm{wt}}(n - a_{\mathrm{wt}})}{n^2}$, since A'_{wt} and A_{wt} are chosen independently. By linearity of expectation:

$$\mathbb{E}\left[|\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B)|\right] = \sum_{u \in U \setminus B} \Pr(\mathsf{Ind}_u = 1) = (n - b)\left(\frac{a_{\mathrm{wt}}(n - a_{\mathrm{wt}})}{n^2}\right) = \frac{a_{\mathrm{wt}}}{n^2}(n - a_{\mathrm{wt}})(n - b). \qquad (22)$$

By (20), (21), and (22), we have that,

$$\mathbb{E}\left[\mathsf{CI}\right] \geq \left(q_{\mathrm{wt}} - \frac{a_{\mathrm{wt}}b}{n}\right) - \frac{a_{\mathrm{wt}}}{n^2}(n - a_{\mathrm{wt}})(n - b) = q_{\mathrm{wt}} - \frac{a_{\mathrm{wt}}}{n}\left(b + \frac{(n - a_{\mathrm{wt}})(n - b)}{n}\right). \qquad (23)$$

Therefore, by (19) and (23) we have that,

$$\mathbb{E}\left[|\mathsf{A}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}})|\right] \geq \frac{a_{\mathrm{rd}}}{n}\left(q_{\mathrm{wt}} - \frac{a_{\mathrm{wt}}}{n}\left(b + \frac{(n-a_{\mathrm{wt}})(n-b)}{n}\right)\right). \tag{24}$$

Combining and simplifying equations (17), (18), and (24), we obtain,

$$\mathbb{E}\left[\mathsf{Conflicting}\right] \leq \frac{a_{\mathrm{rd}}}{n^3}(2a_{\mathrm{wt}}n^2 - na_{\mathrm{wt}}b - q_{\mathrm{wt}}n^2 - a_{\mathrm{wt}}^2 n + a_{\mathrm{wt}}^2 b), \tag{25}$$

and by (15), (16), and (25) we have,

$$\mathbb{E}\left[\mathsf{MaxConflicting}\right] \leq \frac{a_{\mathrm{rd}}}{n^3}\left(n^2 b + 2a_{\mathrm{wt}}n^2 - na_{\mathrm{wt}}b - n^2 q_{\mathrm{wt}} - a_{\mathrm{wt}}^2 n + a_{\mathrm{wt}}^2 b\right). \tag{26}$$

□

Lemma 4.3. *If PO-Consistency holds, then*

$$\mathbb{E}\left[\mathsf{MaxConflicting}\right] = \frac{a_{\mathrm{rd}}}{n^3}\left(n^2 b + 2n^2 a_{\mathrm{wt}} - na_{\mathrm{wt}}b - n^2 q_{\mathrm{wt}} - a_{\mathrm{wt}}^2 n + a_{\mathrm{wt}}^2 b\right). \tag{27}$$

Proof. The logic for the calculations follows that of the proof of Lemma 4.3, except that if PO-Consistency holds, $\mathbb{E}\left[\mathsf{CI}\right] > 0$. Assume the contrary, i.e., $\mathbb{E}\left[\mathsf{CI}\right] \leq 0$ and PO-Consistency holds. Then by (19), $\mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}|\right] \leq 0$. Thus, because $a_{\mathrm{rd}} \geq q_{\mathrm{rd}}$ and $a_{\mathrm{wt}} \geq q_{\mathrm{wt}}$, by (15) and (17) we have,

$$\begin{aligned}
&\mathbb{E}\left[\mathsf{MaxConflicting}\right] \\
&= \mathbb{E}\left[\mathsf{Malevolent}\right] + \mathbb{E}\left[\mathsf{Conflicting}\right] \\
&\geq \mathbb{E}\left[\mathsf{Conflicting}\right] \\
&= \mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B|\right] - \mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}|\right] \\
&\geq \mathbb{E}\left[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B|\right] \\
&\geq \mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] \\
&= \mathbb{E}\left[|\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{C}_{\mathrm{wt}}|\right]. \\
&= \mathbb{E}\left[\mathsf{MinCorrect}\right]
\end{aligned}$$

But this cannot be true because PO-Consistency holds. As such, the inequalities in equations (20), (23), (24), (25), and (26) all become equalities. □

Theorem 4.4. *PO-Consistency holds iff*

$$b < \frac{(a_{\mathrm{rd}}q_{\mathrm{wt}}n - 2a_{\mathrm{rd}}a_{\mathrm{wt}}n + a_{\mathrm{wt}}^2 a_{\mathrm{rd}} + q_{\mathrm{rd}}q_{\mathrm{wt}}n)n}{n^2 a_{\mathrm{rd}} - a_{\mathrm{rd}}a_{\mathrm{wt}}n + a_{\mathrm{wt}}^2 a_{\mathrm{rd}} + q_{\mathrm{rd}}a_{\mathrm{wt}}n}.$$

Proof. We set the value of $\mathbb{E}\left[\mathsf{MinCorrect}\right]$ given in Lemma 4.1 greater than and the largest possible value of $\mathbb{E}\left[\mathsf{MaxConflicting}\right]$ given in Lemma 4.2 and solve for b to obtain the inequality in Theorem 4.4; this inequality therefore implies PO-Consistency. But by Lemma 4.3, PO-Consistency in turn implies that $\mathbb{E}\left[\mathsf{MaxConflicting}\right]$ is equal to its maximum possible value and, as such, PO-Consistency implies the inequality in Theorem 4.4. □

Benign clients (i.e., those that are non-faulty or that fail only by crashing) are different because they can be trusted to follow the read and write protocols listed in Section 3. In particular, we can trust that a client will select a read quorum at random without requiring that the client select the quorum from a randomly-chosen access set; we reflect this in our calculations by not differentiating between read quorums and read access sets (i.e., by setting $a_{\mathrm{rd}} = q_{\mathrm{rd}}$). In addition, we can assume that all quorums used in writes are chosen independently.

Theorem 4.5. *If all clients are benign, then PO-Consistency holds iff*

$$b < \frac{(q_{\mathrm{wt}}n - a_{\mathrm{wt}}n + q_{\mathrm{wt}}a_{\mathrm{wt}})n}{n^2 + a_{\mathrm{wt}}^2}.$$

Proof. We can now assume that all servers in C_{wt} are chosen uniformly at random from $U \setminus B$ since writes are independent. Let $\mathsf{D} = |(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}|$. Instead of the calculations in (20)–(24), we calculate $\mathbb{E}[\mathsf{D}]$ by beginning with $\mathbb{E}[\mathsf{D} \mid \mathsf{C}_{\mathrm{wt}}]$. Consider an indicator random variable Ind_u, such that $\mathsf{Ind}_u = 1$ if $u \in (\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}$, and $\mathsf{Ind}_u = 0$ otherwise. For each $u \in \mathsf{C}_{\mathrm{wt}}$, we have $\Pr[\mathsf{Ind}_u = 1] = \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}}{n^2}$, since A_{rd} and A'_{wt} are chosen independently. By linearity of expectation:

$$\mathbb{E}[\mathsf{D}|\mathsf{C}_{\mathrm{wt}}] = \sum_{u \in \mathsf{C}_{\mathrm{wt}}} \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}}{n^2} = \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}|\mathsf{C}_{\mathrm{wt}}|}{n^2}. \tag{28}$$

Then, by the rules of conditional expectation and (21),

$$\begin{aligned}\mathbb{E}[|(\mathsf{A}_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap \mathsf{C}_{\mathrm{wt}}||] &= \mathbb{E}[\mathbb{E}[\mathsf{D}|\mathsf{C}_{\mathrm{wt}}]] = \mathbb{E}\left[\frac{a_{\mathrm{rd}}a_{\mathrm{wt}}|\mathsf{C}_{\mathrm{wt}}|}{n^2}\right] = \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}}{n^2}\mathbb{E}[|\mathsf{C}_{\mathrm{wt}}|] \\ &= \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}}{n^2}(q_{\mathrm{wt}} - \frac{a_{\mathrm{wt}}b}{n}) = \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}(q_{\mathrm{wt}}n - a_{\mathrm{wt}}b)}{n^3}.\end{aligned} \tag{29}$$

Therefore, by (17), (18), and (29),

$$\mathbb{E}[\mathsf{Conflicting}] = \frac{a_{\mathrm{rd}}a_{\mathrm{wt}}(n^2 - nb - q_{\mathrm{wt}}n + a_{\mathrm{wt}}b)}{n^3}, \tag{30}$$

and by (15), (16), and (30),

$$\mathbb{E}[\mathsf{MaxConflicting}] = \frac{a_{\mathrm{rd}}}{n^3}\left(n^2b + n^2a_{\mathrm{wt}} - na_{\mathrm{wt}}b - na_{\mathrm{wt}}q_{\mathrm{wt}} + a_{\mathrm{wt}}^2b\right). \tag{31}$$

Solving $\mathbb{E}[\mathsf{MinCorrect}] > \mathbb{E}[\mathsf{MaxConflicting}]$ for b, we have the inequality in the theorem. □

As shown in Section 4.5, a construction exhibits decreasing error probability in the limit with increasing n if PO-Consistency holds. Therefore, the remainder of this section is concerned with interpreting the inequalities in Theorems 4.4 and 4.5. Our analysis (summarized in Table 2) shows that the best bounds are provided when: (i) both types of quorums are as large as possible (while still ensuring an available quorum), i.e., $q_{\mathrm{rd}} = q_{\mathrm{wt}} = n - b$; and (ii), given (i), that access sets are as small as possible.

We first consider scenarios in which a read or write can be completed with a single access set because $a_{\mathrm{rd}} = q_{\mathrm{rd}} + b$ ("single-phase reads") and $a_{\mathrm{wt}} = q_{\mathrm{wt}} + b$ ("single-phase writes"). Then, we derive the better bounds that can be achieved if we set $a_{\mathrm{rd}} < q_{\mathrm{rd}} + b$ or $a_{\mathrm{wt}} < q_{\mathrm{wt}} + b$. Finally, we consider scenarios pertaining only to clients that are benign.

Table 2: Lower bounds on n for various configurations.

$\mathbf{n} >$	$= n$	$= n - b$	$= n - 2b$	eq.	Sec. 4.4.x
3.15b	-	$a_{\mathrm{rd}}\ q_{\mathrm{rd}}\ a_{\mathrm{wt}}\ q_{\mathrm{wt}}$	-	(40)	4, 5
3.83b	a_{rd}	$q_{\mathrm{rd}}\ a_{\mathrm{wt}}\ q_{\mathrm{wt}}$	-	(38)	3, 4
4.00b	a_{wt}	$a_{\mathrm{rd}}\ q_{\mathrm{rd}}\ q_{\mathrm{wt}}$	-	(36)	2, 4, 5
4.08b	-	$a_{\mathrm{rd}}\ a_{\mathrm{wt}}\ q_{\mathrm{wt}}$	q_{rd}	(39)	3
4.56b	$a_{\mathrm{rd}}\ a_{\mathrm{wt}}$	$q_{\mathrm{rd}}\ q_{\mathrm{wt}}$	-	(32)	1, 2, 3, 4
4.73b	a_{wt}	$a_{\mathrm{rd}}\ q_{\mathrm{wt}}$	q_{rd}	(33)	1, 3
5.49b	-	$a_{\mathrm{rd}}\ q_{\mathrm{rd}}\ a_{\mathrm{wt}}$	q_{wt}	(37)	2
6.07b	a_{rd}	$q_{\mathrm{rd}}\ a_{\mathrm{wt}}$	q_{wt}	(34)	1, 2
6.19b	-	$a_{\mathrm{rd}}\ a_{\mathrm{wt}}$	$q_{\mathrm{rd}}\ q_{\mathrm{wt}}$	(35)	1

4.4.1 Single-Phase Reads and Writes.

Figure 3(a) plots the results of solving the inequality in Theorem 4.4 for n when $q_{\mathrm{wt}} = a_{\mathrm{wt}} - b$, $q_{\mathrm{rd}} = a_{\mathrm{rd}} - b$, and the sizes of access sets are varied between $n - b$ and n. We observe that the best bound is found when $a_{\mathrm{wt}} = a_{\mathrm{rd}} = n$ and $q_{\mathrm{wt}} = q_{\mathrm{rd}} = n - b$. In this case, we require,

$$c = \left(\frac{5 + \sqrt{17}}{2}\right) \approx 4.561552813, \qquad n > c \cdot b. \tag{32}$$

We make the lower bound on n progressively worse by decreasing the sizes of access sets (and therefore quorums). If we set $a_{\mathrm{rd}} = n - b$ and $a_{\mathrm{wt}} = n$, we find,

$$c = \left(3 + \sqrt{3}\right) \approx 4.732050808, \qquad n > c \cdot b. \tag{33}$$

By decreasing q_{wt} as a result of decreasing a_{wt}, we find that we soon fail to break the $n > 5b$ bound of strict opaque quorum systems [17]. If we set $a_{\mathrm{rd}} = n$ and $a_{\mathrm{wt}} = n - b$, we require,

$$c \approx 6.065103370, \qquad n > c \cdot b. \tag{34}$$

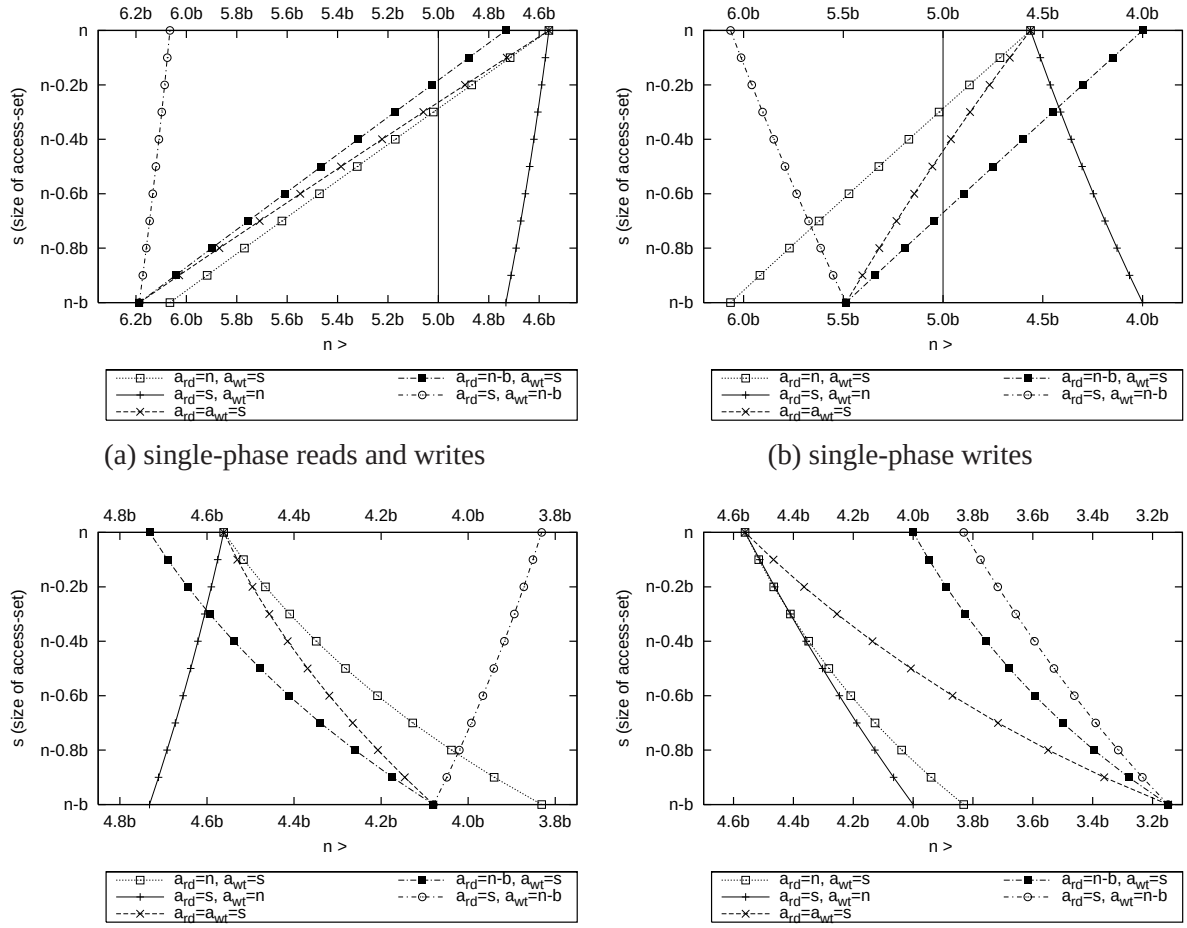
Finally, if we set $a_{\mathrm{rd}} = a_{\mathrm{wt}} = n - b$,

$$c \approx 6.186789391, \qquad n > c \cdot b. \tag{35}$$

4.4.2 Single-Phase Writes

Figure 3(b) plots the results of solving the inequality in Theorem 4.4 for n when $q_{\mathrm{wt}} = a_{\mathrm{wt}} - b$, $q_{\mathrm{rd}} = n - b$, and the sizes of access sets are varied between $n - b$ and n. Because a read access set might not contain an available quorum, we can achieve better bounds than (32) by setting $a_{\mathrm{rd}} < q_{\mathrm{rd}} + b$. The best bound, when $a_{\mathrm{rd}} = n - b$ and $a_{\mathrm{wt}} = n$, is,

$$n > 4b. \tag{36}$$



(a) single-phase reads and writes

(b) single-phase writes

(c) single-phase reads

(d) best bounds

Figure 3: Sizes of access sets to achieve a given lower bound on n for: (a) $q_{\mathrm{rd}} = a_{\mathrm{rd}} - b$ and $q_{\mathrm{wt}} = a_{\mathrm{wt}} - b$; (b) $q_{\mathrm{rd}} = n - b$ and $q_{\mathrm{wt}} = a_{\mathrm{wt}} - b$; (c) $q_{\mathrm{rd}} = a_{\mathrm{rd}} - b$ and $q_{\mathrm{wt}} = n - b$; (d) $q_{\mathrm{rd}} = q_{\mathrm{wt}} = n - b$.

However, as in Section 4.4.1, decreasing q_{wt} by decreasing a_{wt} results in worse bounds, again quickly worse than $n > 5b$. If $a_{\mathrm{rd}} = a_{\mathrm{wt}} = n - b$ we require,

$$c \approx 5.486416764, \qquad n > c \cdot b. \tag{37}$$

The worst bound is when $a_{\mathrm{rd}} = n$ and $a_{\mathrm{wt}} = n - b$, i.e., (34).

4.4.3 Single-Phase Reads

Figure 3(c) plots the results of solving the inequality in Theorem 4.4 for n when $q_{\mathrm{wt}} = n - b$, $q_{\mathrm{rd}} = a_{\mathrm{rd}} - b$, and the sizes of access sets are varied between $n - b$ and n. All of the points in the graph represent an improvement on the $n > 5b$ bound of strict opaque quorum systems. As in Section 4.4.2, we find that we can achieve better bounds than (32), here by decreasing the size of

a_{wt}. For example, if $a_{\mathrm{wt}} = n - b$ and $a_{\mathrm{rd}} = n$, we require,

$$c \approx 3.831177208, \qquad n > c \cdot b. \tag{38}$$

And if $a_{\mathrm{wt}} = a_{\mathrm{rd}} = n - b$, we require,

$$c \approx 4.079595625, \qquad n > c \cdot b. \tag{39}$$

The case where $a_{\mathrm{wt}} = n$ and $a_{\mathrm{rd}} = n - b$ is bound by (33).

4.4.4 Best Bounds

Now consider $q_{\mathrm{wt}} = q_{\mathrm{rd}} = n - b$. This results in better bounds than the scenarios in which the sizes of quorums are smaller. Figure 3(d) plots the results of solving the inequality in Theorem 4.4 for n when $q_{\mathrm{wt}} = q_{\mathrm{rd}} = n - b$, and the sizes of access sets are varied between $n - b$ and n. Because the sizes of quorums are fixed at $n - b$, we can improve on (32) by decreasing the size of a_{wt} or a_{rd}; the best bound, achieved when $a_{\mathrm{rd}} = a_{\mathrm{wt}} = n - b$, is,

$$c \approx 3.147899035, \qquad n > c \cdot b. \tag{40}$$

This represents the assumption in [18], in which all servers are accessed via randomly selected quorums. Otherwise, if $a_{\mathrm{rd}} = n$ and $a_{\mathrm{wt}} = n - b$ we have (38), and if $a_{\mathrm{rd}} = n - b$ and $a_{\mathrm{wt}} = n$, we have (36).

4.4.5 Benign Clients

Theorem 4.5 shows us that in this case, if $a_{\mathrm{wt}} = n$, we require (36), and if $a_{\mathrm{wt}} = n - b$ we require (40). However, as seen in Figure 4, for values of a_{wt} strictly between n and $n - b$, we achieve better maximum ratios of b/n than would a system that can tolerate faulty clients.

4.5 Bounding the Error Probability

Our primary result in this section is Theorem 4.8, which shows that the error probability goes to zero as n grows, assuming that the ratio of each of b, a_{rd}, q_{rd}, a_{wt}, and q_{wt} to n remains constant. In this section, the symbols "θ", "ω" and "Ω" are used as in standard asymptotic notation.

Lemma 4.6. *Let* $\mathbb{E}[\mathsf{MinCorrect}] > \mathbb{E}[\mathsf{MaxConflicting}]$, *and let the ratio of each of* b, a_{rd}, q_{rd}, a_{wt}, *and* q_{wt} *to* n *be fixed. Then,*

$$\begin{aligned} \mathbb{E}[\mathsf{MinCorrect}] &= \theta(n) \\ \mathbb{E}[\mathsf{MaxConflicting}] &= \theta(n) \\ \mathbb{E}[\mathsf{MinCorrect}] - \mathbb{E}[\mathsf{MaxConflicting}] &= \theta(n). \end{aligned}$$

Proof. Given that the ratio of each of b, a_{rd}, q_{rd}, a_{wt}, and q_{wt} to n is constant, we have by (12) and (27) that $\mathbb{E}[\mathsf{MinCorrect}] = \theta(n)$ and $\mathbb{E}[\mathsf{MaxConflicting}] = \theta(n)$, and that if $\mathbb{E}[\mathsf{MinCorrect}] - \mathbb{E}[\mathsf{MaxConflicting}]$ is not identically zero then $\mathbb{E}[\mathsf{MinCorrect}] - \mathbb{E}[\mathsf{MaxConflicting}] = \theta(n)$. So, the result follows from the stipulation that $\mathbb{E}[\mathsf{MinCorrect}] - \mathbb{E}[\mathsf{MaxConflicting}] > 0$. □

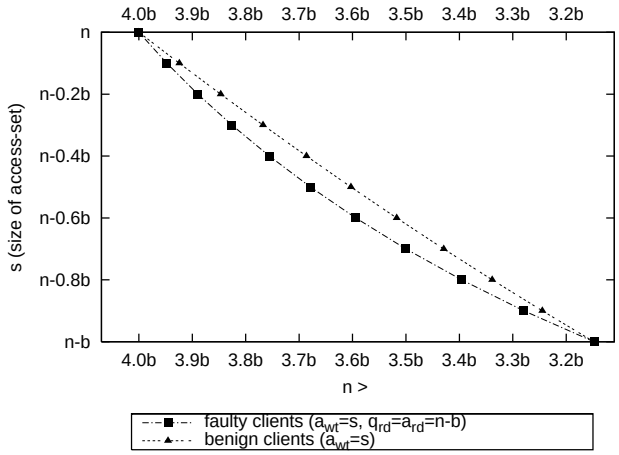


Figure 4: Benign clients vs. faulty clients—sizes of access-sets to achieve a given lower bound on n for $q_{\mathrm{wt}} = n - b$.

Suppose a read operation always returns a value that receives more than r votes, where $\mathbb{E}\left[\mathsf{MaxConflicting}\right] \leq r < \mathbb{E}\left[\mathsf{MinCorrect}\right]$. Then, the error probability, ϵ, is

$$\epsilon = \Pr(\mathsf{MaxConflicting} > r \vee \mathsf{MinCorrect} \leq r). \tag{41}$$

Theorem 4.8 states that if r is chosen so that

$$\begin{aligned} \mathbb{E}\left[\mathsf{MinCorrect}\right] - r &= \theta(n) \quad \text{and} \\ r - \mathbb{E}\left[\mathsf{MaxConflicting}\right] &= \theta(n) \end{aligned} \tag{42}$$

then ϵ decreases as a function of n. For example, r can be set to $(\mathbb{E}\left[\mathsf{MaxConflicting}\right] + \mathbb{E}\left[\mathsf{MinCorrect}\right])/2$.

Our proof of Theorem 4.8 uses the following theorem, which is a simplification of the Molloy and Reed statement [22, p. 172] (c.f., [22, p. 81]) of the McDiarmid Inequality.

Theorem 4.7 ([22])**.** *Let $\mathsf{Z} = \boldsymbol{z}(\Pi_1, \ldots, \Pi_l)$ be a random variable that is a function of a series $\Pi_1, \ldots, \Pi_l$ of independent random variables, where each Π_i takes on a random permutation (bijection) $\pi : \{1, \ldots, |P|\} \to P$ of a finite non-empty set P. Also, for some positive constants δ and μ, let the following conditions hold (where if $\Pi_j = \pi_j$ then the* mapping $\langle i, j, m\rangle$ *indicates that $\pi_j(i) = m$):*

M1 Swapping the mappings of any two elements in a single permutation π_j (i.e., changing $\{\langle i, j, m\rangle, \langle i', j, m'\rangle\}$ to $\{\langle i', j, m\rangle, \langle i, j, m'\rangle\}$, where $i \neq i'$ and $m \neq m'$) changes the value of Z by at most δ.

M2 If $\mathsf{Z} = \boldsymbol{z}(\pi_1, \ldots, \pi_l) = x$, then there exists a set of μx distinct mappings $\{\langle i_1, j_1, m_1\rangle, \ldots, \langle i_{\mu x}, j_{\mu x}, m_{\mu x}\rangle\}$ such that $\boldsymbol{z}(\pi'_1, \ldots, \pi'_l) \geq x$ for any $\pi'_1, \ldots, \pi'_l$ sharing the same set of mappings.

If $d = \omega(\sqrt{\mathbb{E}[\mathsf{Z}]})$ *and* $0 \leq d \leq \mathbb{E}[\mathsf{Z}]$*, then:*

$$\Pr(|\mathsf{Z} - \mathbb{E}[\mathsf{Z}]| \geq d) = 2/e^{\Omega\left(d^2/\mathbb{E}[\mathsf{Z}]\right)}. \tag{43}$$

Theorem 4.8. *Let* MinCorrect, MaxConflicting*, and* r *be defined as above (so PO-Consistency holds) and let the ratio of each of* b*,* a_{rd}*,* q_{rd}*,* a_{wt}*, and* q_{wt} *to* n *be fixed. Then,*

$$\epsilon = 2/e^{\Omega(n)} + 2/e^{\Omega(n)}.$$

Proof. Consider the following method for computing MinCorrect and MaxConflicting. Fix any set of b servers to constitute B. Next, define random variables Π_0, Π_1, Π_2, and Π_3, each taking on a random permutation $\{1, \ldots, |U|\} \to U$, where U is the set of all n servers. Let Q_{rd} be the random variable used in (8), i.e., a random variable taking on the read quorum selected uniformly at random. Then consider the following definitions:

- Define $\mathsf{A}_{\mathrm{wt}} = \{\Pi_0(1), \ldots, \Pi_0(a_{\mathrm{wt}})\}$.
- Define $\mathsf{A}'_{\mathrm{wt}} = \{\Pi_1(1), \ldots, \Pi_1(a_{\mathrm{wt}})\}$.
- Define $\mathsf{A}_{\mathrm{rd}} = \{\Pi_2(1), \ldots, \Pi_2(a_{\mathrm{rd}})\}$.
- Define $\mathsf{Q}_{\mathrm{rd}} = \{\Pi_3(1), \ldots, \Pi_3(a_{\mathrm{rd}})\}$.

Because each permutation is randomly selected (independently of B), so too are A_{wt}, $\mathsf{A}'_{\mathrm{wt}}$, A_{rd}, and Q_{rd}. Define C_{wt} in accordance with Sections 4.3 and 4.4. Specifically, choose its $q_{\mathrm{wt}} - \mathsf{MalWrite}$ servers at random first from $(\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B))$ and only then from $(\mathsf{A}_{\mathrm{wt}} \cap (\mathsf{A}'_{\mathrm{wt}} \setminus B))$. To do this, select the servers according to the random order imposed by Π_0, i.e., select $\Pi_0(i)$ before $\Pi_0(i+1)$ if both servers are in the relevant set ($(\mathsf{A}_{\mathrm{wt}} \setminus (\mathsf{A}'_{\mathrm{wt}} \cup B))$ or $(\mathsf{A}_{\mathrm{wt}} \cap (\mathsf{A}'_{\mathrm{wt}} \setminus B))$). Random selection is sufficient because a faulty client gains no advantage by any other scheme.

Given that we have defined A_{wt}, $\mathsf{A}'_{\mathrm{wt}}$, A_{rd}, Q_{rd}, C_{wt}, and B, we can directly calculate MinCorrect and MaxConflicting using definitions (8) and (11). Consider this in relation to Theorem 4.7 as follows. Swapping any two elements in one permutation can change the value of MinCorrect by at most 1 (by adding or removing a server from the relevant intersection of the sets), and because an additional server added to A_{rd} cannot be both faulty and non-faulty, swapping any two elements in one permutation can similarly change the value of MaxConflicting by at most 1. Therefore, $\delta = 1$ in Condition M1. Additionally, if $\mathsf{MinCorrect} = x$, then the mappings

$$\bigcup_{u \in Q_{\mathrm{rd}} \cap C_{\mathrm{wt}}} \{\langle \Pi_0^{-1}(u), 0, u\rangle, \langle \Pi_3^{-1}(u), 3, u\rangle\}$$

suffice to satisfy Condition M2, where Q_{rd} and C_{wt} are the values taken on by Q_{rd} and C_{wt}, respectively. Therefore, $\mu = 2$ for MinCorrect, and similarly $\mu = 3$ if $\mathsf{Z} = \mathsf{MaxConflicting}$.

To derive a bound on $\Pr(\mathsf{MinCorrect} \leq r)$, we set $d = \mathbb{E}[\mathsf{MinCorrect}] - r$. Because of (42), $d = \theta(n)$ and therefore $d = \omega(\sqrt{\mathbb{E}[\mathsf{MinCorrect}]})$. As such, by Theorem 4.7 and (43) we have,

$$\begin{aligned}
&\Pr(\mathsf{MinCorrect} \leq r) \\
&\quad = \Pr(\mathsf{MinCorrect} \leq \mathbb{E}[\mathsf{MinCorrect}] - d) \\
&\quad = \Pr(\mathbb{E}[\mathsf{MinCorrect}] - \mathsf{MinCorrect} \geq d) \\
&\quad \leq \Pr(|\mathsf{MinCorrect} - \mathbb{E}[\mathsf{MinCorrect}]| \geq d) \\
&\quad = 2/e^{\Omega\left(\theta(n)^2/\theta(n)\right)} \\
&\quad = 2/e^{\Omega(n)}.
\end{aligned}$$

To derive a bound on $\Pr(\mathsf{MaxConflicting} > r)$, we set $d' = r - \mathbb{E}[\mathsf{MaxConflicting}]$ and $d = \mathbf{min}(d', \mathbb{E}[\mathsf{MaxConflicting}])$, to ensure $d \leq \mathbb{E}[\mathsf{MaxConflicting}]$. Again, by Lemma 4.6 and (42), $d = \theta(n)$, and so $d = \omega(\sqrt{\mathbb{E}[\mathsf{MaxConflicting}]})$. Therefore, by Theorem 4.7 and (43) we have,

$$\begin{aligned}
&\Pr(\mathsf{MaxConflicting} > r) \\
&\quad \leq \Pr(\mathsf{MaxConflicting} \geq r) \\
&\quad = \Pr(\mathsf{MaxConflicting} \geq d' + \mathbb{E}[\mathsf{MaxConflicting}]) \\
&\quad = \Pr(\mathsf{MaxConflicting} - \mathbb{E}[\mathsf{MaxConflicting}] \geq d') \\
&\quad \leq \Pr(\mathsf{MaxConflicting} - \mathbb{E}[\mathsf{MaxConflicting}] \geq d) \\
&\quad \leq \Pr(|\mathsf{MaxConflicting} - \mathbb{E}[\mathsf{MaxConflicting}]| \geq d) \\
&\quad = 2/e^{\Omega\left(\theta(n)^2/\theta(n)\right)} \\
&\quad = 2/e^{\Omega(n)}.
\end{aligned}$$

Finally, note the following about (41):

$$\begin{aligned}
&\Pr(\mathsf{MaxConflicting} > r \vee \mathsf{MinCorrect} \leq r) \\
&\quad = \Pr(\mathsf{MaxConflicting} > r) + \Pr(\mathsf{MinCorrect} \leq r) - \\
&\qquad \Pr(\mathsf{MaxConflicting} > r \wedge \mathsf{MinCorrect} \leq r) \\
&\quad \leq \Pr(\mathsf{MaxConflicting} > r) + \Pr(\mathsf{MinCorrect} \leq r).
\end{aligned}$$

□

5 Access-Restriction Protocol

Our analysis in the previous sections assumes that all access sets are chosen uniformly at random by all clients—even faulty clients. Therefore, here we present an access-restriction protocol that is used to enforce this. Recall from Section 3.1 that the need for read access sets to be selected uniformly at random is motivated by repair. As such, protocols that do not involve repair may not require this access-restriction protocol for read operations.

Our protocol must balance conflicting constraints. First, a client may be forced to discard a randomly chosen access set—and choose another—because a given access set (of size less than b servers more than a quorum) might not contain an available quorum. However, in order to support

protocols like Q/U [1] that use opaque quorum systems for single-round writes, we cannot require additional rounds of communication for each operation. This precludes, for example, a protocol in which the servers collectively choose an access set at random and assign it to the client for every operation. As such, a client must be able to choose from multiple access sets without involving the servers for each. Yet, a faulty client should be prevented from discarding access sets in order to choose the one that has the highest probability of causing an error given the current system state. In addition, we should ensure that a faulty client does not benefit from waiting for the system state to change in order to use a previously chosen access set that becomes more advantageous as a result of the change.

In our protocol, the client obtains one or more random values, each called a Verifiable Random Value (VRV), with the participation of non-faulty servers. Each VRV determines a unique, verifiable, ordered sequence of random access sets that the client can use; the client has no control over the sequence. To deter a client from discarding earlier access sets in the sequence for potentially more favorable access sets later in the sequence, the protocol imposes an exponentially increasing cost (in terms of computation) for the ability to use later access sets. The cost is implemented as a *client puzzle* [13]. We couple this with a facility for the propagation of the correct value in the background so that any advantages for a faulty client in the current system state are reduced if the client chooses to delay performing the operation while it explores later access sets. Finally, to deter a client from waiting for the system state to change, we tie the validity of a VRV (and its sequence of access sets) to the state of the system so that as execution proceeds, any unused access sets become invalid.

The remainder of this section is structured as follows: Section 5.1 discusses how a client obtains a VRV; Section 5.2 discusses the client puzzle and how the sequence of access sets is computed from the VRV; Section 5.3 outlines the responsibilities of servers concerning verification of the validity of access sets; and Section 5.4 provides details of the background propagation of correct values.

5.1 Obtaining a VRV

In order to get an access set, the client first must obtain a VRV from the servers. Servers implement a metering policy, in which each server responds to a request for a VRV only after a delay. The delay varies, such that it increases exponentially with the rate at which the client has requested VRVs during some recent interval of time—i.e., a client that has not requested a VRV recently will receive a VRV with little or no delay, whereas a client that has recently requested many VRVs will receive a VRV after a (potentially significant) delay. To offload work from servers to clients (e.g., for scalability), the servers can make it relatively more expensive (in terms of time) to ask for and receive a new VRV than to compute a given number of access sets (potentially for multiple operations) from a single VRV, using the mechanisms described below.

The VRV is characterized by the following properties:

- It can be created only with the consent of non-faulty servers;
- Its validity is tied to the state of the system, in the sense that as the system state evolves (possibly merely through the passage of time), eventually the VRV is invalidated;

- While it is valid, any non-faulty server can verify its validity and so will accept it.

The VRV must be created with the consent of non-faulty servers because otherwise faulty servers might collude to issue multiple VRVs to a faulty client with no delay. Therefore, l, the number of servers required for the issuance of a VRV, must be at least $b+1$. However, of the non-faulty servers in the system, only those among the (at least $l - b$) used to issue a VRV will impose additional delay before issuing an additional VRV. Therefore, to minimize the time to get an additional VRV, a faulty client avoids involving servers that have issued VRVs recently. This strategy maximizes the number of VRVs to which the non-faulty server contributing to the fewest VRVs has contributed. Thus, once k VRVs have been issued, all $n - b$ non-faulty servers have contributed to the issuance of at least $\lfloor k(l-b)/(n-b) \rfloor$ of these k. Since all non-faulty servers have contributed to at least this many VRVs, and the delay is exponential in this number, the time $T(k)$ required for a client to obtain k VRVs is:

$$T(k) = \Omega\left(\exp\left\lfloor \frac{k(l-b)}{n-b} \right\rfloor\right)$$

In practice, $T(k)$ for a client decays during periods in which that client does not request additional VRVs, so that a client that does not request VRVs for a period can obtain one with small delay.

The validity of the VRV (and its sequence of access sets) is tied to the state of the system so that as execution proceeds, any unused access sets become invalid. To implement this, the replication protocol may provide some piece of data that varies with the state of the system—the *Object History Set* in Q/U [1] is an example of this—with which the servers can compute a VRV, but, in the absence of a suitable value from the protocol, the VRV can include a timestamp (assuming that the non-faulty servers have roughly synchronized clocks). The VRV consists of this value together with a digital signature created using a (l, n)-threshold signature scheme (e.g., [24]), i.e., so that any set of l servers can together create the signature, but smaller sets of servers cannot. The signature scheme must be *strongly unforgeable* [4], meaning that an adversary, given a VRV, is not able to find other valid VRVs. This is necessary because otherwise a faulty client would be able to generate variations of a valid VRV until finding one from which to select an access set that causes an error (see below).

5.2 Choosing an Access Set

As motivated above: (i) the VRV determines a sequence of valid access sets; and (ii) a client puzzle must make it exponentially harder to use later access sets in the sequence than earlier ones. In addition, it is desirable for our protocol to satisfy the following requirements:

- Each VRV must determine only a single valid sequence of access sets. This is to prevent a faulty client from choosing a preferred sequence.
- The puzzle solutions must be easy to verify, so that verification costs do not limit the scalability of the system in terms of the number of requests.
- There must be a solution to each puzzle. Otherwise a non-faulty client might be unable to use any access set.

- No server can know the solution to the puzzle beforehand due to the Byzantine fault model. Otherwise, a faulty client could avoid the exponential work by asking a faulty server for the solution.

In our protocol, the sequence of access sets is determined as follows. Let v be a VRV, let **g** be a hash function modeled as a random oracle [7], and let **access_set** be a deterministic operation that, given a seed value, selects an access set of the specified size from the set of all access sets of that size in a uniform fashion. Let the first seed, s_1, be $\mathbf{g}(v)$, and the i'th seed, s_i, be $\mathbf{g}(s_{i-1})$. Then the i'th access set is **access_set**(s_i).

The function **access_set** works as follows for seed value s. Let **h** be a hash function modeled as a random oracle that maps inputs uniformly to its output range. Divide the output range of **h** into n equal sized intervals labeled 1 through n, and let **server**(r) return the server corresponding to the range into which r falls. Let r_1 be $\mathbf{h}(s)$ and let r_i be $\mathbf{h}(r_{i-1})$. Then the i'th candidate server chosen for the access set is **server**(r_i), and candidate servers are added to the access set until the access set contains the required number of unique servers. Because **h** maps inputs uniformly to its output range, each server is effectively selected at random.

In order to use the i'th access set, the client must solve a puzzle of suitable difficulty. This puzzle must be non-interactive [12] to avoid additional rounds of communication. There are many suitable candidate puzzle functions [12]; we use a simple variant of Hashcash [5]. Specifically, let **f** be a hash function modeled as a random oracle. Then, to use the i'th access-set, the client must find a value, w, such that the first ci bits of $\mathbf{f}(v||w)$ equal zero, where '||' represents concatenation and c is a constant greater than or equal to one. The most efficient known way to do so is a brute force search. In general, the client will need to compute $2^{ci}/2 = 2^{ci-1}$ evaluations of **f** to find a w such that the first ci bits of $\mathbf{f}(v||w)$ bits are zero. Larger values of c require more effort to obtain each access set.

5.3 Server Verification

Upon receiving a write request for the i'th access set, each non-faulty server in the chosen access set must verify that it is a member of the access set; for a repair request it must verify that the relevant votes are from servers in the access set of the read operation that gave rise to the repair. In addition, in either case, before accepting the value, each server must verify that the VRV is valid, that the access set corresponds to the i'th access set of the sequence, and that the client has provided a valid solution to a puzzle of difficulty ci. To validate the solution, w, the server need only check that the first ci bits of $\mathbf{f}(v||w)$ are zero.

While the client can obtain additional access sets from the VRV, each access set used is treated as a different operation by servers as stated in Section 3.1; e.g., a write operation using one access set, and then using another access set, is treated as two different writes,[3] so that a faulty client cannot "accumulate" more than a_{wt} servers for its operation through the use of multiple write access sets.

[3]Typically, a Byzantine-fault-tolerant write protocol must already be resilient to partial writes, which is how these writes using different access sets might appear to the service.

5.4 Background Propagation

As described above, servers work to propagate the values of established writes to each other in the background. Our main contribution in this area is our analysis of the threshold number of servers that must propagate a value for it to be accepted by another server. While related Byzantine diffusion protocols (e.g., [16]) use the number $b + 1$, we require a larger number because opaque quorum systems allow that some non-faulty servers may accept conflicting values. We assume an appropriate propagation algorithm (e.g., a variant of an epidemic algorithm [10] such as [16]). At a high level, a non-faulty server has two responsibilities. First, having accepted a write value and returned a response to the client, it periodically informs other servers that it has accepted the value. Second, if it has not yet accepted a value upon learning that a threshold number, p, of servers have accepted the value, it accepts the value. Faulty servers are all assumed to have access to any conflicting value directly without propagation, so we assume no additional constraints on their behavior.

Lemma 5.1. *Let $n < 2q_{\text{wt}} - 2b$ and $p = n - q_{\text{wt}} + b + 1$. Then an established value will be accepted and propagated by at least p non-faulty servers, and no conflicting value can be propagated by p servers (faulty or non-faulty).*

Proof. The established value is accepted by at least $q_{\text{wt}} - b$ non-faulty servers by definition. If we allow that all other servers may forward a conflicting value, then the number of servers that forward a conflicting value is $n - (q_{\text{wt}} - b)$. First, note that $p > n - (q_{\text{wt}} - b)$ by the lemma. In addition, $p \leq q_{\text{wt}} - b$ because,

$$\begin{aligned} & n < 2q_{\text{wt}} - 2b \\ \Leftrightarrow\ & n - (q_{\text{wt}} - b) < q_{\text{wt}} - b \\ \Leftrightarrow\ & n - (q_{\text{wt}} - b) + 1 \leq q_{\text{wt}} - b \\ \Leftrightarrow\ & p \leq q_{\text{wt}} - b \end{aligned}$$

□

For example, if $q_{\text{wt}} = n - b$ and $n > 4b$ we set $p = 2b + 1$. Since the established value will be accepted by at least p non-faulty servers, it will propagate. No conflicting value will propagate.

If the conditions of Lemma 5.1 do not hold, we must allow for some probability of error during propagation. We set p so that it is between the expectations of the minimum number of non-faulty servers that accept an established write ($\mathsf{PCorrect}$), and the maximum number of servers that propagate a conflicting value ($\mathsf{PConflicting}$). We now derive expressions for $\mathsf{PCorrect}$ and $\mathsf{PConflicting}$. First, note that $\mathsf{PCorrect} = |\mathsf{C}_{\text{wt}}|$. Therefore, by (13), (7), and linearity of expectation we have:

$$\begin{aligned} \mathbb{E}\left[\mathsf{PCorrect}\right] &= \mathbb{E}\left[|\mathsf{C}_{\text{wt}}|\right] \\ &= \mathbb{E}\left[q_{\text{wt}} - \mathsf{MalWrite}\right] \\ &= q_{\text{wt}} - \mathbb{E}\left[\mathsf{MalWrite}\right] \\ &= q_{\text{wt}} - \frac{a_{\text{wt}} b}{n} \\ &= \frac{n q_{\text{wt}} - a_{\text{wt}} b}{n}. \end{aligned} \tag{44}$$

PConflicting is the number of servers in the disjoint sets B and $(A'_{\mathrm{wt}} \setminus (C_{\mathrm{wt}} \cup B))$. Therefore,

$$\mathbb{E}\left[\mathsf{PConflicting}\right] = b + \mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \setminus (\mathsf{C}_{\mathrm{wt}} \cup B)|\right]. \tag{45}$$

Note that,

$$\begin{aligned}
&\mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \setminus (\mathsf{C}_{\mathrm{wt}} \cup B)|\right] \\
&= \mathbb{E}\left[|(\mathsf{A}'_{\mathrm{wt}} \setminus B) \setminus \mathsf{C}_{\mathrm{wt}}|\right] \\
&= \mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \setminus B| - |\mathsf{A}'_{\mathrm{wt}} \cap (\mathsf{C}_{\mathrm{wt}} \setminus B)|\right] \\
&= \mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \setminus B|\right] - \mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \cap \mathsf{C}_{\mathrm{wt}}|\right].
\end{aligned} \tag{46}$$

$|\mathsf{A}'_{\mathrm{wt}} \setminus B|$ is a hypergeometric random variable with expectation,

$$\mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \setminus B|\right] = \frac{a_{\mathrm{wt}}(n-b)}{n}. \tag{47}$$

Combining and simplifying equations (46), (47), and (23), we have,

$$\mathbb{E}\left[|\mathsf{A}'_{\mathrm{wt}} \setminus (\mathsf{C}_{\mathrm{wt}} \cup B)|\right] = \frac{2a_{\mathrm{wt}}n^2 - na_{\mathrm{wt}}b - q_{\mathrm{wt}}n^2 - a_{\mathrm{wt}}^2 n + a_{\mathrm{wt}}^2 b}{n^2}. \tag{48}$$

So by (45) and (48),

$$\mathbb{E}\left[\mathsf{PConflicting}\right] = \frac{n^2 b + 2a_{\mathrm{wt}}n^2 - na_{\mathrm{wt}}b - n^2 q_{\mathrm{wt}} - a_{\mathrm{wt}}^2 n + a_{\mathrm{wt}}^2 b}{n^2}. \tag{49}$$

Lemma 5.2. *PO-Consistency* $\Rightarrow \mathbb{E}\left[\mathsf{PCorrect}\right] > \mathbb{E}\left[\mathsf{PConflicting}\right]$.

Proof. Recall that PO-Consistency holds iff $\mathbb{E}\left[\mathsf{MinCorrect}\right] > \mathbb{E}\left[\mathsf{MaxConflicting}\right]$. Next,

$$\begin{aligned}
&\mathbb{E}\left[\mathsf{MinCorrect}\right] > \mathbb{E}\left[\mathsf{MaxConflicting}\right] \\
&\Leftrightarrow \frac{n}{q_{\mathrm{rd}}}\mathbb{E}\left[\mathsf{MinCorrect}\right] > \frac{n}{q_{\mathrm{rd}}}\mathbb{E}\left[\mathsf{MaxConflicting}\right] \\
&\Rightarrow \frac{n}{q_{\mathrm{rd}}}\mathbb{E}\left[\mathsf{MinCorrect}\right] > \frac{n}{a_{\mathrm{rd}}}\mathbb{E}\left[\mathsf{MaxConflicting}\right] \\
&\Leftrightarrow \mathbb{E}\left[\mathsf{PCorrect}\right] > \mathbb{E}\left[\mathsf{PConflicting}\right]
\end{aligned}$$

The final line follows from definitions (44) and (49), and Lemmas 4.1 and 4.3. □

Lemma 5.2 shows that we can set p as described for any system in which PO-Consistency holds.

6 Evaluation

In this section, we analyze error probabilities for concrete system sizes. In addition to validating our results from Section 4, this shows that an access restriction protocol like that of Section 5 can provide significant advantages in terms of worst-case error probabilities.

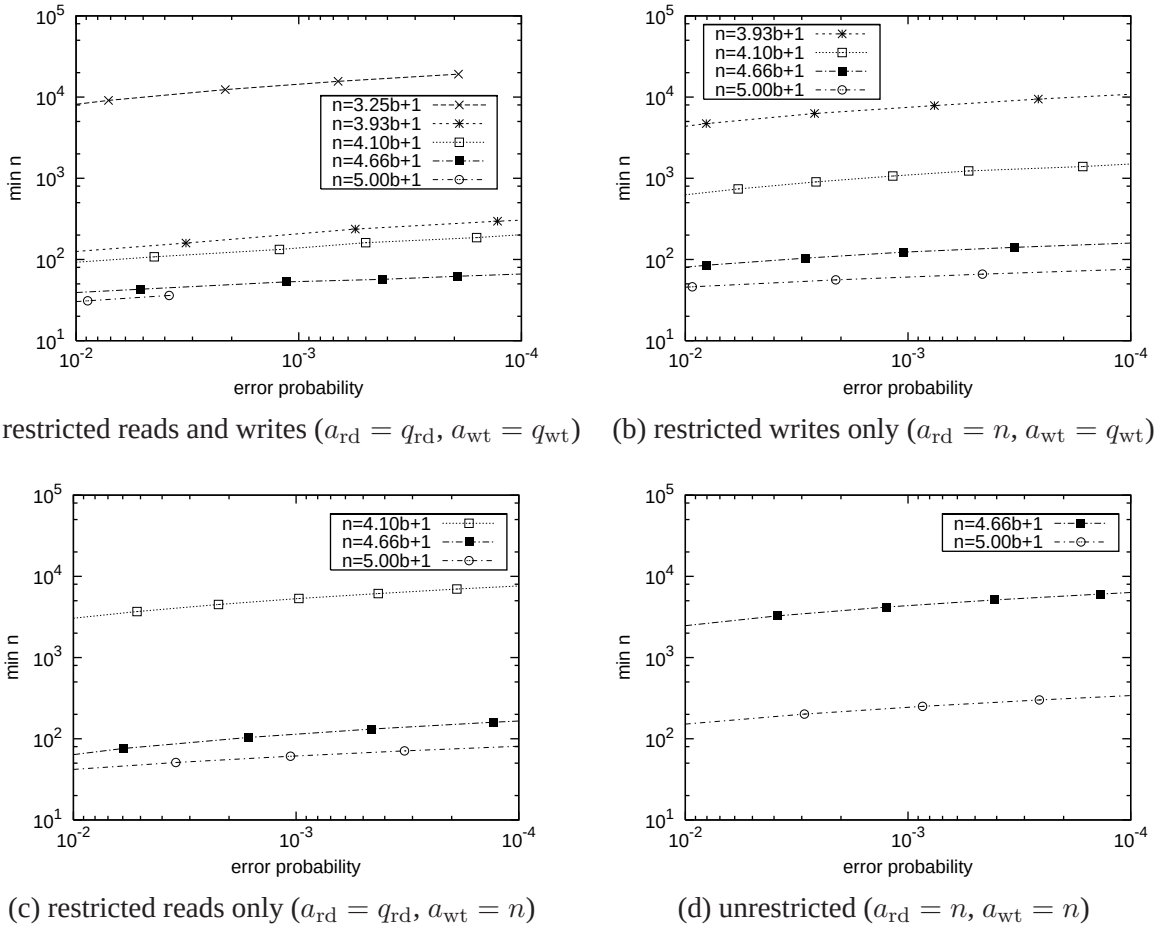
Figure 5 plots the total number of nodes required to achieve a given calculated error probability for each of the configurations that tolerate faulty clients where $q_{\mathrm{wt}} = q_{\mathrm{rd}} = n - b$. Since the *unrestricted* configuration ($a_{\mathrm{rd}} = n$, $a_{\mathrm{wt}} = n$) shown in Figure 5(d) does not require the access-restriction protocol of Section 5, yet yields the best maximum ratios of b to n of all the configurations that provide single-phase reads and writes (Section 4.4.1), we do not evaluate the error probabilities of those other configurations here. In all cases, the error probabilities are worst-case in that they reflect the situation in which all b nodes are in fact faulty. For each configuration, we provide plots for different ratios of n to b, ranging from the maximum b for a given configuration, to $n = 5b + 1$, as a comparison with strict opaque quorum systems. Appendix B provides details of our calculations.

Overall, we find that our constructions can tolerate significantly more than $b = n/5$ faulty servers, while providing error probabilities in the range of 10^{-2} to 10^{-4} for systems with fewer than 50 servers to hundreds of servers. Coupled with the dissemination of correct values between servers (off the critical path), as described in Section 5, the error probability decreases between writes.

Within each figure, we see that to decrease the worst-case error probability, we can either keep the same function of b in terms of n while increasing n, or hold n fixed while decreasing the number of faults the system can tolerate. For example, Figure 5(b) shows that if $b = (n-1)/4.66$, we decrease the worst-case error probability from $\sim 10^{-2}$ to $\sim 10^{-4}$ by increasing the system size from 48 servers to 141 servers. On the other hand, Figure 5(a) shows us that if we keep n fixed at ~ 100 servers, we can provide order of 10^{-3} worst-case error probability with $(n-1)/4.10$ faulty servers, but provide only order of 10^{-2} worst-case error probability for $(n-1)/3.93$ faulty servers.

Considering two figures together, we see that configurations that tolerate a larger b also provide better error probabilities for a given b. For example, Figure 5(a) shows that by restricting reads and writes, a system of approximately 130 servers can tolerate $(n-1)/4.10$ faults with a worst-case error probability on the order of 10^{-3}. By comparison, Figure 5(b) shows that, if we restrict only writes, the same degree of fault tolerance and low error probability requires more than 1000 servers. As such, an access restriction protocol like that of Section 5 provides real benefits in terms of worst-case error probabilities.

While a very large number of servers is required for any configuration to tolerate its theoretical limit on b with small error probability, each configuration can tolerate close to its limit with far fewer servers. For example, Figure 5(a) shows that, if we restrict reads and writes, it requires more than 10,000 servers to tolerate $(n-1)/3.25$ faults with worst-case error probability on the order of 10^{-3}. However, by decreasing the fraction of faults that can be tolerated to $(n-1)/3.93$, we can achieve the same error probability with ~ 200 servers—$1/50$ of the servers. However, this is not a linear function; if we again reduce the fraction of servers we can tolerate by a similar amount to $(n-1)/4.66$, we reduce the minimum n to approximately 50 servers—only by $1/4$.



(a) restricted reads and writes ($a_{\mathrm{rd}} = q_{\mathrm{rd}}$, $a_{\mathrm{wt}} = q_{\mathrm{wt}}$)

(b) restricted writes only ($a_{\mathrm{rd}} = n$, $a_{\mathrm{wt}} = q_{\mathrm{wt}}$)

(c) restricted reads only ($a_{\mathrm{rd}} = q_{\mathrm{rd}}$, $a_{\mathrm{wt}} = n$)

(d) unrestricted ($a_{\mathrm{rd}} = n$, $a_{\mathrm{wt}} = n$)

Figure 5: Number of servers required to achieve given calculated worst-case error probability.

7 Conclusion

First, we have presented probabilistic opaque quorum systems (POQS), a new type of opaque quorum system that we have shown can tolerate up to $n/3.15$ Byzantine servers (compared with $n/5$ Byzantine servers for strict opaque quorum systems) with high probability, while preserving the properties that make opaque quorums useful for optimistic Byzantine-fault-tolerant service protocols. Second, we have presented an optional, novel access-restriction protocol for POQS that provides the ability for servers to constrain clients so that they use randomly selected access sets for operations. With POQS, we expect to create probabilistic optimistic Byzantine fault-tolerant service protocols that tolerate substantially more faults than current optimistic protocols. While strict opaque quorums systems may be more appropriate for smaller systems that require no chance of error, a POQS can provide increased fault tolerance for a given number of nodes, with a worst-case error probability that is bounded and that decreases as the system scales.

References

[1] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie. Fault-scalable Byzantine fault-tolerant services. In *Symposium on Operating Systems Principles*, October 2005.

[2] A. S. Aiyer, L. Alvisi, and R. A. Bazzi. On the availability of non-strict quorum systems. In *DISC 2005*, pages 48–62, 2005.

[3] A. S. Aiyer, L. Alvisi, and R. A. Bazzi. Byzantine and multi-writer k-quorums. In *DISC 2006*, pages 443–458, 2006.

[4] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *EUROCRYPT 2002*, pages 83–107, London, UK, 2002.

[5] A. Back. Hashcash - a denial of service counter-measure. `http://cypherspace.org/hashcash/hashcash.pdf`, August 2002.

[6] R. A. Bazzi. Access cost for asynchronous Byzantine quorum systems. *Distributed Computing*, 14(1):41–48, 2001.

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Conference on Computer and Communications Security*, pages 62–73, 1993.

[8] C. Cachin and S. Tessaro. Optimal resilience for erasure-coded Byzantine distributed storage. In *International Conference on Dependable Systems and Networks*, 2006.

[9] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, 2002.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Principles of Distributed Computing*, pages 1–12, August 1987.

[11] M. Herlihy and J. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3):463–492, 1990.

[12] M. Jakobsson and A. Juels. Proofs of work and bread pudding protocols. In *Communications and Multimedia Security*, pages 258–272, 1999.

[13] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Network and Distributed Systems Security Symposium*, pages 151–165, 1999.

[14] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[15] B. Liskov and R. Rodrigues. Tolerating Byzantine faulty clients in a quorum system. In *International Conference on Distributed Computing Systems*, 2006.

[16] D. Malkhi, Y. Mansour, and M. K. Reiter. Diffusion without false rumors: On propagating updates in a Byzantine environment. *Theoretical Computer Science*, 299(1–3):289–306, 2003.

[17] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[18] D. Malkhi, M. K. Reiter, A. Wool, and R. N. Wright. Probabilistic quorum systems. *Information and Computation*, 170(2):184–206, 2001.

[19] J.-P. Martin and L. Alvisi. Fast Byzantine consensus. *IEEE Transactions on Dependable and Secure Computing*, 3(3):202–215, 2006.

[20] C. McDiarmid. Concentration for independent permutations. *Combinatorics, Probability and Computing*, 11(2):163–178, 2002.

[21] M. Mitzenmacher and E. Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[22] M. Molloy and B. Reed. *Graph Colouring and the Probabilistic Method*. Springer, 2002.

[23] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

[24] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, 2002.

[25] H. Yu. Signed quorum systems. *Distributed Computing*, 18(4):307–323, 2006.

Appendix A Intersection of Two Write Quorums

We show in this section that O-Consistency (from Section 4.1), although phrased in terms of intersection between a read quorum and a write quorum, implies sufficient constraints in terms of two write quorums in the threshold quorum system model that we assume. In particular, as stated in [1], the constraint on the intersection of two write quorums in an opaque quorum system ensures that if one write is established, no conflicting write is established or can be repaired successfully. That is, let a *repairable set*, $R \in \mathcal{R}(Q_{\mathrm{wt}})$, be any set of r servers from Q_{wt}; then,

$$\forall Q_{\mathrm{wt}}, Q'_{\mathrm{wt}} \in \mathcal{Q}_{\mathrm{wt}}, \forall R \in \mathcal{R}(Q'_{\mathrm{wt}}) : Q_{\mathrm{wt}} \cap R \notin B \tag{50}$$

We demonstrate that O-Consistency provides the same guarantee by showing that it implies (50):

$$\begin{aligned}
&\forall Q_{\mathrm{rd}} \in \mathcal{Q}_{\mathrm{rd}}, \forall Q_{\mathrm{wt}} \in \mathcal{Q}_{\mathrm{wt}} : r > \textbf{conflicting}(Q_{\mathrm{rd}}, Q_{\mathrm{wt}}) \\
&\Leftrightarrow \forall Q_{\mathrm{rd}} \in \mathcal{Q}_{\mathrm{rd}}, \forall Q_{\mathrm{wt}} \in \mathcal{Q}_{\mathrm{wt}} : r > |(Q_{\mathrm{rd}} \cap B) \cup (Q_{\mathrm{rd}} \setminus Q_{\mathrm{wt}})| \\
&\Leftrightarrow \forall Q_{\mathrm{rd}} \in \mathcal{Q}_{\mathrm{rd}}, \forall Q_{\mathrm{wt}} \in \mathcal{Q}_{\mathrm{wt}} : r > q_{\mathrm{rd}} - |(Q_{\mathrm{rd}} \cap Q_{\mathrm{wt}}) \setminus B| \\
&\Leftrightarrow r > q_{\mathrm{rd}} - ((q_{\mathrm{rd}} + q_{\mathrm{wt}} - n) + (n - b) - n) \\
&\Leftrightarrow r > n + b - q_{\mathrm{wt}} \\
&\Leftrightarrow q_{\mathrm{wt}} + r - n > b \\
&\Leftrightarrow \forall Q_{\mathrm{wt}}, Q'_{\mathrm{wt}} \in \mathcal{Q}_{\mathrm{wt}}, \forall R \in \mathcal{R}(Q'_{\mathrm{wt}}) : Q_{\mathrm{wt}} \cap R \notin B
\end{aligned}$$

Appendix B Calculating ϵ

To perform our calculations, we use the *R language*, and a dynamic programming approach due to numerous summations of terms. Here, we describe the calculations that we perform.

First, in accordance with Section 4.5, we set r as follows,

$$r = \lceil (\mathbb{E}\left[\mathsf{MinCorrect}\right] + \mathbb{E}\left[\mathsf{MaxConflicting}\right])/2 \rceil.$$

Next, to determine the error probability accurately, we calculate the error probabilities for non-faulty clients (ϵ_1) and faulty clients (ϵ_2) independently. We determine ϵ as,

$$\epsilon = \textbf{max}(\epsilon_1, \epsilon_2).$$

To facilitate this, we define new random variables and functions. While, as described in Section 4.3, a faulty client chooses votes for the incorrect value from the entire read access set taken on by A_{rd}, a non-faulty client uses only the randomly chosen read quorum taken on by Q_{rd} for choosing such votes. Therefore, for a non-faulty client, in place of $\mathsf{MaxConflicting}$ we define $\mathsf{MaxConflicting}'$ (compare with (11)),

$$\begin{aligned}
\mathsf{Malevolent}' &= |\mathsf{Q}_{\mathrm{rd}} \cap B| \\
\mathsf{Conflicting}' &= |\mathsf{Q}_{\mathrm{rd}} \cap (\mathsf{A}'_{\mathrm{wt}} \setminus (\mathsf{C}_{\mathrm{wt}} \cup B))| \\
\mathsf{MaxConflicting}' &= \mathsf{Malevolent}' + \mathsf{Conflicting}'
\end{aligned}$$

Similarly, while the read quorum (taken on by Q_{rd}) that is used by a non-faulty client contains at least MinCorrect votes from non-faulty servers that return the correct value, the read access set from which a faulty client selects votes (taken on by A_{rd}) contains at least MinCorrect$'$ such votes (compare with (8)),

$$\mathsf{MinCorrect}' = |\mathsf{A}_{\mathrm{rd}} \cap \mathsf{C}_{\mathrm{wt}}|$$

Finally, let QStale and AStale be the maximum number of non-faulty servers with stale values in Q_{rd} and A_{rd}, respectively,

$$\begin{aligned}\mathsf{QStale} &= q_{\mathrm{rd}} - \mathsf{MinCorrect} - \mathsf{MaxConflicting}' \\ \mathsf{AStale} &= a_{\mathrm{rd}} - \mathsf{MinCorrect}' - \mathsf{MaxConflicting}\end{aligned}$$

Then, for a non-faulty client, we calculate the error probability as,

$$\begin{aligned}\epsilon_1 &= \Pr(\mathsf{MinCorrect} \leq r \vee \mathsf{MaxConflicting}' > r) \\ &= \Pr(\mathsf{MinCorrect} \leq r \vee q_{\mathrm{rd}} - \mathsf{MinCorrect} - \mathsf{QStale} > r) \\ &= \Pr(\mathsf{MinCorrect} \leq r \vee \mathsf{MinCorrect} < q_{\mathrm{rd}} - r - \mathsf{QStale}) \\ &= \Pr(\mathsf{MinCorrect} \leq \mathbf{max}(r, q_{\mathrm{rd}} - r - \mathsf{QStale} - 1))) \\ &= \sum_z \Pr(\mathsf{MinCorrect} \leq \mathbf{max}(r, q_{\mathrm{rd}} - r - \mathsf{QStale} - 1)) \Pr[\mathsf{QStale} = z]\end{aligned}$$

For a faulty client, we calculate the error probability as,

$$\begin{aligned}\epsilon_2 &= \Pr(\mathsf{MaxConflicting} > r) \\ &= \Pr(a_{\mathrm{rd}} - \mathsf{MinCorrect}' - \mathsf{AStale} > r) \\ &= \Pr(\mathsf{MinCorrect}' < a_{\mathrm{rd}} - r - \mathsf{AStale}) \\ &= \Pr(\mathsf{MinCorrect}' \leq a_{\mathrm{rd}} - r - \mathsf{AStale} - 1) \\ &= \sum_z \Pr(\mathsf{MinCorrect}' \leq a_{\mathrm{rd}} - r - z - 1) \Pr[\mathsf{AStale} = z]\end{aligned}$$

For a hypergeometric random variable $\mathsf{H} \sim \mathbf{hyp}(w, t, d)$ defined by d draws from a population of t elements containing w success elements, we can directly calculate the cumulative distribution function ($\Pr[\mathsf{H} \leq l]$) and the probability mass function ($\Pr[\mathsf{H} = l]$).

Based on the description of MinCorrect in Section 4.4, we have that $(\mathsf{MinCorrect} \mid \mathsf{MalWrite} = m) \sim \mathbf{hyp}(q_{\mathrm{wt}} - m, n, q_{\mathrm{rd}})$. Then,

$$\Pr[\mathsf{MinCorrect} \leq x] = \sum_m \Pr[(\mathsf{MinCorrect} \mid \mathsf{MalWrite} = m) \leq x] \Pr[\mathsf{MalWrite} = m].$$

We calculate $\Pr[\mathsf{MinCorrect}' \leq x]$ in the same fashion.

In Section 4.4 when calculating $\mathbb{E}[\mathsf{MaxConflicting}]$ we assume the worst-case behavior of faulty clients described in Section 4.2. Therefore, to calculate $\Pr[\mathsf{QStale} = z]$, we again assume

that $A_{\mathrm{wt}} \setminus (B \cup A'_{\mathrm{wt}}) \subseteq C_{\mathrm{wt}}$, that all of the servers in $A'_{\mathrm{wt}} \setminus B$ have either the correct value or the conflicting value, and that all of the servers in B return the conflicting value if polled. Therefore, the number of servers in Q_{rd} that return neither the correct value nor the conflicting value are,

$$\mathsf{QStale} = |\mathsf{Q}_{\mathrm{rd}} \setminus (\mathsf{A}_{\mathrm{wt}} \cup \mathsf{A}'_{\mathrm{wt}} \cup B)|.$$

Since $\mathsf{Q}_{\mathrm{rd}} \setminus (\mathsf{A}_{\mathrm{wt}} \cup \mathsf{A}'_{\mathrm{wt}} \cup B) = (\mathsf{Q}_{\mathrm{rd}} \setminus \mathsf{A}_{\mathrm{wt}}) \setminus \mathsf{A}'_{\mathrm{wt}}) \setminus B$, we calculate $\Pr[\mathsf{QStale} = z]$ as follows. Let $\mathsf{W} = |(U \setminus \mathsf{A}'_{\mathrm{wt}}) \setminus B|$ and $\mathsf{V} = |((U \setminus \mathsf{A}_{\mathrm{wt}}) \setminus \mathsf{A}'_{\mathrm{wt}}) \setminus B|$. Since Q_{rd}, A_{wt}, and A'_{wt} are chosen independently, $\mathsf{W} \sim \mathbf{hyp}(n-b, n, n-a_{\mathrm{wt}})$, $(\mathsf{V} \mid \mathsf{W} = w) \sim \mathbf{hyp}(w, n, n-awt)$, and $(\mathsf{QStale} \mid \mathsf{V} = v) \sim \mathbf{hyp}(v, n, q_{\mathrm{rd}})$. As such,

$$\Pr[\mathsf{V} = v] = \sum_{w} \Pr[(\mathsf{V} \mid \mathsf{W} = w) = v] \Pr[\mathsf{W} = w].$$

and,

$$\Pr[\mathsf{QStale} = z] = \sum_{v} \Pr[(\mathsf{QStale} \mid \mathsf{V} = v) = z] \Pr[\mathsf{V} = v].$$

We can calculate $\Pr[\mathsf{AStale} = z]$ in the same fashion.