Analysis of Long-Running Replicated Systems

Sriram Ramabhadran
Dept. of Computer Science & Engineering
University of California, San Diego
sriram@cs.ucsd.edu

Joseph Pasquale
Dept. of Computer Science & Engineering
University of California, San Diego
pasquale@cs.ucsd.edu

***Abstract*— We address the problem of using replication to reliably maintain state in a distributed system for time spans that far exceed the lifetimes of individual replicas. This scenario is relevant for any system comprised of a potentially large and selectable number of replicated components, each of which may be highly unreliable, where the goal is to have enough replicas to keep the system "alive" (meaning at least one replica is working or available) for a certain expected period of time, i.e., the system's lifetime. In particular, this applies to recent efforts to build highly available storage systems based on the peer-to-peer paradigm. We model notions of replica loss and replica repair in such systems by a simple Markov chain model, and derive an expression for the lifetime of the replicated state. We then apply this model to study the impact of practical considerations like storage and bandwidth limits on the system, and describe methods to optimally choose system parameters so as to maximize lifetime. Our analysis sheds light on the efficacy of various replication strategies.**

I. Introduction

Replication is a cornerstone of reliable distributed system design. By replicating functionality across multiple nodes, applications can tolerate the loss of individual nodes in the system, thereby achieving some level of reliability in a system consisting of unreliable components. As the scope and scale of such systems continues to grow, it is of crucial importance to develop sound design principles for problems that are of fundamental importance to reliability engineering. In this paper, we are concerned with one such problem: *How do we reliably maintain state in a replicated system, where the lifetime of the state is required to significantly exceed that of individual nodes in the system?*

We are motivated to study this problem by several recent efforts [10], [17], [6], [14] to build highly available storage systems based on the peer-to-peer paradigm. These systems use replication as a key mechanism to build reliable storage in what is a highly unreliable and failure-prone environment. Storage systems based on the peer-to-peer model must address what happens when participating nodes leave the system, and then either subsequently return, or permanently cease their participation. When a node is not participating in the system, state stored on that node is not available, and the system must ensure that it is available on other nodes that are participating. Achieving availability in the face of a dynamically changing set of participating peers is one of the main challenges in these systems [8].

Replication ensures that the system is able to guard against individual failures. However, to maintain long-term availability, the system must constantly repair replicas lost due to node departure. Thus, there is a notion of repair rate, which captures how proactive the system is in replacing lost replicas. Thus, the two parameters the systems uses to ensure long-term availability of state are the number of replicas, and the repair rate. The metric the system tries to optimize is the lifetime of the state, i.e., how long the system is able to preserve the state in the system before all replicas are lost. How the system tunes these parameters to maximize lifetime is the subject of this paper, in which we address the following questions:

- What is the lifetime of a replicated system, given a certain degree of replication and a certain rate of repair?
- What is the effect of resource constraints on a replicated system, i.e., what happens to the lifetime when storage is limited, repair bandwidth is limited, or both?
- Given the above constraints, what is the optimal way to choose system parameters like the number of replicas and the rate of repair?

Although we have motivated the problem in terms of peer-to-peer storage, this problem of maximizing lifetime is relevent to other distributed systems as well. A good example are computational grid applications, which execute long-running computations over a peer-to-peer system of user PCs, each of which may "fail" (be turned off or have its processes removed/killed by its owner) long before the computation is complete. Other application domains include wireless sensor networks [7] and amorphous computing systems [3]. In summary, maximizing the lifetime of state stored in a replicated system is a fundamental problem that occurs in several contexts.

The rest of the paper is organized as follows. In Section II, we develop a simple Markov model for a replicated system; and obtain an expression for its lifetime. We then analyse the impact of resource constraints on replication strategies, and show how to optimally choose system parameters. In Section III, we use availability numbers from a real distributed system - Planetlab - to validate the model, and apply our results in a realistic setting. In Section IV, we survey related work, and in Section V, we present conclusions.

II. Analysis

A. A basic model

The system consists of some universe of nodes on which replicas can be created. A node participates in the system for

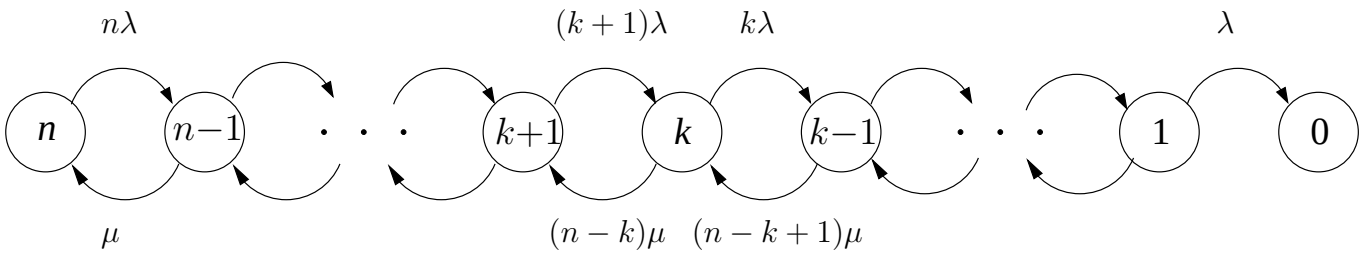

Fig. 1. Markov chain model where each state represents the number of currently available replicas

some duration of time t_p; we assume that t_p is an exponentially distributed random variable with mean $\frac{1}{\lambda}$, where λ is the rate of departure, and thus, $Pr\{t_p > t\} = e^{-\lambda t}$. (Below, we comment on the appropriateness of this assumption). We assume that t_p is independent and identically distributed for all nodes in the system.

Any state replicated on a node is available for the duration of its participation in the system. When a node leaves the system, we assume that its state is lost. Suppose we need to reliably maintain some state S in this system. For reasons of availability, this state is replicated on multiple nodes; let the number of replicas of S be n. We note that n is a parameter that the system can choose, although it may be subject to constraints such as resource capacities. For example, storage limitations may impose a practical upper bound on n. In any case, the system starts out with exactly n replicas of S.

Replication alone is insufficient to reliably maintain S. Over a period of time, node departures decrease the number of replicas of S present in the system. To compensate for this attrition, the system must also use a repair mechanism that creates new replicas to account for lost ones. The repair mechanism must first detect the loss of a replica, and then create a new one by copying S to another node from an existing replica. This whole process may take the system some duration of time t_r. To model repair, we assume that t_r is also an exponentially distributed random variable with mean $\frac{1}{\mu}$, where μ is the rate of repair. As before, this means that $Pr\{t_r > t\} = e^{-\mu t}$.

Finally, we define the *repair ratio* of the system, denoted by γ, as the normalized repair rate, relative to the rate of departure of nodes, i.e.,

$$\gamma = \frac{\mu}{\lambda} \quad (1)$$

Intuitively, γ represents the balance between how fast the system is losing replicas and how fast it is able to create new ones to compensate. We note that, in addition to n, γ is also a configurable parameter in that μ is tunable, i.e., the system can choose how fast it responds to lost replicas. A large γ implies the system aggressively replaces lost replicas, and vice versa. Again, there may be constraints on the choice of γ; for example, the repair time t_r must be at least the time it takes to detect a lost replica of S and copy S across the network.

Regarding the use of exponentials to model both node participation and replica repair, it is common practice in reliability theory [15] to model the failure of components by an exponential distribution; node participation is a similar concept. In Section III, we present data that suggests that uptime of nodes in a real system is exponentially distributed, thus providing some evidence that our assumption is grounded in reality. Modeling the repair time by an exponential distribution is somewhat harder to substantiate. We note that repair time may depend on factors such as timeliness of failure detection and available bandwidth, and is generally subject to some randomness. Again, in Section III, we present data that suggests the downtime of nodes in a real system is exponentially distributed, implying that modeling repair by an exponential is not altogether unrealistic. Last but not least, the memoryless property of the exponential distribution results in our replicated system model being Markovian, and hence mathematically tractable.

B. Markov chain

To analyze the above model, we reduce it to a Markov chain. At any point of time, the system has k $(0 \leq k \leq n)$ functioning replicas; the remaining $n-k$ are being repaired. Thus the system can be modeled by a Markov chain with $n+1$ possible states; the system is in state k if there are k functioning replicas. In state k, any one of the k functioning replicas can fail, in which case the system goes to state $k-1$, or one of the $n-k$ non-functioning replicas is repaired, in which case the system goes to state $k+1$. This is a continuous Markov model; in state k, the system moves to state $k-1$ with rate $k\lambda$ and to state $k+1$ with rate $(n-k)\mu$. Note that state 0 is an absorbing state; the system can no longer recover S when there are no more functioning replicas left. The model is illustrated in Figure 1. In the appendix, we derive an expression for the expected time to failure by considering the discrete embedding of this model.

C. Mean time to failure

Given this basic model, we now address the questions posed in Section I. Specifically, we quantify how long a replicated system can maintain some state S before it is lost permanently due to the dynamics of the underlying population of nodes. In terms of the above model, the relevant metric is the time to failure, i.e., the time it takes for the system, starting with n replicas of S, to reach a state where there are no replicas of S left. We denote this quantity, the *lifetime* of S, by t_s. Clearly, t_s is a random variable whose distribution depends on the

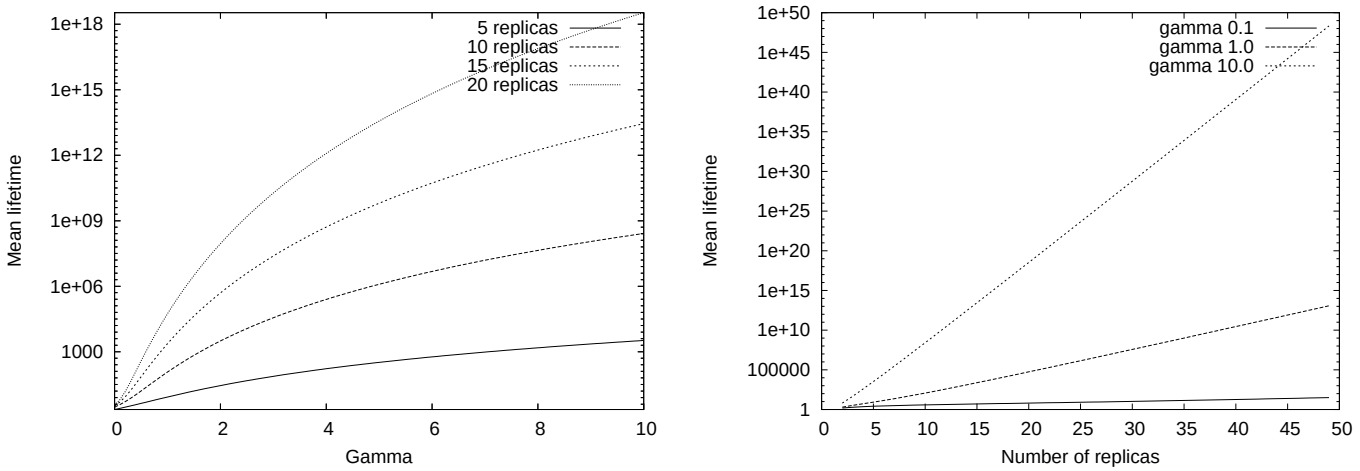

Fig. 2. $P_n(\gamma)$ as a function of (a) repair ratio γ and (b) number of replicas n. Note that the Y-axis is log scale.

n	$P_n(\gamma)$
2	$\frac{3}{2} + \frac{1}{2}\gamma$
3	$\frac{11}{6} + \frac{7}{6}\gamma + \frac{1}{3}\gamma^2$
4	$\frac{25}{12} + \frac{23}{12}\gamma + \frac{13}{12}\gamma^2 + \frac{1}{4}\gamma^3$
5	$\frac{137}{60} + \frac{163}{60}\gamma + \frac{137}{60}\gamma^2 + \frac{21}{20}\gamma^3 + \frac{1}{5}\gamma^4$

TABLE I
$P_n(\gamma)$ FOR SOME SMALL VALUES OF n

system parameters n, λ and μ. In terms of the above Markov chain, t_s is the time to absorption, and state 0 is an absorbing state.

We derive the expected value of t_s to be

$$E[t_s] = \frac{1}{\lambda} P_n(\gamma) \quad (2)$$

$$= \frac{1}{\lambda}\left(c_{0,n}\gamma^0 + c_{1,n}\gamma^1 + \ldots + c_{n-1,n}\gamma^{n-1}\right) \quad (3)$$

where the coefficient of γ^i is given by

$$c_{i,n} = \frac{1}{n}\sum_{j=0}^{n-i-1} \frac{\binom{n}{j}}{\binom{n-1}{i+j}} \quad (4)$$

A detailed derivation is presented in the appendix. We note that general techniques exist to compute time to absorption in matrix form for finite Markov chains; see [13] for example. Also, the mean time to failure of a parallel system of components with repair has been derived in [9]; although this expression is different in form from Equation 3, it can be shown to be identical. However, our derivation, which is based on the Gambler's Ruin Problem [12], [11], is simpler and more intuitive, and we provide it for the sake of completeness [1].

Recall that the average time a node participates in the system is $\frac{1}{\lambda}$. Therefore, $P_n(\gamma)$ represents the expected lifetime of the replicated system in terms of the average lifetime of a single replica. Note that this is purely a function of the repair ratio $\gamma = \frac{\mu}{\lambda}$. We note here that since t_s is a constant times $P_n(\gamma)$ [2], it is sufficient to study the behavior of $P_n(\gamma)$ with respect to n and γ.

Equation 3 shows that the mean lifetime varies polynomially with the repair ratio γ and exponentially with the number of replicas n. This is further illustrated in Figure 2. Figure 2(a) shows the behavior of $P_n(\gamma)$ with respect to γ for different values of n. Likewise Figure 2(b) shows the behavior of P_n with respect to n for different values of γ. Finally, Table I records $P_n(\gamma)$ for a few small values of n. As expected, the mean lifetime increases with both n and γ. We note that when γ is large, $P_n(\gamma)$ is large even for a relatively small n. On the other hand, when γ is small, even a large n results only in a modest $P_n(\gamma)$. In the limiting case, when either $\gamma \longrightarrow \infty$ or $n \longrightarrow \infty$, $P_n \longrightarrow \infty$ as well. The former corresponds to instantaneous repair while the latter corresponds to an unbounded number of replicas. In this case, even for $\gamma = 0$, i.e., the system does not repair lost replicas,

$$P_n(0) = c_{0,n} \quad (5)$$

[1]The Gambler's Ruin Problem is related to ours as follows: it considers what is the probability that a gambler (the system) with a certain stake (the number of replicas) playing repeated games in which there is a certain probability of winning (probability a new replica is generated before another replica fails) will ultimately be ruined (the state disappears due to no replicas surviving), and what is the average playing time (average lifetime of the state). Our problem is more complicated in that the "probability of winning in a single game" in our formulation is not a constant, but changes with the "current stake," i.e., the number of existing replicas. However, we use a framework similar to that of deriving classical results for the Gambler's Ruin, resulting in a closed-form analytical result that is very different in form but functionally equivalent to that of [9]. The advantage of our derivation is that it does not rely on transform methods, making it more intuitive.

[2]λ is a property of the nodes in the system and cannot be tuned by the system.

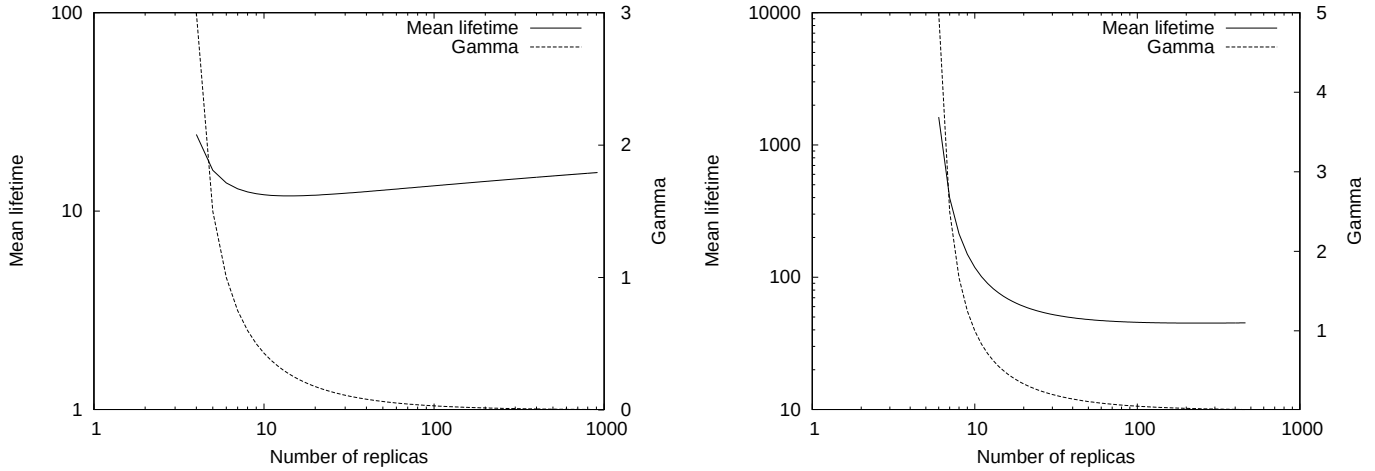

Fig. 3. Mean lifetime ($P_n(\gamma)$) under constrained repair bandwidth for different values of available repair bandwidth n_{max} = 3.0 and 5.0

$$= \frac{1}{n}\sum_{j=0}^{n-1}\frac{n}{n-j} \quad (6)$$

$$= \sum_{j=1}^{j=n}\frac{1}{j} \quad (7)$$

$$= H_n \quad (8)$$

Since the harmonic sum H_n diverges to infinity (albeit slowly), in theory, mean lifetime can be made arbitrarily large by creating a sufficiently large number of replicas, even though there is no repair. We note that, in most scenarios, neither instantaneous repair nor an unbounded number of replicas is likely to be a viable replication strategy, as there are practical considerations that limit the repair rate or the number of replicas. We elaborate on this in the following section.

D. Choosing system parameters

It is clear from Figure 2 that the mean lifetime t_s increases with both n and γ. Therefore, in order to maximize t_s, the system should be designed with both as large as possible, although how they should be increased relative to each other is an interesting question. In this section, we consider the effect of some such factors that arise out of practical considerations, and show how to choose n and γ so as to maximize t_s. These factors include the following:

1) Storage: The number of replicas of S that can be created may be limited by the total storage capacity of the system. This results in the constraint

$$n \leq n_{max} \quad (9)$$

where n_{max} is an upper bound on the number of replicas due to storage limits.

2) Management overhead: Replica repair must include a mechanism to detect the loss of individual replicas. One such mechanism is a group membership protocol that maintains the current set of replicas of S through constant probing. Overhead considerations may limit the frequency of these probes, which implies that detecting the loss of a replica may be associated with some average latency. This has a direct effect on the replica repair time t_r, and may result in a constraint of the form

$$\gamma \leq \gamma_{max} \quad (10)$$

where γ_{max} is an upper bound on the normalized repair rate.

3) Repair bandwidth: Finally there could be a constraint on the bandwidth used by the system in creating new replicas. For example, the system may be required to used no more than a bandwidth of $\bar{c}$ on average for the repair process. The effect of this on n and γ is as follows.

A replica exists on a node for a time t_p after which the node leaves the system. The repair mechanism then creates a new replica after a time t_r. Therefore, on average, a new replica is created after a time $E[t_p + t_r] = E[t_p] + E[t_r] = \frac{1}{\lambda} + \frac{1}{\mu}$. If the size of S is b (bytes), this incurs an overhead of b in terms of repair traffic. If the system started out with n replicas of S, the average rate of repair traffic c is given by

$$c = \frac{nb}{\left(\frac{1}{\lambda}+\frac{1}{\mu}\right)} \quad (11)$$

$$= \frac{nb\lambda}{\left(1+\frac{1}{\gamma}\right)} \quad (12)$$

Applying a constraint of c_{max} on the repair bandwidth,

$$\frac{nb\lambda}{\left(1+\frac{1}{\gamma}\right)} \leq c_{max} \quad (13)$$

or

$$n \leq \frac{c_{max}}{b\lambda}\left(1+\frac{1}{\gamma}\right) \quad (14)$$

$$= d_{max}\left(1+\frac{1}{\gamma}\right) \quad (15)$$

where $d_{max} = \frac{c_{max}}{b\lambda}$. Intuitively, d_{max} is a normalized notion of repair bandwidth. While c_{max} expresses the bandwidth constraint in terms of bytes per second, d_{max} expresses the

same constraint in terms of replicas per node lifetime. For example, $d_{max} = 5.0$ represents a repair bandwidth that is equivalent to creating 5 replicas over an interval of $\frac{1}{\lambda}$, which is the average lifetime of a node. It is easy to verify that if the system creates d_{max} replicas of size b during an interval of $\frac{1}{\lambda}$, this translates to a bandwidth of $d_{max}b\lambda$, which is equal to c_{max}.

Unlike the other constraints, Equation 15 involves both n and γ. When equality holds, as is the case when repair bandwidth is the limiting factor in the system [3], n cannot be increased without decreasing γ, and vice versa. Thus, there exists a tradeoff between n and γ in maximizing t_s. If a *max-repair* policy is used, γ is large as possible and n is correspondingly small. Alternatively, if a *max-replicas* policy is used, n is as large as possible and γ is correspondingly small. Of course, these are two extreme points in the parameter space; n and γ could take on a range values in between as well.

Given the above constraints, we now show how to choose n and γ optimally. Let n_{min} denote the maximum number of replicas that can be created while using max-repair, and without violating the constraint on repair bandwidth. From Equation 15,

$$n_{min} = \lceil d_{max}\left(1 + \frac{1}{\gamma_{max}}\right)\rceil \tag{16}$$

If $n_{max} \leq n_{min}$, the limiting factor in the system is storage. The optimal choice of parameters is trivial: $n = n_{max}$ and $\gamma = \gamma_{max}$. This is because both n and γ are at their maximum possible values, and it is straightforward to see that Equation 15 is satisfied.

On the other hand, if $n_{max} > n_{min}$, the limiting factor in the system is repair bandwidth. The optimal value of n lies between n_{min} and n_{max}, and also satisfies equality in Equation 15. At this point, ideally, an analytic solution can be obtained by substituting for γ in $P_n(\gamma)$, and then optimization techniques can be used to derive the value of n that maximizes t_s. However, the expression for $P_n(\gamma)$ is not tractable; hence, we present numerical results instead. For a given value of n, we can compute the corresponding value of γ from Equation 15, and compute $P_n(\gamma)$. For a given d_{max}, $P_n(\gamma)$ is computed for all values $n > d_{max}$; Figure 3 plots this for $d_{max} = 3.0$ and $d_{max} = 5.0$. The results for two values of d_{max} (3.0 and 5.0) are plotted in Figure 3(a) and Figure 3(b) respectively. Recall that d_{max} is the normalized constraint on repair bandwidth.

The following trends can be observed from Figure 3. $P_n(\gamma)$ is first decreasing, reaches a minimum and then is increasing (to infinity). The minima occurs at $n = 14$ for $d_{max} = 3.0$ and $n = 242$ for $d_{max} = 5.0$. As the value of d_{max} increases, the minima shifts more and more to the right. The slow growth of $P_n(\gamma)$ after it reaches the minima can be attributed to the fact that as the repair rate becomes smaller, the effect of repair diminishes and the increase in $P_n(\gamma)$ is essentially the harmonic growth that occurs in the absence of any repair.

The shape of the graph provides a straightforward way of determining the optimal values of n and γ. The absence of any local maxima implies that, in the interval $[n_{min}, n_{max}]$, $P_n(\gamma)$ is maximum at one of the endpoints. Therefore the optimal value of n is either n_{min} or n_{max}; which one can be determined by evaluating it at both points using Equation 3. The corresponding value of γ is obtained from Equation 15. Thus, given a constraint on repair bandwidth, the optimal replication strategy that maximizes t_s is either max-repair or max-replicas; all intermediate strategies are sub-optimal.

III. Case study

In this section, we study availability data from a real large-scale distributed system - Planetlab [2]. Our objective is two-fold: (1) to validate our model, especially one of the key assumptions that node participation can be modeled by an exponential distribution, and (2) to apply the results of Section II to a realistic setting. Planetlab is an experimental testbed consisting of roughly 500 nodes distributed across the world. It is representative of a node population consisting of moderately powerful desktop machines with, by and large, good network connectivity [4].

A. Availability data

To generate availability numbers for Planetlab, we use the *all pairs ping* dataset [1], which consists of each Planetlab node pinging the others roughly 4 times an hour, over a period of 21 months (October 2003 - June 2005). Each data point in the data set consists of a time stamp, and the result of all pair-wise pings between the nodes. A ping consists of 10 probes; it is successful if at least one probe response was received, and unsuccessful otherwise. In certain cases, presumably due to data collection issues, no data is available for certain pairs.

Our methodology is as follows. At every time for which data is available, we classify each node as being either *up* or *down*. A node is up if at least one other node succeeded in pinging it; a node is down if no other node was able to ping it *and* there were at least 5 unsuccessful ping attempts to that node. A successful ping attempt indicates that a node is up. However, an unsuccessful ping attempt does not necessarily imply that a node is down; network connectivity issues may result in a failed ping. Therefore, we require a minimum number of unsuccessful ping attempts to classify a node as down, to filter out the small number of cases where the sparsity of data may erroneously result in a node being classified as down. By identifying contiguous intervals of time during which a node was either up or down, we compute uptime and downtime numbers for that node.

We plot the cumulative distribution function of uptimes and downtimes obtained in the manner; Figure 4(a) shows uptimes while Figure 4(b) shows downtimes. This is done for the Planetlab node population as a whole, as the data is

[3]We note that increasing either n and γ increases t_s. So in the absence of other constraints, equality will hold for the optimal choice of n and γ.

[4]A majority of the Planetlab nodes are hosted by academic institutions around the world.

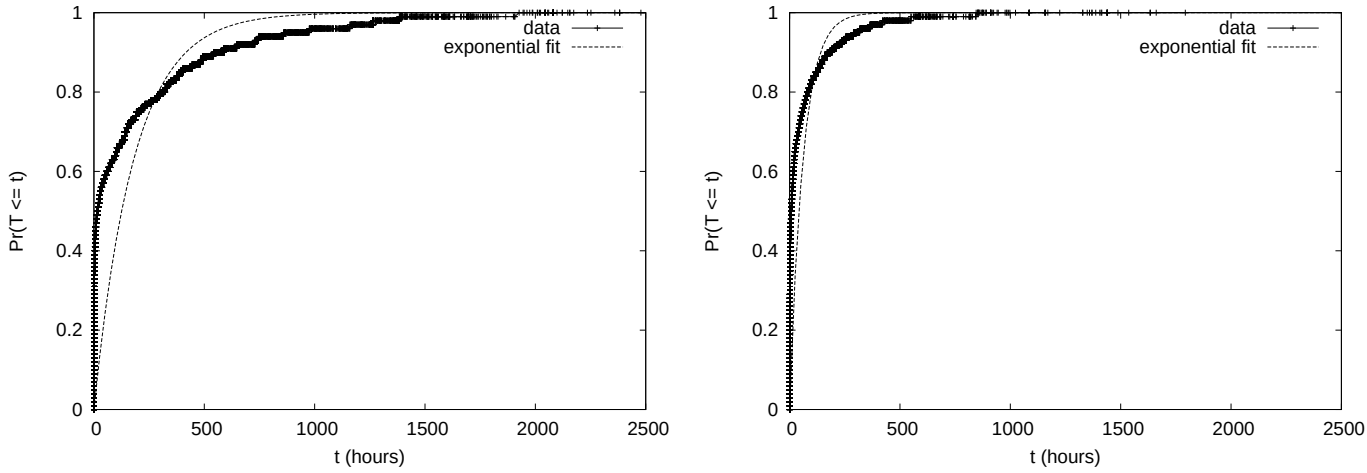

Fig. 4. Cumulative distribution of uptime and downtime for Planetlab nodes

too sparse for this to be done for each individual node. As can be seen from Figure 4, the shape of the curves indicates that an exponential distribution is a reasonable fit for both uptime and downtime. We identify the best fitting exponential distributions using maximum likelihood estimation; these are shown as well. The mean uptime is 181 hours while the mean downtime is 61 hours.

B. Discussion

We now develop a simple example to illustrate our results in the context of a real system; we consider the problem of using replication to reliably maintain state on Planetlab.

Assume the total amount of state to be maintained is 100 GB. From Section III-A, the mean uptime is 181 hours and the mean downtime is 61 hours. Therefore, out of roughly 500 nodes, the mean number of Planetlab nodes up at any point of time is $\frac{181}{181+61} * 500 \approx 374$. To be conservative, we assume the number of nodes participating is 300. If each node contributes 5 GB of storage[5], the total amount of storage is $5 * 300 = 1500$ GB. Therefore, the constraint on storage is given by $n_{max} = \frac{1500}{100} = 15$. If we assume that the system requires a minimum of half an hour on average to detect and repair a lost replica, then the constraint on repair ratio is given by $\gamma_{max} == 181 * 2 = 362$.

Finally, assume that the constraint on repair bandwidth $c_{max} = 4$ Mbps. Since the size of the replicated state is 100 GB and the mean node lifetime is 181 hours, the normalized constraint is then

$$d_{max} = \frac{4\text{Mbps} * 181\text{hours}}{100\text{GB}} = 3.18 \tag{17}$$

Now

$$n_{min} = \lceil 3.18 \left(1 + \frac{1}{362}\right) \rceil = 4 \tag{18}$$

From Section II-D, we know that the optimal value of n is either $n_{min} = 4$ or or $n_{max} = 15$. The former has a mean lifetime of 306 days. while the latter has a mean lifetime of 102 days. Thus, in this case, max-repair is the better replication strategy.

[5]5 GB is the per-slice disk quota on Planetlab nodes

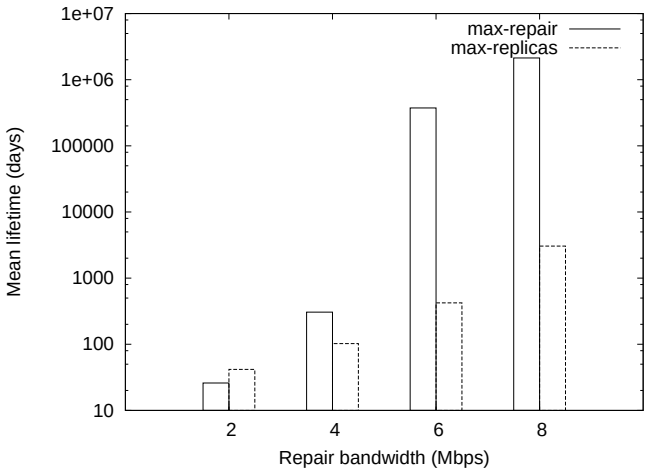

Fig. 5. Mean lifetime as a function of repair bandwidth

Figure 5 shows the mean lifetime of the system for varying repair bandwidths - 2, 4, 6 and 8 Mbps respectively. As can be seen, max-repair is better than max-replicas in all cases but one. In addition, a system-wide repair bandwidth of 6-8 Mbps is sufficient to reliably maintain 100 GB of a state for a very long time; this will scale proportionally as the size of the system increases.

To conclude this section, we consider the question of when max-repair is better than max-replicas, and vice versa. Max-replicas relies on a large number of replicas to overcome a small rate of repair; therefore, one case where it would potentially be better in systems where massive replication is possible, i.e., n_{max} is large. Recall from Section II-D that for low rates of repair, mean lifetime grows very slowly with increasing n. Therefore, this is unlikely to be a viable scenario for the networked systems we are most interested in - peer-to-peer systems and computational grids - where the degree

of replication would be limited to a few tens or hundreds at most. The other regime where max-replicas could be useful is when the available repair bandwidth is small; this can be seen from Figure 5 where for $c_{max} = 2$ Mbps, max-replicas out-performs max-repair. Even so, the mean lifetime is small because n_{max} is small.

Ultimately, max-replicas is likely to be useful only in a regime where n_{max} is large *and* c_{max} is small, i.e., massively replicated systems where communication/coordination between nodes is at a premium. Of particular interest are wireless sensor networks [7], where large numbers of cheap, unreliable wireless devices operate under constraints of limited communication due to issues of radio range and power management. A more tantalizing example is that of massively distributed applications such as *amorphous computing mediums*, which are systems of irregularly placed, asynchronous, locally interacting computational particles that may be sprinkled on a surface or mixed throughout a volume [3]. For more traditional distributed systems, the bottom line is that aggressively repairing a smaller number of replicas is the optimal approach to achieving large system lifetimes.

IV. Related Work

Our work is closely related to two distinct bodies of research: availability research as considered in the systems and networking community, and system reliability theory as considered in the performance and modeling community.

In recent years, the peer-to-peer model has emerged as a paradigm for building large-scale self-managing distributed systems. This has led to several efforts to build highly available distributed storage systems based on this paradigm; prominent among these are CFS [10], PAST [17], TRFS [5], [6], Oceanstore [14], [16] and Farsite [4]. These systems use replication and/or erasure coding to ensure that data is available despite the failure or unavailability of individual nodes in the system. While certainly similar in spirit, our work differs on the following counts. First, as mentioned in Section I, the metric we are interested in is the lifetime of the state in the system, as opposed to its fractional availability. Second, we incorporate notions of resource limitations like storage and bandwidth constraints in our model, and study their impact on the choice of system parameters. Previous studies largely ignore this, focusing instead on choosing parameters solely based on the target level of availability. In this respect, the analysis in [8] is closest to our work; however it uses fractional availability as the metric.

This work is also inspired by work in system reliability theory [15], which attempts to model the failure properties of multi-component systems. In particular our model of Section II-A is similar to that in [9], which deals with analyzing the time-to-failure of a parallel system of components with repair. However, the key difference is that we also attempt to model the effect of practical considerations that are specific to the domain in question, namely large-scale distributed systems.

V. Conclusion

In this paper, we have addressed the question of how to engineer a distributed system that uses replication to reliably maintain state that far outlives the individual nodes on which it resides. For example, such state could be either data, as in the case of peer-to-peer storage, or computation, as in the case of computational grids. Towards this objective, our contributions are as follows:

- We developed a simple Markov model that expresses the essential features of a replicated system, including notions of replica loss and replica repair, in terms of a few key parameters.
- We provided an analysis of the above model, including an expression for the mean lifetime of replicated state in terms of those parameters.
- We developed an analysis of the effect of resource constraints on the system, including a method to optimally choose system parameters so as to maximize lifetime. Our analysis reveals that, for most practical scenarios, it is better to invest the available repair bandwidth in aggressively maintaining a small number of replicas than spreading it thin across a large number of replicas.

References

[1] All pairs ping dataset. *pdos.csail.mit.edu/ strib*.

[2] Planetlab. *www.planet-lab.org*.

[3] H. Abelson, D. Allen, D. Coore, C. Hanson, E. Rauch, G.J. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 2000.

[4] A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. *In Proceedings of USENIX OSDI*, 2002.

[5] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. *In Proceedings of IPTPS*, 2003.

[6] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker. Total recall: System support for automated availability management. *In Proceedings of USENIX NSDI*, 2004.

[7] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan. Upper bounds on the lifetime of sensor networks. *In Proceedings of IEEE ICC*, 2001.

[8] C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. *In Proceedings of HotOS IX*, 2003.

[9] M. Brown. The first passage time distribution for a parallel exponential system with repair. *Reliability and Fault Tree Analysis*, 1974.

[10] F. Dabek, F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. *In Proceedings of ACM SOSP*, 2001.

[11] R. Epstein. *The Theory of Gambling and Statistical Logic*. Academic Press, 1977.

[12] W. Feller. *An Introduction to Probability Theory and Its Applications*. John Wiley and Sons, 1968.

[13] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[14] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinsk, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. *In Proceedings of ACM ASPLOS*, 2000.

[15] M. Rausand and A. Hoyland. *System Reliability Theory: Models, Statistical Methods, and Applications*. John Wiley & Sons, 2004.

[16] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the oceanstore prototype. *In Proceedings of USENIX FAST*, 2003.

[17] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. *In Proceedings of ACM SOSP*, 2001.

APPENDIX

In this section, we derive an expression for the expected lifetime t_s, i.e., the average time it takes for the system to reach state 0, starting in state n. We analyze the continuous Markov model presented in Section II-B by considering its discrete embedding. In the discrete case, the transitions betweens states are labeled with probabilities instead of rates. From state k, there are two possible transitions: to state $k-1$ with probability p_k and to state $k+1$ with probability q_k. The transition probabilities p_k and q_k are functions of λ, μ and k.

The derivation uses the concept of an *epoch*, which is as follows. At the start of an epoch, the system is in state n. At the end of an epoch, the system either returns to state n, in which case the next epoch starts, or the system reaches state 0. The system goes through a number of epochs, and in each epoch except for the last one, returns to state n. In the last epoch, the system goes to state 0. The derivation proceeds in two steps: computing the expected *number* of epochs n_e, and computing the expected *duration* t_e of each epoch. The expected lifetime t_s is then obtained as the product of n_e and t_e.

We note that each epoch is essentially an instance of the Gambler's Ruin Problem. Starting in state n, the system always makes the transition to state $n-1$ upon the first failure. After this, the behavior of a system in the epoch is similar to that of a gambler starting with a fortune of $n-1$ and playing till he makes a fortune of n, or alternatively, is ruined. The key difference is that the transition probabilities are not constant but state dependent.

A. Number of epochs

Let Q_k be the probability that the system reaches state 0 before state n starting in state k. Clearly Q_0 = 1 and $Q_n = 0$. For $0 < k < n$, Q_k satisfies the recurrence

$$Q_k = p_k Q_{k-1} + q_k Q_{k+1}$$

where

$$\begin{aligned} p_k &= \frac{k\lambda}{k\lambda + (n-k)\mu} \\ q_k &= \frac{(n-k)\mu)}{k\lambda + (n-k)\mu} \end{aligned}$$

where p_k and q_k are the transition probabilities in state k. Applying $p_k + q_k = 1$,

$$(p_k + q_k)Q_k = p_k Q_{k-1} + q_k Q_{k+1}$$

and rearranging terms,

$$p_k(Q_{k-1} - Q_k) = q_k(Q_k - Q_{k+1})$$

which yields

$$\begin{aligned} Q_{k-1} - Q_k &= \frac{q_k}{p_k}(Q_k - Q_{k+1}) \\ &= \frac{n-k}{k}\left(\frac{\mu}{\lambda}\right)(Q_k - Q_{k+1}) \\ &= \frac{n-k}{k}\gamma\,(Q_k - Q_{k+1}) \end{aligned}$$

where $\gamma = \frac{\lambda}{\mu}$. The quantity of interest is Q_{n-1}, i.e., the probability that the system reaches state 0 before state n starting in state $n-1$. Denote this quantity by Q^*. Therefore

$$Q_{n-1} - Q_n = Q^*(Q_n = 0)$$

Repeatedly applying the above recurrence equation,

$$\begin{aligned} Q_{n-2} - Q_{n-1} &= \frac{1}{n-1}\gamma\,(Q_{n-1} - Q_n) \\ &= \frac{1}{\binom{n-1}{1}}\gamma^1 Q^* \\ Q_{n-3} - Q_{n-2} &= \frac{2}{n-2}\gamma\,(Q_{n-2} - Q_{n-1}) \\ &= \frac{2}{n-2}\gamma\frac{1}{\binom{n-1}{1}}\gamma^1 Q^* \\ &= \frac{1}{\binom{n-1}{2}}\gamma^2 Q^* \end{aligned}$$

and so on. Therefore, proceeding inductively,

$$Q_{n-k-1} - Q_{n-k} = \frac{1}{\binom{n-1}{k}}\gamma^k Q^*$$

Now

$$Q_0 - Q_n = 1 - 0 = 1$$

and

$$\begin{aligned} Q_0 - Q_n &= \sum_{k=0}^{k=n-1}(Q_{n-k-1} - Q_{n-k}) \\ &= Q^* \sum_{k=0}^{k=n-1}\frac{1}{\binom{n-1}{k}}\gamma^k \end{aligned}$$

which implies

$$Q^* = \left(\sum_{k=0}^{k=n-1}\frac{1}{\binom{n-1}{k}}\gamma^k\right)^{-1}$$

The probability that an epoch will result in the number of replicas going down to 0, i.e., the probability that it will be the last epoch, is Q^*. Therefore, the expected value of the number of epochs is given by

$$n_e = \frac{1}{Q^*}$$

B. Duration of epoch

Let T_k be the expected *time* that elapses before the system reaches either state 0 or state n starting in state k. Clearly $T_0 = 0$ and $T_n = 0$. For $0 < k < n$, T_k satisfies the recurrence

$$T_k = p_k T_{k-1} + q_k T_{k+1} + t_k$$

where

$$t_k = \frac{1}{k\lambda + (n-k)\mu}$$

Applying $p_k + q_k = 1$,

$$(p_k + q_k)T_k = p_k T_{k-1} + q_k T_{k+1} + t_k$$

and rearranging terms,

$$p_k(T_{k-1} - T_k) \quad = \quad q_k(T_k - T_{k+1}) - t_k$$

which yields

$$\begin{aligned}(T_{k-1} - T_k) &= \frac{q_k}{p_k}(T_k - T_{k+1}) - \frac{t_k}{p_k} \\ &= \frac{n-k}{k}\left(\frac{\mu}{\lambda}\right)(T_k - T_{k+1}) - \frac{1}{k\lambda} \\ &= \frac{n-k}{k}\gamma(T_k - T_{k+1}) - \frac{1}{k\lambda}\end{aligned}$$

The quantity of interest is T_{n-1}, i.e., the expected time that elapses before the system reaches either state 0 or state n starting in state $n-1$. Denote this quantity by T^*. Therefore

$$T_{n-1} - T_n \quad = \quad T^*(T_n = 0)$$

Repeatedly applying the above recurrence equation,

$$\begin{aligned}T_{n-2} - T_{n-1} &= \frac{1}{n-1}\gamma\left(T_{n-1} - T_n\right) - \frac{1}{n-1}\frac{1}{\lambda} \\ &= \frac{1}{\binom{n-1}{1}}\gamma^1 T^* - \frac{1}{n\lambda}\frac{\binom{n}{1}}{\binom{n-1}{1}}\gamma^0 \\ T_{n-3} - T_{n-2} &= \frac{2}{n-2}\gamma\left(T_{n-2} - T_{n-1}\right) - \frac{1}{n-2}\frac{1}{\lambda} \\ &= \frac{1}{\binom{n-1}{2}}\gamma^2 T^* - \frac{1}{n\lambda}\frac{\binom{n}{1}}{\binom{n-1}{2}}\gamma^1 - \frac{1}{n\lambda}\frac{\binom{n}{2}}{\binom{n-1}{2}}\gamma^0\end{aligned}$$

and so on. Therefore, proceeding inductively,

$$T_{n-k-1} - T_{n-k} \quad = \quad \frac{1}{\binom{n-1}{k}}\gamma^k T^* - \frac{1}{n\lambda}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}$$

Now

$$T_0 - T_n = 0 - 0 = 0$$

and

$$\begin{aligned}T_0 - T_n &= \sum_{k=0}^{k=n-1}\left(T_{n-k-1} - T_{n-k}\right) \\ &= T^*\left(\sum_{k=0}^{k=n-1}\frac{1}{\binom{n-1}{k}}\gamma^k\right) - \frac{1}{n\lambda}\sum_{k=0}^{k=n-1}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j} \\ &= \frac{T^*}{Q^*} - \frac{1}{n\lambda}\sum_{k=0}^{k=n-1}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\end{aligned}$$

which implies

$$T^* \quad = \quad Q^*\left(\frac{1}{n\lambda}\sum_{k=0}^{k=n-1}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\right)$$

The expected value of the epoch length is given by

$$\begin{aligned}t_e &= t_n + T^* \\ &= \frac{1}{n\lambda} + Q^*\left(\frac{1}{n\lambda}\sum_{k=0}^{k=n-1}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\right) \\ &= \frac{Q^*}{n\lambda}\left(\frac{1}{Q^*} + \sum_{k=0}^{k=n-1}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\right) \\ &= \frac{Q^*}{n\lambda}\left(\sum_{k=0}^{k=n-1}\frac{1}{\binom{n-1}{k}}\gamma^k + \sum_{k=0}^{k=n-1}\sum_{j=1}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\right) \\ &= \frac{Q^*}{n\lambda}\left(\sum_{k=0}^{k=n-1}\sum_{j=0}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\right)\end{aligned}$$

C. Expected lifetime

The expected value of the lifetime is given by

$$\begin{aligned}t_s &= n_e\, t_e \\ &= \frac{1}{n\lambda}\sum_{k=0}^{k=n-1}\sum_{j=0}^{j=k}\frac{\binom{n}{j}}{\binom{n-1}{k}}\gamma^{k-j}\end{aligned}$$

Rearranging terms to write t_s as a polynomial in γ gives us our final expression

$$\begin{aligned}t_s &= \frac{1}{\lambda}P_n(\gamma) \\ &= \frac{1}{\lambda}\left(c_{0,n}\gamma^0 + c_{1,n}\gamma^1 + \ldots + c_{n-1,n}\gamma^{n-1}\right)\end{aligned}$$

where the coefficient of γ^i is given by

$$c_{i,n} = \frac{1}{n}\sum_{j=0}^{j=n-i-1}\frac{\binom{n}{j}}{\binom{n-1}{i+j}}$$