6-2015

Replica Placement for Availability in the Worst Case

Peng Li

Debin GAO
Singapore Management University, dbgao@smu.edu.sg

Mike Reiter

DOI: https://doi.org/10.1109/ICDCS.2015.67

2015 IEEE 35th International Conference on Distributed Computing Systems

Replica Placement for Availability in the Worst Case

Peng Li
VMWare
Palo Alto, CA, USA

Debin Gao
Singapore Management University
Singapore

Michael K. Reiter
University of North Carolina
Chapel Hill, NC, USA

***Abstract*—We explore the problem of placing object replicas on nodes in a distributed system to maximize the number of objects that remain available when node failures occur. In our model, failing (the nodes hosting) a given threshold of replicas is sufficient to disable each object, and the adversary selects which nodes to fail to minimize the number of objects that remain available. We specifically explore placement strategies based on combinatorial structures called t-packings; provide a lower bound for the object availability they offer; show that these placements offer availability that is c-competitive with optimal; propose an efficient algorithm for computing combinations of t-packings that maximize their availability lower bound; and provide parameter selection strategies to concretely instantiate our schemes for different system sizes. We compare the availability offered by our approach to that of random replica placement, owing to the popularity of the latter approach in previous work. After quantifying the availability offered by random replica placement in our model, we show that our combinatorial strategy yields placements with better availability than random replica placement for many realistic parameter values.**

I. Introduction

Consider the problem of deploying *replicas* of *objects* onto a system of physical *nodes* so as to ensure the survival of as many objects as possible when node failures occur. This general problem occurs in practice in many computing contexts: the "objects" might be virtual machines, files, or servers, and the "replicas" could be whole object copies or merely components used in the implementation of the object. The survival of an object is achieved provided that fewer than a given threshold number of its replicas were placed on the nodes that fail. This threshold might range from all of the object replicas to only a few. The question we address in this paper is: How should the object replicas be placed on the nodes (aside from the obvious requirement that the replicas of an object all be placed on different nodes)?

Upon encountering this problem for the first time, it might not be immediately obvious that the placement matters. But consider the possibility that all of the failed nodes host replicas of mostly the same objects. This scenario might fail objects that require many replica failures to do so, but it fails fewer objects than it otherwise could if each object fails when only a few of its replicas do. Alternatively, suppose the failed nodes host replicas of mostly different objects. Then, many objects might fail if only few object replica failures suffice to fail each object, but fewer objects might fail if many replica failures per object are required. As this contrast suggests, the placement certainly matters and depends not only on the number of nodes, the number of objects, the number of node faults, and the replicas per object, but also the number of an object's replicas' failures that prove fatal to the object.

We are not the first to study the problem of object replica placement for availability (see Sec. II for a discussion of related work), but to our knowledge, our treatment is novel in at least two ways. First, we consider a *worst-case* adversary that fails a specified number of nodes *with knowledge of how object replicas were placed on nodes*, so as to maximize the number of objects failed. This is in contrast to failures that occur probabilistically, for example. Second, by decoupling the number of replicas per object from the number of replica failures that disable each object, our framework allows for treatment of a wide variety of object configurations, such as objects that are accessed using majority quorums (e.g., [19], [17]) so that a majority of available replicas is required for the object to survive, or objects for which even just a single surviving replica suffices to keep the object available (e.g., in the primary-backup(s) approach [7]).

Our study is also general by virtue of what it leaves unspecified. While we label nodes, replicas and objects as "failed" or not, we remain agnostic to the fault model [36] (crash, Byzantine, etc.). Similarly, the protocols run among object replicas or for objects to interact with others are not our concern here. Rather, we simply assume that a node failure fails all of the object replicas it hosts, and that an object fails once a specified number of its object replicas do. We also do not constrain the means by which the adversary fails the nodes it chooses to, whether that be disabling them by denial-of-service attacks, leveraging vulnerabilities in object replicas they host, physically attacking the nodes, etc.

In this context, we make the following contributions:

- We study the viability of block designs for replica placements. Specifically, we first leverage t-packings (e.g., see [26]), a relaxation of Steiner systems, as a replica placement strategy. We provide a lower bound for the availability of these replica placements and show that they already offer availability that is c-competitive with optimal placements (for a factor c that we specify). This suggests that t-packings are a useful starting point for constructing placement strategies.
- We develop a placement strategy that improves on the use of t-packings in isolation by combining them. We present an efficient algorithm to compute combinations of individual t-packings that maximize our lower bound on availability (among any such combination) for a given number of node failures. We further show that for a range of practical parameter values, the placement strategy derived for a given target number of failures provides good availability even for different numbers of failures.
- We develop as our primary comparison point a placement strategy of randomly placing replicas on nodes subject to a

1063-6927/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDCS.2015.67

load-balancing requirement, owing to the popularity of this strategy in previous work. We characterize availability for this placement strategy in our adversarial model, and then we develop an expression for the limit of this measure as the number of objects grows. We further show that this limit already closely reflects reality for practical parameter values and relatively small numbers of objects, allowing it to be used as a basis to compare to the availabilities offered by our strategies based on t-packings. In this way, we show that our constructions based on t-packings provide better availability for ranges of practical parameter values than does random replica placement.

As discussed above, the replica placement strategies that we explore build from t-packings, and some of our analysis depends on use of *maximum* t-packings (also called t-designs). Based on current knowledge of t-designs (which we briefly survey in Sec. III-C), we limit our attention to replication scenarios involving up to five replicas per object. Fortunately, this decision is not limiting for practical replication scenarios in data centers: VM replication for fault tolerance typically uses two (e.g., [38]) and many file systems default to three or four replicas per file or related structure (as in GFS [18], Hadoop [37], and FARSITE [2]).

II. Related Work

Replica placement for availability (or durability) has been extensively studied in various fields (e.g., [5], [15], [4], [41], [31], [39], [3], [34]), sometimes in conjunction with other concerns. All related work we have located focuses on leveraging node failure distributions, especially their independence and/or heterogeneity as would be common in peer-to-peer storage and computing, for example. Here we make no assumptions about node failure distributions, allowing them to be controlled by an arbitrary adversary constrained only by the number of nodes he can fail. This renders our analysis both simpler in many cases and, at the same time, very general.

We nevertheless draw from this work where possible. Notably, Yu and Gibbons [39] explored the following question: If each node fails independently with fixed probability and if all replicas of an object must fail for the object to fail, then what placement strategy offers the highest probability of success for operations involving multiple objects, a given number of which must be available for the operation to succeed? Their finding that we most directly leverage here is their identification of random replica placements as offering close to the best probability of operation success when an operation can tolerate some object failures. Together with the widespread use and empirical study of random placements (e.g., [35], [2], [4], [18], [40], [24]), this finding motivates our choice of random replica placement as a comparison point for our proposed placement strategies. That said, for drawing this comparison we need to develop our own analysis of the availability of random placements, since we focus on a worst-case adversary that can choose which nodes to fail; this analysis might be of interest in its own right. Our work also differs from Yu and Gibbons' in that we do not consider multi-object operations, asking instead only how many objects remain available, but we do so while permitting an object to remain available only if a specified number of its replicas survive (versus just one of them). Note that equating our "objects" to their "operations" and our "replicas" to their "objects" (each with only one replica) does not yield the same problem—even setting aside our different adversarial models—since replicas of the same object in our case must be placed on different nodes, while their objects do not.

As discussed in Sec. I, the cornerstone of the replica placement strategy we develop is a t-packing. To our knowledge, we are the first to explore the use of t-packings for replica placement in distributed systems. That said, such block designs have found application in several diverse domains, as surveyed elsewhere (e.g., [11], [9], [33]). The most conceptually related use of block designs to ours is their use in constructing quorum systems (e.g., [29]). Quorum systems, however, must intersect, whereas we have no such requirement here for object placements, a fact that we leverage.

III. Overlap-Based Placement Strategies

The strength of random replica placement in diminishing the likelihood that random node failures will fail many objects (see Sec. II) derives from it inducing low *inter-object correlation* [39], a measure that reflects the overlaps of objects' replica placements. However, random placement induces small overlaps only probabilistically, allowing the possibility that *targeted* node failures could still impact many objects. In this section we explore "overlap-based" placement strategies that manage these overlaps explicitly. We will return to analyzing the impact of targeted node failures on random placements in Sec. IV and compare to our overlap-based strategies there.

b	The number of objects
r	The number of replicas per object
s	The number of an object's replicas whose failure fails the object; $1 \leq s \leq r$
n	The number of nodes
k	The number of failed nodes; $s \leq k < n$
π	A placement
$\mathcal{O}$	The set of all objects; $\lvert\mathcal{O}\rvert = b$
$\mathcal{N}$	The set of all nodes; $\lvert\mathcal{N}\rvert = n$

Fig. 1: Notation

Before continuing, we first define some notation used in the rest of this document (see Fig. 1). We presume a system of n nodes denoted by the set $\mathcal{N}$ ($|\mathcal{N}| = n$). These nodes will host a set $\mathcal{O}$ of b objects ($|\mathcal{O}| = b$), each replicated r times. This hosting is represented by a *placement* $\pi : \mathcal{O} \rightarrow \mathbf{2}^{\mathcal{N}}$, where $\mathbf{2}^{\mathcal{N}}$ is the power set of $\mathcal{N}$. Specifically, for each $\mathsf{obj} \in \mathcal{O}$, $\pi(\mathsf{obj})$ is a subset of $\mathcal{N}$ of size $|\pi(\mathsf{obj})| = r$ that indicates the nodes on which replicas of obj are located. We use k to denote the number of nodes that fail. If $\mathcal{K} \subseteq \mathcal{N}$ is the set of k failed nodes, then an object obj is said to fail if and only if $|\pi(\mathsf{obj}) \cap \mathcal{K}| \geq s$. This gives rise to the following natural definition of the availability of a placement π.

Definition 1: For any fixed placement π, let $\mathit{Avail}(\pi)$ denote the number of available objects, minimized over all sets $\mathcal{K}$ of (potentially failed) nodes where $|\mathcal{K}| = k$. In other words,

$$\mathit{Avail}(\pi) = \min_{\substack{\mathcal{K} \subseteq \mathcal{N}: \\ |\mathcal{K}| = k}} |\{\mathsf{obj} \in \mathcal{O} : |\pi(\mathsf{obj}) \cap \mathcal{K}| < s\}|$$

A. The Simple(x, λ) Placement Strategy

Our intuition for developing a replica placement strategy so as to maximize availability is simply to limit the number of objects whose replicas overlap on the same nodes "too much." This intuition is captured in the Simple(x, λ) strategy, which limits overlaps of more than x nodes to at most λ objects. We limit our attention to $x < s$, since once $x \geq s$, arbitrarily many objects can overlap on s nodes in a Simple(x, λ) placement, meaning that failures of those nodes could fail arbitrarily many objects.

Definition 2: The Simple(x, λ) placement strategy locates object replicas on nodes using a placement π so that for all $\mathcal{N}' \subseteq \mathcal{N}$ where $|\mathcal{N}'| = x+1$ and all $\mathcal{O}' \subseteq \mathcal{O}$, if $\mathcal{N}' \subseteq \pi(\mathsf{obj})$ for every $\mathsf{obj} \in \mathcal{O}'$, then $|\mathcal{O}'| \leq \lambda$.

In words, Definition 2 says that no $(x+1)$-subset $\mathcal{N}'$ of nodes can host (replicas of) more than λ objects in common. So, for example, if $\lambda = 1$, then the replicas of any two objects can overlap on at most x nodes.

It is important to note that a Simple(x, λ) placement exists only for limited values of b, once n and r are fixed. Specifically, borrowing results from design theory—where a Simple(x, λ) is otherwise known as a $(x+1)$-(n, r, λ)-*packing* (e.g., [26])—we have:

Lemma 1 (e.g., [26]): A Simple(x, λ) placement exists only if $b \leq \left\lfloor \lambda \binom{n}{x+1} / \binom{r}{x+1} \right\rfloor$.

While $b \leq \left\lfloor \lambda \binom{n}{x+1} / \binom{r}{x+1} \right\rfloor$ is necessary for a Simple(x, λ) placement, it is not sufficient. To achieve a sufficient condition, we select an $n_x \leq n$ and a μ_x that divides λ (i.e., $\mu_x \mid \lambda$). The values n_x and μ_x are chosen so that $\mu_x \binom{n_x}{x+1} / \binom{r}{x+1}$ is integral and, moreover, a Simple(x, μ_x) placement exists for any $b \leq \mu_x \binom{n_x}{x+1} / \binom{r}{x+1}$ objects. Then, a Simple(x, λ) placement on n_x nodes can be obtained by "copying" the Simple(x, μ_x) placement λ/μ_x times.

Observation 1: If there exist an $n_x \leq n$ and a $\mu_x \mid \lambda$ so that a Simple(x, μ_x) placement exists for all $b \leq \mu_x \binom{n_x}{x+1} / \binom{r}{x+1}$, then a Simple($x$, λ) placement exists for all $b \leq \lambda \binom{n_x}{x+1} / \binom{r}{x+1}$.

Observation 2: Placing replicas on only $n_x \leq n$ nodes can lead to a load-imbalanced system, but only slightly if we can find a suitable $n_x \approx n$. If we cannot, then we can instead identify values $n_{x1}, \ldots, n_{xm}$ such that $\sum_{i=1}^{m} n_{xi} \leq n$ but $\sum_{i=1}^{m} n_{xi} \approx n$, and then extend the results below to account for building a Simple(x, λ) placement on $\sum_{i=1}^{m} n_{xi}$ nodes for any $b \leq \sum_{i=1}^{m} \lambda \binom{n_{xi}}{x+1} / \binom{r}{x+1}$ objects from a Simple(x, λ) placement on each chunk of n_{xi} nodes.

The extension in Observation 2 is straightforward but tedious, and so we defer its discussion to Sec. III-C. For now, we simply assume that a suitable n_x and μ_x exist and can be found to support Observation 1. We also adopt the convention that, given n_x, μ_x, r, s, and b, λ is chosen minimally, so that

$$(\lambda - \mu_x) \frac{\binom{n_x}{x+1}}{\binom{r}{x+1}} < b \leq \lambda \frac{\binom{n_x}{x+1}}{\binom{r}{x+1}} \tag{1}$$

We now briefly characterize the availability of Simple(x, λ) placements, to justify their use as a building block for a more useful placement strategy in Sec. III-B. The key observation in characterizing the availability of Simple(x, λ) placements is that the availability can be lower-bounded by applying Lemma 1 to packing s-sized sets of replicas into the k failed nodes, as shown in the following lemma. Due to space limitations, the proofs of all results have been deferred to Appendix B.

Lemma 2: For any Simple(x, λ) placement π, *Avail*(π) $\geq$ *lbAvail*$^{\mathsf{si}}(x, \lambda)$ where

$$\textit{lbAvail}^{\mathsf{si}}(x, \lambda) = b - \left\lfloor \lambda \frac{\binom{k}{x+1}}{\binom{s}{x+1}} \right\rfloor \tag{2}$$

lbAvail$^{\mathsf{si}}(x, \lambda)$ is a tight lower bound for only some parameter values, as indicated in Fig. 2. In this figure, *Avail*(π) was calculated explicitly after placing objects according to a Simple(x, λ) placement π and then simulating the worst k failures.

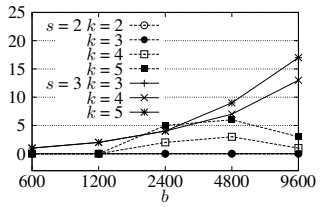


Fig. 2: *Avail*(π) − *lbAvail*$^{\mathsf{si}}(x, \lambda)$ for $n = 71$, $x = 1$, and $r = 3$

This lower bound for *Avail*(π), together with Eqn. 1, permits us to relate *Avail*(π) to the availability of *any* placement π'—and so, in particular, the placement offering the best availability.

Theorem 1: Let $c = \left[1 - \frac{\binom{r}{x+1}\binom{k}{x+1}}{\binom{n_x}{x+1}\binom{s}{x+1}}\right]^{-1}$ and $\alpha = c\mu_x \frac{\binom{k}{x+1}}{\binom{s}{x+1}}$. If $\binom{r}{x+1}\binom{k}{x+1} < \binom{n_x}{x+1}\binom{s}{x+1}$ and so $c > 1$, then for any number b of objects, any Simple(x, λ) placement π, and any other placement π',

$$\textit{Avail}(\pi') < c \cdot \textit{Avail}(\pi) + \alpha$$

In this respect, Simple(x, λ) placements are "c-competitive" (c.f., [6]) with placements yielding the best availability, for any number b of objects (while other parameters, except λ, are held constant).

To see an illustration of Theorem 1, suppose that $s = r$ so that $\binom{r}{x+1}$ and $\binom{s}{x+1}$ cancel. Then,

$$1 - \frac{\binom{r}{x+1}\binom{k}{x+1}}{\binom{n_x}{x+1}\binom{s}{x+1}} = 1 - \frac{k(k-1)\cdots(k-x)}{n_x(n_x-1)\cdots(n_x-x)} \geq 1 - \left(\frac{k}{n_x}\right)^{x+1}$$

So, for example, if $\left(\frac{k}{n_x}\right)^{x+1} \approx 0.2$, then under these conditions, the availability of a Simple(x, λ) placement converges to $\approx 80\%$ of what any placement could achieve as $b \longrightarrow \infty$. On the other hand, under other conditions (such as when s is small relative to r), this constant factor can be less favorable.

B. The Combo($\langle \lambda_x \rangle_{x \in [s]}$) Placement Strategy

The previous section illustrated the potential utility of Simple(x, λ) placements, but we stopped short of suggesting

exactly how to select x. To see why this may not be straightforward, consider a fixed n, r, s, and k, but consider increasingly large values of b. On the one hand, if x is held constant, then the value λ must grow linearly with b, due to Eqn. 1. This, however, implies that the (lower bound on) availability in Lemma 2 also *diminishes* linearly. On the other hand, if x is increased so that λ need not be, then this increases the values of b that can be accommodated exponentially (assuming each $n_x \approx n$ and $r \ll n$); to accommodate some values of b, though, this huge increase is unnecessary and results in a larger penalty to availability than increasing λ would have.

Let $[s] = \{0, 1, \ldots, s-1\}$ and so $\langle \lambda_x \rangle_{x \in [s]} = \langle \lambda_0, \ldots, \lambda_{s-1} \rangle$. In this section we develop a new placement strategy, called $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$, that provides us the flexibility to tune parameters $\langle \lambda_x \rangle_{x \in [s]}$ corresponding to the possible values of $x \in [s]$, to best match a given b. That is, $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ takes a value λ_x corresponding to each $x \in [s]$, subject to the constraint

$$b \leq \sum_{x=0}^{s-1} \lambda_x \frac{\binom{n_x}{x+1}}{\binom{r}{x+1}} \tag{3}$$

and then divides the objects over placements $\mathsf{Simple}(0, \lambda_0), \ldots, \mathsf{Simple}(s-1, \lambda_{s-1})$. Eqn. 3 ensures that $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ can place all b objects, since each $\mathsf{Simple}(x, \lambda_x)$ placement can accommodate $\lambda_x \binom{n_x}{x+1} / \binom{r}{x+1}$ of them (see Observation 1).

Definition 3: A $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement strategy locates object replicas on nodes by placing up to $\lambda_x \binom{n_x}{x+1} / \binom{r}{x+1}$ objects according to a $\mathsf{Simple}(x, \lambda_x)$ placement for each $x \in [s]$.

Lemma 3: For any $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement π, $\mathit{Avail}(\pi) \geq \mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})$ where

$$\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]}) = b - \sum_{x=0}^{s-1} \left\lfloor \lambda_x \frac{\binom{k}{x+1}}{\binom{s}{x+1}} \right\rfloor \tag{4}$$

1) Computing a $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ Placement to Maximize $\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})$: To construct a $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement that achieves high availability for a given value of k, we take it as our goal to select $\langle \lambda_x \rangle_{x \in [s]}$ so as to maximize the lower bound $\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})$ subject to Eqn. 3. This problem lends itself to the following recurrence for $\mathsf{lbav}(x', b')$, which denotes this maximum value of $\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [x'+1]})$ for b' objects placed using placements $\mathsf{Simple}(0, \lambda_0)$, …, $\mathsf{Simple}(x', \lambda_{x'})$ under any selection of $\lambda_0, \ldots, \lambda_{x'}$.

$$\forall x', \forall b' \leq 0 : \ \mathsf{lbav}(x', b') = 0 \tag{5}$$

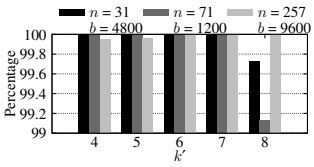
$$\forall b' > 0 : \ \mathsf{lbav}(0, b') = \max\left\{0, b' - \left\lfloor \left(\left\lceil \frac{b'}{\mu_0} \frac{r}{n_0} \right\rceil \mu_0 \right) \frac{k}{s} \right\rfloor \right\} \tag{6}$$

$$\forall x' > 0, \forall b' > 0 : \ \mathsf{lbav}(x', b') = \max_{0 \leq d \leq \left\lceil \frac{b'}{\mu_{x'}} \frac{\binom{r}{x'+1}}{\binom{n_{x'}}{x'+1}} \right\rceil} \left\{ \begin{array}{l} \mathsf{lbav}\left(x' - 1, b' - d\mu_{x'} \frac{\binom{n_{x'}}{x'+1}}{\binom{r}{x'+1}}\right) + \\ \min\left\{b', d\mu_{x'} \frac{\binom{n_{x'}}{x'+1}}{\binom{r}{x'+1}}\right\} - \left\lfloor d\mu_{x'} \frac{\binom{k}{x'+1}}{\binom{s}{x'+1}} \right\rfloor \end{array} \right\} \tag{7}$$

In words, Eqn. 5 encodes that zero availability can be offered if there are no objects ($b' \leq 0$). Eqn. 6 encodes that when $x' = 0$ the availability that can be achieved for $b' > 0$ objects is that resulting from setting $\lambda_0 = \left\lceil (b'/\mu_0) \binom{r}{1} / \binom{n_0}{1} \right\rceil \mu_0 = \left\lceil (b'/\mu_0)(r/n_0) \right\rceil \mu_0$ and using Lemma 2 (or simply 0 if this value turns out to be negative). Finally, Eqn. 7 encodes that when $x' > 0$, availability can be maximized by considering every option for $\lambda_{x'} = d\mu_{x'}$ and, for each option, adding the availability contributed by this setting of $\lambda_{x'}$ (i.e., $\min\left\{b', \lambda_{x'} \binom{n_{x'}}{x'+1} / \binom{r}{x'+1}\right\} - \left\lfloor \lambda_{x'} \binom{k}{x'+1} / \binom{s}{x'+1} \right\rfloor$) to the availability that can be achieved for the remaining $b' - \lambda_{x'} \binom{n_{x'}}{x'+1} / \binom{r}{x'+1}$ objects by optimally setting $\langle \lambda_x \rangle_{x \in [x']}$ (i.e., $\mathsf{lbav}\left(x' - 1, b' - \lambda_{x'} \binom{n_{x'}}{x'+1} / \binom{r}{x'+1}\right)$).

As such, $\mathsf{lbav}(s-1, b)$ for a given number k of failed nodes is the maximum $\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})$ that a $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement π can achieve. This recurrence gives rise to the natural dynamic programming algorithm (see [12, Ch. 6]) for choosing $\langle \lambda_x \rangle_{x \in [s]}$ that runs in $O(sb)$ time, treating all other parameters as constants.

2) Sensitivity to Choice of k: A potential disadvantage of the $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement strategy, or more precisely of the algorithm described in Sec. III-B1 to configure $\langle \lambda_x \rangle_{x \in [s]}$ to maximize $\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})$, is that it does so only for the specified value k. A natural concern is that a $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement π configured for k node failures might fare poorly when subjected to $k' \neq k$ failures, at least in comparison to its availability were it configured for k' failures. This could occur if the $\langle \lambda_x \rangle_{x \in [s]}$ resulting from the configuration with k and those values resulting from configuration with k' were different.



$r = 5$, $s = 3$, $k = 6$

Fig. 3: $\frac{\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})}{\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda'_x \rangle_{x \in [s]})}$ expressed as a percentage

We have explored parameter spaces of interest to identify settings for which $\langle \lambda_x \rangle_{x \in [s]}$ would be different when configured for k or k' failed nodes, and then compared the resulting availability lower bounds. Fig. 3 shows a representative example. This figure plots the ratio $\frac{\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda_x \rangle_{x \in [s]})}{\mathit{lbAvail}^{\mathsf{co}}(\langle \lambda'_x \rangle_{x \in [s]})}$ expressed as a percentage for a $\mathsf{Combo}(\langle \lambda_x \rangle_{x \in [s]})$ placement configured for k node failures and a $\mathsf{Combo}(\langle \lambda'_x \rangle_{x \in [s]})$ placement configured for k' node failures. As such, when $k' = k$ this ratio will be 100%, for example. As this plot indicates, for some parameter values, this ratio dips below 100%, though for cases we have explored, this ratio remains high.

C. Parameter Selection

In this section we explore alternatives for instantiating our constructions for given values of n, r, x, and λ. Per Observation 1, creating a $\mathsf{Simple}(x, \lambda)$ placement for n nodes can be achieved by identifying an $n_x \leq n$ and a μ_x that divides λ, for which $\mu_x \binom{n_x}{x+1} / \binom{r}{x+1}$ is integral and, moreover, a $\mathsf{Simple}(x, \mu_x)$ placement exists for any $b \leq \mu_x \binom{n_x}{x+1} / \binom{r}{x+1}$

n	r = 2	3	4	5
31	$n_1 = 31$	$n_1 = 31$ [23] $n_2 = 31$	$n_1 = 28$ [13] $n_2 = 28$ [21] $n_3 = 31$	$n_1 = 25$ [13] $n_2 = 26$ [20] $n_3 = 23$ [32] $n_4 = 31$
71	$n_1 = 71$	$n_1 = 69$ [23] $n_2 = 71$	$n_1 = 70$ [13] $n_2 = 70$ [21] $n_3 = 71$	$n_1 = 65$ [13] $n_2 = 65$ [10] $n_3 = 71$ [32] $n_4 = 71$
257	$n_1 = 257$	$n_1 = 255$ [23] $n_2 = 257$	$n_1 = 256$ [13] $n_2 = 256$ [21] $n_3 = 257$	$n_1 = 245$ [13] $n_2 = 257$ [30] $n_3 = 243$ [32] $n_4 = 257$

Fig. 4: Values of n_x used in this paper

objects. For such an n_x and μ_x, a Simple(x, μ_x) placement corresponds to a $(x+1)$-(n_x, r, μ_x)-*design* [26]. The study of the existence of such constructs is a fundamental question in design theory (e.g., [23]).

The need for μ_x to divide λ can be discharged if $\mu_x = 1$, in which case a $(x+1)$-(n_x, r, μ_x)-design is a Steiner system. Letting q be any prime power and for any $d \geq 2$, known infinite designs include [10]: $x+1=2$, $r=q$, and $n_x = q^d$; $x+1=3$, $r=q+1$, and $n_x = q^d+1$; $x+1=2$, $r=q+1$, and $n_x = q^d+\cdots+q+1$; $x+1=2$, $r=q+1$, and $n_x = q^3+1$; and $x+1=2$, $r=2^d$, and $n_x = 2^{d+d'}+2^d-2^{d'}$ for any $d' > d$. In addition, there are numerous known finite designs for $x < 5$, as surveyed by Colbourn and Mathon [10]. Known designs of Steiner systems suffice to implement Simple(x, λ) for a wide array of practical parameter values, including all of the parameter settings investigated in this paper. Fig. 4 shows the Steiner systems used in our evaluations. Note that when $x+1=r$, the constraints for a Steiner system are vacuously satisfied by sets of size r.

As discussed in Observation 2, if a suitable $n_x \approx n$ cannot be found, then an alternative is to deconstruct the n nodes into "chunks" of size n_{x1}, ..., n_{xm}, each admitting a Simple(x, μ_{xi}) placement, and to build a Simple(x, μ_x) placement for $\mu_x = \mathsf{lcm}\{\mu_{x1}, \ldots, \mu_{xm}\}$ on $\sum_{i=1}^{m} n_{xi}$ nodes by building a Simple(x, μ_x) placement on each chunk of n_{xi} nodes individually. This observation introduces a wide range of placement options for arbitrary n. This is demonstrated in Fig. 5 for $\mu_x = 1$, which explores possible placements when even only $m = 3$. Each CDF shows the fraction of n values in the range $[50, 800]$ for which the "capacity gap" is at most the value on the horizontal axis, where the "capacity gap" is the difference between the ideal capacity (i.e., $\left\lfloor \mu_x \binom{n}{x+1} / \binom{r}{x+1} \right\rfloor$) and the capacity achievable (using concrete Steiner systems) by decomposing n into up to $m=3$ chunks (i.e., $\sum_{i=1}^{m} \mu_{xi} \binom{n_{xi}}{x+1} / \binom{r}{x+1}$ with each $\mu_{xi} = 1$), expressed as a fraction of the ideal capacity. As shown there, in the cases $r \in \{2, 3, 4\}$, a very low (i.e., good) capacity gap can be achieved for nearly all system sizes n and all values of x. This is not the case for $r = 5$, however, where only about 10% of the system sizes n admit constructions (of which we are aware) for $x = 2$ or $x = 3$ with up to $m = 3$ chunks that yield a reasonably small capacity gap.

One way to address difficult cases like these (i.e., $r = 5$ along with $x = 2$ or $x = 3$) is to simply select one's system size n from the fraction of possible system sizes for which

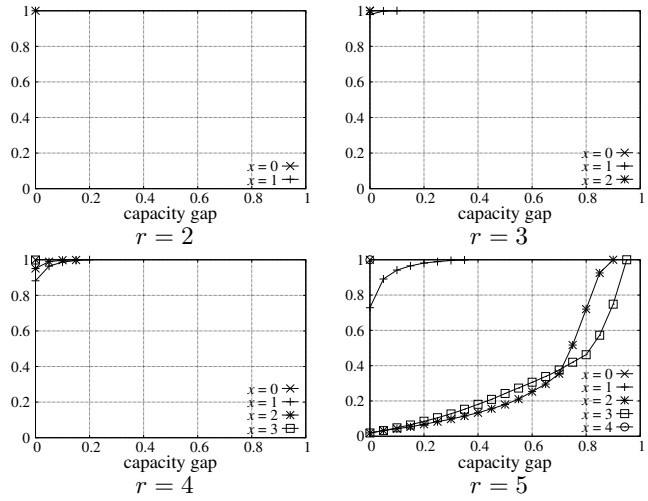


Fig. 5: CDFs showing the fraction of system sizes $n \in [50, 800]$ for which the capacity gap (indicated on the horizontal axis, where lower is better) can be achieved using up to $m = 3$ Steiner systems ($\mu_{xi} = 1$)

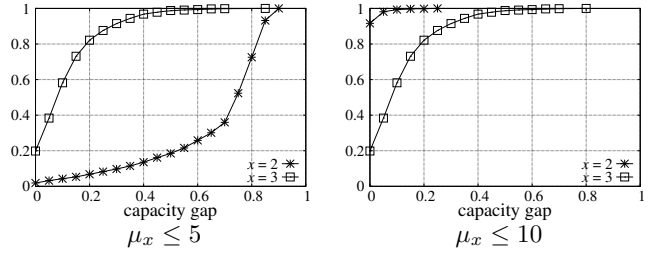


Fig. 6: Re-plot of Fig. 5 for $r = 5$ and $x \in \{2, 3\}$, but allowing $\mu_x = \mathsf{lcm}\{\mu_{x1}, \ldots, \mu_{xm}\} \geq 1$

a small capacity gap can be achieved. Another alternative is to increase m. A third alternative is to expand consideration to $\mu_x > 1$, in which case numerous additional constructions are possible. Trivially, for any $x+1 \leq r$, the collection of all r-subsets of n_x nodes suffices as a Simple(x, μ_x) placement for $\mu_x = \binom{n_x-x-1}{r-x-1}$. There are many other classes of $(x+1)$-(n_x, r, μ_x)-designs with $\mu_x > 1$, as have been surveyed elsewhere [25], [1], [22]. In particular, Khosrovshahi and Laue [22, Table 4.3.7] survey a number of infinite designs for $3 \leq x+1 \leq 5$.

To see the power of permitting $\mu_x > 1$ for realistic parameter settings, in Fig. 6 we re-plot the $x = 2$ and $x = 3$ cases for $r = 5$ but allowing μ_x to be $\mu_x \leq 5$ (left) or $\mu_x \leq 10$ (right). As can be seen in Fig. 6, allowing $\mu_x \leq 5$ yields significant improvements in the $x = 3$ case, and permitting $\mu_x \leq 10$ additionally improves the $x = 2$ case dramatically. As such, permitting even modest growth of μ_x can greatly shrink the capacity gap in difficult cases.

IV. Comparison to Random Replica Placement

As discussed in Sec. II, Yu and Gibbons [39] highlighted random replica placements as being very effective in ensuring the completion of multi-object operations when some objects'

loss could be tolerated, in a system model where nodes fail independently with fixed probability. Given this result and the more general prominence of random replica placement in the research literature, we compare the availability offered by our Combo($\langle\lambda_x\rangle_{x\in[s]}$) placement strategy to that offered by random replica placement. Specifically, we compare to a random placement strategy that (as in Yu and Gibbon's work) is load-balanced, where the average number of replicas per node is $\ell = \frac{rb}{n}$.

Definition 4: The Random placement strategy locates object replicas using a placement chosen uniformly at random from all placements that locate at most $\lceil\ell\rceil$ replicas on each node.

A. The Worst-Case Availability of Random

Evaluating the worst-case availability of Random placement is more subtle than for our previous placements, since *any* (load-balanced) placement can result from this placement strategy. So, in the truly worst case, Random would produce the worst possible placement for availability. That said, Random would do so with very low probability, and so this does not provide a representative view of how Random fares.

A more representative evaluation would take into account the *expected* behavior of the placement strategy. In some sense, the work of Yu and Gibbons [39] did so, but they did not take into account the *worst-case* behavior of the *adversary*. That is, their adversary failed nodes independently with a fixed probability, but ours adaptively chooses which nodes to fail based on the placement. So, to quantify the availability offered by Random in this worst case, we start by defining the *vulnerability* of Random:[1]

Definition 5: For any f, the vulnerability of Random, denoted $\mathit{Vuln}^{\mathsf{rnd}}(f)$, is the expected number of pairs $(\mathcal{K},\mathcal{F})$ where $\mathcal{K}\subseteq\mathcal{N}$, $|\mathcal{K}| = k$, $\mathcal{F}\subseteq\mathcal{O}$, $|\mathcal{F}|\geq f$, and at least s replicas of each object in $\mathcal{F}$ are placed on the nodes in $\mathcal{K}$. The expectation is taken with respect to the random choices made by the Random placement strategy.

If $\mathit{Vuln}^{\mathsf{rnd}}(f)\geq 1$, then in expectation, there will be a set of k nodes that, if failed, will fail a set of at least f objects. It is then natural to define the number of objects that are *probably available* as follows:[2]

Definition 6: In a Random placement, the number of objects that are probably available is

$$\mathit{prAvail}^{\mathsf{rnd}} = b - \max\{f : \mathit{Vuln}^{\mathsf{rnd}}(f)\geq 1\}$$

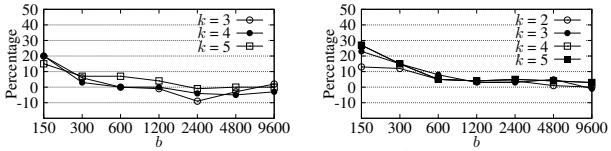
We now quantify the probable availability of Random.

Theorem 2: As $\ell\longrightarrow\infty$, $\mathit{Vuln}^{\mathsf{rnd}}(f)\longrightarrow$

$$\binom{n}{k}\binom{n}{r}^{-b}\left[\sum_{f'=f}^{b}\binom{b}{f'}\alpha(n,k,r,s)^{f'}\left(\binom{n}{r}-\alpha(n,k,r,s)\right)^{b-f'}\right]$$

where $\alpha(n,k,r,s) = \sum_{s'=s}^{\min\{r,k\}}\binom{k}{s'}\binom{n-k}{r-s'}$.

To confirm that this limit approaches reality quickly, in Fig. 7 we show $\mathit{prAvail}^{\mathsf{rnd}} - \mathit{avgAvail}^{\mathsf{rnd}}$ as a percentage of $\mathit{avgAvail}^{\mathsf{rnd}}$, where $\mathit{avgAvail}^{\mathsf{rnd}}$ is the empirical average of the number of objects available in 20 simulations of Random placements under a worst-case choice of k nodes to fail. In other words, these graphs show the percentage error in $\mathit{prAvail}^{\mathsf{rnd}}$. These graphs show that this percentage error is generally at 10% or below when b reaches 600, and this holds true for the other parameter values we have evaluated. So, in drawing our comparisons between Random and Combo($\langle\lambda_x\rangle_{x\in[s]}$) placements, we will generally restrict our attention to $b\geq 600$, so as to be fair to Random.



$n = 31,\ r = 5,\ s = 3$ $\qquad$ $n = 71,\ r = 5,\ s = 2$

Fig. 7: $\mathit{prAvail}^{\mathsf{rnd}} - \mathit{avgAvail}^{\mathsf{rnd}}$ as a percentage of $\mathit{avgAvail}^{\mathsf{rnd}}$

Fig. 8 plots $\frac{1}{b}\mathit{prAvail}^{\mathsf{rnd}}$, i.e., $\mathit{prAvail}^{\mathsf{rnd}}$ as a fraction of b, for various values of s, r, and n when $b = 38400$. Plotted as a fraction of b, the curves look very similar for the various values of b that we have explored. One takeaway from these graphs is that the case $s = 1$ performs quite poorly relative to larger s (notice the vertical axes are not the same scale), and we prove in Appendix A that this is true for Random placements in general. Indeed, these graphs illustrate that as s grows to approach r, $\mathit{prAvail}^{\mathsf{rnd}}$ improves dramatically.

B. Quantitative Comparison Results

In this section we compare Combo and Random placements using $\mathit{lbAvail}^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) - \mathit{prAvail}^{\mathsf{rnd}}$ as a measure, i.e., using a Combo($\langle\lambda_x\rangle_{x\in[s]}$) placement computed to maximize $\mathit{lbAvail}^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]})$ (Sec. III-B1). We use the limit of $\mathit{Vuln}^{\mathsf{rnd}}(f)$ in Theorem 2 to calculate $\mathit{prAvail}^{\mathsf{rnd}}$ as defined in Definition 6. The measure $\mathit{lbAvail}^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) - \mathit{prAvail}^{\mathsf{rnd}}$ is conservative in the sense that $\mathit{lbAvail}^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]})$ is a *lower*

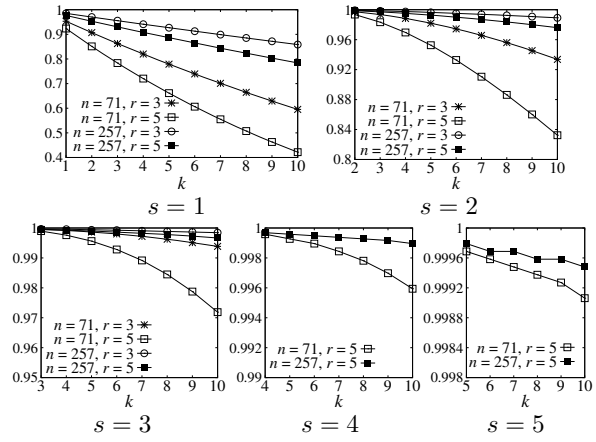


Fig. 8: $\frac{1}{b}\mathit{prAvail}^{\mathsf{rnd}}$ for $b = 38400$

[1] Definitions 5–6 trivially generalize to any randomized placement strategy.

[2] Unlike the alternative of simply using the expected value of the maximum, over all $\mathcal{K}$, $|\mathcal{K}| = k$, of the number of objects with s replicas on the nodes in $\mathcal{K}$, the definition of $\mathit{prAvail}^{\mathsf{rnd}}$ in Definition 6 yields an integral value. This simplifies our analysis in the rest of the paper.

bound, whereas *prAvail* $^{\mathsf{rnd}}$ is only a probabilistic estimate of the number of objects that remain available under Random and so it is not guaranteed.

Here and elsewhere in this paper, we use $n \in \{31, 71, 257\}$, both because these values span a reasonably wide range and because suitable $n_x \approx n$ and μ_x can be found for them without resorting to Observation 2. (These are by no means the only values that meet these criteria, though.) In particular, this means that the Combo$(\langle\lambda_x\rangle_{x\in[s]})$ placements represented in this section have concrete implementations. The selection of each n_x (with $\mu_x = 1$) is detailed in Sec. III-C.

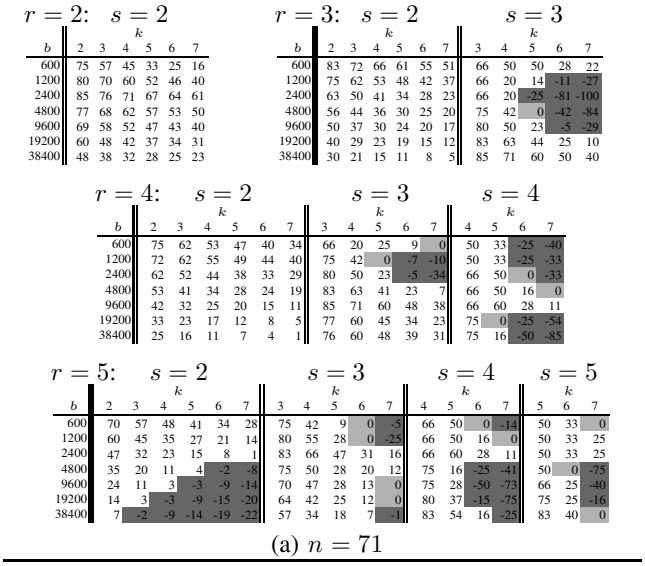
A summary of results is given in Fig. 9, where the top (Fig. 9a) shows the results with $n = 71$ and the bottom (Fig. 9b) shows the results with $n = 257$. Each portion shows a table for $2 \le r \le 5$ and $2 \le s \le r$. (The case $s = 1$ is further discussed in Appendix A.) The number k of failed nodes is ranged over $s \le k \le 7$ in the $n = 71$ case, and over $s \le k \le 8$ in the $n = 257$ case; both ranges encompass a substantial rate of node failures. In each table, the number b of objects begins at $b = 600$ and is repeatedly doubled until it reaches $b = 38400$. Each table entry indicates *lbAvail* $^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) - $ *prAvail* $^{\mathsf{rnd}}$ as a percentage of the maximum possible improvement $b - $ *prAvail* $^{\mathsf{rnd}}$ that could be achieved over *prAvail* $^{\mathsf{rnd}}$. To ease readability, cells where *lbAvail* $^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) >$ *prAvail* $^{\mathsf{rnd}}$ (and so Combo "wins") are colored white; cells where *lbAvail* $^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) =$ *prAvail* $^{\mathsf{rnd}}$ (neither Combo nor Random "wins") are colored light gray; and cells where *lbAvail* $^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) <$ *prAvail* $^{\mathsf{rnd}}$ (Random "wins") are colored dark gray.

It is evident that Combo "wins" most of the time, and the percentage by which it does so is often very substantial. For example, the table in the very upper-left corner of Fig. 9a indicates that in the case $n = 71$, $r = 2$, $s = 2$, $b = 2400$ and $k = 2$, Combo *guarantees* to preserve the availability of 85% of the objects that will probably fail under Random.

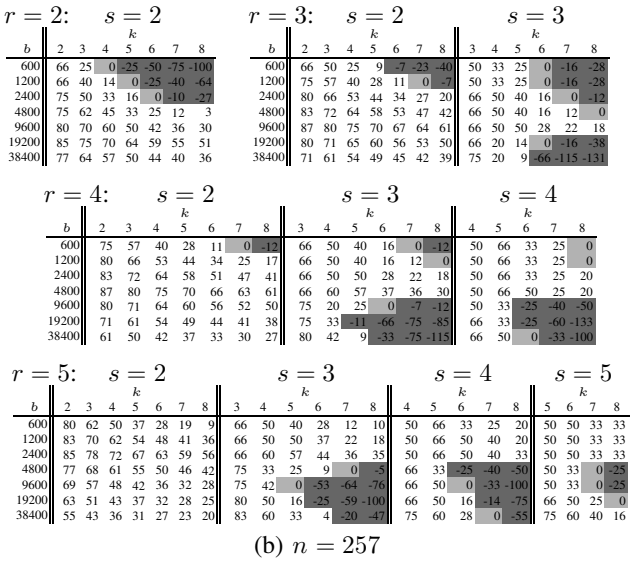
C. Breakdown of Combo Placements

Recall that each Combo$(\langle\lambda_x\rangle_{x\in[s]})$ placement combines individual Simple(x, λ_x) placements. In this section we detail how individual Simple(x, λ_x) placements contribute to the Combo$(\langle\lambda_x\rangle_{x\in[s]})$ placements computed via the algorithm described in Sec. III-B1, or more specifically how they contribute to the results showing the improvement of Combo placements over Random placements in Sec. IV-B.

We demonstrate these contributions through Fig. 10, which isolates three cases: $n = 31$ (Fig. 10a), $n = 71$ (Fig. 10b), and $n = 257$ (Fig. 10c). In this figure, each table corresponds to $r = s = 3$. The rightmost sub-table of each table represents the Combo$(\langle\lambda_x\rangle_{x\in[s]})$ placement and, in the case $n = 71$ or $n = 257$, is an exact copy of the table in Fig. 9a or Fig. 9b, respectively, for $r = s = 3$. The other tables in its row represent Simple(x, λ_x) placements. These figures do not show Simple$(0, \lambda_0)$ placements to save space, since they contribute so minimally. As before, a white table cell indicates that our placement achieves better availability than a Random placement; a light gray cell indicates that both provide equal availability (setting aside the conservative nature of the comparison, see Sec. IV-B); and a dark gray cell indicates that Random provides (potentially) better availability.



(a) $n = 71$

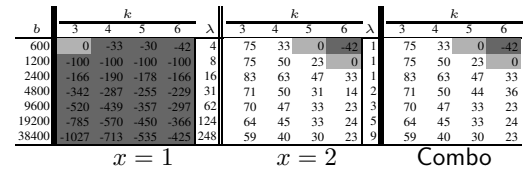


(b) $n = 257$

Fig. 9: *lbAvail* $^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]})$ − *prAvail* $^{\mathsf{rnd}}$ for a Combo$(\langle\lambda_x\rangle_{x\in[s]})$ placement (computed as in Sec. III-B) as a percentage of the maximum possible improvement $b - $ *prAvail* $^{\mathsf{rnd}}$

We draw the following observations from Fig. 10.

- When b grows and n and x are held constant, Simple(x, λ) availability improves relative to Random until λ has to grow to satisfy Eqn. 1. This can be seen, for example, in the $x = 1$ sub-table of Fig. 10c. As shown there, while λ can remain at 1 (see rightmost column of leftmost sub-table), Simple(x, λ) (and so Combo, as shown in the rightmost sub-table) "wins" more, but its performance diminishes as λ grows.
- One way to offset the need to grow λ is to adjust x, since when $k \approx s$, doing so impacts availability only a small amount (Eqn. 2) but can allow a Simple(x, λ) placement to accommodate many more objects (assuming $n \gg r$). This is shown clearly in, e.g., Fig. 10a and Fig. 10b, where



b	k 3	4	5	6	λ	k 3	4	5	6	λ	k 3	4	5	6
600	0	-33	-30	-42	4	75	33	0	-42	1	75	33	0	-42
1200	-100	-100	-100	-100	8	75	50	23	0	1	75	50	23	0
2400	-166	-190	-178	-166	16	83	63	47	33	1	83	63	47	33
4800	-342	-287	-255	-229	31	71	50	31	14	2	71	50	44	36
9600	-520	-439	-357	-297	62	70	47	33	23	3	70	47	33	23
19200	-785	-570	-450	-366	124	64	45	33	24	5	64	45	33	24
38400	-1027	-713	-535	-425	248	59	40	30	23	9	59	40	30	23

$x = 1$ $x = 2$ Combo

(a) $n = 31$

b	k 3	4	5	6	7	λ	k 3	4	5	6	7	λ	k 3	4	5	6	7
600	66	50	50	28	22	1	66	0	-66	-185	-288	1	66	50	50	28	22
1200	33	20	14	-11	-27	2	66	20	-42	-122	-218	1	66	20	14	-11	-27
2400	-33	-60	-62	-81	-100	4	66	20	-25	-81	-150	1	66	20	-25	-81	-100
4800	-75	-100	-130	-150	-157	7	75	42	0	-42	-84	1	75	42	0	-42	-84
9600	-160	-225	-230	-242	-237	13	80	50	23	-5	-29	1	80	50	23	-5	-29
19200	-316	-354	-361	-362	-348	25	83	63	44	25	10	1	83	63	44	25	10
38400	-614	-614	-564	-525	-493	50	85	71	60	50	40	1	85	71	60	50	40

$x = 1$ $x = 2$ Combo

(b) $n = 71$

b	k 3	4	5	6	7	8	λ	k 3	4	5	6	7	8	λ	k 3	4	5	6	7	8
600	50	33	25	0	-16	-28	1	50	-33	-150	-300	-483	-700	1	50	33	25	0	-16	-28
1200	50	33	25	0	-16	-28	1	50	-33	-150	-300	-483	-700	1	50	33	25	0	-16	-28
2400	66	50	40	16	0	-12	1	66	0	-100	-233	-400	-600	1	66	50	40	16	0	-12
4800	66	50	40	16	12	0	1	66	0	-100	-233	-337	-522	1	66	50	40	16	12	0
9600	66	50	50	28	22	18	1	66	0	-66	-185	-288	-409	1	66	50	50	28	22	18
19200	33	20	14	0	-16	-38	2	66	20	-42	-100	-191	-330	1	66	20	14	0	-16	-38
38400	0	-60	-18	-66	-115	-131	4	75	20	9	-66	-169	-250	1	75	20	9	-66	-115	-131

$x = 1$ $x = 2$ Combo

(c) $n = 257$

Fig. 10: $\mathit{lbAvail}^{\mathsf{si}}(x, \lambda) - \mathit{prAvail}^{\mathsf{rnd}}$ for $\mathsf{Simple}(x, \lambda)$ placements and $\mathit{lbAvail}^{\mathsf{co}}(\langle\lambda_x\rangle_{x\in[s]}) - \mathit{prAvail}^{\mathsf{rnd}}$ for best $\mathsf{Combo}(\langle\lambda_x\rangle_{x\in[s]})$ placement (rightmost sub-table in each row) when $r = s = 3$, as a percentage of the maximum possible improvement $b - \mathit{prAvail}^{\mathsf{rnd}}$

moving from $x = 1$ to $x = 2$ relieves the pressure on λ to increase, allowing the advantages of Combo to be preserved as b grows.

- Another way to slow the growth of λ is to increase n. For a fixed number b of objects and as n grows, Combo placements will increasingly place objects using $\mathsf{Simple}(x, \lambda_x)$ placements for smaller x. To see this, compare the contributions of $x = 1$ and $x = 2$ to the resulting Combo placement in Fig. 10a and Fig. 10c. This can be explained by observing that as n and so each n_x grows, the smallest x that suffices to achieve Eqn. 1 can shrink while keeping λ the same. This, in turn, yields better availability (Eqn. 2).
- Even at specific parameter values, Combo can outperform $\mathsf{Simple}(x, \lambda)$ for any single x. This is illustrated in Fig. 10a, for example, in which at $b = 4800$ and $k \in \{5, 6\}$, the Combo table includes entries (44 and 36) that exceed the corresponding entries of any of the $\mathsf{Simple}(x, \lambda_x)$ tables in its row. This occurs at a value of b at which $\mathsf{Simple}(2, \lambda_2)$ must increase λ_2 from $\lambda_2 = 1$ to $\lambda_2 = 2$ to satisfy Eqn. 1. In this case, it turns out to be better to build Combo using a $\mathsf{Simple}(2, 1)$ placement in conjunction with a $\mathsf{Simple}(1, 2)$ placement to satisfy Eqn. 3, rather than using a $\mathsf{Simple}(2, 2)$ placement alone.

D. Limitations of Combo Relative to Random

While Combo often outperforms Random for realistic parameter values as shown in Sec. IV-B, it can also be somewhat more difficult to use since, as we have formulated it here, it requires estimates of the number b of objects that will be placed and of the number k of failed nodes to plan for. Random requires neither of these as inputs. Projecting a value for k is commonplace in distributed computing practice, and our algorithm for computing a Combo placement is not especially sensitive to that estimate anyway, as discussed in Sec. III-B; so, we do not expect this requirement to be onerous. Moreover, in some cases b can be reasonably projected, either due to particulars of the application or due to capacity limits per node that restrict the number of replicas that can be placed on it and, thus, the number of objects that can be deployed on the n nodes. For other cases, an algorithm to adapt our placements as new objects come and go would be an interesting advance; we leave investigation of such an algorithm to future work.

V. Conclusion

In this paper we explored replica placement based on t-packings, here called $\mathsf{Simple}(x, \lambda)$ placements, for maximizing the availability of objects in the face of the worst k node failures out of n nodes. We showed that a $\mathsf{Simple}(x, \lambda)$ placement provides availability that is c-competitive with optimal, for a specified constant c (for constant n, k, replicas r, and fatality threshold s). We then devised a placement strategy called $\mathsf{Combo}(\langle\lambda_x\rangle_{x\in[s]})$ that combines multiple $\mathsf{Simple}(x, \lambda)$ placements, and a dynamic programming algorithm that selects $\langle\lambda_x\rangle_{x\in[s]}$ so as to maximize our lower bound on the availability of the resulting Combo placement for a chosen k. We showed that a resulting Combo placement is not very sensitive to the value of k with which it is configured, however, for the parameter values we explored. Finally, we demonstrated and dissected the improvements offered by Combo over Random replica placement, based on our analysis of the expected availability supported by Random in our worst-case model.

Our algorithms leverage t-packings for parameters for which maximum t-packings (also called t-designs, see Sec. III-C) exist, meaning that based on current knowledge, realistically our results are limited to $r \leq 5$. Fortunately, this suffices for a wide array of data center applications in practice. Our work does, however, provide further impetus to advance t-packing construction.

Acknowledgements: This work was supported in part by NSF grant 1330599 and the National Natural Science Foundation of China (grant 61303213).

References

[1] Abel, R.J.R., Greig, M.: BIBDs with small block size, chap. 3. In: Colbourn and Dinitz [8], 2nd edn. (2007)

[2] Adya, A., et al.: FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In: 5th Symp. Operating Systems Design & Implementation (Dec 2002)

[3] Bernard, S., Le Fessant, F.: Optimizing peer-to-peer backup using lifetime estimations. In: 2009 EDBT/ICDT Workshops (Mar 2009)

[4] Bhagwan, R., Savage, S., Voelker, G.M.: Replication strategies for highly available peer-to-peer storage systems. Tech. Rep. CS2002-0726, Department of Computer Science and Engineering, University of California, San Diego (Nov 2002)

[5] Bolosky, W.J., Douceur, J.R., Ely, D., Theimer, M.: Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In: 2000 ACM Intern. Conf. Measurement and Modeling of Computer Systems (2000)

[6] Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)

[7] Budhiraja, N., Marzullo, K., Schneider, F.B., Toueg, S.: The primary-backup approach, chap. 8. In: Mullender [28], 2nd edn. (1993)

[8] Colbourn, C.J., Dinitz, J.H. (eds.): Handbook of Combinatorial Designs. Chapman Hall/CRC, 2nd edn. (2007)

[9] Colbourn, C.J., Dinitz, J.H., Stinson, D.R.: Applications of combinatorial designs to communications, cryptography, and networking. In: Lamb, J.D., Preece, D.A. (eds.) Surveys in Combinatorics, 1999. Cambridge University Press (1999)

[10] Colbourn, C.J., Mathon, R.: Steiner systems, chap. 5. In: Colbourn and Dinitz [8], 2nd edn. (2007)

[11] Colbourn, C.J., Van Oorschot, P.C.: Applications of combinatorial designs in computer science. ACM Computing Surveys 21(2) (Jun 1989)

[12] Dasgupta, S., Papadimitriou, C., Vazirani, U.: Algorithms. McGraw-Hill (2008)

[13] Dinitz, J.H., Stinson, D.R.: A brief introduction to design theory, chap. 1. In: [14] (1992)

[14] Dinitz, J.H., Stinson, D.R. (eds.): Contemporary Design Theory: A Collection of Surveys. Wiley-Interscience (1992)

[15] Douceur, J., Wattenhofer, R.: Competitive hill-climbing strategies for replica placement in a distributed file system. In: 15th Intern. Symp. Distributed Computing (2001)

[16] Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 1. John Wiley & Sons, Inc., 3rd edn. (1968)

[17] Garcia-Molina, H., Barbara, D.: How to assign votes in a distributed system. Journal of the ACM 32 (Oct 1985)

[18] Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. In: 19th ACM Symp. Operating Systems Principles (Oct 2003)

[19] Gifford, D.K.: Weighted voting for replicated data. In: 7th ACM Symp. Operating System Principles (1979)

[20] Hanani, H., Hartman, A., Kramer, E.S.: On three-designs of small order. Discrete Mathematics 45(1) (1983)

[21] Hanani, M.: On quadruple systems. Canad. J. Math. 12 (1960)

[22] Khosrovshahi, G.B., Laue, R.: t-designs with $t \geq 3$, chap. 4. In: Colbourn and Dinitz [8], 2nd edn. (2007)

[23] Lindner, C.C., Rodger, C.A.: Design Theory, chap. 1. CRC Press (2008)

[24] MacCormick, J., et al.: Kinesis: A new approach to replica placement in distributed storage systems. ACM Transactions on Storage 4 (Jan 2009)

[25] Mathon, R., Rosa, A.: 2-(v, k, λ) designs of small order, chap. 1. In: Colbourn and Dinitz [8], 2nd edn. (2007)

[26] Mills, W.H., Mullin, R.C.: Coverings and packings, chap. 9. In: Dinitz and Stinson [14] (1992)

[27] Mitzenmacher, M., Upfal, E.: Probability and Computing. Cambridge University Press (2005)

[28] Mullender, S. (ed.): Distributed Systems. Addison-Wesley, 2nd edn. (1993)

[29] Ng, W.K., Ravishankar, C.V.: Coterie templates: A new quorum construction method. In: 15th Intern. Conf. Distributed Computing Systems (May 1995)

[30] Ogilvy, C.S.: Excursions in Geometry, chap. 3-4. Dover (1990)

[31] On, G., Schmitt, J., Steinmetz, R.: Quality of availability: Replica placement for widely distributed systems. In: 11th Intern. Conf. Quality of Service (2003)

[32] Ostergard, P.R., Pottonen, O.: There exists no Steiner system $S(4, 5, 17)$. Journal of Combinatorial Theory, Series A 115(8) (2008)

[33] Raghavarao, D., Padgett, L.V.: Balanced incomplete block designs — applications. In: Block Designs: Analysis, Combinatorics and Applications, chap. 5. World Scientific (2005)

[34] Rzadca, K., Datta, A., Buchegger, S.: Replica placement in P2P storage: Complexity and game theoretic analyses. In: 30th IEEE Intern. Conf. Distributed Computing Systems (Jun 2010)

[35] Santos, J.R., Muntz, R.R., Ribeiro-Neto, B.: Comparing random data allocation and data striping in multimedia servers. In: 2000 ACM Intern. Conf. Measurement and Modeling of Computer Systems (2000)

[36] Schneider, F.B.: What good are models and what models are good?, chap. 2. In: Mullender [28], 2nd edn. (1993)

[37] Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop distributed file system. In: 26th IEEE Symp. Mass Storage Systems and Technologies (2010)

[38] VMWare, Inc.: Protecting mission-critical workloads with VMware fault tolerance. http://www.vmware.com/resources/techresources/1094 (Feb 2009)

[39] Yu, H., Gibbons, P.B.: Optimal inter-object correlation when replicating for availability. In: 26th ACM Symp. Principles of Distributed Computing (2007)

[40] Yu, H., Gibbons, P.B., Nath, S.: Availability of multi-object operations. In: 3rd USENIX Symp. Networked Systems Design & Implementation (2006)

[41] Yu, H., Vahdat, A.: Minimal replication cost for availability. In: 21st ACM Symp. Principles of Distributed Computing (2002)

APPENDIX A
THE $s = 1$ CASE

In our comparisons between $\mathsf{Combo}(\langle\lambda_x\rangle_{x\in[s]})$ and Random placements in Sec. IV-B, we deferred the case $s = 1$. In this case, a $\mathsf{Combo}(\langle\lambda_x\rangle_{x\in[s]})$ placement is just a $\mathsf{Simple}(0, \lambda_0)$ placement. Our analysis in this paper applies to the $s = 1$ case, and a comparison using $\mathit{lbAvail}^{\mathsf{co}}(\lambda_0) - \mathit{prAvail}^{\mathsf{rnd}}$ as in Sec. IV-B indicates that Random slightly outperforms $\mathsf{Simple}(0, \lambda_0)$ in this measure, for the parameter values we tested. Nevertheless, we relegated this case to the appendix simply because both Random and $\mathsf{Simple}(0, \lambda_0)$ perform poorly in this case. The following lemma, proved in Appendix B, formalizes this claim for Random placements.

Lemma 4: Suppose $s = 1$, $k < n/2$, and $\ell = \frac{rb}{n}$. Then,

$$\mathit{prAvail}^{\mathsf{rnd}} \leq b\left(1 - \frac{1}{b}\right)^{k\lfloor\ell\rfloor}$$

To see one implication of this lemma, recall that $(1 - \frac{1}{b})^b$ converges to e^{-1} as $b \longrightarrow \infty$. So, for large enough b, $\mathit{prAvail}^{\mathsf{rnd}}$ is at most approximately $b(e^{-r/n})^k$. In terms of parameter values tested elsewhere in this paper, Fig. 11 shows how $\frac{1}{b}\mathit{prAvail}^{\mathsf{rnd}} = (1 - 1/b)^{k\lfloor\ell\rfloor}$ behaves for small numbers of node failures (c.f., the $s = 1$ case of Fig. 8). Fig. 11 is plotted for $b = 38400$, but plots for $b = 2400$ and $b = 9600$ are virtually indistinguishable. This graph shows that the availability of Random placements, as a fraction of b, decays essentially linearly in the number k of failed nodes, with a slope that grows smaller as n increases or r decreases (since each node then hosts fewer object replicas).

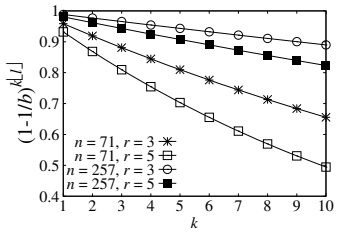


Fig. 11: $\left(1 - \frac{1}{b}\right)^{k\lfloor\ell\rfloor}$ for various n and r, as a function of k ($b = 38400$)

APPENDIX B
PROOFS

Proof of Lemma 2: An upper bound on the number of objects that become unavailable due to the failure of nodes in $\mathcal{K}$ is simply the number of objects for which s replicas can be packed onto the nodes $\mathcal{K}$ under the constraints of a $\mathsf{Simple}(x,$

λ) placement, i.e., in a Simple(x, λ) placement using only s replicas per object (versus r) and only k nodes (versus n). Adapting Lemma 1 accordingly, we get that at most $\left\lfloor \lambda \binom{k}{x+1} / \binom{s}{x+1} \right\rfloor$ objects become unavailable. □

Proof of Theorem 1: First note that Eqn. 1 implies

$$\frac{\lambda}{b} < \frac{\binom{r}{x+1}}{\binom{n_x}{x+1}} + \frac{\mu_x}{b} \tag{8}$$

By Lemma 2,

$$\begin{aligned}
\frac{\mathit{Avail}(\pi)}{\mathit{Avail}(\pi')} &\geq \frac{b - \left\lfloor \lambda \frac{\binom{k}{x+1}}{\binom{s}{x+1}} \right\rfloor}{b} \\
&\geq \frac{b - \lambda \frac{\binom{k}{x+1}}{\binom{s}{x+1}}}{b} \\
&= 1 - \frac{\lambda}{b} \frac{\binom{k}{x+1}}{\binom{s}{x+1}} \\
&> 1 - \left(\frac{\binom{r}{x+1}}{\binom{n_x}{x+1}} + \frac{\mu_x}{b} \right) \frac{\binom{k}{x+1}}{\binom{s}{x+1}}
\end{aligned}$$

where the last step is substituting Eqn. 8. Rearranging, we get

$$\mathit{Avail}(\pi') - c \cdot \mathit{Avail}(\pi) < \left(\frac{\mathit{Avail}(\pi')}{b} \right) \alpha \leq \alpha$$

where c and α are as given in the theorem statement. □

Proof of Lemma 3: Under a Combo($\langle \lambda_x \rangle_{x \in [s]}$) placement, each Simple(x, λ_x) placement accounts for placing at most $\lambda_x \frac{\binom{n_x}{x+1}}{\binom{r}{x+1}}$ objects, of which up to $\left\lfloor \lambda_x \frac{\binom{k}{x+1}}{\binom{s}{x+1}} \right\rfloor$ might be rendered unavailable by k node failures, as in Lemma 2. So, at most $\sum_{x=0}^{s-1} \left\lfloor \lambda_x \frac{\binom{k}{x+1}}{\binom{s}{x+1}} \right\rfloor$ objects can be rendered unavailable in total by k node failures. Since only b objects can be placed, the result follows. □

Proof of Theorem 2: Consider a variant Random′ of the Random placement in which the r replicas of each object are placed on r distinct nodes selected uniformly at random, but without limiting the number of replicas placed at each node. Let $X_{\mathsf{nd},\mathsf{obj}}$ be an indicator random variable defined as $X_{\mathsf{nd},\mathsf{obj}} = 1$ if a replica of obj is placed at nd and $X_{\mathsf{nd},\mathsf{obj}} = 0$ otherwise. Let $L_{\mathsf{nd}} = \sum_{\mathsf{obj} \in \mathcal{O}} X_{\mathsf{nd},\mathsf{obj}}$; i.e., L_{nd} is a random variable denoting the number of replicas placed at node nd.

While Random enforces that the number of replicas placed on each node is at most $\lceil \ell \rceil$, Random′ allows more. Specifically, for a fixed nd, $\{X_{\mathsf{nd},\mathsf{obj}}\}_{\mathsf{obj} \in \mathcal{O}}$ are independent, identically distributed Bernoulli random variables; i.e., $X_{\mathsf{nd},\mathsf{obj}} \sim B(\frac{r}{n})$ for each obj $\in \mathcal{O}$. Therefore, $\mathbb{E}(L_{\mathsf{nd}}) = \frac{br}{n} = \ell$ and, applying well-known Chernoff bounds (see, e.g., [27, Corollary 4.6]),

$$\mathbb{P}\left(|L_{\mathsf{nd}} - \ell| \geq \delta \ell\right) \leq 2e^{-\ell \delta^2/3}$$

for any $0 < \delta < 1$. Consequently, the distribution of object replicas to nodes under Random′ (quickly) approaches the distribution induced by Random as $\ell \longrightarrow \infty$, and so we can reason about the asymptotic distribution induced by Random using the one induced by Random′.

Let $\mathbf{failedNodes}(\mathcal{K})$ denote the event that set $\mathcal{K} \subseteq \mathcal{N}$ is the complete set of failed nodes, and $\mathbf{failedObjs}(\mathcal{F})$ denote the event that the set $\mathcal{F} \subseteq \mathcal{O}$ is the complete set of objects that failed due to the failure of the nodes in $\mathcal{K}$. Now, under Random′,

$$\begin{aligned}
&\mathbb{P}\big(\mathbf{failedObjs}(\mathcal{F}) \;\big|\; \mathbf{failedNodes}(\mathcal{K})\big) \\
&= \left[\prod_{\mathsf{obj} \in \mathcal{F}} \mathbb{P}\left(\begin{smallmatrix} \text{obj replicas placed} \\ \text{on } s' \geq s \text{ nodes in} \\ \mathcal{K} \text{ and } r - s' \text{ others} \end{smallmatrix} \right) \right] \left[\prod_{\mathsf{obj} \in \mathcal{O} \setminus \mathcal{F}} \mathbb{P}\left(\begin{smallmatrix} \text{obj replicas placed} \\ \text{on } s' < s \text{ nodes in} \\ \mathcal{K} \text{ and } r - s' \text{ others} \end{smallmatrix} \right) \right] \\
&= \left[\prod_{\mathsf{obj} \in \mathcal{F}} \sum_{s'=s}^{\min\{r,k\}} \frac{\binom{k}{s'}\binom{n-k}{r-s'}}{\binom{n}{r}} \right] \left[\prod_{\mathsf{obj} \in \mathcal{O} \setminus \mathcal{F}} \sum_{s'=0}^{s-1} \frac{\binom{k}{s'}\binom{n-k}{r-s'}}{\binom{n}{r}} \right] \\
&= \binom{n}{r}^{-b} \left(\sum_{s'=s}^{\min\{r,k\}} \binom{k}{s'}\binom{n-k}{r-s'} \right)^{f} \left(\sum_{s'=0}^{s-1} \binom{k}{s'}\binom{n-k}{r-s'} \right)^{b-f} \\
&= \binom{n}{r}^{-b} \alpha(n,k,r,s)^f \left(\binom{n}{r} - \alpha(n,k,r,s) \right)^{b-f}
\end{aligned} \tag{9}$$

To complete the proof, for any $\mathcal{K} \subseteq \mathcal{N}$, $|\mathcal{K}| = k$, and any $\mathcal{F} \subseteq \mathcal{O}$, define an indicator random variable $X_{\mathcal{K},\mathcal{F}}$ as follows: $X_{\mathcal{K},\mathcal{F}} = 1$ if $\mathcal{F}$ is the set of objects failed when the nodes $\mathcal{K}$ fail, and $X_{\mathcal{K},\mathcal{F}} = 0$ otherwise. The expected value of $X_{\mathcal{K},\mathcal{F}}$ is then $\mathbb{E}(X_{\mathcal{K},\mathcal{F}}) = \mathbb{P}\big(\mathbf{failedObjs}(\mathcal{F}) \;\big|\; \mathbf{failedNodes}(\mathcal{K})\big)$. By linearity of expectation, $\mathit{Vuln}^{\mathsf{rnd}}(f)$ is then:

$$\mathit{Vuln}^{\mathsf{rnd}}(f) = \mathbb{E}\left(\sum_{\substack{\mathcal{K} \subseteq \mathcal{N}: \\ |\mathcal{K}| = k}} \sum_{\substack{\mathcal{F} \subseteq \mathcal{O}: \\ |\mathcal{F}| \geq f}} X_{\mathcal{K},\mathcal{F}} \right) = \sum_{\substack{\mathcal{K} \subseteq \mathcal{N}: \\ |\mathcal{K}| = k}} \sum_{\substack{\mathcal{F} \subseteq \mathcal{O}: \\ |\mathcal{F}| \geq f}} \mathbb{E}\left(X_{\mathcal{K},\mathcal{F}}\right)$$

Plugging in Eqn. 9 yields the result. □

Proof of Lemma 4: In choosing k nodes to fail, our adversary is guaranteed to be able to fail $k \lfloor \ell \rfloor$ replicas (because $k < n/2$) and, since $s = 1$, every object with one or more replicas in these $k \lfloor \ell \rfloor$ replicas. Let F denote the number of failed objects after the adversary induces k node failures. Contrast this scenario to sampling $k \lfloor \ell \rfloor$ objects with replacement from the objects $\mathsf{obj}_1, \ldots, \mathsf{obj}_b$, and let Y be a random variable capturing the number of *distinct* objects sampled. We claim that $\mathbb{P}(F \geq f) \geq \mathbb{P}(Y \geq f)$ due to two key differences between our adversary's scenario and simply sampling objects at random with replacement: First, since a placement places only one replica per object on any node, once the adversary selects a node to fail, it is *guaranteed* no repetitions of the same object on that node. Second, our adversary can encounter at most r replicas of any object (versus up to $k \lfloor \ell \rfloor$ when sampling objects uniformly at random with replacement). As such,

$$\mathbb{E}(F) = \sum_{f=1}^{\infty} \mathbb{P}(F \geq f) \geq \sum_{f=1}^{\infty} \mathbb{P}(Y \geq f) = \mathbb{E}(Y)$$

Since $\mathbb{E}(Y) = b\left[1 - \left(1 - \frac{1}{b}\right)^{k\lfloor \ell \rfloor}\right]$ (e.g., [16, p. 31]), when $f = b\left[1 - \left(1 - \frac{1}{b}\right)^{k\lfloor \ell \rfloor}\right]$ we have $\mathit{Vuln}^{\mathsf{rnd}}(f) \geq 1$, and thus $\mathit{prAvail}^{\mathsf{rnd}} \leq b\left(1 - \frac{1}{b}\right)^{k\lfloor \ell \rfloor}$. □