Voting Class – an Approach to Achieving High Availability for Replicated Data

Jim Tang
Department of Computer Science
Memorial University of Newfoundland
St. John’s, Nfld, A1C 5S7, Canada

Abstract
In a distributed system, data are often replicated to increase the availability in the face of node and communication failures. However, updates to replicated data must be properly controlled to avoid data inconsistency. This can adversely affect the availability. In this paper, we propose an approach to the design of replica control schemes which can provide higher availability than currently existing voting schemes. The approach is based on the observation that the existing voting schemes actually belong to a general class, called voting class. This class of voting schemes can be represented in a very simple and uniform way. Thus a designer can choose the optimal scheme within the voting class by evaluating each of them and choose the one which maximizes the availability.

1 Introduction
In a distributed system, data are often replicated to increase the availability in the face of node and communication failures. If a data item is replicated at more than one sites, then it may still be available even if some sites crash. However, replication may cause data inconsistency due to uncontrolled updates to the copies of a data item [6]. For example, suppose an account is replicated at two branches of a bank and the initial balance of this account is $100. If these two branches cannot communicate with each other due to failures (called partition failures), then allowing simultaneous withdrawal of $100 at each branch would result in an overdraw.

Several replica control schemes have been proposed to maintain data consistency in the presence of failures [1,2,4,5,8-12,15,16]. A pessimistic scheme maintains data consistency by prohibiting some updates from being processed when a system is partitioned [1,2,4,8-12,15,16]. In contrast, an optimistic scheme never rejects updates. Any inconsistency that may consequently arise is detected and resolved when failures are repaired [5,6]. An optimistic scheme does not limit availability in the presence of failures. However, resolving data inconsistency may involve complicated resolution procedures.

1.1 Voting schemes and their limitations
The best-known pessimistic scheme is voting [7,17]. In its simplest form, a voting scheme assigns to each copy of a data item a number of votes. A group is allowed to perform updates only if it possesses a majority of the votes of the entire system. (Such a group is called a majority group.) Aside from its simplicity, this kind of voting schemes has an additional merit that if the votes associated with any copy reflect the rate at which updates are submitted for that copy, then the majority updates submitted to the system are ensured to be submitted to the majority group. However, these voting schemes are very vulnerable to successive partitions. For example, suppose a network partitions into two groups, with one containing a set of sites that is just enough for a majority. Then, any subsequent partition of this group results in the data being unavailable for update operations.

One reason that a voting scheme is vulnerable to successive failures is that it is static in the sense that the majority is always defined with respect to the entire system. Some attempts have been made to make the definition somewhat dynamic. For example, in [8], mechanisms were introduced to adjust dynamically the eligibility for a group to process updates. Nevertheless the mechanisms still disallow a group to process updates if its size falls below the majority of the entire system.

Recently, a new kind of voting schemes have been proposed to prevent loss of majority group, as far as possible, in the presence of successive partitions [4,9,15]. (These schemes have been called dynamic voting schemes while the previous ones static voting schemes.) The basic idea can be generalized as follows. At any time, there is a unique nodal set, called base set. The base sets, which may vary from time to time, are used to derive “majority groups”, according to some derivation criteria. (Note to distinguish between arbitrary set
of nodes and connected set of nodes, we use the term 'group' to mean the latter. Also note the term 'majority' may no longer bear its traditional meaning. We prefer, however, not to change the terminology for simplicity.) No other set can possibly derive majority groups. To overcome vulnerability to successive failures, the definition of such a time-dependent base set should increase the chance that after any successive failures, some groups are majority groups in the sense that they are derivable from the base set. This is in contrast with any traditional voting schemes where the base set is set permanently to the entire nodal set at design time. To coordinate data consistency and base set variability, dynamic voting schemes delicately define a base set at any time instance to be the nodal set of the most recent majority group. In the schemes proposed in [4,9], a group is a majority group if and only if it contains a majority subset of the base set. (Actually, the protocol of majority determination is invoked when an update attempts to commit.) Another scheme that is similar to dynamic voting schemes in spirit was proposed in [2]. The authors introduced techniques for dynamically reassigning votes to sites in the majority group upon failures such that if this group partitions subsequently, one of the newly formed groups will have a majority of votes. It has been proven that, under a special stochastic model (see section 5), dynamic voting schemes are better than static voting schemes in terms of availability [9].

The increased availability provided by the above dynamic voting schemes relies on high update frequency of data items. However, high update frequency of data items may not always be ensured in a large database system. To improve this weakness, a different kind of dynamic voting schemes is introduced in [15]. While the definition of a base set remains the same as before, the way a majority group is derived from a base set in this scheme is quite different. Simply speaking, in this scheme, each site maintains a local view of the network structure, which needs not to match the actual network structure. To check for majority, first a partition group forms a group view and then based on the group view compares its votes individually with the votes of every other partition group, rather than one plus half of the total votes, in the base set. It will elect itself as a majority group if it finds that its votes are the highest. (For easy reference, we call the dynamic voting schemes in [4,9] simple dynamic voting schemes and the one in [15] refined dynamic voting scheme.) Under the same model used in [9], it can be proven that the refined dynamic voting schemes provide higher availability than the simple dynamic voting schemes [15].

While dynamic voting schemes (both simple and refined) enhance data availability by means of increasing the likelihood that a majority group exists, they have the side effect of tending to decrease the size of a majority group. If such a side effect is not properly controlled, it may adversely affect the availability in some situations. To illustrate this point, imagine an extreme case when a data is replicated at 100 sites. (Each copy is assigned one vote for simplicity.) and a simple dynamic voting scheme is employed for replica control. Now consider the following scenario. The system partitions into two groups, with one consisting of 51 sites and the other 49 sites. An update is processed in the former group, making the copies in the latter group out of date. Now the 51-site group partitions into a 26-site group and a 25-site group. An update is processed in the 26-site group, making the copies in the 25-site group out of date. This process repeats itself, with each partition generating a smaller majority group in the previous majority group. Eventually, the majority group consists only of 2 sites. At this time none of the other 98 sites can process updates even if presumably they have merged into a single group, which is 48 times larger than the other one. This is because the 98-site group remains isolated from the 2-site group and contains no up-to-date copies. In this situation, more than 95 percent of the updates submitted to the system have to be aborted. Hence the availability of the system in such a situation is severely affected. Although the overall availability of a system cannot be judged based only on a single scenario, this example does suggest that such a side effect is a negative factor that may counter-balance the advantages of dynamic voting schemes.

1.2 An extension to the existing voting schemes

The problem posed in the last paragraph (called 'tiny majority problem') is intrinsic to any dynamic voting schemes, because they have defined a 'majority group' in terms only of its relative size, but ignored entirely its absolute size. To remedy such a deficiency, we must properly combine both aspects into the notion of a majority group. One approach to absolute-size control is restricting directly the minimum size of any majority groups. Alternatively, we can restrict the minimum size of a base set, indirectly controlling the size of majority groups.

Note that although both approaches control ultimately the sizes of majority groups, they have subtle difference. Consider a system employing a simple dynamic voting scheme for replica control. If we restrict the minimum size of base sets, then a most recent majority group will be disqualified for being a base set if its size is below the minimum. Thus it cannot be used by any nodes to derive majority groups. But any group, irrespective of its size, can be a majority group as soon as it is derivable from some base set. On the other hand, if we restrict the minimum size of majority groups, then a most recent majority group is always a base set. (Recall the definition of a base set in a simple dynamic voting scheme.) But a small enough group is not qualified for being a majority group even if it is derivable from some base set. However, since they control the (absolute) sizes of majority groups, both approaches support to achieve the following overall effects. That is, on one hand, a group is a majority group without possessing a majority of the votes of the entire system, and on the other hand, the size of the majority will never become excessively small. Such effects should be made optimal in the sense that the vulnerability to successive failures and the tiny majority problems are overcome.
to the extent that the best joint effect can be produced. In this paper, we will only discuss how we can use the first approach to achieve this overall effect. Some discussion about the second approach can be found in [10], where the authors developed a mechanism by which a base set is lower bounded by 3 in size. We will give a more general discussion of the second approach and certain comparisons of the two on a forthcoming paper.

One of the merits of the first approach is that it can be realized very easily. The implementation incurs almost no extra overhead beyond what dynamic voting schemes have incurred. Actually all that is needed is a value, called lower bound, associated with each data item, plus a more elaborate majority test. The majority test ensures that, among other things, 'too small' a group will never be a majority group and a 'large enough' group is always a majority group.

In addition to its simplicity, the approach is interesting in that, by varying the lower bound over a predefined interval, a class, called voting class, can be generated among which the static voting schemes and (simple) dynamic voting schemes are just two ordinary members. This means that the schemes in the voting class can be represented in a very simple and uniform way. Thus for any particular network, it is possible for a designer to choose a scheme in a voting class that maximizes the availability. (In what follows, whenever the term 'dynamic voting scheme' is used, it is referred to a simple dynamic voting scheme.)

The rest of this paper is organized as follows. In section 2, we give a simple description about the system model. In section 3, we present how lower bounds are embodied into dynamic voting schemes, outline the correctness proof and give an example to illustrate how the new scheme works. In section 4, we give a more detailed characterization for the notion of voting class. In section 5, we show the evaluation results concerning the availabilities for the schemes in some voting classes and compare the availability of optimal schemes with some of the existing voting schemes. We conclude this paper by summarizing the major results.

2 Outline of the Model

The system consists of n sites interconnected by communication links. A single site is replicated at every site. The file can be updated by executing an update transaction (or update for short). An update is a read operation followed by a write operation. To be considered correct, the concurrent execution of updates must be one copy serializable [3].

Sites and links can fail. A failed component simply stops functioning. Due to the failure, the system may be partitioned into isolated partition groups. However, we assume no failures occur while a transaction execution is in progress. (Since otherwise some recovery algorithms, such as the one introduced in [13, 14], can be adopted to provide such a transparency.)

Each site runs a concurrency control algorithm that guarantees (copy-level) serializability of concurrent updates within a partition group. To guarantee one copy (data-level) serializability, each site must in addition run a replica control algorithm, such as the one presented in section 3.

Practically, the decision as to the acceptance or final commitment of an update submitted to a site must be made co-operatively by both concurrency control and replica control algorithms. Since this paper is concerned only with replica controls, we assume that the conditions for committing an update imposed by concurrency control algorithms are always satisfied. In other words, transaction executions at copy level appear to be serial within a partition group.

Note that the assumption of single replicated file is solely for the purpose of simplifying the presentation. In case of multiple file system (or a database system) all that is needed is to apply the arguments to every file (or data item) that is updated by a single transaction.

3 Voting with Lower Bound

3.1 Basic concepts

In the following, we use fi to represent the copy of file f that resides at site i.

Associated with fi is an integer L, called lower bound for f. The value of lower bound for f is determined at design time. This value is maintained by fi for all i throughout the execution.

In addition, associated with each copy fi, are two integers, ni called version number of the copy, and UVi, called update group votes. The version number of a copy is the number of updates that have been performed to the file until the last update to that copy. Initially, every copy has version number 1. (This is because we assume every copy is updated by an initial update when the system is created.) Thus if two copies have different version numbers, the one with the larger version number is the more recent copy than the other. In particular, the copy with the largest version number in the system is the most up-to-date copy. Such a copy is called a current copy. A partition group consisting of current copies is called current group. The update group votes associated with a copy is the number of votes used by the copy to check for majority of its containing group. It is initialized to the number of total votes in the system. Whenever a group of copies commits an update, the update group votes associated with every copy in the group is updated to the total votes of the committing group. Thus the update group votes associated with a copy is always equal to the total votes of the most recent majority group that contains it.

3.2 Update protocol

For simplicity, We assume that each copy of the file is assigned one vote. The generalization that different copies have different votes is straightforward. When an update arrives at site i, site i initiates a consensus procedure in its contain-
ing group. The operations that are performed at site $i$ are specified as follows.
1. Solicit $v_j$ and $UV_j$ from every site $j$ that is reachable.
2. Upon receiving $v_j$ and $UV_j$ from every reachable site $j$, it constructs the following set:
   \[ G = \{ j : \text{response has been received from site } j \} \]
   \[ M = \max_{j \in G} \{ v_j = M & j \in G \} \]
   \[ I = \{ j \colon v_j = M & j \in G \} \]
   \[ UV = UV_j \text{ for any } j \in I \]
3. Based on the sets constructed at step 2, site $i$ checks for majority for $G$:
   - $G$ is a majority group if
   - \( |I| \geq |UV/2| + 1 \) and \( |G| \geq L \) or
   - \( |G| \geq \max\{\lceil n/2 \rceil + 1, n - L + 1 \} \).
4. If $G$ is a majority group, then the update is processed in $G$ as follows.
   - i. Read copy $f_i$ for some $i \in I$.
   - ii. Write copy $f_i$ for all $j \in G$.
   - iii. Perform the operations: $v_j \leftarrow M + 1$ and $UV_j \leftarrow |G|$
   - otherwise, the update is rejected. The majority test specified at step 3 is different from any existing voting schemes. It has the following implications. When the size of a group is 'moderate', the number of current copies contained in this group predominates the test for its majority. However, if its size is larger than a specific value (condition C2) or smaller than $L$, the lower bound, then it is forced to be or not to be a majority group, respectively. It is this elaboration in majority test that makes it possible to achieve higher availability than existing voting schemes.

3.3 Outline of the Correctness Proof

The correctness of the algorithm can be argued as follows. Suppose $T_1$ and $T_2$ are two updates that successively commit in group $G_1$ and $G_2$, respectively. Thus both groups pass majority test. If both of them satisfy condition C1, then the arguments of the correctness proof for dynamic voting schemes apply. If this is not the case, then at least one of the two groups satisfies condition C2. Therefore these two groups must overlap.

3.4 Examples

In this section, we will give some examples that best explain the working of a voting scheme with lower bound $L$.

**Example 1**

Suppose a file is replicated at seven sites (i.e., $n = 7$). Suppose also it has been determined that $L = 3$. Initially, $v_i = 1$ and $UV_i = 7$ for all $i$, $1 \leq i \leq 7$. Now a partition failure occurs that isolates group \{1,2,3,4\} from group \{5,6,7\}. The network state is shown in Figure 1. Now, suppose an update arrives at site 1. We have
   \[ G = \{1,2,3,4\} \]
   \[ M = 1 \]
   \[ I = \{1,2,3,4\} \]
   \[ UV = 7 \]

Figure 1: Network state after the first partition.

Now the group \{1,2,3,4\} partitions into two groups \{1,2,3\} and \{4\}. Figure 2 shows the network state at this point. Suppose at this time, an update arrives at site 1. We have the following:
   \[ G = \{1,2,3\} \]
   \[ M = 2 \]
   \[ I = \{1,2,3\} \]
   \[ UV = 4 \]

Since $|I| = 3 \geq |UV/2| + 1$ and $|G| \geq L$, the majority test succeeds. Thus the update is accepted. Accordingly, version numbers and update group votes are modified as $v_j = 3$ and $UV_j = 3$ for $1 \leq j \leq 3$.

Figure 2: Network state after the second partition.

Now group \{1,2,3\} splits into \{1,2\} and \{3\}. The network state at this moment is shown in Figure 3. When an update arrives at site 1, we have
   \[ G = \{1,2\} \]
   \[ M = 3 \]
   \[ I = \{1,2\} \]
   \[ UV = 3 \]

Since $|I| = |UV/2| + 1$ and $|G| < L$, condition C1 is not true. Since $|G| < \max\{\lceil n/2 \rceil + 1, n - L + 1 \} = 5$, C2 is not true. Thus the majority test fails. Thus group \{1,2\} is

\[ v_j \]
\[ UV_j \]

Figure 3: Network state after the third partition.
not a majority group. (It is easy to verify that, under this situation, the majority test allows no group to be a majority group.)

Note the difference between this scheme and dynamic voting. Under a dynamic voting, group \{1,2\} would continue being a majority group.

Now suppose sites 3,4,5,6 and 7 merge together. Let G = \{3,4,5,6,7\}. Figure 4 is the network state at this moment. Now, an update arrives at site 3. Since |G| = 5 \geq \max\{\lceil n/2 \rceil + 1, n - L + 1\} = 5, G is a majority group.

Again, a dynamic voting scheme would behave differently in this situation: G would not be allowed to process updates despite its large size.

4 Voting Class

The lower bound $L$ in the specification of update protocol can be assigned different values. Each lower bound defines a unique voting scheme. For a specific system, the set of voting schemes defined by all integer $L$ as a lower bound is called a *voting class* for the system (or simply voting class if no confusion is possible). In this section, we will take a close look at the structure of voting class.

4.1 Some properties of a voting class

In many places in the following context, we will have to argue that a specific partition group is (or is not) a majority group under a voting scheme. We now give an informal description of a proposition that is slightly different nevertheless equivalent to condition $C_1$ specified in the majority test of our voting scheme.

It can be easily proven that for any voting scheme in a voting class, if $G$ contains at least one current copy, then condition $C_1$, $|I| \geq \lceil UV/2 \rceil + 1$, is true if and only if the number of current copies contained in $G$ is more than half of the total number of current copies (some of them may fail) in the *entire system*. This proposition is true because if $G$ contains some current copies, then $|I|$ and $UV$ must be the number of current copies contained in $G$ and the entire system, respectively.

We will use this modified version of condition $C_1$ frequently in the following context, deeming it is easier to illustrate as an informal argument. In the subsequent discussion, we will use $S(L)$ to denote the voting scheme with lower bound $L$.

The following two assertions describe some characteristics of a voting class. For simplicity, we will give only informal arguments for the assertions.

**Assertion 1:** For every $L$, $L \leq 1$, $S(L)$ is a dynamic voting scheme and for every $L$, $L \geq \lceil n/2 \rceil + 1$, $S(L)$ is a static voting scheme.

First assume $L \leq 1$. Since $I$ is a subset of $G$, we have $|G| \geq |I|$. If $|I| \geq \lceil UV/2 \rceil + 1$, then $|G| \geq 1 \geq L$. This means $C_1$ is true. Since we also have $C_1 \Rightarrow |I| \geq \lceil UV/2 \rceil + 1$. $C_1 \equiv |I| \geq \lceil UV/2 \rceil + 1$. On the other hand, $C_2 \equiv |G| \geq n - L + 1$. If $C_2$ is true, it must be the case that $|G| = n$. (This happens only when $L = 1$.) This implies that $G$ is the group consisting of all sites. Thus $I$ is the set of all current copies. By the proposition stated above, we have $|I| \geq \lceil UV/2 \rceil + 1$. This means that $C_2 \Rightarrow C_1$. Thus $C_2$ can be dropped from the conditions for majority test. The scheme becomes exactly a dynamic voting scheme [9].

We now assume $L \geq \lceil n/2 \rceil + 1$. We have $n - L + 1 \leq n - \lceil n/2 \rceil \leq \lceil n/2 \rceil + 1$. Thus $C_2 \equiv |G| \geq \lceil n/2 \rceil + 1$. So $C_1 \Rightarrow |G| \geq L \Rightarrow C_2$. This means that $C_1$ can be dropped from the conditions for the majority test. Thus the scheme is precisely a static voting scheme.

**Assertion 2:** $S(L_1) \neq S(L_2)$ for every $L_1$ and $L_2$, where $1 \leq L_1, L_2 \leq \lceil n/2 \rceil + 1$, and $L_1 \neq L_2$.

Without loss of generality, assume $L_1 < L_2$. Thus $L_2 \geq 2$.

We need to show that, when employed by some replicated file system for replica control, $S(L_1)$ and $S(L_2)$ exhibit different behaviors for some sequence of failure and merge events. The different behaviors can be, for example, that after the occurrence of the last event in the sequence, $S(L_1)$ and $S(L_2)$ permit different groups to be majority groups.

We assume that $n$ is an odd number and the $n$ sites contained in the network are 1,2,...,n, which are connected through a common bus with two end points site 1 and site n.

In the case $L_2 \geq 3$, a sequence of events that makes $S(L_1)$ and $S(L_2)$ choose different majority groups is given in Table 1.

The left column in the table is a sequence of failure and merge events, where edge $e_i$ denotes the link between site $i$ and site $i + 1$, for all $i, 1 \leq i \leq n - 1$. Initially, there is only one partition group \{1,2,...,n\} in the system. After each failure, one more partition group is generated. After the last repair event, the system consists only of two partition groups. The sets of partition groups generated as a result of failure or repair events are independent of the schemes employed by the system. (This is reflected in Table 1 by the fact that the sets of groups at the middle column and that at the right column are identical if they are in the same row.) However, a copy may have different states under different schemes. For example, after $e_{L_2-1}$ fails, $L_2$ remains current under the scheme $S(L_2)$ but becomes non-current under the
Table 1: Different behaviors of $S(L_1)$ and $S(L_2)$ for the same failure-repair sequence.

<table>
<thead>
<tr>
<th>Events</th>
<th>When $S(L_1)$ is used</th>
<th>When $S(L_2)$ is used</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{n-1}$ fails</td>
<td>${1,\ldots,n-1},{n}$</td>
<td>${1,\ldots,n-1},{n}$</td>
</tr>
<tr>
<td>$e_{n-2}$ fails</td>
<td>${1,\ldots,n-2},{n-1},{n}$</td>
<td>${1,\ldots,n-2},{n-1},{n}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$e_{L_2}$ fails</td>
<td>${1,\ldots,L_2},{L_2+1},\ldots,{n}$</td>
<td>${1,\ldots,L_2},{L_2+1},\ldots,{n}$</td>
</tr>
<tr>
<td>$e_{L_2-1}$ fails</td>
<td>${1,\ldots,L_2-1},{L_2},\ldots,{n}$</td>
<td>${1,\ldots,L_2-1},{L_2},\ldots,{n}$</td>
</tr>
<tr>
<td>all failed links but $e_{L_2-1}$ repair</td>
<td>${1,\ldots,L_2-1},{L_2,L_2+1,\ldots,n}$</td>
<td>${1,\ldots,L_2-1},{L_2,L_2+1,\ldots,n}$</td>
</tr>
</tbody>
</table>

scheme $S(L_1)$. (In Table 1, the boldface letters denote the current copies.)

After merges occur, the group $\{L_2,\ldots,n\}$ is a majority group under the scheme $S(L_2)$ since the cardinality of this group is $n-L_2+1 \geq \lceil n/2 \rceil + 1$, condition $C_2$ is satisfied.

However, it is not a majority group under the scheme $S(L_1)$: since we have $I=\{L_2\},\{L_2\} = \{1,\ldots,L_2\} \setminus \{1\}$, thus $C_1$ is not true; on the other hand, the cardinality of the group is $n-(L_2-1)$, and we have the following relation:

$$\max\{\lceil n/2 \rceil +1,n-(L_2-1)\} \leq \max\{\lceil n/2 \rceil +1,n-L_1+1\},$$

thus $C_2$ is not true.

In case that $L_2=2$, most of the arguments are the same. The only difference is the arguments for C1 being false. That is, under $S(L_1)$, group $\{L_2,\ldots,n\}$ contains single current copy $L_2$, but the total number of current copies in the system is 2. By the proposition stated previously, C1 is false.

The assertion 2 tells us that every voting scheme behaves differently from every other voting scheme in the voting class. The different behaviors are caused by different lower bounds. (Note that all schemes in a voting class have identical vote assignment.) Thus it is of interest to know how availability varies over a voting class. In particular, from the viewpoint of a designer, it is desirable to locate the voting scheme in a voting class that provides the maximum data availability among all voting schemes in it. (We call the lower bound for the voting scheme with maximum availability optimal lower bound in $[1,\lceil n/2 \rceil +1]$.)

In general, the value of optimal lower bound depends on the characteristics of a particular network. It is usually related to such factors as network topology, the long term alive probabilities of nodes and links, the update submission rate at each site, etc. In any case, it can be determined at the design time. For example, a designer first comes up with a stochastic model which he/she thinks is closest to the environment in question. Then he/she solves this model to get the availability for each integer as a lower bound in $[1,\lceil n/2 \rceil +1]$. The optimal lower bound is the one for which the obtained availability is the highest.

5 A Sample Derivation of Optimal Schemes in a Voting Class

As mentioned in the last section, the optimal lower bound for a voting class can be obtained by using enumeration. Because of the uniformity of the representations of the schemes in a voting class, the task for the enumeration is greatly eased. As can be seen in the appendix, when using a Markov Model for evaluation, for example, we can use a single form of linear systems of equations for almost all schemes in a voting class. In this section, we will show some of the evaluation results concerning the availabilities of the schemes with different lower bound in the voting class.

5.1 Availability measure

The availability is the limit as $t$ approaches to infinity of the probability that an update submitted to an arbitrary site at time $t$ succeeds. This measure takes into account the likelihood that both a majority group exists and an update is submitted to the majority group.

5.2 A stochastic model

The stochastic model we will use to obtain the optimal lower bound is the same as that assumed in [9,12]. The assumptions made to such a model are stated in the following.

1. The communication links between sites are infallible. Any alive site can communicate with any other alive site.
2. The failures at various sites form independent Poisson processes with failure rate $\lambda$.
3. The repairs at various sites form independent Poisson processes with repair rate $\mu$.
4. Updates are instantaneous.
5. Updates are frequent. That is, after every failure or repair, an update always arrives at a functioning site and is processed before the next failure or repair.

A general justification for the use of this model can be
found in [9,12]. Here we wish to say a bit more only about the first assumption. The first assumption is a strong assumption which models high-connectivity networks. Hence its generality is limited. The reasons we choose this model are two fold. Firstly, it is simple enough to enable us to maintain a Markov Process with manageable number of states. Secondly, our intention is to show a sample derivation of the optimal lower bound. The derivations based on different models will basically follow the same procedure. Thus the degree of the generality of the model used for the sample derivation should not be the dominating factor in determining the significance of the results. Since the assumed model is the only model with which people have quantitatively evaluated the availability of several schemes that have gained the popularity in the replica control research community, we are interested in knowing if the obtained optimal lower bound is better than the previous superior.

5.3 The availabilities of the schemes in a voting class

The system is formulated as a Markov Process. The state diagram for this process and the detailed procedure in solving this process is given in the appendix. The following three tables contain the evaluation results of the availability for each scheme in the voting class. Table 2,3 and 4 are for systems consisting of 5, 7 and 9 sites, respectively. In each of these tables, a column contains the availabilities of all voting class members for a particular repair/failure ratio, and a row contains the availabilities of a particular voting class member for varies repair/failure ratio. Each column contains a boldface number which denotes the availability of the optimal lower bound for the corresponding repair/failure ratio. It can be seen from these tables that the static voting schemes (with lower bounds 3, 4 and 5, respectively, in the three tables) lie at the bottom of the voting class under every repair/failure ratio. The results for the performance of the dynamic voting schemes (with lower bound 1 in all three tables) are also not very encouraging. When the repair/failure ratio is low, the dynamic voting schemes rank middle-low. As the ratio increases, its rank increases to middle-high. However, under no circumstances it can reach the highest ranking. In a system of 5 sites, the optimal lower bound is 2 for all 5 different ratios. In case of 7 sites, the optimal lower bound is 3 for the

<table>
<thead>
<tr>
<th>Lower Bound</th>
<th>Availability for Different Repair/failure Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>0.4697599</td>
</tr>
<tr>
<td>2</td>
<td>0.5021532</td>
</tr>
<tr>
<td>3</td>
<td>0.4672912</td>
</tr>
</tbody>
</table>

Table 3: Availability under different lower bounds for a system of seven sites.

<table>
<thead>
<tr>
<th>Lower Bound</th>
<th>Availability for Different Repair/failure Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>0.5350492</td>
</tr>
<tr>
<td>2</td>
<td>0.5420461</td>
</tr>
<tr>
<td>3</td>
<td>0.5464969</td>
</tr>
<tr>
<td>4</td>
<td>0.4672912</td>
</tr>
</tbody>
</table>

Table 4: Availability under different lower bounds for a system of nine sites.

<table>
<thead>
<tr>
<th>Lower Bound</th>
<th>Availability for Different Repair/failure Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>0.5670846</td>
</tr>
<tr>
<td>2</td>
<td>0.5814343</td>
</tr>
<tr>
<td>3</td>
<td>0.5728458</td>
</tr>
<tr>
<td>4</td>
<td>0.5340316</td>
</tr>
<tr>
<td>5</td>
<td>0.4663523</td>
</tr>
</tbody>
</table>

Table 2: Availability under different lower bounds for a system of five sites.

Table 3: Availability under different lower bounds for a system of seven sites.

Table 4: Availability under different lower bounds for a system of nine sites.
repair/failure ratio 1.4, 2 for other four values. In the case that the system consists of 9 sites, the optimal lower bound is 3 for the repair/failure ratio 1.4 and 2.2, 2 for the other three values.

It can also be seen from the table that, as the repair/failure ratio increases, the discrepancy in availabilities for different lower bounds decreases drastically. This can be explained partially by the special nature of the chosen stochastic model. In this model, repair is especially favorable for availability, since as long as there is a majority group, a repair always contributes to increasing availability. The rapid increase in availability for all lower bounds in turn narrows their discrepancy. This will not be the case, however, for a model where a majority group and a minority group can coexist. (In fact, this is the case in most application environments.) For this kind of models, the repair in a minority group may not contribute at all to increasing availability. (Consider, for example, the following scenario. The system is in a state where there are two groups, one being a majority group and the other a minority group. A site recovers in the minority group, but fails again before this group merges with the majority group. Thus the recovery of this site virtually does not occur.)

6 Conclusion

An approach to increasing availability for replicated file system is discussed. It is based on the observation that high availability can be gained by improving vulnerability of static voting schemes to successive failures and neutralizing the side effects of tiny majority problem for dynamic voting schemes. To this end, the minimum allowable size of majority groups in dynamic voting schemes is properly controlled. This is done by attaching a lower bound to a data item and employing a more elaborate majority test scheme. The attachment of lower bounds reveals an interesting fact that the existing voting schemes, both static and dynamic, belong to a general class called voting class. The voting class is formed by varying the lower bound over a predefined interval \([1, n/2 + 1]\). The existence of such a voting class is of practical significance because, on one hand, it provides a designer with an easy way to obtain an "optimal" voting scheme within the class, and on the other hand, implementation of such an optimal scheme involves almost no additional runtime overhead beyond that of dynamic voting schemes. By using a known stochastic model, we showed how to obtain the optimal lower bound and compared it with the existing voting schemes.

To simplify the presentation, we have omitted the discussion about the following two aspects that can be embodied into the definition of a voting class. A voting class of so embodied may provide higher availability, but may involve more overhead in its implementation. We believe that these aspects can be incorporated into the definition of our voting class without much difficulties if doing so is preferred by a designer.

1. For every scheme in our voting class, techniques can be used to break ties (For example, one suggested in [11]).
2. The conditions in majority test for simple dynamic voting schemes is used as part of the conditions for defining a voting class. Thus the approach is in favor of high update frequency for data items. If high update frequency cannot be ensured, the conditions in majority test for refined dynamic voting schemes can be employed to replace that for simple dynamic voting schemes in the definition of our voting class.

References

Appendix. Evaluation of Markov Process

The state diagram for the Markov Process for a scheme with lower bound \( m \geq 2 \) in the voting class is depicted in Figure 5. Each state in the diagram is denoted by a triple of integers, \((X,Y,Z)\), where \( X \) is the number of alive current copies, \( Y \) is the total number of (alive or failed) current copies, and \( Z \) is the number of alive non-current copies. The arcs between the states denote state transitions. The number attached to each arc is the transition rate for that arc.

In the diagram, we use \( A_{m+i} \) to denote a state \((m+i,m+i,0)\), where \( 0 \leq i \leq n-m \), and \( A_j \) to denote a state \((j,m,i)\), where \( 0 \leq j \leq m-1 \).

Note that for this model, there is only one partition group in the system at any time. (In the subsequent discussion in this section, whenever we say 'the partition group', we refer to this unique partition group in the system.) If the system is in state \( A_{j+i} \), where \( 0 \leq j \leq m-1 \), the partition group consists of \( i+j \) copies among which \( j \) copies are current. The remaining \( n-(i+j) \) copies are failed. Among these failed copies, \( j \) copies are current. On the other hand, if the system is in state \( A_{m+i} \), the partition group consists of \( m+i \) copies and all are current. Furthermore, all remaining copies failed and none of them are current. Thus this partition group must be a majority group.

Initially, the system is in state \( A_m \), meaning the partition group consists of all sites in the system. Assume that the system is in state \( A_{m+i} \), where \( 0 < i < n-m \), and one of \( m+i \) current copies fail. Immediately after the failure occurs, the partition group consists of \( m+i-1 \) current copies out of \( m+i \). Since \( i > 0 \), \( m+i-1 \geq m \). Since \( m+i-1 \geq \lceil \frac{m+i}{2} \rceil +1 \), updates arriving at this group (recall the assumption about high update frequency) commit. Thus all copies in this group stay current but the remaining \( n-(m+i) \) copies become non-current. This implies a state transition \( A_{m+i} \to A_{m+i+1} \).

Now assume that the system is in state \( A_{n} \). If one of \( m \) current copies fail, the partition group in the system is no longer a majority group since its size falls below the lower bound \( m \). Thus the arriving update is rejected. The \( m \) current copies in state \( A_m \) stay current now but only \( m-1 \) are alive. That is, the transition \( A_m \to A_{m-1} \) occurs.

Now assume that the system is in state \( A_{j+i} \), where \( 0 \leq j < n-m \). In this case, there are \( j \) alive current copies, \( m-j \) failed current copies and \( i \) failed non-current copies. One of the following state transitions may occur.

- When \( j > 0 \), if one of the \( j \) alive current copies fails, then \( A_j \to A_{j-1} \).
- When \( i > 0 \), if one of \( i \) alive non-current copies fails, then \( A_{j+i} \to A_{j+i-1} \).

When one of the failed copies recover, the next state to be entered depends on the value of \( j \) and the currentness of the recovered copy. The following is a detailed explanation.

If \( [m/2]+1 \leq j \leq m-1 \), the partition group in the system consists of more than \( [m/2] \) current copies. The total number of (alive or failed) current copies is \( m \). Therefore, it must be the case that \( i < n-j \), otherwise the cardinality of the partition group would be \( j+i \geq m \) which is the lower bound. Thus the group would be a majority group and the system would be in state \( A_{j+i} \), which contradicts the assumption that the system is in state \( A_{j+i} \). (This is why for every state \( A_{j+i} \), where \( [m/2]+1 \leq j \leq m-1 \), in the state diagram, \( i \) can be at most \( m-j-1 \).)

When \( i < m-j-1 \), the transition \( A_j \to A_{j+i} \) if one of the \( n-m-i \) failed non-current copies recover. When the system is in state \( A_{j+i} \), the recovery of any failed copies results in the partition group consisting of \( m \) copies. Furthermore, since this partition group contains more than \( [m/2] \) current copies, it is a majority partition group. This implies the transition \( A_{j+i} \to A_m \).

Now assume \( 0 \leq j \leq [m/2] \). If \( j \leq n-m-j \), the recovery of any previously failed non-current copies results in transition \( A_j \to A_{j+i+1} \). When the system is in state \( A_{j+n-m-j} \), the recovery of any copy results in the partition group consisting of \( n-m+1 \) copies. The partition group becomes a majority since \( C_2 \) can be satisfied. This implies transition \( A_{j+n-m-j} \to A_{n-m+i} \).

Now we further consider state \( A_{[m/2]+i} \), where the number of the current copies in the partition group is only one short of satisfying condition \( C_1 \). If \( i \geq [m/2]-1 \) and one of the \( [m/2] \) failed current copies recover (Note \( [m/2] = m-[m/2] \)), then \( C_1 \) is satisfied. This is because, immediately after the recovery, the partition group contains \( [m/2]+i+1 \geq m \) sites among which \( [m/2]+1 \) is current. This means the transition \( A_{[m/2]+i} \to A_{[m/2]+i+1} \) occurs.
A horizontal (vertical) arc for which no rate is shown has the same rate as the same-directioned bottom-most horizontal (leftmost vertical) arc in the same column (row).

For each state, a linear equation can be constructed by setting the flow out to the flow in for that state. The steady state probabilities of all the states can then be computed by solving this linear system of equations. In the following, rows are counted bottom up. Thus row zero is the bottom most row above which comes row one, and so on.

For sake of simplicity, we also use the symbol $A_{ij}$ ($A_{i}$) to denote the steady state probability of state $A_{ij}$ ($A_{i}$). Thus the availability of the system is

$$\sum_{i=0}^{n-m} \frac{m+i}{n} A_{m+i}$$

For each state, a linear equation can be constructed by setting the flow out to the flow in for that state. The steady state probabilities of all the states can then be computed by solving this linear system of equations. In the following, rows are counted bottom up. Thus row zero is the bottom most row above which comes row one, and so on.

The equations for the states at row $0$ are:

$$m \mu + (n-m) \mu A_{0,0} - \lambda A_{1,0} - \lambda A_{0,1} = 0$$

For each $i$, $1 \leq i \leq n - m - 1$, $(i \lambda + (n-m-i) \mu + m \mu) A_{0,i} - (n-m-i+1) \mu A_{0,i+1} - (i+1) \lambda A_{0,i+1} - \lambda A_{i,0} = 0$

$((n-m) \lambda + m \mu) A_{0,n-m} - \mu A_{0,n-m-1} = 0$

The equations for the states at row $j$, $1 \leq j \leq \lfloor m/2 \rfloor - 1$ are:

$$((m-j) \mu + (n-m) \mu + j \lambda) A_{j,0} - (m-j+1) \mu A_{j-1,0} - \lambda A_{j+1,0} - (j+1) \lambda A_{j+1,0} = 0$$

For each $i$, $1 \leq i \leq n - m - j - 1$, $(i \lambda + j \lambda + (n-m-i) \mu + (m-j) \mu) A_{j,i} - ((n-m-i+1) \mu) A_{j,i+1} - ((m-j+1) \mu) A_{j-1,i} - (i+1) \lambda A_{j+1,i} - (j+1) \lambda A_{j+1,i} = 0$

$((n-m-j) \lambda + j \lambda + m \mu) A_{j,n-j} - (j+1) \mu A_{j,n-j-1} - (m+1-j) \mu A_{j-1,n-j-1} - \lambda A_{j+1,n-j-1} = 0$

The equations for the states at row $\lfloor m/2 \rfloor$ are:

$$((m-j) \mu + (n-m) \mu + j \lambda) A_{j,0} - (m-j+1) \mu A_{j-1,0} - \lambda A_{j+1,0} - (j+1) \lambda A_{j+1,0} = 0$$
For each $i, 1 \leq i \leq m - j - 2, (i + j + (n - m - i)\mu + (m-j)\mu)A_{i,j} - ((n-m-i+1)\mu)A_{i+1,j} - ((m-j+1)\mu)A_{i-1,j} - 
(i+1)\lambda A_{i+1} - (j+1)\lambda A_{j+1} = 0
((m-j-1)\lambda + j\lambda + (n-m+1)\mu)A_{i,m-j-1} - (n-2m+j+1)\mu A_{i,m-j-2} - (m-j+1)\mu A_{i,m-j-1} = 0

The equations for the states at row $m-1$ are:

$((m-1)\lambda + \mu + (n-m)\mu)A_{m-1,0} - 2\mu A_{m-2,0} - \lambda A_{m-1,1} = 0$

The equations for the states at the top row are:

$((n-m)\mu + m\lambda)A_{m+0} - (m+1)\mu A_{m+1} - (n-m+1)\mu A_{m-1,0} - (n-m+1)\mu A_{m-2,1} = 0$

For each $i, 1 \leq i \leq n - 2m, ((m+i)\lambda + (n-m-i)\mu)A_{n+i} - (n-m+1-i)\mu A_{n+i-1} - (m+i+1)\lambda A_{m+i+1} - (m/2)\mu A_{m/2} - (m/2)^2 = 0$

$((n-m+1)\lambda + (m-1)\mu)A_{n-m+1} - m\mu A_{n-m} - (n-m+2)\lambda A_{n-m+2} - m\mu A_{n-m-1} - ... - m\mu A_{n-m} = 0$

The state diagram in Figure 5 is not applicable to the case when $m = 1$, since the system can never enter the state $(X,1,Y)$. This is because in a system where a dynamic voting scheme (i.e., the voting scheme with $m = 1$) is employed, the total number of current copies can never fall below two. (Recall our assumption that each replicated copy possesses one vote.) The state diagram when $m = 1$ is shown in Figure 6. The top row consists of all states in which there is a majority group. This Markov chain can be easily solved. (The detail is omitted.) The availability is calculated using the following formula:

$$\sum_{i=1}^{n} A_i$$

Note that the state diagram in Figure 5 is not applicable to the case when $m = 1$, since the system can never enter the state $(X,1,Y)$. This is because in a system where a dynamic voting scheme (i.e., the voting scheme with $m = 1$) is employed, the total number of current copies can never fall below two. (Recall our assumption that each replicated copy possesses one vote.) The state diagram when $m = 1$ is shown in Figure 6. The top row consists of all states in which there is a majority group. This Markov chain can be easily solved. (The detail is omitted.) The availability is calculated using the following formula:

$$\sum_{i=1}^{n} A_i$$

For each $i, n-2m+2 \leq i \leq n - m$, $((m+i)\lambda + (n-m-i)\mu)A_{m+i} - (n-m+1-i)\mu A_{m+i-1} - (m+i+1)\lambda A_{m+i+1} = 0$

Note that the state diagram in Figure 5 is not applicable to the case when $m = 1$, since the system can never enter the state $(X,1,Y)$. This is because in a system where a dynamic voting scheme (i.e., the voting scheme with $m = 1$) is employed, the total number of current copies can never fall below two. (Recall our assumption that each replicated copy possesses one vote.) The state diagram when $m = 1$ is shown in Figure 6. The top row consists of all states in which there is a majority group. This Markov chain can be easily solved. (The detail is omitted.) The availability is calculated using the following formula:

$$\sum_{i=1}^{n} A_i$$

Figure 6: The state diagram for a scheme with $m = 1$. 

156