Optimal Voting Strategy Against Rational Attackers

Li Wang, Zheng Li, Shangping Ren*
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{lwang64, zli80, ren}@iit.edu

Kevin Kwiat†
Cyber Science Branch
Air Force Research Laboratory, AFRL/RIGG
Rome, NY 13441
Email: kwiatk@rl.af.mil

***Abstract*—Voting algorithms are often used to improve a system's reliability through fault tolerance. However, when both the reliability of individual voters and the existence of rational attackers are taken into consideration, the number of voters that participate in an actual voting process determines the fault-and-attack tolerance performance of voting algorithms. In this paper, we are to find an optimal voting strategy (i.e., the optimal number of participating voters) against rational attackers whose goal is to defect the system by strategically compromising individual voters across the system. We model the problem of deciding the number of participating voters against rational attackers as a two-person zero-sum game problem and provide solutions based on the results from this well-known game problem. A set of experiments are performed to illustrate the voting strategy's performance in the presences of rational attackers.**

***Index Terms*—Attacker-defender Problem; Voting Strategy; Game Thoery; Reliability**

I. Introduction

For many safety-critical systems, such as air traffic control systems, fly-by-wire systems, etc., ensuring their reliability is often paramount. However, the system's reliability can be compromised when it is under cyber attacks. Furthermore, even for uncompromised components, they rarely have 100% reliability [11]. Hence, in order to improve the system's reliability, redundancy approaches, such as N-Modular Redundancy [20], and N-Version Programming [3], [6] are often used. With these approaches, systems can still perform correctly even when some of the replicas become faulty or have been compromised.

To achieve reliability in the presence of possible replica failures, voting algorithms are often used. There are many different voting algorithms. For instance, *Unanimity voting* [12] generates a result when all the replicas are in agreement. This type of voting algorithm is used when reaching agreement by all replicas is needed. However, it does not tolerate any replica failures. *Majority voting* [26] algorithm takes the majority as its final value, while *Plurality voting* [12] is a less strict form of majority voting. It requires *m-out-of-n* replicas to agree on the same result, where m is less than a strict majority of n (i.e., $m < \lceil \frac{n+1}{2} \rceil$).

In [11], a blending of fault and attack tolerance of three majority voting algorithms: Majority Rule (MR), Random Dictator (RD), and Random Troika (RT), is studied. For MR, it needs all the replicas to vote, while RD randomly chooses one replica's value as its result. Although RT also belongs to the majority voting family, the selection of the Troika is rather random, and the final result is decided by the majority of the three chosen replicas. The conclusion in [11] states that none of these three voting algorithms is always superior to the others when the number of uncompromised replicas and each replica's reliability vary. Therefore, in order to maximize system reliability, different voting strategies should be applied. In this paper, a voting strategy is defined as choosing the number of replicas that participate in voting.

The selection of the voting strategy is a challenging issue when the system is attacked by a rational attacker because the attacker's strategy is almost always unknown in advance. In addition, earlier work [11] only considers three special majority voting algorithms in which the number of votes involved are n (MR), 1 (RD), or 3 (RT), respectively. In fact, the number of voters in majority voting can range from 1 to n, where n refers to the total number of replicas available.

In this paper, we aim to find out the optimal number of participating voters in majority voting when the system is attacked by a rational attacker. In particular, we consider a scenario in which a system is composed of multiple clusters, and each cluster consists of the same number of replicas. We assume that the attacker only makes the effort to compromise the system when the system is in operation, and the attacker has to decide how many clusters to attack to minimize the number of surviving clusters. The defender has to decide the number of participating voters in each cluster that maximize the number of surviving clusters.

The rest of the paper is organized as follows. Section II discusses the related work. In Section III, we formally define the optimal voting strategy problem and provide the solution to the problem in Section IV. The experimental results are shown and discussed in Section V. Finally, we conclude and point out future work in Section VI.

II. Related Work

The analysis of attacker-defender problems have been studied for many years. Wang et al. [29], [30] discussed the attacker-defender problem and analyzed how to hide the location of the core components and allocate resources to maximize a system's availability, when the system is under cyber attack. In [1], Bier et al. applied game theory to identify

*The work is supported by NSF CAREER Award (CNS0746643)
†Approved for Public Release; Distribution Unlimited: 88ABW-20113364, JUN 13, 2011

978-1-4577-1891-5/11/$26.00 ©2011 IEEE

optimal defenses against intentional threats to system reliability. In their system model, they considered both series and parallel systems, and illustrated how the optimal allocation of defensive investments depends on the structure of the system, the cost-effectiveness of infrastructure protection investments, and the adversary's goals and constraints.

In [17], Levitin et al. proposed three defense approaches (i.e., false targets, protection, and replication) to minimize system damage when both the defender and the attacker have limited resources, and illustrated the minimum number of genuine components needed to meet the system demand and how the defender and attacker choosed their strategies when the contest intensity changes. In [16], they analyzed how to allocate resources between deploying false targets and enhancing object protection when protecting a single object. They concluded that the optimal number of false targets does not depend on the attacker's resources but only depends on the relative target cost. In [15], the approaches of protection and redundancy are provided to reduce the expected damages caused by the attacks. The vulnerability of each system element is determined by an attacker-defender contest success function, and the expected damage caused by the attack is evaluated as the system's unsupplied demand. In addition, the attacker-defender problem in multi-state systems is also studied in [13], [14], [18], [19].

Our work differs from those discussed above in two aspects. First, in our work, we assume that attackers are given a fixed amount of time to compromise the system. This assumption is reasonable for mission critical applications that only operate for a certain period of time, such as battle field applications. In the work discussed above, attackers are assumed to have a fixed amount of attack resources. A second difference is the defender's model. In earlier work, defenders are given a fixed amount of defensive resources that are distributed according to different defensive approaches. However, in our work, the defender has no extra resources, but can only choose different ways to vote to maximize the system's expected number of surviving clusters.

In [9], Hardekopf et al. proposed a decentralized voting algorithm that improves system dependability and protects the system from faults and hostile attacks. In [27], Tong et al. showed how to make optimal weight assignments for the majority voting strategy in systems and also proposed a new effective vote assignment algorithms which aimed to maximize system reliability. In [5], Davcev proposed a dynamic weighted voting scheme for consistency and recovery control of replicated files in distributed systems. More weighted voting schemes are discussed in [2], [8], [24].

There are two differences between our work with those discussed above. First, we consider a centralized voting scheme rather than a decentralized voting scheme in our medel. Second, in our system model, up to n components can be involved to form the voting components. However, the previous work's voting machanism is based on fixed number of voting components.

III. Preliminaries and Problem Formulation

A. Preliminaries and Assumptions

We assume the system consists of n independent clusters, and each of them is composed of s replicas. The reliability of each individual uncompromised replica is p. For each cluster, its reliability is the probability that a correct final result is obtained through a voting process. Figure 1 shows the configuration of a cluster in a system. We use s to denote the cluster size, i.e., the total number of replicas in the cluster; and f refers to the number of compromised replicas in the cluster; and k denotes the number of participating voting replicas. In this paper, the terms "voters", "voting replicas", "participating voters", and "replicas that vote" are synonymous.

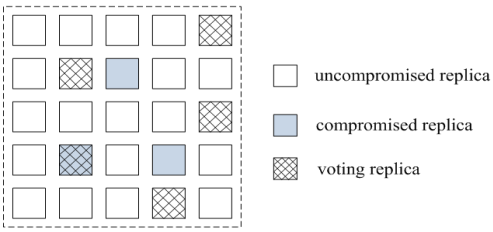


Fig. 1: The cluster configuration

We further assume that the system is to operate for D time units, and it is only during the system operation time that the attacker can attempt to compromise the system. For an attacker, the time needed to compromise a system component depends on the component's vulnerabilities and the attacker's skill level [21], [22]. We assume that the attacker can successfully compromise m replicas within that D time units. The attacker knows the the number of clusters, the number of replicas in each cluster, and the replicas' reliability in the system, but the attacker does not know how many replicas are chosen to participate in voting within a cluster.

For the attacker, his strategic decision is to decide how many clusters to attack. Assume the attacker chooses h ($h \leq n$) clusters to attack and evenly distributes the effort among the clusters, which indicates that there are at least $f = \lfloor \frac{m}{h} \rfloor$ compromised replicas in the h chosen clusters, and no compromised components in the remaining $n - h$ clusters. When m is not evenly divided by h, say that the division ends with remainder q, that is $q = m \bmod h$. Under this scenario, the attacker will compromise $\lfloor \frac{m}{h} \rfloor + 1$ replicas in the first q out of h clusters, and compromise $\lfloor \frac{m}{h} \rfloor$ replicas in the remaining $h - q$ clusters.

The defender aims to maximize the number of surviving clusters (i.e., the clusters which can produce a correct result under attack). The defender's strategy is to decide the number of participating voters k ($1 \leq k \leq s$) for each cluster, and the result produced by the cluster is through the majority voting from the chosen participating voters.

B. The Reliability of a Cluster not under Attack

When the cluster is not under attack, none of the replicas in the cluster is compromised. Assume the defender chooses

k ($1 \leq k \leq s$) replicas to vote in each cluster, where s is the size of each cluster. The final result is decided by the majority of the results from the k chosen replicas. In order to have a correct result, at least $\lceil \frac{k+1}{2} \rceil$ replicas must vote correctly. We use $\theta(k)$ to refer to the probability that the cluster is able to produce a correct result when it is not under attack, and we have

$$\theta(k) = \sum_{i=\lceil \frac{k+1}{2} \rceil}^{k} \binom{k}{i} p^i (1-p)^{k-i} \tag{1}$$

where k refers to the number of voting replicas, p is each replica's reliability.

C. The Reliability of a Cluster under Attack

When the cluster is under attack, assume f out of s replicas are compromised by the attacker, and the defender chooses k replicas to vote. In order to produce a correct result, the number of uncompromised replicas in the k replicas needs to be at least $\lceil \frac{k+1}{2} \rceil$. Therefore, the minimum number of uncompromised replicas in the k replicas can not be smaller than $\lceil \frac{k+1}{2} \rceil$. However, when the number of compromised replicas in the cluster is small (i.e., $f < \lfloor \frac{k-1}{2} \rfloor$), the minimum number of uncompromised replicas in that k replicas will be $k - f$, which is larger than $\lceil \frac{k+1}{2} \rceil$. Therefore, the minimum number of uncompromised replicas in the k replicas is $max(\lceil \frac{k+1}{2} \rceil, k - f)$.

Similarly, when the number of compromised replicas f in the cluster is large (i.e., $f > s - k$), the number of uncompromised replicas in the k replicas can not exceed $min(k, s - f)$. Therefore, if the number of uncompromised replicas is i, we have $max(\lceil \frac{k+1}{2} \rceil, k-f) \leq i \leq min(k, s-f)$.

Assume j out of i uncompromised replicas vote correctly, in order to produce a correct result, we must have $\lceil \frac{k+1}{2} \rceil \leq j \leq i$. We use $\varphi(k, f)$ to denote the probability that the cluster is able to produce a correct result under f failures when k replicas are involved in the majority voting, and we have

$$\varphi(k,f) = \frac{\sum_{i=max(\lceil \frac{k+1}{2} \rceil, k-f)}^{min(k,s-f)} \binom{s-f}{i} \binom{f}{k-i} \sum_{j=\lceil \frac{k+1}{2} \rceil}^{i} \binom{i}{j} p^j (1-p)^{i-j}}{\binom{s}{k}} \tag{2}$$

where $\frac{\sum_{i=max(\lceil \frac{k+1}{2} \rceil, k-f)}^{min(k,s-f)} \binom{s-f}{i}\binom{f}{k-j}}{\binom{s}{k}}$ is the probability of choosing k participating voting replicas among which the number of uncompromised replicas is between $max(\lceil \frac{k+1}{2} \rceil, k - f)$ and $min(k, s - f)$.

D. Problem Formulation

Assume there are n clusters in the system and the number of replicas in each cluster is s. The attacker can compromise m replicas in total and chooses h clusters to attack. The defender decides upon the k participating voting replicas in each cluster. We use $T(k, h)$ to denote the expected number of clusters which are able to produce a correct result, based on the analysis above, we have

$$\begin{aligned} T(k,h) = (n-h)\theta(k) + (h-q) \times \varphi(k,f) \\ + q \times \varphi(k, f+1) \end{aligned} \tag{3}$$

where $f = \lfloor \frac{m}{h} \rfloor$, and $q = m \bmod h$.

Our goal is to decide an optimal value k to maximize $T(k, h)$ given that for any k the attacker chooses h that minimizes the value $T(k, h)$. That is

$$\max_k \min_h T(k,h) \tag{4}$$

IV. Determining the Optimal Voting Strategy

For a defender, as he can choose any number of participants to perform majority voting in the cluster with size s, hence the number of possible voting strategies is $d = s$.

The total number of replicas the attacker can successfully compromise is m, and these compromised replicas can be scattered among at least $\lceil \frac{m}{s} \rceil$ clusters. As the number of clusters in the system is n, hence the number of clusters being attacked, h, is in the range of $[\lceil \frac{m}{s} \rceil, n]$. Therefore, the total number of possible attack strategies is $g = n - \lceil \frac{m}{s} \rceil + 1$.

Once the strategy of both defender and attacker is decided, the expected number of surviving clusters $T(k, h)$, i.e., the clusters which produce a correct result under attack, can be calculated according to equation 3.

The gain of the defender is the loss of the attacker, and vice versa. A game in which one player wins what the other player loses is called a two-person zero-sum game [25]. The optimal voting strategy problem can be mapped to the two-person zero-sum game problem.

It has been proved that the two-person zero-sum game is equivalent to the linear programming problem [10]. Therefore, in order to find out the optimal voting strategy, we transform the optimal voting problem into a linear programming problem. In our work, we simply follow the steps proposed in [31] to perform the transformation. Here, we first present the steps in Algorithm 1 and provide the detailed explanation and discussion afterward.

If the defender uses strategy x and the attacker uses strategy y, the expected number of surviving clusters for this pair of strategies is $E(x, y) = \sum_{i=1}^{d} \sum_{j=1}^{g} x_i t_{ij} y_j$ [7]. For any strategy x that the defender chooses, the attacker's goal is to choose an attack strategy y that minimizes the expected number of surviving clusters. In this case, the defender can expect to have at least $\min_y E(x, y)$ surviving clusters. Therefore, the defender aims to select his particular strategy x to maximize $\min_y E(x, y)$.

It has been proved that $\min_y E(x, y) = \min_j \sum_{i=1}^{d} x_i t_{ij}$ [28]. Therefore, no matter what the attacker's strategy is, the defender is assured of obtaining at least $\min_j \sum_{i=1}^{d} x_i t_{ij}$. Let X be the lower bound for each of the j summations, that is $\forall j = 1, 2, \ldots, g$, $\sum_{i=1}^{d} x_i t_{ij} \geq X$. Then, the defender's goal

Algorithm 1 DETERMINE THE OPTIMAL VOTING STRATEGY

Input:

n: total number of clusters in the system;
s: total number of replicas in each cluster;
p: the reliability of each replica;
m: total number of replicas the attacker can compromise;

Output:

x: the defender's optimal voting strategy;
X: the expected number of surviving clusters;

1: $d \leftarrow s$
2: $g \leftarrow n - \lceil \frac{m}{s} \rceil + 1$
3: Create a $d \times g$ matrix $T = (t_{ij})$, where t_{ij} denotes the expected number of surviving clusters if defender chooses its ith voting strategy and attacker chooses its jth attack strategy.
4: Create a d-dimensional vector $x = (x_1, \ldots, x_d)^T$ and a g-dimensional vector $y = (y_1, \ldots, y_g)^T$ to denote the probability of choosing the ith voting strategy by defender, and jth attack strategy by attacker, respectively, where $\sum_{i=1}^{d} x_i = 1$, $\sum_{j=1}^{g} y_j = 1$.
5: Create the equivalent linear programming problem for the optimal voting problem.
6: Use simplex algorithm to solve the linear programming problem, and get x and X.

is equivalent to maximizing X, and we have

$$maximize \;\; X \tag{5}$$

subject to:

$$\sum_{i=1}^{d} x_i = 1 \tag{6}$$

$$\sum_{i=1}^{d} t_{ij} x_i \geq X, \quad \forall j = 1, 2, \ldots, g \tag{7}$$

$$x_i \geq 0, \quad \forall i = 1, 2, \ldots, d \tag{8}$$

The attacker, on the other hand, tries to minimize the expected number of surviving clusters by choosing an optimal attack strategy against the defender's strategies. By using a similar analysis above, the attacker's goal is to minimize $\max_x E(x, y)$, where $\max_x E(x, y) = \max_i \sum_{j=1}^{g} y_j t_{ij}$. Let Y be the upper bound for each of the i summations, that is $\forall i = 1, 2, \ldots, d$, $\sum_{j=1}^{g} y_j t_{ij} \leq Y$, the attacker's goal is equivalent to minimizing Y, and we have

$$minimize \;\; Y \tag{9}$$

subject to:

$$\sum_{j=1}^{g} y_j = 1 \tag{10}$$

$$\sum_{j=1}^{g} t_{ij} y_j \leq Y, \quad \forall i = 1, 2, \ldots, d \tag{11}$$

$$y_j \geq 0, \quad \forall j = 1, 2, \ldots, g \tag{12}$$

After solving these two linear programming problems by using the simplex algorithm proposed in [4], we will get X, Y, x, and y, where both X and Y refer to the expected number of surviving clusters, x and y contain the probability of each strategy taken by the defender and attacker, respectively. Actually, according to Von Neuman's Minimax Theorem [23], for both defender and attacker, if x and y are their optimal strategies, we can have $X = Y$.

If there exists one element in vector $x = (x_1, \ldots, x_d)^T$ which is equal to 1, it means its corresponding strategy is always the best no matter how the attacker's strategy is chosen. A similar conclusion can be made for the attacker if there exists one element in vector $y = (y_1, \ldots, y_g)^T$ which is equal to 1. If no such element exists in the probability vector, it indicates that no deterministic decision can be obtained.

The following example illustrates the process of determining the optimal voting replicas in the cluster.

Example 1 Assume a distributed system has $n = 10$ clusters, and each cluster consists of $s = 25$ replicas, the reliability of each replica is $p = 0.9$. The total number of replicas the attacker can compromise is $m = 25$. For the defender, we assume that only an odd number of replicas is chosen to vote, therefore, the total number of voting strategies is $d = \lceil \frac{s}{2} \rceil = 13$. For the attacker, the total number of attack strategies is $g = n - \lceil m/s \rceil + 1 = 10$.

Table 1 displays all possible outcomes of an attack on 10 clusters with 10 scenarios of attack strategies and 13 possible defense choices. The values in the table represent the expected number of surviving clusters given the defense choice and the attack strategy.

Based on the results in the table, if the attacker attacks 8 out of 10 clusters ($h = 8$), and the defender chooses 13 out of 25 replicas to vote ($k = 13$), then the expected number of clusters which generate a correct result is 9.9716, and obviously, this strategy is not optimal for the attacker because if the defender sticks to the same voting strategy (i.e., choosing 13 out of 25 replicas to vote), then the attacker will prefer to attack 2 out of 10 clusters to decrease the expected number of surviving clusters to 8.6389, rather than attacking 8 out of 10 clusters. On the other hand, if the attacker sticks to its strategy, the defender would like to choose 25 replicas to vote and increase the expected number of surviving clusters to 9.9998. Therefore, for both attacker and defender, their initial strategies of attacking 8 out of 10 clusters and choosing 13 out of 25 replicas to vote is not optimal against each other.

In order to decide the optimal strategy in this example, we set the probability of the defender taking the ith strategy is x_i, the probability of the attacker taking the jth strategy is y_j. After transforming the problem into the linear programming problem, we solve this linear programming problem by using simplex algorithm, and we have $X = Y = 8.7384$, $x = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$, and $y = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0)^T$, which indicates that the fourth defense strategy is the defender's optimal strategy, and that is to choose 7 out of 25 replicas to vote, and the attacker's optimal strategy is the second strategy which is to attack 2 out

TABLE I: The result matrix for defender and attacker

	h = 1	h=2	h = 3	h = 4	h = 5	h = 6	h = 7	h = 8	h = 9	h = 10
k = 1	8.1	8.1	8.1	8.1	8.1	8.1	8.1	8.1	8.1	8.1
k = 3	8.748	8.6216	8.7621	8.8665	8.9417	8.9938	9.0322	9.066	9.0894	9.1076
k = 5	8.923	8.7326	9.0231	9.2195	9.3464	9.4248	9.478	9.5228	9.5514	9.5718
k = 7	8.9754	8.7384	9.1734	9.4364	9.5863	9.6671	9.7173	9.7574	9.781	9.7964
k = 9	8.992	8.7135	9.2868	9.5912	9.7407	9.8099	9.8493	9.8788	9.8951	9.9048
k = 11	8.9973	8.6785	9.3856	9.7081	9.8418	9.894	9.9215	9.9407	9.9508	9.9563
k = 13	8.9991	8.6389	9.4769	9.7971	9.9066	9.9425	9.9601	9.9716	9.9774	9.9803
k = 15	8.9997	8.5957	9.5629	9.8633	9.9467	9.9696	9.9802	9.9867	9.9898	9.9913
k = 17	8.9999	8.5482	9.6437	9.9111	9.9706	9.9843	9.9904	9.9939	9.9955	9.9962
k = 19	9	8.4949	9.7182	9.9441	9.9842	9.9921	9.9954	9.9972	9.998	9.9983
k = 21	9	8.433	9.7846	9.966	9.9918	9.9961	9.9979	9.9988	9.9991	9.9993
k = 23	9	8.357	9.8413	9.9799	9.9958	9.9981	9.999	9.9994	9.9996	9.9997
k = 25	9	8.2542	9.8873	9.9885	9.9979	9.9991	9.9996	9.9998	9.9998	9.9999

of 10 clusters. For this case, the expected number of surviving clusters is 8.7384.

In order to show that their strategies are optimal against each other, we will traverse all the defender's strategies and check whether a better defense strategy is available when the attacker's optimal strategy is fixed. The attacker's optimal strategy can also be checked in a similar way. Based on the result table, we can see that when attacker chooses the second strategy, that is $h = 2$, no better defense strategy for defender is available, and when defender chooses the second strategy, that is $k = 7$, no better attack strategy for attacker is available, either. Therefore, for both attacker and defender, their strategies are optimal against each other.

V. Experimental Results

In this section, we describe three sets of experiments. The first set of experiments is to investigate the relationship between the defender's optimal voting strategy, the number of replicas in each cluster, and the total number of replicas the attacker can compromise. The second set of experiments shows the relationship between the defender's optimal voting strategy and a replica's reliability. The third set of experiments shows how the number of compromised replicas and replica's reliability affect the maximum number of surviving clusters.

For the first set of experiments, we assume that there are 10 clusters, the number of replicas in each cluster is 17, and the reliability of each replica is 0.9. That is, we have $n = 10, s = 17, p = 0.9$. In addition, we assume the defender chooses odd number of replicas to vote. When the total number of replicas the attacker can compromise increases from 1 to 45, Figure 2a shows the change of the optimal voting strategy.

In Figure 2a, the length of the bar indicates the probability of the strategy the defender will take. For example, when $m = 5$, the optimal voting strategy is to choose $k = 17$ replicas to vote because the length of the bar for that strategy is 1, and the length of the bar for other strategies is 0.

When $1 \leq m \leq 7$, the defender's optimal strategy is to choose all the replicas in the cluster to vote. This is because the number of compromised replicas is smaller than the majority in each cluster. For this case, according to equation 1 and equation 2, no matter whether the cluster is attacked or not, the more voters that are involved, means higher reliability for the defender. Therefore, it is preferable to choose all the replicas in the cluster to vote.

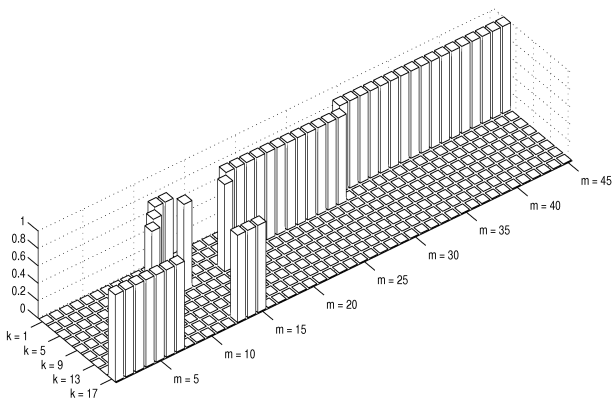
When $8 \leq m \leq 12$, the optimal voting strategy changes dramatically. This is because the attacker attacks only one cluster when $m < 15$ (this is shown in Figure 4a). Therefore, more than half of replicas in that attacked cluster are compromised. If the defender continues to choose all the replicas to vote, the unattacked clusters will have the highest reliability, but the reliability of that attacked cluster, is nevertheless, close to 0. Therefore, the defender must choose a certain number of voters to reach a balance between attacked cluster and unattacked clusters.

However, when $13 \leq m \leq 15$, the defender's optimal strategy is to choose all the replicas to vote, which indicates that the gain of choosing a larger number of voters in the unattacked clusters is larger than the loss of choosing a smaller number of voters in the attacked cluster.

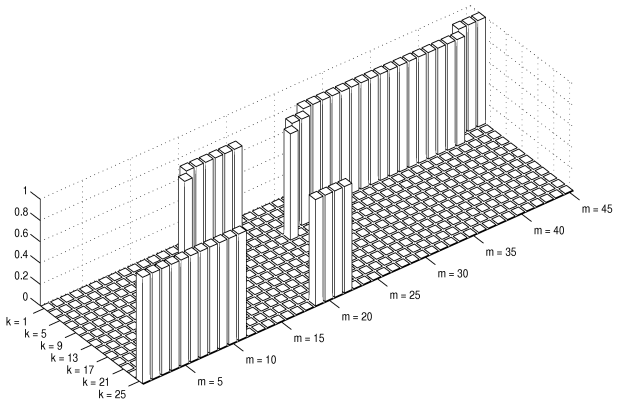
When m becomes larger (i.e., $m > 15$), the defender will choose fewer replicas to vote. The reason lies in the fact that the number of compromised replicas in the attacked clusters are larger than the majority of the total number of replicas. Although higher reliability will be obtained in the unattacked clusters if more replicas are chosen to vote, the gain is smaller than the loss in the attacked clusters. Therefore, the defender shall choose a small number of replicas to vote.

A similar conclusion can be made for the case in which the number of replicas in each cluster is $s = 25$, and the optimal voting strategy is shown in Figure 2b. The main difference between these two cases (that is, $s = 17$ and $s = 25$) is the voting strategy change points. The main reason for the difference lies in the fact that when s becomes larger, the cluster can tolerate more component failures.

The second set of experiments is performed to investigate the relationship between a defender's optimal voting strategy and replicas' reliability. We also assume the defender will choose an odd number of replicas to perform majority voting. Here we set $n = 10$ and $s = 25$. The replicas' reliability, p, changes from 0.9, 0.7 to 0.4. The results are shown in Figure 2b, Figure 3a, and Figure 3b, respectively. From these results, we can see that when the replicas' reliability is over 0.5, the optimal strategy changes with m. However, when the

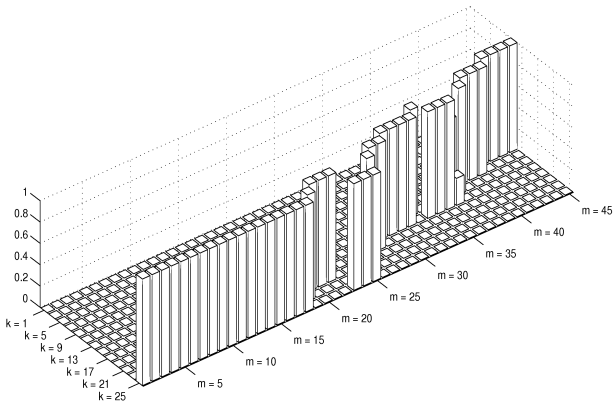


(a) System configuration (n = 10, s = 17, p = 0.9)

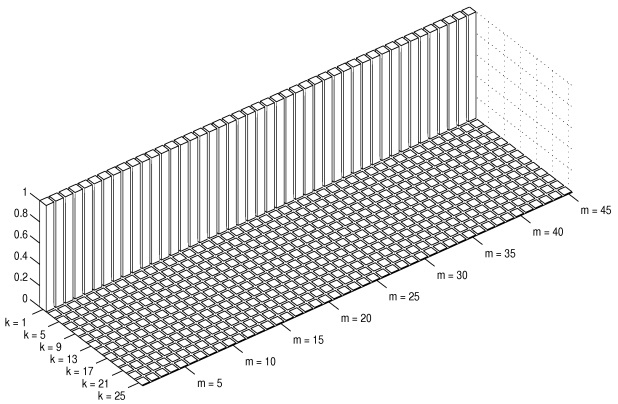


(b) System configuration (n = 10, s = 25, p = 0.9)

Fig. 2: The relationship between optimal voting strategy k and the total number of replicas m the attacker can compromise.

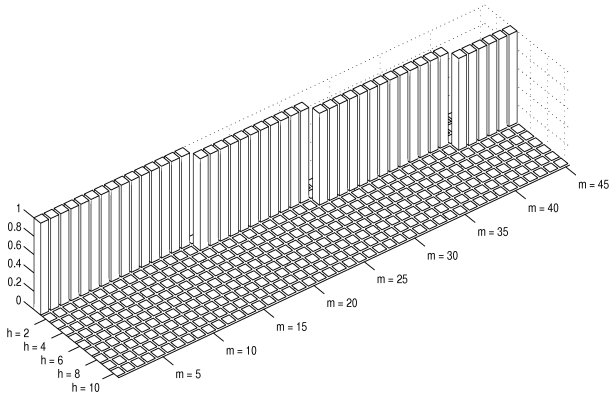


(a) System configuration (n = 10, s = 25, p = 0.7)

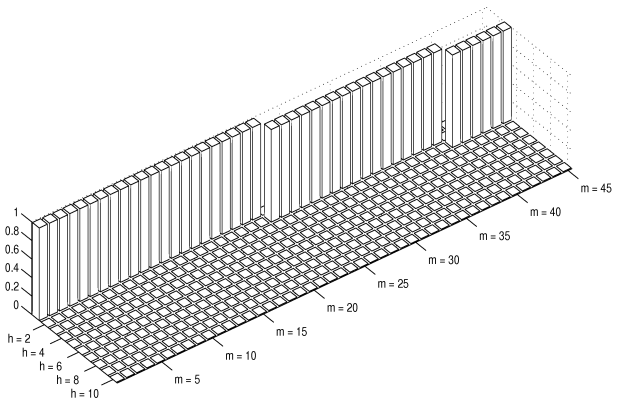


(b) System configuration (n = 10, s = 25, p = 0.4)

Fig. 3: The relationship between optimal voting strategy k and the reliability of the replicas p.



(a) System configuration (n = 10, s = 17, p = 0.9)



(b) System configuration (n = 10, s = 25, p = 0.9)

Fig. 4: The relationship between optimal attack strategy h and the total number of replicas m the attacker can compromise.

replicas' reliability is under 0.5, the optimal voting strategy is always to choose one replica to vote. This is because when the replica's reliability is under 0.5, the more replicas that are involved, lowers the reliability in both the attacked and unattacked clusters.

If there is no strategy whose probability is equal to 1 (see Figure 3a when $m = 37$), this indicates that there is no deterministic strategy for the defender. For this case, the defender can choose any voting strategy whose probability is not equal to 0. However, in this case, the maximum number of surviving clusters may not be obtained.

The third experiment shows the relationship between the number of surviving clusters, the number of replicas the attacker can compromise, and the replicas' reliability. In this experiment, we have $n = 10$, $s = 25$. Figure 5 shows that the number of surviving clusters increases when the replicas' reliability increases, and the number of surviving clusters decreases when the number of replicas the attacker can compromise becomes larger. It is worth noting that when the replicas' reliability is under 0.5, the percentage between the expected number of surviving clusters and the total number of clusters is always under 50% even when $m = 0$. This indicates that having more replicas in a voting scheme with low reliability does not improve a cluster's reliability.

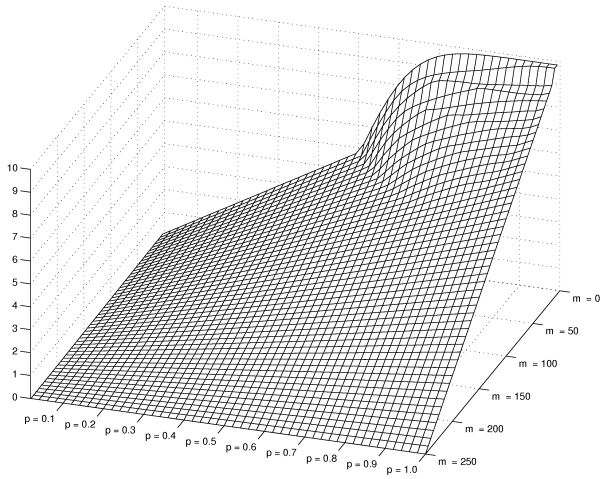


Fig. 5: The relationship between the maximum number of surviving clusters T, the replica's reliability p, and the number of compromised replicas m.

VI. Conclusion

For different voting algorithms, their fault-and-attack tolerance performance differs when the reliability of the replicas and the number of compromised replicas change. In this paper, we aimed to find the optimal number of participating voting replicas in a majority voting scheme when the system is under rational attacks. We began by providing the system model and assumptions, then the voting problem was analyzed and formulated. We also presented a solution to decide the optimal number of voting replicas to maximize the expected number of clusters that can produce a correct result under attack. Finally, three sets of experiments are performed to investigate the relationship between the voting strategy, the attacker strategy, and the system reliability.

Moreover, in this paper, the replicas are assumed to be homogeneous. For the future work, we will study the system reliability when both replicas' reliability and their voting weights are heterogeneous, which indicates that we will decide how many replicas should be chosen from two different replica groups in the cluster to maximize the number of surviving clusters.

References

[1] V. M. Bier, A. Nagaraj, and V. Abhichandani. Protection of simple series and parallel systems with components of different values. *Reliability Engineering & System Safety*, 87(3):315 – 323, 2005.

[2] J. J. Bloch, D. S. Daniels, and A. Z. Spector. A weighted voting algorithm for replicated directories. *J. ACM*, 34:859–909, Oct. 1987.

[3] L. Chen and A. Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. In *Digest of Papers FTCS-8: Eighth Annual International Conference on Fault Tolerant Computing*, pages 3 – 9, June 1978.

[4] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[5] D. Davcev. A dynamic voting scheme in distributed systems. *Software Engineering, IEEE Transactions on*, 15(1):93 –97, Jan. 1989.

[6] D. E. Eckhardt and L. D. Lee. Fundamental differences in the reliability of n-modular redundancy and redundancy and n-version programming. *J. Syst. Softw.*, 8(4):313–318, 1988.

[7] S. I. Gass. *Linear programming: methods and applications (5th ed.)*. McGraw-Hill, Inc., New York, NY, USA, 1984.

[8] D. K. Gifford. Weighted voting for replicated data. In *Proceedings of the seventh ACM symposium on Operating systems principles*, SOSP '79, pages 150–162, New York, NY, USA, 1979. ACM.

[9] B. Hardekopf, K. Kwiat, and S. Upadhyaya. A decentralized voting algorithm for increasing dependability in distributed systems. In *5th World Multiconference on Systemic, Cybernetics and Informatics (SCI2001*, 2001.

[10] H. Kuhn and A. Tucker. *Contributions to the Theory of Game*, volume 1. Princeton University Press, 1950.

[11] K. Kwiat, A. Taylor, W. Zwicker, D. Hill, S. Wetzonis, and S. Ren. Analysis of binary voting algorithms for use in fault-tolerant and secure computing. In *Computer Engineering and Systems (ICCES), 2010 International Conference on*, pages 269 –273, 2010.

[12] G. Latif-Shabgahi, J. Bass, and S. Bennett. A taxonomy for software voting algorithms used in safety-critical systems. *Reliability, IEEE Transactions on*, 53(3):319 – 328, Sep. 2004.

[13] G. Levitin. Optimal multilevel protection in series-parallel systems. *Reliability Engineering & System Safety*, 81(1):93 – 102, 2003.

[14] G. Levitin, Y. Dai, M. Xie, and K. L. Poh. Optimizing survivability of multi-state systems with multi-level protection by multi-processor genetic algorithm. *Reliability Engineering & System Safety*, 82(1):93 – 104, 2003.

[15] G. Levitin and K. Hausken. Protection vs. redundancy in homogeneous parallel systems. *Reliability Engineering & System Safety*, 93(10):1444 – 1451, 2008.

[16] G. Levitin and K. Hausken. False targets efficiency in defense strategy. *European Journal of Operational Research*, 194(1):155–162, April 2009.

[17] G. Levitin and K. Hausken. Redundancy vs. protection vs. false targets for systems under attack. *Reliability, IEEE Transactions on*, 58(1):58 –68, March 2009.

[18] G. Levitin and A. Lisnianski. Optimal separation of elements in vulnerable multi-state systems. *Reliability Engineering & System Safety*, 73(1):55 – 66, 2001.

[19] G. Levitin and A. Lisnianski. Optimizing survivability of vulnerable series-parallel multi-state systems. *Reliability Engineering & System Safety*, 79(3):319 – 331, 2003.

[20] L. Mancini and M. Koutny. Formal specification of n-modular redundancy. In *CSC '86: Proceedings of the 1986 ACM fourteenth annual conference on Computer science*, pages 199–204, 1986.

[21] M. A. McQueen, W. F. Boyer, M. A. Flynn, and G. A. Beitel. Time-to-compromise model for cyber risk reduction estimation. In *First Workshop on Quality of Protection*, 2005.

[22] M. A. McQueen, W. F. Boyer, M. A. Flynn, and G. A. Beitel. Time-to-compromise model for cyber risk reduction estimation. *Quality of Protection*, 23:49–64, 2006.

[23] J. V. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[24] L. Nordmann and H. Pham. Weighted voting systems. *Reliability, IEEE Transactions on*, 48(1):42 –49, March 1999.

[25] T. Raghavan. Zero-sum two-person games. In R. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 2, pages 735–768. Elsevier, May 1994.

[26] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4:180–209, June 1979.

[27] Z. Tong and R. Kain. Vote assignments in weighted voting mechanisms. *Computers, IEEE Transactions on*, 40(5):664 –667, May 1991.

[28] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, second edition, 2001.

[29] L. Wang, Y. Leiferman, S. Ren, K. Kwiat, and X. Li. Improving complex distributed software system availability through information hiding. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 452–456, New York, NY, USA, 2010. ACM.

[30] L. Wang, S. Ren, K. Yue, and K. Kwiat. Optimal resource allocation for protecting system availability against random cyber attacks. In *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, volume 1, pages 477 –482, March 2011.

[31] P. Zafra. *Linear Programming and Two-Person Zero-Sum Games*. Wiley Encyclopedia of Operations Research and Management Science, Feb. 2011.