Write Markers for Probabilistic Quorum Systems

Michael G. Merideth[1] and Michael K. Reiter[2]

[1] Carnegie Mellon University, Pittsburgh, PA, USA
[2] University of North Carolina, Chapel Hill, NC, USA

Abstract. Probabilistic quorum systems can tolerate a larger fraction of faults than can traditional (strict) quorum systems, while guaranteeing consistency with an arbitrarily high probability for a system with enough replicas. However, the masking and opaque types of probabilistic quorum systems are hampered in that their optimal load—a best-case measure of the work done by the busiest replica, and an indicator of scalability—is little better than that of strict quorum systems. In this paper we present a variant of probabilistic quorum systems that uses *write markers* in order to limit the extent to which Byzantine-faulty servers act together. Our masking and opaque probabilistic quorum systems have asymptotically better load than the bounds proven for previous masking and opaque quorum systems. Moreover, the new masking and opaque probabilistic quorum systems can tolerate an additional 24% and 17% of faulty replicas, respectively, compared with probabilistic quorum systems without write markers.

1 Introduction

Given a universe U of servers, a *quorum system* over U is a collection $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ such that each $Q_i \subseteq U$ and

$$|Q \cap Q'| > 0 \tag{1}$$

for all $Q, Q' \in \mathcal{Q}$. Each Q_i is called a *quorum.* The intersection property (1) makes quorums a useful primitive for coordinating actions in a distributed system. For example, if clients perform writes at a quorum of servers, then a client who reads from a quorum will observe the last written value. Because of their utility in such applications, quorums have a long history in distributed computing.

In systems that may suffer Byzantine faults [1], the intersection property (1) is typically not adequate as a mechanism to enable consistent data access. Because (1) requires only that the intersection of quorums be non-empty, it could be that two quorums intersect only in a single server, for example. In a system in which up to $b > 0$ servers might suffer Byzantine faults, this single server might be faulty and consequently, could fail to convey the last written value to a reader, for example.

For this reason, Malkhi and Reiter [2] proposed various ways of strengthening the intersection property (1) so as to enable quorums to be used in Byzantine environments. For example, an alternative to (1) is

$$|Q \cap Q' \setminus B| > |Q' \cap B| \tag{2}$$

for all $Q, Q' \in \mathcal{Q}$, where B is the (unknown) set of all (up to b) servers that are faulty. In other words, the intersection of any two quorums contains more non-faulty servers than the faulty ones in either quorum. As such, the responses from these non-faulty servers will outnumber those from faulty ones. These quorum systems are called *masking* systems.

Opaque quorum systems, have an even more stringent requirement as an alternative to (1):

$$|Q \cap Q' \setminus B| > |(Q' \cap B) \cup (Q' \setminus Q)| \tag{3}$$

for all $Q, Q' \in \mathcal{Q}$. In other words, the number of correct servers in the intersection of Q and Q' (i.e., $|Q \cap Q' \setminus B|$) exceeds the number of faulty servers in Q' (i.e., $|Q' \cap B|$) together with the number of servers in Q' but not Q. The rationale for this property can be seen by considering the servers in Q' but not Q as "outdated", in the sense that if Q was used to perform an update to the system, then those servers in $Q' \setminus Q$ are unaware of the update. As such, if the faulty servers in Q' behave as the outdated ones do, their behavior (i.e., their responses) will dominate that from the correct servers in the intersection ($Q \cap Q' \setminus B$) unless (3) holds.

The increasingly stringent properties of Byzantine quorum systems come with costs in terms of the smallest system sizes that can be supported while tolerating a number b of faults [2]. This implies that a system with a fixed number of servers can tolerate fewer faults when the property is more stringent as seen in Table 1, which refers to the quorums just discussed as *strict*. Table 1 also shows the negative impact on the ability of the system to disperse load amongst the replicas, as discussed next.

Naor and Wool [3] introduced the notion of an *access strategy* by which clients select quorums to access. An access strategy $p : \mathcal{Q} \to [0, 1]$ is simply a probability distribution on quorums, i.e., $\sum_{Q \in \mathcal{Q}} p(Q) = 1$. Intuitively, when a client accesses the system, it does so at a quorum selected randomly according to the distribution p.

The formalization of an access strategy is useful as a tool for discussing the load dispersing properties of quorums. The *load* [3] of a quorum system, $\mathcal{L}(\mathcal{Q})$, is the probability with which the busiest server is accessed in a client access, under the best possible access strategy p. As listed in Table 1, tight lower bounds have been proven for the load of each type of strict Byzantine quorum system. The load for opaque quorum systems is particularly unfortunate—systems that utilize opaque quorum systems cannot effectively disperse processing load across more servers (i.e., by increasing n) because the load is at least a constant. Such Byzantine quorum systems are used by many modern Byzantine-fault-tolerant protocols, e.g., [4–9] in order to tolerate the arbitrary failure of a subset of their replicas. As such, circumventing the bounds is an important topic.

One way to circumvent these bounds is with *probabilistic quorum systems*. Probabilistic quorum systems relax the quorum intersection properties, asking them to hold only with high probability. More specifically, they relax (2) or (3), for example, to hold only with probability $1-\epsilon$ (for ϵ, a small constant), where probabilities are taken with respect to the selection of quorums according to an access strategy p [10, 11]. This technique yields masking quorum constructions tolerating $b < n/2.62$ and opaque quorum constructions tolerating $b < n/3.15$ as seen in Table 1. These bounds hold in the sense that for any $\epsilon > 0$ there is an n_0 such that for all $n > n_0$, the required intersection property ((2) or (3) for masking and opaque quorum systems, respectively) holds with probability at least $1-\epsilon$. Unfortunately, probabilistic quorum systems alone do not materially improve the load of Byzantine quorum systems.

In this paper, we present an additional modification, *write markers*, that improves on the bounds further. Intuitively, in each update access to a quorum of servers, a write marker is placed at the accessed servers in order to evidence the quorum used in that access. This write marker identifies the quorum used; as such, faulty servers not in this quorum cannot respond to subsequent quorum accesses as though they were.

As seen in Table 1, by using this method to constrain how faulty servers can collaborate, we show that probabilistic masking quorum systems with load $O(1/\sqrt{n})$ can be achieved, allowing the systems to disperse load independently of the value of b. Further, probabilistic opaque quorum systems with load $O(b/n)$ can be achieved, breaking the constant lower bound on load for opaque systems. Moreover, the resilience of probabilistic masking quorums can be improved an additional 24% to $b < n/2$, and the resilience of probabilistic opaque quorum systems can be improved an additional 17% to $b < n/2.62$.

Table 1. Improvements due to write markers. (**Bold** entries are properties of particular constructions; others are lower bounds.)

Non-Byzantine:	**load**		**faults**	
strict	$\Omega(1/\sqrt{n})$	[3]	$< n$	
Masking:	**load**		**faults**	
strict	$\Omega(\sqrt{b/n})$	[2]	$< n/4.00$	[12]
probabilistic	$\Omega(b/n)$	[10]	$< \mathbf{n/2.62}$	[11]
write markers	$\mathbf{O(1/\sqrt{n})}$	[here]	$< \mathbf{n/2.00}$	[here]
Opaque:	**load**		**faults**	
strict	$\geq 1/2$	[2]	$< n/5.00$	[2]
probabilistic	unproven		$< \mathbf{n/3.15}$	[11]
write markers	$\mathbf{O(b/n)}$	[here]	$< \mathbf{n/2.62}$	[here]

The probability of error in probabilistic quorums requires mechanisms to ensure that accesses are performed according to the required access strategy p if the clients cannot be trusted to do so. Therefore, we adapt one such mechanism, the access-restriction protocol of probabilistic opaque quorum systems [11], to accomodate write markers. Thus, as a side benefit, our implementation forces faulty clients to follow the access strategy. With this, we provide a protocol to implement write markers that tolerates Byzantine clients.

Our primary contributions are (i) the identification and analysis of the benefits of write markers; and (ii) a proposed implementation of write markers that handles the complexities of tolerating Byzantine clients. Our analysis yields the following results:

Masking Quorums: We show that the use of write markers allows probabilistic masking quorum systems to tolerate up to $b < n/2$ faults when quorums are of size $\Omega(\sqrt{n})$. Setting all quorums to size $\rho\sqrt{n}$ for some constant ρ, we achieve a load that is asymptotically optimal for any quorum system, i.e., $\rho\sqrt{n}/n = O(1/\sqrt{n})$ [3].

This represents an improvement in load and the number of faults that can be tolerated. Probabilistic masking quorums without write markers can tolerate up to $b < n/2.62$ faults [11] and achieve load no better than $\Omega(b/n)$ [10]. In addition, the maximum number of faults that can be tolerated is tied to the size of quorums [10]. Thus, without write markers, achieving optimal load requires tolerating fewer faults. Strict masking quorum systems can tolerate (only) up to $b < n/4$ faults [2] and can achieve load $\Omega(\sqrt{b/n})$ [12].

Opaque Quorums: We show that the use of write markers allows probabilistic opaque quorum systems to tolerate up to $b < n/2.62$ faults. We present a construction with load $O(b/n)$ when $b = \Omega(\sqrt{n})$, thereby breaking the constant lower bound of 1/2 on the load of strict opaque quorum systems [2]. Moreover, if $b = O(\sqrt{n})$, we can set all quorums to size $\rho\sqrt{n}$ for some constant ρ, in order to achieve a load that is asymptotically optimal for any quorum system, i.e., $\rho\sqrt{n}/n = O(1/\sqrt{n})$ [3].

This represents an improvement in load and the number of faults that can be tolerated. Probabilistic opaque quorum systems without write markers can tolerate (only) up to $b < n/3.15$ faults [11]. Strict opaque quorum systems can tolerate (only) up to $b < n/5$ faults [2]; these quorum systems can do no better than constant load even if $b = 0$ [2].

2 Definitions and System Model

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 faults [1]). 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). However, for our implementation of write markers, as is typical for many Byzantine-fault-tolerant protocols (c.f., [4–6, 9]), we assume that faulty clients and servers are computationally bound such that they cannot subvert standard cryptographic primitives such as digital signatures.

Communication. Write markers require no communication assumptions beyond those of the probabilistic quorums for which they are used. For completeness, we summarize the model of [11], which is common to prior works in probabilistic [10] and signed [13] 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.

Access set; access strategy; operation. 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 [14] of the execution.

In the introduction, we discussed *access strategies* as probability distributions on quorums used for operations. For the remainder of the paper, we follow [11] in strictly generalizing the notion of access strategy to apply instead to *access sets* from which quorums are chosen. An access set is a set of servers from which the client selects a quorum. If the client is non-faulty, we assume that this selection is done uniformly at random. We adopt the access strategy that all access sets are chosen uniformly at random (even by faulty clients). In Section 4, we adapt a protocol to support write markers from one in [11] that approximately ensures this access strategy. Our analysis allows that access sets may be larger than quorums, though if access sets and quorums are of the same size, then our protocol effectively forces even faulty clients to select quorums uniformly at random as discussed in the introduction. In our analysis, all access sets used for reads and writes are of constant size a_{rd} and a_{wt} respectively. All quorums used for reads and writes are of constant size q_{rd} and q_{wt} respectively.

Candidate; conflicting; error probability; established; participant; qualified; vote. Each write yields a corresponding *candidate* at some number of servers. A candidate is an abstraction used in part to ensure that two distinct write operations are distinguishable from each other, even if the corresponding data values are the same. A candidate is *established* once it is accepted by all of the non-faulty servers in some write quorum of size q_{wt} within the write access set of size a_{wt}. In opaque quorum systems, property (3) anticipates that different non-faulty servers each may hold a different candidate due to concurrent writes. A candidate that is characterized by the property that a non-faulty server would accept either it or a given established candidate, but not both, is called a *conflicting* candidate. Two candidates may conflict because, e.g., they both bear the same timestamp. In either masking or opaque quorum systems, a faulty server may try to forge a conflicting candidate. No non-faulty server accepts two candidates that conflict with each other.

A server can try to *vote* for some candidate (e.g., by responding to a read operation) if the server is a *participant* in voting (i.e., if the server is a member of the client's read access set). However, a server becomes *qualified* to vote for a particular candidate only if the server is a member of the client's write access set selected for the write operation for which it votes. Non-faulty clients wait for responses from a read quorum of size q_{rd} contained in the read access set of size a_{rd}. An *error* is said to occur in a read operation when a non-faulty client fails to observe the latest value or a faulty client obtains sufficiently many votes for a conflicting value.[3] The *error probability* is the probability of this occurring.

Behavior of faulty clients. We assume that faulty clients seek to maximize the error probability by following specific strategies [11]. This is a conservative assumption; a client cannot increase—but may decrease—the probability of error by failing to follow these strategies. At a high level, the strategies are as follows: a faulty client, which may be completely restricted in its choices: (i) when establishing a candidate, writes the candidate to as few non-faulty servers as possible to minimize the probability that it is observed by a non-faulty client; and (ii) writes a conflicting candidate to as many servers as will accept it (i.e., faulty servers plus, in the case of an opaque quorum system, any non-faulty server that has not accepted the established candidate) in order to maximize the probability that it is observed.

3 Analysis of Write Markers

Intuitively, when a client submits a write, the candidate is associated with a write marker. We require that the following three properties are guaranteed by an implementation of write markers:

W1. Every candidate has a write marker that identifies the access set chosen for the write;
W2. A verifiable write marker implies that the access set was selected uniformly at random (i.e., according to the access strategy);
W3. Every non-faulty client can verify a write marker.

When considering a candidate, non-faulty clients and servers verify the candidate's write marker. Because of this verification, no non-faulty node will accept a vote for a candidate unless the issuing server is qualified to vote for the candidate. Since each write access set is chosen uniformly at random (W2), the faulty servers that can vote for a candidate, i.e., the faulty qualified servers, are therefore a random subset of the faulty servers.

Thus, write markers remove the advantage enjoyed by faulty servers in strict and traditional-probabilistic masking and opaque quorum systems, where any

[3] Faulty clients may be able to affect the system with such votes in some protocols [11].

faulty participant can vote for any candidate—and therefore can collude to have a conflicting, potentially fabricated candidate chosen instead of an established candidate. This aspect of write markers is summarized in Table 2, which shows the impact of write markers in terms of the abilities of faulty and non-faulty servers to vote for a given candidate.

3.1 Consistency Constraints

Probabilistic quorum systems must satisfy constraints similar to those of strict quorum systems (e.g., (2), (3)), but only with probability $1 - \epsilon$. As with strict quorum systems, the purpose of these constraints is to guarantee that operations can be observed consistently in subsequent operations by receiving enough votes.

First, the constraints must ensure in expectation that a non-faulty client can observe the latest established candidate if such a candidate exists. Let Q_{rd} represent a read quorum chosen uniformly at random, i.e., a random variable, from a read access set itself chosen uniformly at random. (Think of this quorum as one used by a non-faulty client.) Let Q_{wt} represent a write quorum chosen by a potentially faulty client; Q_{wt} must be chosen from A_{wt}, an access set chosen uniformly at random. (Think of Q_{wt} as a quorum used for an established candidate.) Then the threshold r number of votes necessary to observe a value must be less than the expected number of non-faulty qualified participants, which is

Table 2. Ability of a server to vote for a given candidate: • (traditional quorums); ⋆ (write markers).

Type of server	**Vote**
Non-faulty qualified participant	• ⋆
Faulty qualified participant	• ⋆
Non-faulty non-qualified participant	
Faulty non-qualified participant	•

$$\mathbb{E}\left[\left|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B\right|\right]. \tag{4}$$

The use of write markers has no impact here on (4) because $(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B$ contains no faulty servers. However, write markers do enable us to set r smaller, as the following shows.

Second, the constraints must ensure that a conflicting candidate (which is in conflict with an established candidate as described in Section 2) is, in expectation, not observed by any client (non-faulty or faulty). In general, it is important for all clients to observe only established candidates so as to enable higher-level protocols (e.g., [4]) that employ repair phases that may affect the state of the system within a read [11]. Let $\mathsf{A}'_{\mathrm{rd}}$ and $\mathsf{A}'_{\mathrm{wt}}$ represent read and write access sets, respectively, chosen uniformly at random. (Think of $\mathsf{A}'_{\mathrm{wt}}$ as the access set used by a faulty client for a conflicting candidate, and of $\mathsf{A}'_{\mathrm{rd}}$ as the access set used by a faulty client for a read operation. How faulty clients can be forced to choose uniformly at random is described in Section 4.) We consider the cases for masking and opaque quorums separately:

Probabilistic Masking Quorums. In a masking quorum system, (2) dictates that only faulty servers may vote for a conflicting candidate. Using write markers, we require that the faulty qualified participants alone cannot produce sufficient votes for a candidate to be observed in expectation. Taking (4) into consideration, we require:

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|(\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap B|\right]. \tag{5}$$

Contrast this with (2) and with the consistency requirement for traditional probabilistic masking quorum systems [10] (adapted to consider access sets), which requires that the faulty participants (qualified or not) cannot produce sufficient votes for a candidate to be observed in expectation:

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|\mathsf{A}'_{\mathrm{rd}} \cap B|\right]. \tag{6}$$

Intuitively, the intersection between access sets can be smaller with write markers because the right-hand side of (5) is less than the right-hand side of (6) if $a_{wt} < n$.

Probabilistic Opaque Quorums. With write markers, we have the benefit, described above for probabilistic masking quorums, in terms of the number of faulty participants that can vote for a candidate in expectation. However, as shown in (3), opaque quorum systems must additionally consider the maximum number of non-faulty qualified participants that vote for the same conflicting candidate in expectation. As such, instead of (5), we have:

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|(\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap B|\right] + \mathbb{E}\left[|\left((\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B\right) \setminus \mathsf{Q}_{\mathrm{wt}}|\right]. \tag{7}$$

Contrast this with the consistency requirement for traditional probabilistic opaque quorums [11]:

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|\mathsf{A}'_{\mathrm{rd}} \cap B|\right] + \mathbb{E}\left[|\left((\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B\right) \setminus \mathsf{Q}_{\mathrm{wt}}|\right]. \tag{8}$$

Again, intuitively, the intersection between access sets can be smaller with write markers because the right-hand side of (7) is less than the right-hand side of (8) if $a_{wt} < n$.

3.2 Implied Bounds

In this subsection, we are concerned with quorum systems for which we can achieve error probability (as defined in Section 2) no greater than a given ϵ for any n sufficiently large. For such quorum systems, there is an upper bound on b in terms of n, akin to the bound for strict quorum systems.

Intuitively, the maximum value of b is limited by the relevant constraint (i.e., either (5) or (7)). Of primary interest are Theorem 1 and its corollaries, which demonstrate the benefits of write markers for probabilistic masking quorum systems, and Theorem 2 and its corollaries, which demonstrate the benefits of write markers for probabilistic opaque quorum systems. They utilize Lemmas 1 and 2, which together present basic requirements for the types of quorum systems with which we are concerned. Due to space constraints, proofs of the lemmas and theorems appear only in a companion technical report [15].

Define $\mathsf{MinCorrect}$ to be a random variable for the number of non-faulty servers with the established candidate, i.e., $\mathsf{MinCorrect} = |(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|$ as indicated in (4).

Lemma 1. *Let $n - b = \Omega(n)$. For all $c > 0$ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$ and $q_{rd}q_{wt} - n = \Omega(1)$, it is the case that $\mathbb{E}[\mathsf{MinCorrect}] > c$ for all n sufficiently large.*

Let r be the threshold, discussed in Section 3.1, for the number of votes necessary to observe a candidate. Define $\mathsf{MaxConflicting}$ to be a random variable for the maximum number of servers that vote for a conflicting candidate. For example: due to (5), in masking quorums with write markers, $\mathsf{MaxConflicting} = |(\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap B|$; and due to (7), in opaque quorums with write markers, $\mathsf{MaxConflicting} = |(\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap B| + |((\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B) \setminus \mathsf{Q}_{\mathrm{wt}}|$.

Lemma 2. *Let the following hold,*[4]

$$\mathbb{E}[\mathsf{MinCorrect}] - \mathbb{E}[\mathsf{MaxConflicting}] > 0,$$
$$\mathbb{E}[\mathsf{MinCorrect}] - \mathbb{E}[\mathsf{MaxConflicting}] = \omega(\sqrt{\mathbb{E}[\mathsf{MinCorrect}]}).$$

Then it is possible to set r such that,

$$\textit{error probability} \to 0 \quad \textit{as } \mathbb{E}[\mathsf{MinCorrect}] \to \infty.$$

Here and below, a suitable setting of r is one between $\mathbb{E}[\mathsf{MinCorrect}]$ and $\mathbb{E}[\mathsf{MaxConflicting}]$, inclusive. The remainder of the section is focused on determining, for each type of probabilistic quorum system, the upper bound on b and bounds on the load that Lemmas 1 and 2 imply.

Theorem 1. *For all ϵ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$, $q_{rd}q_{wt} - n = \Omega(1)$, and*

$$b < \frac{q_{rd}q_{wt}n}{q_{rd}a_{wt} + a_{rd}a_{wt}},$$

any such probabilistic masking quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large.

Corollary 1. *Let $a_{rd} = q_{rd}$ and $a_{wt} = q_{wt}$. For all ϵ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$, $q_{rd}q_{wt} - n = \Omega(1)$, and*

$$b < n/2,$$

any such probabilistic masking quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large.

[4] ω is the little-oh analog of Ω, i.e., $f(n) = \omega(g(n))$ if $f(n)/g(n) \to \infty$ as $n \to \infty$.

In other words, with write markers, the size of quorums does not impact the maximum fraction of faults that can be tolerated when quorums are selected uniformly at random (i.e., when $a_{rd} = q_{rd}$ and $a_{wt} = q_{wt}$).

Corollary 2. *Let $a_{rd} = q_{rd}$, $a_{wt} = q_{wt}$, and $b < n/2$. For all ϵ there is a constant $\rho > 1$ such that if $q_{rd} = q_{wt} = \rho\sqrt{n}$, any such probabilistic masking quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large, and has load*

$$\rho\sqrt{n}/n = O(1/\sqrt{n}).$$

Theorem 2. *For all ϵ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$, $q_{rd}q_{wt} - n = \Omega(1)$, and*

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

any such probabilistic opaque quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large.

Corollary 3. *Let $a_{rd} = q_{rd}$ and $a_{wt} = q_{wt}$. For all ϵ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$, $q_{rd}q_{wt} - n = \Omega(1)$, and*

$$b < \frac{q_{wt}n}{q_{wt} + n},$$

any such probabilistic opaque quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large.

Comparing Corollary 3 with Corollary 1, we see that in the opaque quorum case q_{wt} cannot be set independently of b.

Corollary 4. *Let $a_{rd} = q_{rd}$, $a_{wt} = q_{wt}$, and $b < (q_{wt}n)/(q_{wt} + n)$. For all ϵ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$ and $q_{rd}q_{wt} - n = \Omega(1)$, any such probabilistic opaque quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large, and has load*

$$\Omega(b/n).$$

Corollary 5. *Let $b = \Omega(\sqrt{n})$. For all ϵ there is a constant $d > 1$ such that for all a_{rd}, a_{wt}, q_{rd}, q_{wt} where $a_{rd} = a_{wt} = q_{rd} = q_{wt} = lb$ for a value l such that $c' \geq l > n/(n-b)$ for some constant c', $(lb)^2 > dn$ and $(lb)^2 - n = \Omega(1)$, any such probabilistic opaque quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large, and has load*

$$O(b/n).$$

Corollary 6. *Let $a_{rd} = q_{rd}$ and $a_{wt} = q_{wt} = n - b$. For all ϵ there is a constant $d > 1$ such that for all q_{rd}, q_{wt} where $q_{rd}q_{wt} > dn$, $q_{rd}q_{wt} - n = \Omega(1)$, and*

$$b < n/2.62,$$

any such probabilistic opaque quorum system employing write markers achieves error probability no greater than ϵ given a suitable setting of r for all n sufficiently large.

4 Implementation

Our implementation of write markers provides the behavior assumed in Section 3, even with Byzantine clients. Specifically, it ensures properties W1–W3. (Though, technically, it ensures W2 only approximately in the case of opaque quorum systems, in which, as we explain below, a faulty server might be able to create a conflicting candidate using a write marker for a stale, i.e., out-of-date, access set—but to no advantage.)

Because clients may be faulty, we cannot rely on, e.g., digital signatures issued by them to implement write markers. Instead, we adapt mechanisms of our access-restriction protocol for probabilistic opaque quorum systems [11]. The access-restriction protocol is designed to ensure that all clients follow the access strategy. It already enables non-faulty *servers* to verify this before accepting a write. And, since it is the only way of which we are aware for a probabilistic quorum system to tolerate Byzantine clients when write markers are of benefit (i.e., when the sizes of write access sets are restricted), its mechanisms are appropriate.

The relevant parts of the preexisting protocol work as follows [11]. From a pre-configured number of servers, a client obtains a *verifiable recent value* (VRV), the value of which is unpredictable to clients and b or fewer servers prior to its creation. This VRV is used to generate a pseudorandom sequence of access sets. Since a VRV can be verified using only public information, both it and the sequence of access sets it induces can be verified by clients and servers. Non-faulty clients simply choose the next unused access set for each operation.[5] However, a faulty client is motivated to maximize the probability of error. If the use of the next access set in the sequence does not maximize the probability of error given the current state of the system (i.e., the candidates accepted by the servers), such a client may try to skip ahead some number of access sets. Alternatively, such a client might try to wait to use the next access set until the state of the system changes. If allowed to follow either strategy, such a client would circumvent the access strategy because its choice of access set would not be independent from the state of the system.

Three mechanisms are used together to coerce a faulty client to follow the access strategy. First, the client must perform exponentially increasing work in expectation in order to use later access sets. As such, a client requires exponentially

[5] Non-faulty clients should choose a new access set for each operation to ensure independence from the decisions of faulty clients [11].

increasing time in expectation in order to choose a later access set. This is implemented by requiring that the client solve a client puzzle [16] of the appropriate difficulty. The solution to the puzzle is, in expectation, difficult to find but easy to verify. Second, the VRV and sequence of access sets become invalid as the non-faulty servers accept additional candidates, or as the system otherwise progresses (e.g., as time passes). Non-faulty servers verify that an access set is still valid, i.e., not stale, before accepting it. Thus, system progress forces the client to start its work anew, and, as such, makes the work solving the puzzle for any unused access set wasted. Finally, during the time that the client is working, the established candidate propagates in the background to the non-faulty servers that are non-qualified (c.f., [17]). This decreases the window of vulnerability in which a given access set in the sequence is useful for a conflicting write by making non-qualified servers aware that (i) there is an established candidate (so that they will not accept a conflicting candidate) and (ii) that the state of the system has progressed (so that they will invalidate the current VRV if appropriate).

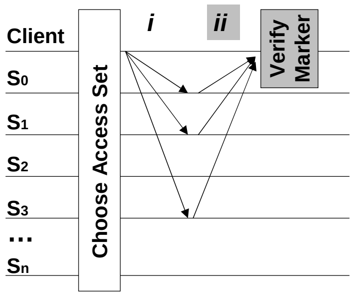

Fig. 1. Read operation with write markers: messages and stages of verification of access set. (Changes in gray.)

The impact of these three mechanisms is that a non-faulty *server* can be confident that the choice of write access set adheres (at least approximately) to the access strategy upon having verified that the access set is valid, current, and is accompanied by an appropriate puzzle solution.

For write markers, we extend the protocol so that, as seen in Figure 1, *clients* can also perform verification. This requires that information about the puzzle solution and access set (including the VRV used to generate it) be returned by the servers to clients. (As seen in Figure 2 and explained below, this information varies across masking and opaque quorum systems.) In the preexisting access-restriction protocol, this information is verified and discarded by each server. For write markers, this information is instead stored by each server in the verification stage as a write marker. It is sent along with the data value as part of the candidate to the client during any read operation. If the server is non-faulty—a fact of which a non-faulty client cannot be certain—the access set used for the operation was indeed chosen according to the access strategy because the server performed verification before accepting the candidate. However, because the server may be faulty, the client performs verification as well; it verifies the write marker and that the server is a member of the access set. This allows us to guarantee points W1–W3. As such, faulty non-qualified servers are unable to vote for the candidates for which qualified servers can vote.

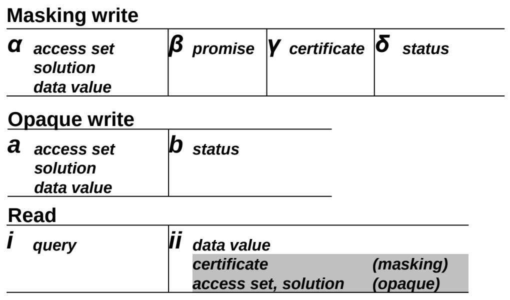

Fig. 2. Message types. (Write marker emphasized with gray.)

Figures 1, 2, 3, and 4 illustrate relevant pieces of the preexisting protocol and our modifications for write markers in the context of read and write operations in probabilistic masking and opaque quorum systems. The figures highlight that the additions to the protocol for write markers involve saving the write markers and returning them to clients so that clients can also verify them.

The differences in the structure of the write marker for probabilistic opaque and masking quorum systems mentioned above results in subtly different guarantees. The remainder of the section discusses these details.

4.1 Probabilistic Opaque Quorums

As seen in Figure 2 (message *ii*), a write marker for a probabilistic opaque quorum system consists of the write-access-set identifier (including the VRV) and the solution to the puzzle that unlocks the use of this access set. Unlike a non-faulty server that verifies the access set at the time of use, a non-faulty client cannot verify that an access set was not already stale when the access set was accepted by a faulty server. Initially, this may appear problematic because it is clear that, given sufficient time, a faulty client will eventually be able to solve the puzzle for its preferred access set to use for a conflicting write—this access set may contain all of the servers in B. In addition, the faulty client can delay the use of this access set because non-faulty clients will be unable to verify whether it was already stale when it was used.

Fortunately, because non-faulty servers will not accept a stale candidate (i.e., a candidate accompanied by a stale access set), the fact that a stale access set may be accepted by a faulty server does not impact the benefit of write markers for opaque quorum systems. In general, consistency requires (7), i.e.,

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|(\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap B|\right] + \mathbb{E}\left[|\,((\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \setminus B) \setminus \mathsf{Q}_{\mathrm{wt}}|\right].$$

However, only faulty servers will accept a stale candidate. Therefore, if the candidate was stale when written to A'_{wt}, no non-faulty server would have accepted it. Thus, in this case, the consistency constraint is equivalent to,

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|(\mathsf{A}'_{\mathrm{rd}} \cap \mathsf{A}'_{\mathrm{wt}}) \cap B|\right].$$

However, this is (6), the constraint on probabilistic masking quorum systems without write markers. In effect, a faulty client must either: (i) use a recent access set that is therefore chosen approximately uniformly at random, and be limited by (7); or (ii), use a stale access set and be limited by (6). If quorums are the sizes of access sets, both inequalities have the same upper bound on b (see [15]); otherwise, a faulty client is disadvantaged by using a stale access set because a system that satisfies (6) can tolerate more faults than one that satisfies (7), and is therefore less likely to result in error (see [15]). Even if the access set contains all of the faulty servers, i.e., $B \subset A'_{wt}$, then this becomes,

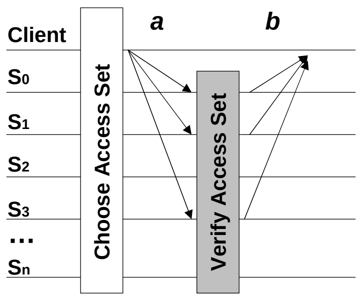

Fig. 3. Write operation in opaque quorum systems: messages and stages of verification of write marker. (Changes in gray.)

$$\mathbb{E}\left[|(\mathsf{Q}_{\mathrm{rd}} \cap \mathsf{Q}_{\mathrm{wt}}) \setminus B|\right] > \mathbb{E}\left[|\mathsf{A}'_{\mathrm{rd}} \cap B|\right].$$

4.2 Probabilistic Masking Quorums

Protocols for masking quorum systems involve an additional round of communication (an echo phase, c.f., [8] or broadcast phase, c.f., [18]) during write operations in order to tolerate Byzantine or concurrent clients. This round prevents non-faulty servers from accepting conflicting data values, as assumed by (2). In order to write a data value, a client must first obtain a *write certificate* (a quorum of replies that together attest that the non-faulty servers will accept no conflicting data value). In contrast to optimistic protocols that use opaque quorum systems, these protocols are pessimistic.

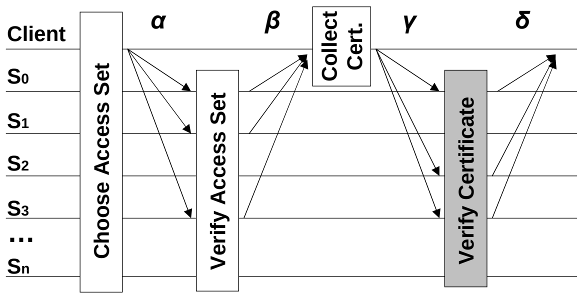

Fig. 4. Write operation in masking quorum systems: messages and stages of verification of write marker. (Changes in gray.)

This additional round allows us to prevent clients from using stale access sets. Specifically, in the request to authorize a data value (message α in Figure 2 and Figure 4), the client sends the access set identifier (including the VRV), the solution to the puzzle enabling use of this access set, and the data value. We require that the certificate come from servers in the access set that is chosen for the write operation. Each server verifies the VRV and that the puzzle solution enables use of the indicated access set before returning authorization (message β in Figure 2 and Figure 4). The non-faulty servers that contribute to the certificate all implicitly agree that the access set is not stale, for otherwise they would not agree to the write. This certificate (sent to each server in message γ in Figure 2 and Figure 4) is stored along with the data value as a write marker. Thus, unlike in probabilistic opaque quorum systems, a verifiable write marker in a probabilistic masking quorum system implies that a stale access set was not used. The reading client verifies the certificate (returned in message *ii* in Figure 1 and Figure 2) before accepting a vote for a candidate. Because a writing client will be unable to obtain a certificate for a stale access set, votes for such a candidate will be rejected by reading clients. Therefore, the analysis in Section 3 applies without additional complications.

5 Additional Related Work

Probabilistic quorum systems were explored in the context of dynamic systems with non-uniform access strategies by Abraham and Malkhi [19]. Recently, probabilistic quorum systems have been used in the context of security for wireless sensor networks [20] as well as storage for mobile ad hoc networks [21]. Lee and Welch make use of probabilistic quorum systems in randomized algorithms for distributed read-write registers [22] and shared queue data structures [23].

Signed quorum systems presented by Yu [13] also weaken the requirements of strict quorum systems but use different techniques. However, signed quorum systems have not been analyzed in the context of Byzantine faults, and so they are not presently affected by write markers.

Another implementation of write markers was introduced by Alvisi et al. [24] for purposes different than ours. We achieve the goals of (i) improving the load, and (ii) increasing the maximum fraction of faults that the system can tolerate by using write markers to prevent some faulty servers from colluding. In contrast to this, Alvisi et al. use write markers in order to increase accuracy in estimating the number of faults present in Byzantine quorum systems, and for identifying faulty servers that consistently return incorrect results. Because the implementation of Alvisi et al. does not prevent faulty servers from lying about the write quorums of which they are members, it cannot be used directly for our purposes. In addition, our implementation is designed to tolerate Byzantine clients, unlike theirs.

6 Conclusion

We have presented write markers, a way to improve the load of masking and opaque quorum systems asymptotically. Moreover, our new masking and opaque probabilistic quorum systems with write markers can tolerate an additional 24% and 17% of faulty replicas, respectively, compared with the proven bounds of probabilistic quorum systems without write markers. Write markers achieve this by limiting the extent to which Byzantine-faulty servers may cooperate to provide incorrect values to clients. We have presented a proposed implementation of write markers that is designed to be effective even while tolerating Byzantine-faulty clients and servers.

References

1. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems **4**(3) (July 1982) 382–401
2. Malkhi, D., Reiter, M.: Byzantine quorum systems. Distributed Computing **11**(4) (1998) 203–213
3. Naor, M., Wool, A.: The load, capacity, and availability of quorum systems. SIAM Journal on Computing **27**(2) (1998) 423–447
4. Abd-El-Malek, M., Ganger, G.R., Goodson, G.R., Reiter, M.K., Wylie, J.J.: Fault-scalable Byzantine fault-tolerant services. In: Symposium on Operating Systems Principles. (October 2005)
5. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Symposium on Operating Systems Design and Implementation. (1999)
6. Goodson, G.R., Wylie, J.J., Ganger, G.R., Reiter, M.K.: Efficient Byzantine-tolerant erasure-coded storage. In: International Conference on Dependable Systems and Networks. (June 2004)
7. Kong, L., Manohar, D., Subbiah, A., Sun, M., Ahamad, M., Blough, D.: Agile store: Experience with quorum-based data replication techniques for adaptive Byzantine fault tolerance. In: IEEE Symposium on Reliable Distributed Systems. (2005) 143–154
8. Malkhi, D., Reiter, M.K.: An architecture for survivable coordination in large distributed systems. IEEE Transactions on Knowledge and Data Engineering **12**(2) (2000) 187–202
9. Martin, J.P., Alvisi, L.: Fast Byzantine consensus. IEEE Transactions on Dependable and Secure Computing **3**(3) (2006) 202–215
10. Malkhi, D., Reiter, M.K., Wool, A., Wright, R.N.: Probabilistic quorum systems. Information and Computation **170**(2) (2001) 184–206
11. Merideth, M.G., Reiter, M.K.: Probabilistic opaque quorum systems. In: International Symposium on Distributed Computing. (2007)
12. Malkhi, D., Reiter, M.K., Wool, A.: The load and availability of Byzantine quorum systems. SIAM Journal of Computing **29**(6) (2000) 1889–1906
13. Yu, H.: Signed quorum systems. Distributed Computing **18**(4) (2006) 307–323
14. Herlihy, M., Wing, J.: Linearizability: A correctness condition for concurrent objects. ACM Transactions on Programming Languages and Systems **12**(3) (1990) 463–492
15. Merideth, M.G., Reiter, M.K.: Write markers for probabilistic quorum systems. Technical Report CMU-CS-07-165R, Computer Science Department, Carnegie Mellon University (November 2008)
16. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: Network and Distributed Systems Security Symposium. (1999) 151–165
17. Malkhi, D., Mansour, Y., Reiter, M.K.: Diffusion without false rumors: On propagating updates in a Byzantine environment. Theoretical Computer Science **299**(1–3) (2003) 289–306

18. Martin, J.P., Alvisi, L., Dahlin, M.: Minimal Byzantine storage. In: International Symposium on Distributed Computing. (2002)
19. Abraham, I., Malkhi, D.: Probabilistic quorums for dynamic systems. Distributed Computing **18**(2) (2005) 113 – 124
20. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key predistribution scheme for wireless sensor networks. ACM Transactions on Information and System Security **8**(2) (2005) 228–258
21. Luo, J., Hubaux, J.P., Eugster, P.T.: Pan: providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. In: International symposium on mobile ad hoc networking and computing. (2003) 1–12
22. Lee, H., Welch, J.L.: Applications of probabilistic quorums to iterative algorithms. In: International Conference on Distributed Computing Systems. (April 2001) 21–30
23. Lee, H., Welch, J.L.: Randomized shared queues applied to distributed optimization algorithms. In: International Symposium on Algorithms and Computation. (December 2001)
24. Alvisi, L., Malkhi, D., Pierce, E., Reiter, M.K.: Fault detection for Byzantine quorum systems. IEEE Transactions on Parallel and Distributed Systems **12**(9) (2001) 996–1007