Fault Detection for Byzantine Quorum Systems
(Extended Abstract)

Lorenzo Alvisi* Dahlia Malkhi† Evelyn Pierce‡ Michael Reiter§

Abstract

In this paper we explore techniques to detect Byzantine server failures in replicated data services. Our goal is to detect arbitrary failures of data servers in a system where each client accesses the replicated data at only a subset (quorum) of servers in each operation. In such a system, some correct servers can be out-of-date after a write and thus can return values other than the most up-to-date value in response to a client's read request, thus complicating the task of determining the number of faulty servers in the system at any point in time. We initiate the study of detecting server failures in this context, and propose two statistical approaches for estimating the number of faulty servers based on responses to read requests.

1 Introduction

Data replication is a well-known means of protecting against data unavailability or corruption in the face of data server failures. When servers can suffer Byzantine (i.e., arbitrary) failures, the foremost approach for protecting data is via *state machine replication* [Sch90], in which every correct server receives and processes every request in the same order, thereby producing the same output for each request. If the client then accepts a value returned by at least $t + 1$ servers, then up to t arbitrary server failures can be masked. Numerous systems have been built to support this approach (e.g., [PG89, SESTT92, Rei94, KMM98]).

To improve the efficiency and availability of data access while still protecting the integrity of replicated data, the use of *quorum systems* has been proposed. Quorum systems are a family of protocols that allow reads and updates of replicated data to be performed at only a subset (quorum) of the servers. In a t*-masking quorum*

---

*Department of Computer Science, University of Texas, Austin, Texas; lorenzo@cs.utexas.edu. This work was funded in part by a NSF CAREER award (CCR-9734185), a DARPA/SPAWAR grant number N66001-98-8911 and a NSF CISE grant (CDA-9624082)

†AT&T Labs—Research, Florham Park, New Jersey; dalia@research.att.com

‡Department of Computer Science, University of Texas, Austin, Texas; tumlin@cs.utexas.edu

§Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey; reiter@research.bell-labs.com

system, the quorums of servers are defined such that any two quorums intersect in at least $2t+1$ servers [MR97a, MRW97]. In a system with a maximum of t faulty servers, if each read and write operation is performed at a quorum, then the quorum used in a read operation will intersect the quorum used in the last preceding write operation in at least $t+1$ correct servers. With appropriate read and write protocols, this intersection condition ensures that the client is able to identify the correct, up-to-date data [MR97a].

A difficulty of using quorum systems for Byzantine fault tolerance is that *detecting* responsive but faulty servers is hard. In state machine replication, any server response that disagrees with the response of the majority immediately exposes the failure of the disagreeing server to the client. This property is lost, however, with quorum systems: because some servers remain out of date after any given write, a contrary response from a server in a read operation does not necessarily suggest the server's failure. Therefore, we must design specific mechanisms to monitor the existence of faults in a quorum-replicated system, e.g., to detect whether the number of failures is approaching t.

In this paper, we initiate the study of Byzantine fault detection methods for quorum systems by proposing two statistical techniques for estimating the number of server failures in a service replicated using a t-masking quorum system. Both of our methods estimate the total number of faulty servers from responses to a client's read requests executed at a quorum of servers, and are most readily applicable to the *threshold* quorum construction of [MR97a], in which a quorum is defined as any set of size $\lceil \frac{n+2t+1}{2} \rceil$. The first method has the advantage of requiring essentially no change to the read and write protocols proposed in [MR97a]. The second method does require an alteration of the read and write protocols, but has the advantages of improved accuracy and specific identification of a subset of the faulty servers. Furthermore, the fault identification protocol of the second method is applicable without alteration to all types of t-masking quorum systems, and indeed to other types of Byzantine quorum systems as proposed in [MR97a].

Both methods set an *alarm line* $t_a < t$ and issue a warning whenever the number of server failures exceeds t_a. We show how the system can use information from each read operation to statistically test the hypothesis that the actual number of faults f in the system is at most t_a. As we will show, if t_a is correctly selected and read operations are frequent, both methods can be expected to issue warnings in a timely fashion, i.e., while it is still the case that $f < t$. The service can then be repaired (or at least disabled) before the integrity of the data set is compromised.

As an initial investigation into the statistical monitoring of replicated data, this paper adopts a number of simplifying assumptions. First, we perform our statistical analysis in the context of read operations that are concurrent with no write operations, as observing partially completed writes during a read substantially complicates the task of inferring server failures. Second, we assume that clients are correct; distinguishing a faulty server from a server into which a faulty client has written incorrect data raises issues that we do not consider here. Third, we restrict our attention to techniques that modify the read and write protocols only minimally or not at all and that exploit data gathered from a single read only, without aggregating data across multiple reads. (As we will show in this paper, a surprising amount of information can be obtained without such aggregation.) Each of these assumptions represents an area for possible future research.

The goal of our work is substantially different from that of various recent works that have adapted *failure detectors* [CT96] to solve consensus in distributed systems that can suffer Byzantine failures [MR97b, DS97, KMM97]. These works focus on the specification of abstract failure detectors that enable consensus to be solved. Our goal here is to develop techniques for detecting Byzantine failures specifically in the context of data replicated using quorum systems, without regard to abstract failure detector specifications or the consensus problem. Lin et al. [LRM98] analyze the process of gradual infection of a system by malicious entities. Their analysis attempts to project when failures exceed certain thresholds by extrapolating from observed failures onto the future, on the basis of certain a priori assumptions about the communication patterns of processes and the infection rate of the system. Our methods do not depend on these assumptions, as they do not address the propagation of failures in the system; rather, they attempt to measure the current number of failures at any point in time.

To summarize, the contributions of this paper are twofold: we initiate the direction of fault monitoring and detection in the context of Byzantine quorum systems; and we propose two statistical techniques for performing this detection for t-masking quorum systems under the conditions described above. The rest of this paper is organized as follows. In Section 2 we describe our system model and necessary background. In Sections 3–4 we present and analyze our two statistical methods using exact formulae for alarm line placement in relatively small systems. In Section 5 we present an asymptotic analysis for estimating appropriate alarm line placement in larger systems for both methods. We conclude in Section 6.

2 Preliminaries

2.1 System model

Our system model is based on a *universe* U of n data servers. A *correct* server is one that behaves according to its specification, whereas a *faulty* server deviates from its specification arbitrarily (Byzantine failure). We denote the maximum allowable number of server failures for the system by t, and the actual number of faulty servers in the system at a particular moment by f. Because our goal in this paper is to detect faulty servers, we stipulate that a faulty server *does* in fact deviate from its I/O specification, i.e., it returns something other than what its specification would dictate (or it returns nothing, though unresponsive servers are ignored in this paper and are not the target of our detection methods). It is hardly fruitful to attempt to detect "faulty" servers whose visible behavior is consistent with correct execution.

Our system model also includes some number of clients, which we assume to be correct. Clients communicate with servers over point-to-point channels. Channels are reliable, in the sense that a message sent between a client and a correct server is eventually received by its destination. In addition, a client can authenticate the channel to a correct server; i.e., if the client receives a message from a correct server, then that server actually sent it.

2.2 Masking quorum systems

We assume that each server holds a copy of some replicated variable Z, on which clients can execute *write* and *read* operations to change or observe its value, respectively. The protocols for writing and reading Z employ a *t-masking quorum system* [MR97a, MRW97], i.e., a set of subsets of servers $\mathcal{Q} \subseteq 2^U$ such that $\forall Q_1, Q_2 \in \mathcal{Q}$, $|Q_1 \cap Q_2| \geq 2t+1$. Intuitively, if each read and write is performed at a quorum of servers, then the use of a t-masking quorum system ensures that a read quorum Q_2 intersects the last write quorum Q_1 in at least $t+1$ correct servers, which suffices to enable the reader to determine the last written value. Specifically, we base our methods on *threshold masking quorum systems* [MR97a], defined by $\mathcal{Q} = \{Q \subseteq U : |Q| = \lceil \frac{n+2t+1}{2} \rceil\}$; i.e., the quorums are all sets of servers of size $\lceil \frac{n+2t+1}{2} \rceil$. These systems are easily seen to have the t-masking property above.

We consider the following protocols for accessing the replicated variable Z, which were shown in [MR97a] to give Z the semantics of a *safe* variable [Lam86]. Each server u maintains a timestamp T_u with its copy Z_u of the variable Z. A client writes the timestamp when it writes the variable. These protocols require that different clients choose different timestamps, and thus each client c chooses its timestamps from some set $\mathcal{T}_c$ that does not intersect $\mathcal{T}_{c'}$ for any other client c'. Client operations proceed as follows.

Write: For a client c to write the value v to Z, it queries each server in some quorum Q to obtain a set of value/timestamp pairs $A = \{ <Z_u, T_u> \}_{u \in Q}$, chooses a timestamp $T \in \mathcal{T}_c$ greater than the highest timestamp value in A and greater than any timestamp it has chosen in the past, and updates Z_u and T_u at each server u in some quorum Q' to v and T, respectively.

Read: For a client to read a variable Z, it queries each server in some quorum Q to obtain a set of value/timestamp pairs $A = \{<Z_u, T_u>\}_{u \in Q}$. From among all pairs returned by at least $t+1$ servers in Q, the client chooses the pair $<v, T>$ with the highest timestamp T, and then returns v as the result of the read operation. If there is no pair returned by at least $t+1$ servers, the result of the read operation is $\perp$ (a null value).

In a write operation, each server u updates Z_u and T_u to the received values $<v, T>$ only if T is greater than the present value of T_u; this convention guarantees the serializability of concurrent writes. As mentioned in Section 1 we consider only reads that are not concurrent with writes. In this case, the read operation will never return $\perp$ (provided that the assumed maximum number of failures t is not exceeded).

2.3 Statistical building blocks

The primary goal of this paper is to draw conclusions about the number f of faulty servers in the system, specifically whether f exceeds a selected alarm threshold t_a, where $0 \leq t_a < t$, using the responses obtained in the read protocol of the previous subsection. To do this, we make extensive use of a statistical technique called *hypothesis testing*. To use this technique, we establish two hypotheses about our universe of servers. The first of these is an *experimental hypothesis* H_E that represents a condition to be tested for, e.g., that f exceeds the alarm threshold t_a, and the second is a *null hypothesis* H_0 complementing it. The idea behind hypothesis testing is to examine experimental results (in our case, read operations) for conditions that suggest the truth of the experimental hypothesis, i.e., conditions that would be "highly unlikely" if the null hypothesis were true. We define "highly unlikely" by choosing a *rejection level* $0 < \alpha < 1$ and identifying a corresponding *region of rejection* for H_0, where the region of rejection is the maximal set of possible results that suggest the truth of H_E (and thus the falsity of H_0) and whose total probability given H_0 is at most α. For the purposes of our work, H_E will be $f > t_a$, and H_0 will always be $f = t_a$. (Note that although these hypotheses are not strictly complementary, the region of rejection for H_0 encompasses that of every hypothesis $f = t'_a$, where $0 < t'_a < t_a$; therefore the rejection level of the truly complementary hypothesis $f \leq t_a$ is bounded by that of H_0. This treatment of the null hypothesis is a standard statistical procedure.)

In this paper we will typically choose t_a to be strictly less than the maximum assumed number t of failures in the system, for the reason that it is of little use to detect a dangerous condition after the integrity of the data has been compromised. The "safest" value for t_a is 0, but a higher value may be desirable if small numbers of faults are common and countermeasures are expensive.

In order for our statistical calculations to be valid, we must be able to treat individual quorums and the intersection between any two quorums as random samples of the universe of servers. Given our focus on a quorum system consisting of all sets of size $\lceil \frac{n+2t+1}{2} \rceil$, this can be accomplished by choosing quorums in such a way that each quorum (not containing unresponsive servers) is approximately equally likely to be queried for any given operation.

As in any statistical method, there is some possibility of false positives (i.e., alarms sent when the fault level remains below t_a) and false negatives (failure to detect a dangerous fault level before the threshold is exceeded). As we will show, however, the former risk can be kept to a reasonable minimum, while the latter can be made essentially negligible.[1]

[1]Except in catastrophically unreliable systems. Neither our method nor any other of which we are aware will protect against sudden near-simultaneous Byzantine failures in a sufficiently large number (e.g., greater than t) of servers.

3 Diagnosis using justifying sets

Our first method of fault detection for threshold quorum systems uses the read and write protocols described in Section 2.2. As the random variable for our statistical analysis, we use the size of the *justifying set* for a read operation, which is the set of servers that return the value/timestamp pair $<v, T>$ chosen by the client in the read operation. The size of the justifying set is at least $2t+1$ if there are no faulty servers, but can be as small as $t+1$ if $f = t$. The justifying set may be as large as $\lceil \frac{n+2t+1}{2} \rceil$ in the case where the read quorum is the same as the quorum used in the last completed write operation.

Suppose that a read operation is performed on the system, and that the size of the justifying set for that read operation is x. We would like to discover whether this evidence supports the hypothesis that the number of faults f in the system exceeds some value t_a, where $t_a < t$. We do so using a formula for the probability distribution for justifying set sizes; this formula is derived as follows.

Suppose we have a system of n servers, with a quorum size of q. Given f faulty servers in the system, the probability of exactly j failures in the read quorum can be expressed by a *hypergeometric distribution* as follows:

$$\frac{\binom{f}{j}\binom{n-f}{q-j}}{\binom{n}{q}}$$

Given that the number of failures in the read quorum is j, the probability that there are exactly x correct servers in the intersection between the read quorum and the previous write quorum is formulated as follows: the number of ways of choosing x correct servers from the read quorum is $\binom{q-j}{x}$, and the number of possible previous write quorums that intersect the read quorum in exactly those correct servers (and some number of incorrect ones) is $\binom{n-q+j}{q-x}$. The probability that the previous write quorum intersects the read quorum in exactly this way is therefore:

$$\frac{\binom{q-j}{x}\binom{n-q+j}{q-x}}{\binom{n}{q}}$$

To get the overall probability that there are exactly x correct servers in the intersection between the read and most recent write quorums, i.e., that the justifying set size ($size$) is x, we multiply the conditional probability given j failures in the read quorum by the probability of exactly j failures in the read quorum, and sum the result for $j = 0$ to f:

$$P(size = x) = \sum_{j=0}^{f} \frac{\binom{q-j}{x}\binom{n-q+j}{q-x}\binom{f}{j}\binom{n-f}{q-j}}{\binom{n}{q}^2} \tag{1}$$

This formula expresses the probability that a particular read operation in a t-masking quorum system will have a justifying set size of x given the presence of f faults.

x	$P(size = x \mid f = 0)$	x	$P(size = x \mid f = 0)$
51	.000243	64	0.000500
52	.002922	65	7.92×10^{-05}
53	.015880	66	9.68×10^{-06}
54	.051857	67	9.03×10^{-07}
55	.114087	68	6.33×10^{-08}
56	.179687	69	3.26×10^{-09}
57	.210160	70	1.20×10^{-10}
58	.186867	71	3.05×10^{-12}
59	.128273	72	5.03×10^{-14}
60	.068649	73	5.02×10^{-16}
61	.028810	74	2.65×10^{-18}
62	.009504	75	5.89×10^{-21}
63	.002464	76	3.10×10^{-24}

Table 1: Probability distribution on justifying set sizes for Example 1

For a given rejection level α, then, the region of rejection for the null hypothesis $f = t_a$ is defined as $x \leq highreject$, where $highreject$ is the maximum value such that:

$$\sum_{x=t+1}^{highreject} \sum_{j=0}^{t_a} \frac{\binom{q-j}{x}\binom{n-q+j}{q-x}\binom{t_a}{j}\binom{n-t_a}{q-j}}{\binom{n}{q}^2} \leq \alpha$$

The left-hand expression above represents the *significance level* of the test, i.e., the probability of a false positive (false alarm).

If there are in fact $f' > t_a$ failures in the system, the probability of detecting this condition on a single read is:

$$\sum_{x=t+1}^{highreject} \sum_{j=0}^{f'} \frac{\binom{q-j}{x}\binom{n-q+j}{q-x}\binom{f'}{j}\binom{n-f'}{q-j}}{\binom{n}{q}^2}$$

If we denote this value by γ, then the probability that k consecutive reads *fail* to detect the condition is $(1-\gamma)^k$. As shown in the following examples, k need not be very large for this probability to become negligible.

Example 1: Consider a system of $n = 101$ servers, a quorum size $q = 76$, and a fault tolerance threshold $t = 25$. In order to test whether there are *any* faults in the system, we set $t_a = 0$, so that the null hypothesis H_0 is $f = 0$ and the experimental hypothesis H_E is $f > 0$. Plugging these numbers into (1) over the full range of x yields the results in Table 1. For all other values of x not shown in Table 1, the probability of a justifying set of size x given $f = 0$ is zero.

f	P(detection)	f	P(detection)
1	.046772	11	.867154
2	.093352	12	.909989
3	.160471	13	.941069
4	.246231	14	.962708
5	.345534	15	.977185
6	.451337	16	.986505
7	.556213	17	.992282
8	.653732	18	.995733
9	.739333	19	.997720
10	.810618	20	.998823

Table 2: Probability of detecting $f > 0$ in Example 1

Since $P(51)+P(52)+P(53) \approx 0.019$, while $P(51)+P(52)+P(53)+P(54) \approx 0.071$, the region of rejection for $\alpha = 0.05$ is defined as $x \leq 53$; if a read operation has a justifying set of size 53 or less, the client rejects the null hypothesis and concludes that there are faults in the system. This test has a *significance level* of 0.019; that is, there is a probability of 0.019 that the client will detect faults when there are none. (If this level of risk is unacceptable for a particular system, α can be set to a lower value, thus creating a smaller region of rejection.)

Suppose that there are actually f failures in the system. The probability that this experiment will detect the presence of failures during any given read is:

$$\sum_{x=26}^{53} \sum_{j=0}^{f} \frac{\binom{76-j}{x}\binom{25+j}{76-x}\binom{f}{j}\binom{101-f}{76-j}}{\binom{101}{76}^2}$$

Table 2 shows these values for $1 \leq f \leq 20$.

Although the probability of detecting faults during a given read in this system is relatively low for very small values of f, it would appear that this test is reasonably powerful. Even for fault levels as low as 4 or 5, a client can reasonably expect to detect the presence of failures within a very few reads; e.g., if $f = 5$, then the probability of detecting that $f > t_a$ in only 6 reads is already $1 - (1 - .345534)^6 = .921$. As the fault levels rise, the probability of such detection within a single read approaches near-certainty.

Example 2: Consider a much smaller system consisting of $n = 61$ servers, with a quorum size $q = 46$ and a fault tolerance threshold $t = 15$. Furthermore, suppose that the administrator of this system has decided that no action is called for if only a few failures occur, so that t_a is set at 5 rather than 0. Given $\alpha = 0.05$, the region of rejection for the null hypothesis $H_0 : f = t_a$ is $x \leq 27$. The probabilities of detecting this condition for actual values of f between 8 and 12 inclusive are shown in Table 3.

f	$P(\text{detection})$
8	.070210
9	.130284
10	.213058
11	.314905
12	.428527

Table 3: Probability of detecting $f > 5$ in Example 2

As one might expect, error conditions are more difficult to detect when they are more narrowly defined, as the contrast between examples 1 and 2 shows. Even in the latter experiment, however, a client can reasonably expect to detect a serious but non-fatal error condition within a small number of reads. For $f = 12$, the probability that the alarm is triggered within six read operations is $1 - (1 - 0.428527)^6$, approximately 96.5 percent. The probability that it is triggered within ten reads is over 99.6 percent. We can therefore reasonably consider this technique to be a useful diagnostic in systems where read operations are significantly more frequent than server failures, particularly if the systems are relatively large.

While the ability to detect faulty servers in threshold quorum systems is a step forward, this method leaves something to be desired. It gives little indication of the specific number of faults that have occurred and provides little information toward identifying which servers are faulty. In the next section we present another diagnostic method that addresses both these needs.

4 Diagnosis using quorum markers

The diagnostic method presented in this section has two distinct functions. First, it uses a technique similar to that of the previous section to estimate the fault distribution over the whole system, but with greater precision. Second, it pinpoints specific servers that exhibit detectably faulty behavior during a given read. The diagnostic operates on an enhanced version of the read/write protocol for masking quorum systems: the write marker protocol, described below.

4.1 The write marker protocol

The *write marker protocol* uses a simple enhancement to the read/write protocol of Section 2.2: we introduce a *write quorum marker* field to all variables. That is, for a replicated variable Z, each server u maintains, in addition to Z_u and T_u, a third value W_u, which is the name of the quorum (e.g., an n-bit vector indicating the servers in the quorum) used to complete the write operation in which Z_u and T_u were last written. The write protocol proceeds as in Section 2.2, except that in the last step, in addition to updating Z_u and T_u to v and T at each server u in a quorum Q', the client also updates W_u with (the name of) Q'. Specifically, to update Z_u,

T_u, and W_u at all (correct) servers in Q', the client sends a message containing $<v, T, Q'>$ to each $u \in Q'$. Because communication is reliable (see Section 2), the writer knows that Z_u, T_u and W_u will be updated at all correct servers in Q'. As before, each server u updates Z_u, T_u, and W_u to the received values $<v, T, Q'>$ only if T is greater than the present value of T_u.

The read protocol proceeds essentially as before, except that each server returns the triple $<Z_u, T_u, W_u>$ in response to a read request. From among all triples returned from at least $t + 1$ servers, the client chooses the triple with the highest timestamp.

Below we describe two ways of detecting faults by making use of the set of triples returned by the servers.

4.2 Statistical fault detection

Our revised statistical technique uses the quorum markers to determine the set S of servers whose returned values would match the accepted triple in the absence of faults, and the set S' of servers whose returned values actually do match that triple. Because of the size-based construction of threshold quorum systems and the random selection of the servers that make up the quorum for a given operation, the set S can be considered a random sample of the servers, of which $|S \setminus S'|$ are known to be faulty. Taking a random variable y to be the number of faulty servers in the sample, we can use similar calculations to those in Section 3 to analyze with greater precision the probability that f exceeds t_a.

As shown in Section 3, the probability of finding y faults in a sample of size s given a universe of size n containing f faults is expressed by the hypergeometric formula:

$$\frac{\binom{f}{y}\binom{n-f}{s-y}}{\binom{n}{s}}$$

For a rejection level α, the region of rejection for the hypothesis $f = t_a$ is therefore defined by the lowest value $lowreject$ such that:

$$\sum_{y=lowreject}^{s} \frac{\binom{t_a}{y}\binom{n-t_a}{s-y}}{\binom{n}{s}} \leq \alpha$$

Again, the left-hand expression represents the parameterized probability of a false alarm.

For this method, experiments in which $t_a = 0$ are a degenerate case. The presence of *any* faults in the intersection set is visible and invalidates the null hypothesis; the probability of a false positive in such cases is zero, as the formula above confirms. Likewise, as the number of faults increases, the probability of detecting faults within one or two reads rapidly approaches certainty.

f	$P(\text{detection})$	f	$P(\text{detection})$
1	.564356	11	.999951
2	.812673	12	.999982
3	.920528	13	.999993
4	.966751	14	.999997
5	.986289	15	.999999
6	.994430	16	.999999
7	.997772	17	.999999
8	.999123	18	.999999
9	.999660	19	.999999
10	.999870	20	.999999

Table 4: Probability of detecting $f > 0$ in Example 3

f	$P(\text{detection})$
8	.492173
9	.648616
10	.773168
11	.862716
12	.921818

Table 5: Probability of detecting $f > 5$ in Example 4

Example 3: Consider again the system of $n = 101$ servers, with a fault tolerance threshold of $t = 25$, a quorum size of $q = 76$, and $t_a = 0$, and suppose that a given read quorum overlaps the previous write quorum in $s = 57$ servers (the most likely overlap, with a probability of about 0.21). The probability of alarm on a single read operation for various values of $f < t$, is shown in Table 4.

A comparison of this table with Table 2 illustrates the dramatically higher precision of the write-marker method over the justifying set method. This precision has additional advantages when t_a is set to a value greater than 0.

Example 4: Consider again the system of $n = 61$ servers, with a fault tolerance threshold of $t = 15$, a quorum size of $q = 46$, and $t_a = 5$, and suppose that a given read quorum overlaps the previous write quorum in the most common intersection size $s = 34$ servers. The region of rejection for the null hypothesis $f = 5$, calculated using the formula above, is $y \geq 5$. The probability of alarm on a single read operation for various values of f, $t_a < f < t$, is shown in Table 5.

Again, the increased strength of the write-marker method is evident (see Table 3).

Like the method presented in Section 3, the write-marker technique also has the advantage of flexibility. If we wish to minimize the risk of premature alarms (i.e., alarms that are sent without the alarm threshold being exceeded) we may choose a smaller α at the risk of somewhat delayed alarms. In fact, the greater precision of this method decreases the risks associated with such a course: even delayed alarms can be expected to be timely.

4.3 Fault identification

The write marker protocol has an even stronger potential as a tool for fault detection: it allows the client to identify specific servers that are behaving incorrectly. By keeping a record of this list, the client can thereafter select quorums that do not contain these servers. This allows the system to behave somewhat more efficiently than it would otherwise, as well as gathering the information needed to isolate faulty servers for repair so that the system's integrity is maintained.

The fault identification algorithm accepts as input the triples $\{<Z_u, T_u, W_u>\}_{u\in Q}$ that the client obtained from servers in the read protocol, as well as the triple $<v, T, W>$ that the client chose as the result of the read operation. It then computes the set $S \setminus S'$ where $S = Q \cap W$ and S' is the set of servers that returned $<v, T, W>$ in the read operation. The servers in $S \setminus S'$ are identified as faulty.

Note that the fault identification protocol does not depend in any way on the specific characteristics of threshold quorum systems, and is easily seen to be applicable to masking quorum systems in general.

5 Choosing alarm lines for large systems

The analysis of the previous two sections is precise but computationally cumbersome for very large systems. A useful alternative is to estimate the performance of possible alarm lines by means of bound analysis. In this section we present an asymptotic analysis of the techniques of Sections 3 and 4 that shows how to choose an alarm line value for arbitrarily large systems.

Let us denote the read quorum Q, the write quorum Q', the set of faulty servers by B, and the hypothesized size of B (i.e., the alarm line) by t_a. We define a random variable $X = |(Q \cap Q') \setminus B|$, which is the justifying set size. We can compute the expectation of X directly. For each server $u \notin B$ define an indicator random variable I_u such that $I_u = 1$ if $u \in (Q \cap Q') \setminus B$ and $I_u = 0$ otherwise. For such u we have $P(I_u = 1) = q^2/n^2$ since Q and Q' are chosen independently. By linearity of expectation,

$$E[X] = \sum_{u\in U\setminus B} E[I_u] = \sum_{u\in U\setminus B} P(I_u = 1) = (n - t_a)\frac{q^2}{n^2}.$$

Intuitively, the distribution on X is centered around its expectation and decreases exponentially as X moves farther away from that expectation. Thus, we should be able to show that X grows smaller than its expectation with exponentially decreasing probability. A tempting approach to analyzing this would be to use Chernoff bounds, but these do not directly apply because the selection of individual servers in Q (similarly, Q') is not independent. In the analysis below, we thus use a more powerful tool, *martingales*, to derive the anticipated Chernoff-like bound.

We bound the probability $P(X < k)$ using the method of bounded differences, by defining a suitable Doob martingale sequence and applying Azuma's inequality (see [MR95, Ch. 4.4] for a good exposition of this technique; Appendix A provides a brief introduction). Here, a Doob martingale sequence of conditional random variables is defined by setting X_i, $0 \leq i \leq q$, to be the expected value of X after i selections are made in each of Q and Q'. Then, $X = X_q$ and $E[X] = X_0$, and it is not difficult to see that $|X_i - X_{i-1}| \leq 2$ for all $1 \leq i \leq q$. This yields the following bound (see Appendix A).

$$P(X < E[X] - \delta) \leq 2e^{-\frac{\delta^2}{8q}}$$

We use this formula and our desired rejection level α to determine a δ such that $P(X < E[X] - \delta) \leq \alpha$. This probability value is our probability of a false alarm and can be diminished by decreasing α and recalculating δ. The value $E[X] - \delta$ defines our region of rejection (see Section 2.3).

In order to analyze the probability that our alarm is triggered when the number of faults in the system is $t' > t_a$, we define a second random variable X' identical to X except for the revised failure hypothesis. This gives us:

$$E[X'] = (n - t')\frac{q^2}{n^2} < (n - t_a)\frac{q^2}{n^2} = E[X]$$

An analysis similar to the above provides the following bound:

$$P(X' > E[X'] + \delta') \leq 2e^{-\frac{\delta'^2}{8q}}$$

To summarize, these bounds can now be used as follows. For any given alarm line t_a, and any desired confidence level α, we can compute the minimum δ to satisfy $2e^{-\frac{\delta^2}{8q}} \leq \alpha$. We thus derive the following test: An alarm is triggered whenever the justifying set size is less than $(n-t_a)\frac{q^2}{n^2} - \delta$. The analysis above guarantees that this alarm will be triggered with false positive probability at most our computed bound $2e^{-\frac{\delta^2}{8q}} \leq \alpha$. If, in fact, f faults occur and f is sufficiently larger than t_a, then there exists $\delta' > 0$ such that $E[X'] + \delta' = E[X] - \delta$. Then, by the analysis above, the probability of triggering the alarm is greater than $1 - 2e^{-\frac{\delta'^2}{8q}}$.

In the case of the write marker protocol, we can tighten the analysis by using the (known) intersection size between Q and Q' as follows. Define $S = Q \cap Q'$, $s = |S|$, and a random variable $Y = |S \setminus B|$. Y has a hypergeometric distribution on s, $n - t_a$, and n, and $E[Y] = s(n - t_a)/n$. The appropriate Doob martingale sequence in this case defines Y_i, $0 \leq i \leq s$, to be the expected value of Y after i selections are made in S. Then, $|Y_i - Y_{i-1}| \leq 1$, and so to set the region of rejection we can use

$$P(Y < E[Y] - \delta) \leq 2e^{-\frac{\delta^2}{2s}}.$$

6 Conclusion

In this paper, we have presented two methods for probabilistic fault diagnosis for services replicated using t-masking quorum systems. Our methods mine server responses to read operations for evidence of server failures, and if necessary trigger an alarm to initiate appropriate recovery actions. Both of our methods were demonstrated in the context of the threshold construction of [MR97a], i.e., in which the quorums are all sets of size $\lceil \frac{n+2t+1}{2} \rceil$, but our techniques of Section 4 can be generalized to other masking quorum systems, as well. Our first method has the advantage of requiring no modifications to the read and write protocols proposed in [MR97a]. The second method requires minor modifications to these protocols, but also offers better diagnosis capabilities and a precise identification of faulty servers. Our methods are very effective in detecting faulty servers, since faulty servers risk detection in every read operation to which they return incorrect answers.

Future work will focus on generalizations of these techniques, as well as uses of these techniques in a larger systems context. In particular, we are presently exploring approaches to react to server failures once they are detected.

References

[CT96] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[DS97] A. Doudou and A. Schiper. Muteness detectors for consensus with Byzantine processes. Technical Report TR97-230, Department of Computer Science, École Polytechnic Fédérale de Lausanne, October 1997.

[KMM97] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. Solving consensus in a Byzantine environment using an unreliable failure detector. In *Proceedings of the International Conference on Principles of Distributed Systems*, pages 61–75, December 1997.

[KMM98] K. P. Kihlstrom, L. E. Moser and P. M. Melliar-Smith. The SecureRing protocols for securing group communication. In *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, pages 317–326, January 1998.

[Lam86] L. Lamport. On interprocess communication (part II: algorithms). *Distributed Computing* 1:86–101, 1986.

[LRM98] M. J. Lin, A. Ricciardi and K. Marzullo. On the resilience of multicasting strategies in a failure-propagating environment. UT Austin TR-1998-003.

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.

[MR97a] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing* 11(4):203–213, 1998.

[MR97b] D. Malkhi and M. Reiter. Unreliable intrusion detection in distributed computation. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 116–124, June 1997.

[MRW97] D. Malkhi, M. Reiter, and A. Wool. The load and availability of Byzantine quorum systems. In *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing*, pages 249–257, August 1997.

[Rei94] M. K. Reiter. Secure agreement protocols: Reliable and atomic group multicast in Rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 68–80, November 1994.

[PG89] F. M. Pittelli and H. Garcia-Molina. Reliable scheduling in a TMR database system. *ACM Transactions on Computer Systems*, 7(1):25–60, February 1989.

[Sch90] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.

[SESTT92] S. K. Shrivastava, P. D. Ezhilchelvan, N. A. Speirs, S. Tao, and A. Tully. Principal features of the VOLTAN family of reliable node architectures for distributed systems. *IEEE Transactions on Computers*, 41(5):542–549, May 1992.

A Martingales

In this appendix, we provide a brief introduction to martingales, which summarizes only the necessary definitions and results from the more thorough treatment found in [MR95, Ch. 4.4].

Definition 1 *A* martingale *sequence is a sequence of random variables $X_0, X_1, \ldots$ such that for all $i > 0$,*

$$E[X_i \mid X_0, \ldots, X_{i-1}] = X_{i-1}$$

Our goal in constructing martingale sequences in Section 5 is to apply the following theorem.

Theorem 1 *Let $X_0, X_1, \ldots$ be a martingale sequence such that for each k,*

$$|X_k - X_{k-1}| \leq c$$

where c is independent of k. Then, for $t \geq 0$ and $\delta > 0$,

$$P(|X_t - X_0| \geq \delta) \leq 2e^{-\frac{\delta^2}{2tc^2}}$$

The particular method that we use for constructing martingale sequences employs the notion of a *filter* over a finite sample space Ω, which is a nested sequence of event-sets $F_0 \subseteq F_1 \subseteq \cdots \subseteq F_k$ where $F_0 = \{\emptyset, \Omega\}$, $F_k = 2^\Omega$, and for $0 \leq i \leq k$, F_i is closed under complement and union. Intuitively, each F_i can be thought of as being generated by a partition of Ω into disjoint events, where F_{i+1} is generated by a more refined partition than F_i. In Section 5, each block (event) of the partition generating F_i is defined by the first i choices of servers in each of two quorums. We then apply the following theorem to construct a Doob martingale:

Theorem 2 *Let $F_0, \ldots, F_k$ be a filter, let X be any random variable, and define $X_i = E[X \mid F_i]$, i.e., X_i is the expected value of X conditioned on the events in F_i. Then $X_0, \ldots, X_k$ is a martingale.*