The virtue of dependent failures in multi-site systems

Flavio P. Junqueira
flavio@cs.ucsd.edu

Keith Marzullo
marzullo@cs.ucsd.edu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA - USA

Abstract

A multi-site system consists of a collection of locally-managed networks called sites, each containing a collection of processors. In this paper, we present the multi-site threshold model*: a new failure model for multi-site systems that accounts for failures of sites. Using this model, we then derive quorum constructions that have two interesting features. First, they generate quorum systems that are more available than majority quorums. Second, our constructions reduce quorum sizes compared to the majority construction, thus reducing load on the processors and increasing the total capacity of the system. To show that our model is realistic, we discuss the implementability of this new model, motivating with data from a real system.*

1 Introduction

Determining the failure model is an integral part of the design of any fault-tolerant distributed system. Understanding how components fail is important not only for the design of algorithms and mechanisms, but also to enable proper evaluation of a system. In this paper, we argue that assuming a single threshold on the number of faulty components along with independence of failures is not adequate as a failure model for an important class of distributed systems. The class of systems we discuss is exemplified by grids. Grids, like BIRN [4], Geon [7], and TeraGrid [15], are *multi-site systems*: they consist of a collection of locally-managed networks (sites), each containing a collection of processors. There are other multi-site systems that are not normally considered grids, such as PlanetLab [13].

In such systems, failures are more often the common case than the exception. One kind of failure familiar to users are *partitions*, in which two sites become inaccessible to each other. If partitions can occur, then replicating state using quorum consensus (either directly, or as part of an agreement protocol like Paxos [11]) seems like a bad idea; one cannot construct a coterie in which it is ensured that there is always a quorum that can be constructed, where a coterie is a set of minimal quorums that intersect each other [6]. Because of this, perhaps it is not surprising that grids often use replication techniques with consistency properties weaker than one-copy serializability or simply no replication at all.

In fact, there has been little work done in determining failure models for multi-site systems. From a protocol point of view, an ideal failure model strikes a balance between completeness and simplicity. If it is too simple, then it does not characterize failure patterns with enough accuracy, but if it is too complete, it becomes complex to be useful in developing flexible protocols. For example, quorums formed from a majority of the processors is based on a simple failure model, and such quorums have optimal availability when processors fail independently and with the same failure probability $p < 0.5$ [2, 3]. The availability of majority systems, however, can be seriously impacted by correlation of failures [1, 16]. A partition is an example of a correlated failure: when two sites fail, then the communication between the two sites fails while the communication within the sites does not. Another example is a catastrophic site failure, such as an A/C failure, that potentially causes all the processors in the site to crash. A quorum construction to overcome the traditional coterie construction based on majorities was proposed in [17]. This approach, however, guarantees consistency probabilistically and assumes a uniform distribution of servers across the Internet.

In this paper, we consider a dependent failure model, as opposed to an independent and identically distributed one, that is simple but which we believe better captures the failure patterns that multi-site systems exhibit. We call it the *multi-site threshold model*. First, we give a characterization of the failure model (Section 2) that is based on a set of threshold values. We then describe methods for constructing quorums (Section 3) for this model that uses a compact

representation of quorums, minimizes the number of participating sites in any quorum, and guarantees consistency deterministically. We then compare the availability of the quorums in this construction with quorums constructed of majorities. To some extent, this is not a fair comparison because failures are not independent by assumption, but it gives us an idea of how much better we can do by taking correlation of failures into account. Finally, based on reported availability figures for BIRN and our own experiences, we argue that the failure model is in fact plausible (Section 4).

2 The multi-site threshold model

A multi-site system is a collection of processors $\mathcal{P}$ and a collection $\mathcal{S}$ of sites such that $\mathcal{S}$ is a partition of $\mathcal{P}$, where processors within the same site are connected by a local area network and processors in different sites are connected by a wide area network, such as the Internet or the Internet2. The processors represent the computing resources available in the system, whereas the sites correspond to the organization of these processors. In a multi-site system deployed on the Internet, a site represents an institution that hosts processors that are part of the system.

We differentiate two types of failures in a multi-site system: site failures and processor failures. Site failures correspond to all the processors of a particular site becoming unavailable. A processor fails when it stops providing service, *i.e.*, we assume a crash model for processor failures.

Typically, fault-tolerant systems are designed using the *threshold model*: out of n components at most t can fail. This model is appropriate when failures are independent and identically distributed. As a principle learned from the threshold model, for every part of a system in which the replicated components fail independently, using a threshold is an appropriate way of modeling failures. We therefore propose a model with two parameters: f_s and F_p. The parameter f_s corresponds to a threshold on the number of sites that can be simultaneously unavailable, and F_p is a vector of thresholds, one for each site. We use $F_p[i]$ to denote the threshold on the number of processor failures in site S_i when S_i is available.

This failure model is dependent because it enables a separation of failures that cause the simultaneous unavailability of multiple processors from the ones that cause processors to fail independently. The use of thresholds as opposed to a more expressive model, as the one in [9], is due both to the similar availability characteristics of sites and to the homogeneity of processors in a single site. If there are either sites or processors that are unreliable compared to others, then these unreliable components can be simply ignored. We discuss further the implementability of the model in Section 4.

Compared to the threshold model, our model enables more flexibility in expressing the sets of processors that can fail. Because sites can have different numbers of processors and different sites can have different thresholds, the minimum number of available processors at a given time is actually dependent upon the sites available, the number of processors in these sites, and their threshold on the number of faulty processors. In fact, such minimum sets are *survivor sets* in the model of [9].

In the following section, we describe quorum constructions for the multi-site threshold model.

3 Choosing quorums

Recall that a coterie $\mathcal{Q}$ is a collection of quorums such that every quorum is a subset of $\mathcal{P}$, no quorum is strictly contained in another, and every pair of quorums intersect. We say that a coterie is available when there is at least one quorum in this coterie that contains only available processors.

From the multi-site threshold model, it is clear that we have to use quorum systems that span multiple sites if $f_s \geq 1$. However, this principle is not sufficient for achieving high availability, as processors in an available site can fail. We now describe two constructions that improve over *Majority*, where Majority denotes the set of quorums composed of majorities of processors.

Consider the following construction. Let f_s and F_p be values such as in the model described in Section 2. To determine quorums, we first pick $2f_s+1$ sites out of $\mathcal{S}$. Then, for each S_i, we choose a quorum system such that the quorums are composed only of processors of S_i using the threshold $F_p[i]$. We select a subset S' of S_i of size $2F_p[i]+1$, and we choose the quorums for S_i to be all the subsets of S' of size $F_p[i]+1$. A quorum in $\mathcal{Q}$ is then composed of quorums from f_s+1 sites.

To illustrate the use of this construction, suppose that $|\mathcal{S}| = 3$, $f_s = 1$, $|S_i| = 3$, and $F_p[i] = 1$, $i \in \{1,2,3\}$. First, we select the sites to use. Since $2f_s+1 = 3$, we use all the sites. A quorum for a site S_i is composed of two processors of S_i. From the construction, a quorum in $\mathcal{Q}$ is hence composed of four processors, two from a site S_i and two from a site S_j, $i \neq j$.

We call this construction *Qsite*. Qsite shares many of the features of the hierarchical constructions in [10] and [5]. These constructions, however, do not consider placement of replicas. In fact, the construction in [10] assumes independent and identical probability of failure for processes.

	t = 1		t = 2	
f_s	**Majority**	**Qsite**	**Majority**	**Qsite**
1	5	4	8	6
2	8	6	13	9
3	11	8	18	12
4	14	10	23	15

Table 1: Quorum sizes.

The construction in [5] goes a step further and assumes that replicas are spread across many sites, but it assumes that the set of replicas is given, and hence it does not propose a strategy for replica placement.

We now compare Qsite with Majority, using quorum sizes as a metric. For this comparison, we assume that all the sites have the same threshold t on the number of faulty processors, and both constructions use the same processors. Table 1 shows quorum sizes for different values of f_s and t. The main observation is that Qsite requires fewer processors in all the cases, and the difference between the two constructions increases with the value of f_s. Using fewer processors in each quorum reduces the load handled by any particular processor. Assuming that quorums are uniformly chosen by clients, having smaller quorums implies that processors have to handle fewer requests. Load is inversely proportional to the capacity, and by reducing load we are actually increasing the total capacity of the system, where the capacity is the number of requests the system can handle per unit of time [12].

For both constructions, Majority and Qsite, the quorum system is available as long as there are f_s+1 sites available. However, because Majority uses larger quorums, it tolerates fewer processor failures. For example, suppose that $f_s = t = 1$. From the table, we have that Majority uses quorums of size 5. This implies that Majority not only requires that two sites are available, but also that at least one of the sites contains no faulty processors. A coterie generated by Qsite does not have this same constraint, and it is available as long as there are two sites available, each site containing some majority of processors available.

Of course, the number of processors per quorum increases with the value of f_s and the values in $F_p[i]$ for both constructions. We can alleviate this problem by sacrificing availability, but still obtaining better availability and load compared to Majority.

Suppose that $f_s = 2$ and $F_p[i] = 1$ for every $S_i \in \mathcal{S}$. We construct $\mathcal{Q}$ using four sites instead of five as in the previous construction. For this, we assume that $\mathcal{S}$ contains at least four sites, and each site contains at least three processors. As for Qsite, we first select quorums for each site. Suppose that we are to use sites S_a, S_b, S_c, and S_d. For each site S_i, $i \in \{a, b, c, d\}$, we select three processors. Now let Q_i be the set of quorums for site S_i such that each quorum contains two out of the three processors selected from S_i. We then have the following quorum system:

$$\begin{aligned}\mathcal{Q} = \; & \{q_a q_b q_c : q_a \in Q_a \wedge q_b \in Q_b \wedge q_c \in Q_c\} \cup \\ & \cup \{q_a q_d : q_a \in Q_a \wedge q_d \in Q_d\} \cup \\ & \cup \{q_b q_d : q_b \in Q_b \wedge q_d \in Q_d\} \cup \\ & \cup \{q_c q_d : q_c \in Q_c \wedge q_d \in Q_d\}\end{aligned}$$

where $q_i q_j$ denotes the union of q_i and q_j.

It is easy to observe that any pair of quorums in $\mathcal{Q}$ intersect, and that no quorum is strictly contained in another. According to this construction, if there is a single site unavailable, then $\mathcal{Q}$ is available. As for Majority, if a single site is unavailable, then the coterie is available only if at least one of the available sites has all processors available. If there are two sites S_i and S_j unavailable, then $\mathcal{Q}$ is available as long as $i, j \in \{a, b, c\}$. Using Majority, however, requires seven processors and the set of quorums is unavailable whenever there are two sites unavailable.

Similar constructions can be obtained for other values of f_s.

4 Failures in multi-site systems

We have shown that the multi-site threshold model has useful properties. In this section, we argue why we believe it is also realistic.

4.1 Site failures

To understand how sites fail in a multi-site system, we studied the failure data of a particular system, the BIRN Grid [4]. We obtained monthly availability data for 15 BIRN sites from January 2004 through August 2004.[1] The monthly availability of a site is given by:

$$\text{Availability} = \frac{\text{Total hours} - \text{Unplanned outages}}{\text{Total hours}} \times 100$$

where "Total hours" is the total number of hours in a month deducted the scheduled down time, and "Unplanned outages" is the total number of hours that the site was not available not considering scheduled down time.

According to this availability data, a site becoming unavailable is a surprisingly common event. On average, each site did not have 100% availability on 5 out of the 8 months, which implies that in a given month several sites have unplanned outages and become unavailable as a consequence.

[1] This data is consistently collected by the BIRN staff, and made available through their web page. To determine availability of a site, they use pings and notifications from the SRB (Storage Resource Broker) service.

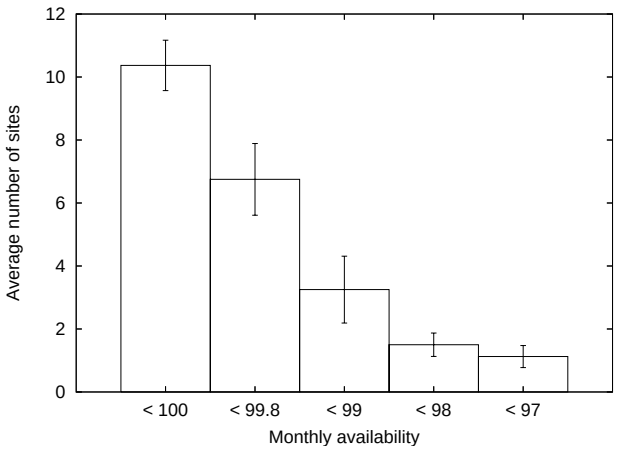

Figure 1: Number of sites with availability under α, for various values of α. The error bars correspond to the standard error for each point.

Figure 1 summarizes the availability of sites. For each month, we counted the number of sites that had availability below some value α, for different values of α. We then computed the average across the eight months for each value. This average is what we plot in Figure 1. From the figure, on average over 10 sites do not have 100% availability in a month.

According to our definition of availability, if the availability of a site is 99%, then it was down for approximately 7 hours in that month, and hence every 1% of unavailability corresponds to 7 hours of unplanned down time. From Figure 1, there is on average at least one site with availability less than 97%, which implies that such a site is unavailable for over 21 hours. In fact, we observed availability values as low as 79%.

Because we aim at constructing coteries that leverage the existence of multiple sites, we computed the average worst-case down time. Table 2 shows the worst-case unavailability for different numbers of sites averaged across the eight months. In more detail, for each month we determine the x sites, $x \in \{1, 2, 3, 4\}$, with lowest availability, and then compute how many minutes during that month we expect the x sites to be simultaneously unavailable. To compute such numbers, we assume that the events causing the sites to be unavailable happen independently. For each value of x, we then average the number of minutes obtained across the eight months. We observe that the average worst case varies from over 55 hours for a single site to a fraction of a minute for four sites.

Back to the model of Section 2, one can determine the value of f_s by looking at the values of Table 2 and choosing the one that corresponds to requirements of the application. For example, if an application sets f_s to zero and uses a single site, then it will experience, considering the worst case, 55 hours of unavailability in a month on average.

Number of sites	Unavailability in minutes
1	3288 (979)
2	87 (33)
3	1.9 (1.0)
4	0.017 (0.009)

Table 2: Average worst-case unavailability across eight months with standard error in parentheses.

In trying to determine the causes of low monthly availabilities in multi-site systems, we identified a few causes for a site to be unavailable, observed in BIRN sites, in TeraGrid sites, and in a local computer cluster. These causes are:

1. Software incompatibility/misconfiguration;
2. Power failures;
3. Failure of shared resources (e.g. storage);
4. Broken pipes causing floodings;
5. Local campus network problems;
6. Loss of air-conditioning.

Note that the order in this list is arbitrary. We are currently attempting to further quantify these failures.

4.2 Processor failures

Even when sites remain available, individual processors can become unavailable due to, for example, hardware faults. Within a site, it is often the case that most or all the processors have the same hardware and software platforms because of the difficulty in managing a heterogeneous environment. We hence assume that the reliability of processors within a site is uniform and independent. Of course, this assumption may be violated by viruses and worms, but their effects are outside the scope of this work. Replication techniques for coping with Internet pathogens are discussed in [8].

In multi-site systems, it is uncommon to have several unavailable nodes at a time in a single site, although it is fairly common to have several sites with faulty processors simultaneously.[2] This happens because the failure of any processor triggers a repair process on the site, and the probability of a processor being repaired is usually higher than the probability of a new failure for a production system.

[2]Note that failures affecting most or all of the nodes of a site are modeled with site failures.

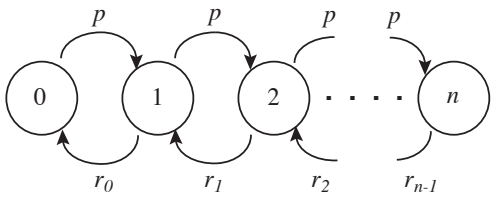

Figure 2: Model for a single site with n processors.

We can then model failures in sites using a *Markov chain* [14]. Instead of modeling the whole system, we model sites individually. By the characteristics of the systems we are assuming, sites operate independently, and the operation of the processors at a site has little or no influence on the operation of processors at another site. As a consequence, sites change states concurrently. In addition, modeling the whole system as opposed to modeling sites individually causes the number of states to increase significantly with the number of sites.

To model failures in a site, we assume that processors fail independently. Thus, states correspond to the number of faulty processors in that site, and the probability of undergoing a transition from a state with f faulty processors to a state with $f + 1$ processors is p. On the other hand, repair transitions (the ones from $f + 1$ to f) may have probabilities that change with the value of f. For example, resources to repair processors can be progressively allocated as more processors fail. As a result, the repair probability remains constant or even increases with the value of f. If there is a fixed amount of resources allocated to repair processors, and these resources are used independently of the number of failures, then the repair probability decreases with the value of f because these resources have to be shared among the faulty processors.

Figure 2 depicts the chain we just described. Assuming that no transition probability is zero, we have that this chain is irreducible and ergodic. According to the model, processors fail independently, but the probability of repair (undergoing a transition from state $f + 1$ to f) may change with the value of f. In our model, we use r_f to denote the probability that the site undergoes a transition from state $f + 1$ to state f.

Repairs in different sites happen independently, and therefore the probability of a repair transition does not increase with failures in different sites. That is, if a processor fails in site S_i and another in site S_j, $i \neq j$, they do not mutually affect their repair probabilities. This probability is affected, according to the previous description, if $S_i = S_j$.

Using this model, we can easily compute a threshold on the number of failures for each site. First, we need to determine a target degree of reliability $\mathcal{R}$, which is the probability that the number of simultaneous processor failures in any site is higher than expected. Because our model is an irreducible ergodic Markov chain, we can compute the limiting probabilities of all states [14]. That is, the probability of being at a state j after a long time has elapsed, independent of the initial state i ($\lim_{n\to\infty} P_{ij}^n$). Using these limiting probabilities, we can determine a threshold for each site: the threshold for a site S_i is the number of failures associated to the first state that has a limiting probability smaller than $\mathcal{R}$.

To illustrate the process of obtaining thresholds for the sites of a system, we give two examples. Let $\mathcal{S}$ be a collection of sites such that each site has three processes. Suppose that in both examples, the probabilities of failure and repair are the same across all the sites. In the first example, these probabilities are as follows: $p = 0.01$, $r_0 = 0.3$, $r_1 = 0.4$ and $r_2 = 0.5$. Computing the limiting probabilities, we have the following:

$$\begin{aligned}
\lim_{n\to\infty} P_{i0}^n &= 0.96695 \\
\lim_{n\to\infty} P_{i1}^n &= 0.03223 \\
\lim_{n\to\infty} P_{i2}^n &= 0.00080 \\
\lim_{n\to\infty} P_{i3}^n &= 0.00002
\end{aligned}$$

If $\mathcal{R}$ is 0.001, for instance, we have that the threshold is one for every site. For the second example, we maintain the value of a failure probability, but we reverse the order of the repair probabilities. We then have that the probabilities for this example are: $p = 0.01$, $r_0 = 0.5$, $r_1 = 0.4$, $r_2 = 0.3$. Computing the limiting probabilities, we have that:

$$\begin{aligned}
\lim_{n\to\infty} P_{i0}^n &= 0.97989 \\
\lim_{n\to\infty} P_{i1}^n &= 0.01960 \\
\lim_{n\to\infty} P_{i2}^n &= 0.00049 \\
\lim_{n\to\infty} P_{i3}^n &= 0.00002
\end{aligned}$$

Comparing these two examples, we observe that the limiting probabilities for one or more failures are smaller in the second example than in the first, and this is because repair probabilities are higher in the second for fewer failures. Thus, repair probabilities are ideally high for fewer processor failures, although they may degrade as the number of processor failures increase. From this observation, it seems that allocating as much resources as possible for a first processor failure is a better strategy than the one of allocating resources progressively.

5 Future work

Although we have discussed quorum constructions that improve over the traditional majority construction, we have not shown that these constructions are optimal. Thus, one of our goals is to determine optimal constructions for settings in which failures are not independent. We conjecture, however, that different classes of systems require different models, and as a consequence have different optimal constructions. As an illustration, the model proposed here matches well the characteristics of homogeneous multi-site systems, but certainly not the ones of heterogeneous systems. In particular, if processors of a site are not equally reliable, then it may be more appropriate to use a more expressive model that directly states minimal subsets of faulty processors instead of using a threshold. The solution for this particular instance happens to be trivial, as we can easily adapt the multi-site threshold model to use the more expressive model of [9] for processor failures and perhaps for site failures if sites are not equally reliable.

The failure characterization that we presented in Section 4 is based on information from one system, and we need to validate with information from other systems. One of the possibilities is that the properties observed are particular to the BIRN style of operation. We have, however, observed, but not quantified, similar properties in PlanetLab.[3] Regarding processor failures, we have to determine suitable values for the failures probabilities for the Markov model. This task requires more detailed information than we obtained to date.

Finally, our preliminary results show that multi-site systems can benefit from models that allow for more expressive descriptions of failures such as the one we proposed in this paper. It remains to study alternative models to determine whether there are other models that are more expressive than the traditional threshold model and that provide a more accurate representation. Further investigating the benefit to algorithms for problems other than quorum consensus is also one of our goals.

Acknowledgements

We would like to express our gratitude to Karan Bhatia, Jeffrey Grethe, and Mark James for their assistance with the BIRN system. We also would like to thank Geoff Voelker for helpful discussions and suggestions, and the anonymous reviewers for their helpful comments. Support for this work was provided by AFOSR MURI Contract F49620-02-1-0233.

[3] PlanetLab does show other failure behaviors, such as highly correlated failures due to high demands that occur near submission deadlines of certain conferences.

References

[1] Y. Amir and A. Wool. Evaluating quorum systems over the Internet. In *Proceedings of the 26th IEEE FTCS*, pages 26–37, Sendai, Japan, June 1996.

[2] Y. Amir and A. Wool. Optimal availability quorum systems: Theory and practice. *Information Processing Letters*, 65(5):223–228, Mar. 1998.

[3] D. Barbara and H. Garcia-Molina. The reliability of voting mechanisms. *ACM Transactions on Computers*, 36(10):1197–1208, Oct. 1987.

[4] The Biomedical Informatics Research Network (BIRN). http://www.nbirn.net.

[5] J.-M. Busca, M. Bertier, F. Belkouch, P. Sens, and L. Arantes. A performance evaluation of a quorum-based state-machine replication algorithm for computing grids. In *Proceedings of the 16th IEEE SBAC-PAD'04*, Foz do Iguaçú, PR, Brazil, Oct. 2004.

[6] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, Oct. 1985.

[7] The Geosciences Network. http://www.geongrid.org/.

[8] F. Junqueira, R. Bhagwan, A. Hevia, K. Marzullo, and G. M. Voelker. Surviving Internet catastrophes. In *Proceedings of USENIX Annual Tech. Conference, General Track*, pages 45–60, Anaheim, CA, Apr. 2005.

[9] F. Junqueira and K. Marzullo. Synchronous Consensus for dependent process failures. In *Proceedings of the 23rd IEEE ICDCS*, pages 274–283, Providence, RI, May 2003.

[10] A. Kumar. Hierarchical Quorum Consensus: A new algorithm for managing replicated data. *IEEE Transactions on Computers*, 40(9):996–1004, Sept. 1991.

[11] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, May 1998.

[12] M. Naor and A. Wool. The load, capacity, and availability of quorum systems. *SIAM Journal on Computing*, 27(2):423–447, Apr. 1998.

[13] The Planetlab testbed. http://www.planet-lab.org/.

[14] S. Ross. *Introduction to probability models*. Harcourt Academic Press, 2000.

[15] The TeraGrid project. http://www.teragrid.org/.

[16] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Beyond availability: Towards a deeper understanding of machine failure characteristics in large distributed systems. In *Proceedings of the 1st USENIX WORLDS*, San Francisco, CA, Dec. 2004.

[17] H. Yu. Signed quorum systems. In *Proceedings of the 23rd ACM PODC*, pages 246–255, St. Johns, Newfoundland, Canada, July 2004.