

Augmenting Overlay Trees for Failure Resiliency

Jeremy Silber*, Sambit Sahu*, Jatinder Singh[†] and Zhen Liu*

*IBM T.J. Watson Research Center
{jsilber, sambits, zhenl}@us.ibm.com

[†]Stanford University
jatinder@stanford.edu

Abstract—Overlay trees typically use directed trees as efficient structures for disseminating information, but their single-path structure means that just one node failure results in the disconnection of all descendants, possibly a significant portion of the graph. The addition of extra “backup” links to a directed tree can provide alternate data paths that significantly reduce the number of nodes disconnected when some set of nodes are removed from the graph. We investigate several deterministic and randomized algorithms for adding such backup links to a directed tree and analyze the connectedness of the resulting graphs when nodes in the network fail with some random probability. We present closed-form approximations and simulation measurements for the connectivity of these augmented trees in networks ranging from hundreds to hundreds of thousands of nodes. We also identify and measure the costs of adding backup links, using simulations and real-world measurements from overlays constructed using PlanetLab latency data. We find that, with node failure rates up to 10%, deterministic backup link selection policies offer comparable resiliency to random backup links with significantly lower overhead and resource usage.

I. INTRODUCTION

Multicast overlay networks typically use directed trees to distribute copies of data among a large number of receivers. Both internal nodes and leaf nodes are typically users’ computers on the public Internet, and so both the nodes and the connections between them are prone to failures and unexpected unavailability. The absence of redundant links in directed trees makes them extremely vulnerable to node (or edge) failures. Removing any node from an overlay tree breaks the connectedness of the network, leaving all descendant nodes disconnected from the tree’s root. This vulnerability is of real concern; one user switching off his PC should not cause hundreds of others to lose connectivity to the overlay network. Overlay reconfiguration due to disconnected nodes requires considerable protocol overhead, and has a significant effect on the long-term performance of overlay networks.

When using tree topologies for propagating information in a network, we would like to minimize sensitivity to such catastrophic failures. We investigate the “augmentation” of a tree by addition of redundant links, such that each node in the graph be assigned one or more “backup” parent nodes in addition to its parent in the original tree. This is conceptually equivalent to resilience schemes used in optical networks, and some overlay networks[5]. In the event that a node loses connection to its tree parent, it may activate a “backup link” to one of these backup nodes. These backup connections may be created only as needed, or created ahead of time and activated

when needed. Costs associated with finding backup parents, creating and maintaining backup links, and the possible loads on the network when nodes switch over to backup connections make it important to choose backup parents carefully.

Of the several backup parent selection policies we evaluate, there are two categories: randomized and deterministic. Randomized strategies involve the selection of backup parents randomly from among some subset of the nodes in the network. Deterministic strategies select one or more backup parents with some specified relationship to the node in question. Previous research has shown some beneficial characteristics of randomized strategies, but without analysis and comparison to deterministic alternatives.

Our primary objective in assigning backup parent nodes is to increase the resiliency of the network to random failures. We introduce *connectivity* as a statistical measure of how well a network handles node failures. The connectivity of a network describes the proportion of the nodes which are still connected to the tree root after some failures have occurred. The resiliency gained by augmenting a tree can be measured by comparing its connectivity to that of the original tree given the same set of node failures. We derive mathematical approximations for connectivity in trees and augmented trees in Section IV, and observe connectivity results from simulations in Section V.

When comparing backup parent selection strategies, we must compare not only their resulting resilience to failures, but also the costs associated with creating, maintaining, and using the backup links. In Section III, we identify and discuss three measures of this cost: length of backup paths, (un)fairness of backup load distribution, and backup path maintenance overhead. Section V reports measured backup path lengths in augmented overlay trees created using real-world data from the PlanetLab All-Pairs Ping project[16], and backup load distribution measured through simulation.

II. RELATED WORK

Tree-based data delivery has been extensively studied in the literature for group communication, with the receivers organized in a tree structure rooted at the source node. RMTP[12], TMTP[18], and STORM[17] manage tree-based reliable group communication in the network layer, while Narada[8] and NICE[4] use application-level forwarding for compatibility with existing network infrastructure. In both cases, end-hosts form a spanning tree among the nodes in the network, where

an edge in the tree denotes a point-to-point connection between two end-systems.

PRM[5] proposes a combination of proactive and reactive mechanisms for handling failures in an application-level multicast tree, probabilistically forwarding data and informational headers along randomly selected backup links. In BitTorrent[9], Slurpie[14], and Bullet[10] the use of random peer nodes is advocated to improve data transfer rates. BitTorrent uses a centralized “Tracker” component to randomly select a list of peer nodes to serve content to a downloader. Slurpie seeks to improve this performance by biasing the random choice toward nodes closer to the downloader. Bullet improves the incoming bandwidth to a node by choosing a set of random nodes as its peer for the download such that load is distributed more evenly across the nodes. The random peer selection component of each of these approaches requires either a central registry of all the participating nodes or a protocol for distributing random node addresses to all nodes. This requirement can prove prohibitive in large networks. It is not clear whether these random policies offer significant advantages over deterministic policies for choosing these peer connections. In this study, we compare the resiliency provided by several deterministic peer-selection policies to random choice from among all peers.

SplitStream[7] and CoopNet[13] address multicast tree failures by spreading data over multiple trees (possibly using loss-resilient coding). Both systems provide a centralized algorithm for constructing a *forest* of trees to minimize the disruption caused by any individual node failure, and heuristically attempt to create trees with low-latency links. In contrast, our approach takes a tree created by any algorithm (e.g. one with minimal end-to-end latency, or maximum end-to-end bandwidth) and heuristically reinforces the tree against node failures.

Many studies in network survivability [2][3] have provided graph theoretic results guaranteeing that a network survives a given type of failure. In our study, we focus on a more statistical metric: the fraction of nodes that remain connected as a result of a given node failure process.

The field of social networks has approached graph connectivity problems from a sociological angle, and provides several results on random augmentation of various types of regular graphs [6], [15], [1].

III. PROBLEM FORMULATION

To evaluate the response of a network to node failures, we utilize a simple model of node failures and recovery mechanisms. The node failure model is binary, with each node either operational or failed, according to independently random processes. In addition to its parent in the tree, each node has a (possibly empty) list of backup parents stored locally (a network edge between the node and a backup parent is referred to as a “backup connection” or “backup link”). When a node’s parent fails, the node is disconnected from the root. It can detect the failure and take steps to reconnect by activating a backup link. After repairs have completed, a node

is determined to be connected to the root if and only if 1) it has not failed and 2) one or more of its tree or backup parents is connected to the root. We assume that the root node never fails (such a case would offer no insight into building better tree augmentation structures).

Nodes are organized in full, k -ary trees, which have a degree of k (k children) at each internal node, and leaf nodes only at the lowest level of the tree. Our most basic network is just this unaltered tree, where each node (other than the root) has one parent and no backup parents, thus no repair process is possible. We also augment the basic tree by adding backup parents at each node according to different policies, creating backup links that can be used to repair the connections if the node becomes disconnected.

A. Performance Metrics

We use the following metrics to evaluate the performance and costs of the augmented trees under different levels of node failure:

1) *Connectivity*: We consider the most basic measure of a network’s ability to operate despite failures to be the connectivity C , defined as the ratio of number of nodes reachable from the root and $n - 1$, where n is the total number of nodes in the tree.

The difference between the connectivity of a specific network and that of a simple tree (under the same probability of node failure) is a direct measure of the benefit of the backup links. Higher connectivity at the same level of failure distinguishes a more failure-resilient network.

2) *Backup Path Lengths*: In overlay networks, some graph edges are much more costly than others. The simplest measure of this cost is the latency experienced by messages traversing the link. All else being the same, shorter backup path lengths are preferable to longer ones for several reasons. First, shorter backup paths can be initiated and activated faster than longer ones; the three-way handshaking of TCP requires time proportional to the path RTT, as would the transmission of a message sent to wake an existing backup link from an idle state. Second, shorter backup paths mean lower propagation delays. Minimal end-to-end delivery time is almost always preferable, and becomes critical in the case of real-time or interactive applications. Finally, shorter backup paths usually entail fewer routing hops on the underlying network, signifying a more efficient usage of network resources.

A reasonable overlay tree construction algorithm would take the cost of each connection into account when building the tree, usually to minimize (subject to various constraints and approximations) the average or maximum path length from the root to all nodes. Given this type of tree, we believe that some backup link selection policies will tend to choose shorter backup links than others. For example, we expect that links to “cousins” in the tree will, on average, be shorter than links randomly selected from the entire network.

3) *Backup Load Distribution*: The backup link selection mechanism determines how requests are distributed to backup parents after failures occur; a mechanism that causes many

nodes to make backup requests to a few backup parents can create significant problems. Incoming backup requests may easily increase a node's degree past the point where it does not have enough bandwidth to send data to all its children.

To measure the extent to which the load of backup requests is distributed fairly among the nodes of the network, we examine the distribution of node degrees after a set of failures has occurred and repairs have taken place. More uniform distributions of load are generally preferable, and the presence of outliers with exceedingly large degrees is especially concerning. To quantify this cost, we record the maximum degree of a node in the network after repairs, and average this result across trials

4) *Backup Path Maintenance Overhead*: The final measure of backup link cost is the overhead necessary to select the backup links and update them as the graph changes. This overhead is difficult to measure directly, but is related to the number of backup parents each node has, the size of the candidate set from which a node must choose its backup parents, and the distance from the node to the members of this candidate set. Making and maintaining more backup links requires more time and resources, so the number of backup links is a good first-order estimate of this cost. Candidate set size is also important: if each node must choose from among a large set of backup nodes, then a correspondingly large amount of information probably must be sent to that node. Similarly, if a node's backup links are very distant in the tree, information about them must be propagated correspondingly far (i.e. over many hops in the original tree), and thus the distance from the node to the candidate backup parents affects cost.

Randomized strategies in particular can have significant overhead, since the backup parent assignments requires a uniformly random selection from among the entire list of nodes. This requires either knowledge at one or more assigning nodes of the entire set of nodes, or a mechanism by which random subsets of node address information may be transmitted to all network nodes. The first option imposes limits on the scalability of the network, while the second requires bandwidth overhead.

Propagation overhead may be mitigated by the addition of one or more specialized nodes, such as the tracker in the Bittorrent protocol [9], but this adds complexity and imposes scalability limits. Specialized algorithms exist to mitigate the cost of random registry dissemination, like the random-node propagation algorithm used in [10], but some amount of overhead is inherent in the approach.

We are not able to measure these costs in a general way (in the context of a specific overlay implementation, the control overhead required by each policy would be a good estimator). We use the number of backup links as a first-order estimate, and so we only directly compare augmentation schemes against others that use the same number of backup links per node. While we cannot quantify the protocol overheads inherent in random policies, we believe that they are significant.

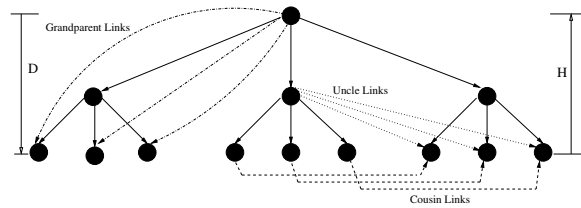


Fig. 1. Diagram of the augmentation policies investigated.

B. Augmentation Policies

We investigate the following augmentation policies in this study (in addition to the simple tree which served as a baseline). Each node's backup parent set is determined according to the policy given. b is a constant representing the number of backup edges directed toward a typical node.

- **Grandparent** - A node's backup parent set contains the parent of the node's parent. The children of the root node have no grandparent, and thus no backup parents. Since each node has at most one grandparent, b cannot be greater than 1.
- **Uncle** - A node's backup parent set contains the b siblings of the node's parent. The children of the root node have no uncles, and thus no backup parents. Since the parent of a node has only $k - 1$ siblings, b cannot be greater than $k - 1$.
- **Cousin** - A node's backup parent set contains one child of each of b of the node's uncles. The children of the root node have no cousins, and thus no backup parents. Since a node can have only $k - 1$ uncles, b cannot be greater than $k - 1$.
- **Random** - Adds edges to every node from b nodes selected uniformly at random from among the entire set of nodes.

The deterministic policies selected would be straightforward to implement with strictly local information at each node, since each uses a nearby relative in the tree. Backup parent address information would only have to travel a limited number of hops on the overlay network, and so should not add significant overhead to overlay control protocols. We selected one policy which selected backup parents from the node's tree level (cousin), one with backup parents one level higher than the node (uncle), and one with backup parents two levels higher than the node (grandparent).

IV. CONNECTIVITY APPROXIMATION

In this section, we derive analytical expressions to approximate network connectivity for both regular trees and regular trees augmented according to the deterministic policies presented in the previous section. In our analysis, the nodes in the topology are assumed to fail independently and are taken to have an identical failure distribution. The structural symmetry of the network at each level of the tree is used to estimate the expected fraction of nodes that remain connected under random node failures. We also assume that the disconnection of a node is independent of disconnection of its chosen backup

node. These assumptions allow us to write recursive relations to derive simple expressions for connectivity. We validate these analytical expressions with simulation results.

A. Regular Tree

We observe that by virtue of symmetry in a regular tree-based structure, the probability of a node being connected is identical for all nodes at a specific depth.

The probability of a node at depth d being connected depends on the connectivity status of the parent node and the backup nodes, if present. We denote by $\mathcal{P}(d)$, the probability that a node at depth d is connected. The tree root ($d = 1$) is assumed not to fail, so $\mathcal{P}(1) = 1$. The number of nodes at depth d equals $k^{(d-1)}$ and is denoted by $n(d)$. The number of connected nodes in a tree-based structure of total depth D is then given by

$$Nc(D) = \sum_{d=2}^D \mathcal{P}(d)n(d) \quad (1)$$

We define the connectivity of a tree-based structure of depth D to be the fraction of nodes in the structure that are connected. The number of nodes (excluding the root) in a tree of depth D is $N(D) = \frac{k^D - k}{k - 1}$. Then the connectivity $C(D)$ can be expressed as

$$C(D) = \frac{Nc(D)}{N(D)} \quad (2)$$

We next evaluate the connectivity expression for various tree-based structures. Node failure is modeled as a Bernoulli event with probability p_n . Nodes in the topology are assumed to fail independently.

For a simple tree structure, a node at a given depth in the tree is connected if and only if it has not failed and its parent is connected. If p_n is the node failure probability,

$$\mathcal{P}(d) = \bar{p}_n \mathcal{P}(d-1), d \geq 3 \quad (3)$$

With the boundary condition, $\mathcal{P}(2) = \bar{p}_n$, we derive from (1), (2) and (3) the expression for connectivity of a simple tree structure:

$$C(D) = \frac{(k\bar{p}_n)^{D-1} - 1}{k\bar{p}_n - 1} \frac{k-1}{k^D - k} \quad (4)$$

B. Regular Tree with Augmentation

Next we consider the case where one backup link is introduced symmetrically at each node in a regular tree. Figure 1 shows sample backup links introduced in a tree of height and depth 3, as per the Uncle, Cousin and Grandparent policies.

For the Cousin policy, a node at depth d is connected if it has not failed and either its parent (at depth $d-1$) or the cousin (at depth d) is connected. That is,

$$\mathcal{P}(d) = \bar{p}_n(1 - (1 - \mathcal{P}(d-1))(1 - \mathcal{P}(d))), d \geq 3 \quad (5)$$

For a cousin tree-based structure with given depth D and degree k , equations (1), (2) and (5) can be numerically solved using the boundary condition $\mathcal{P}(2) = \bar{p}_n$, to obtain connectivity.

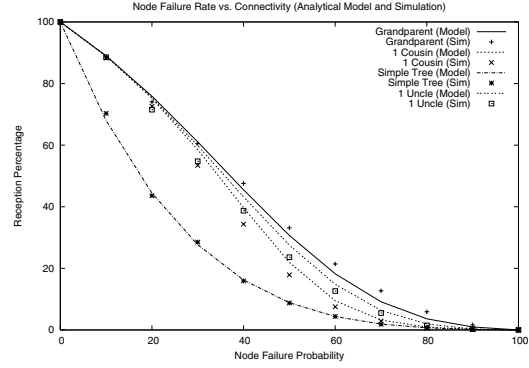


Fig. 2. Validation of analytical model against simulation results, with $k = 4$, $d = 5$.

For the Uncle policy, a node at depth d is connected if it has not failed and either its parent or uncle (both at depth $d-1$) is connected. The probability of a node at depth d being connected is then given by,

$$\mathcal{P}(d) = \bar{p}_n(1 - (1 - \mathcal{P}(d-1))(1 - \mathcal{P}(d-1))), d \geq 3 \quad (6)$$

Proceeding in a similar fashion, for the Grandparent policy, a node is connected if it has not failed and either its parent or the grandparent (depth $d-2$) is connected. That is,

$$\mathcal{P}(d) = \bar{p}_n(1 - (1 - \mathcal{P}(d-1))(1 - \mathcal{P}(d-2))), d \geq 3 \quad (7)$$

With slightly different boundary conditions $\mathcal{P}(2) = \bar{p}_n$ and $\mathcal{P}(1) = 1$, (1), (2), and (7) provide a numerically solvable approximation formula for a tree with Grandparent backup links.

C. Model Validation

The connectivity expressions derived in this section are compared with simulation data in Figure 2. The model proves accurate for the simple tree and for the augmented trees with node failure rates up to 20%. After that point, the assumption of independence between node disconnection and backup parent disconnection seems to produce inconsistencies. The analytical expressions should be useful for approximating connectivity for low failure rates in very large networks, where simulation is difficult or impossible. To avoid any estimation error, however, the remainder of this paper discusses results from the simulator.

V. EXPERIMENTAL RESULTS

To validate our approximation equations and analyze characteristics not feasible in the numerical models, we created a network connectivity simulator. The simulator generates simple trees and augments them according to the backup policies listed in Section III-B. This forms a set of augmented trees, each of which has a different set of edges among the same set of nodes. In each trial, a subset of the nodes is chosen to fail, and these nodes (and incident edges) are removed from each of the augmented trees. The simulator determines the set of nodes still reachable from the root through tree or backup

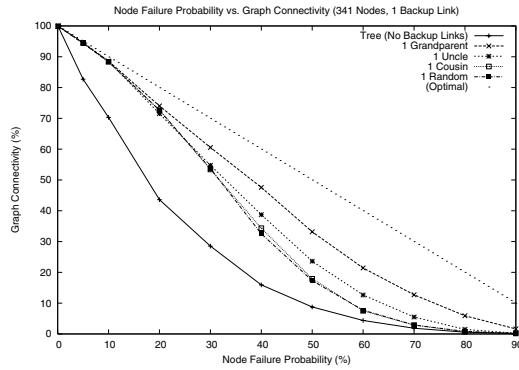


Fig. 3. Connectivity benefit of 1 backup link. Averaged over 100 simulations with 4 tree children per node, depth of 5 (341 nodes).

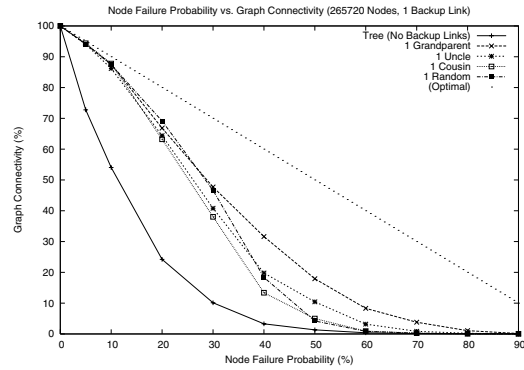


Fig. 5. Resiliency benefit of 1 backup link. Averaged over 20 simulations with 3 tree children per node, depth of 8 (3,280 nodes).

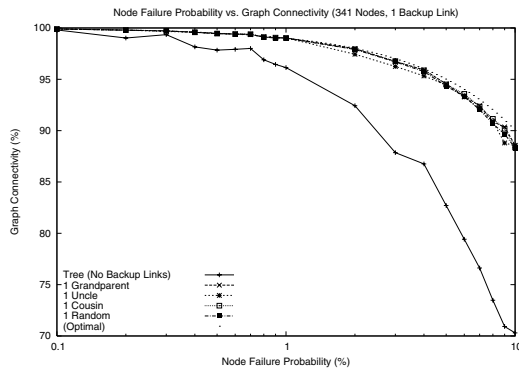


Fig. 4. Detail of Figure 3, plotted on a log scale.

links, and records the resulting degree of each node (i.e. the number of tree and backup children of each node).

Most results presented below are from simulations on networks of 341 nodes, based on a tree with 4 children per node and depth 5. We believe that this network size is representative of a medium-sized overlay, and is similar in size to the PlanetLab network (from which we obtain some data for our analysis in Section V-B). We also describe results obtained from a network approximately 10 times larger (3280 nodes, 3 children per node, depth 8), and one approximately 1000 times larger (265,720 nodes, 3 children per node, depth 12). Since no real-world latency information was available for networks of this scale, backup path length analysis was not possible for these cases. Sections V-A, V-C, and V-B present results which are summarized in Section V-D.

A. Connectivity

Figure 3 shows the connectivity achieved by a simple tree and the same tree augmented with one backup link per node according to the policies given in Section III-B. The dotted line marked 'Optimal' represents the percentage of nodes which are still functional, and therefore the maximum connectivity attainable by any backup policy. The tree used in this case contains 341 nodes, has a depth of 5, and 4 children per internal node. Each data point reflects the average measured over 100 independent trials.

The first result to note is the steep decline in connectivity experienced by the simple tree. At a node failure rate of five percent, almost twenty percent of nodes are disconnected from the root. This degradation is clearer in Figure 4, which plots the same results for node failure rates between 0 to 10% (on a logarithmic scale). Such sensitivity to relatively low rates of node failure rates makes it very difficult to maintain connectivity in a tree-based overlay.

The backup policies all show similar connectivity when node failure is in the 0 to 10% range, but begin diverging somewhat after that point. At node failure rates exceeding 20%, the grandparent policy results in the highest connectivity scores. At rates above 30% the uncle policy diverges somewhat from the cousin and random policies, though it converges again near 80% node failure. Throughout the entire range, these simple backup policies show profound connectivity improvements over a simple tree. The gains are especially high for node failure rates below 50%, and connectivity for all backup policies is near optimal for node failure rates below 10%. This suggests that real-world overlay networks can benefit a great deal from just one backup link established according to any of the given policies.

In larger networks (Figures 5 and 6), we observe similarly shaped resiliency curves, with some interesting variations. In the 3280-node network, the random policy performs somewhat better for failure probabilities below 40%, then falls back in line with the cousin policy. In the very large network (265,720 nodes), the random policy outperforms even the grandparent policy for failure rates below 40%. We believe that as tree depth increases, the random policy's ability to connect across many tree levels proves useful. For failure rates less than 10%, however, 1 backup parent selected according to any policy results in equivalent connectivity.

B. Backup Path Length Analysis

To analyze the relative lengths of backup paths created by the given tree augmentation policies, we used them to augment overlay trees created using experimental data from the PlanetLab[11] All Pairs Ping project[16]. From this data we created a ping matrix, where each (non-missing) entry

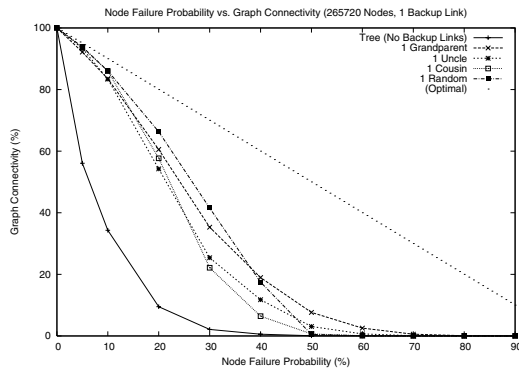


Fig. 6. Resiliency benefit of 1 backup link. Averaged over 20 simulations with 3 tree children per node, depth of 12 (265,720 nodes).

$X[i][j]$ corresponds to the minimum latency observed from node i to node j . We then generated overlay trees using a simple greedy algorithm, which begins with a random root node and assigns the closest k children to each parent until all nodes are connected. After running each of the backup-link augmentation policies on the resulting overlay trees, we measured the length ($X[a][b]$) of each backup link ($a \rightarrow b$) and averaged over all backup links. Table I shows representative results for trees over the PlanetLab dataset with k children per node and depth d .

Type	(k,d)			
	(2,8)	(3,6)	(4,5)	(5,5)
Cousin	44.1	53.0	52.7	52.9
Uncle	63.9	69.1	77.1	79.5
Grandparent	73.5	83.6	93.2	98.1
Random	91.9	91.3	91.0	91.6

TABLE I

AVERAGE LENGTH OF BACKUP PATHS IN PLANETLAB OVERLAY (MS),
AVERAGED OVER 100 DIFFERENT OVERLAY ROOTS.

Most significantly, we see that purely random backup selection results in the longest backup paths, while the deterministic approach of choosing cousin nodes results in the shortest. This follows logically from the idea that, with a tree construction algorithm that makes a reasonable attempt to minimize delays, nearby relatives in the tree should be relatively close in the underlying network. It is reasonable to think that cousins which a fixed distance from each other in the overlay tree will be closer in the underlying network than randomly (un)related nodes that will be on average much further away in the overlay. In fact, the average backup path length seems to increase as distance in the overlay tree increases.

Backup path length analysis for the 3,280- and 265,720-node networks was not possible, since we had no source of such a large all-pairs ping dataset. However, we believe the differences between the deterministic and randomized policies noted above would be even more exaggerated. Since more points in the underlying network would be members of the overlay, nearby relatives in the overlay tree would be even closer. Meanwhile, random links would still be arbitrarily long.

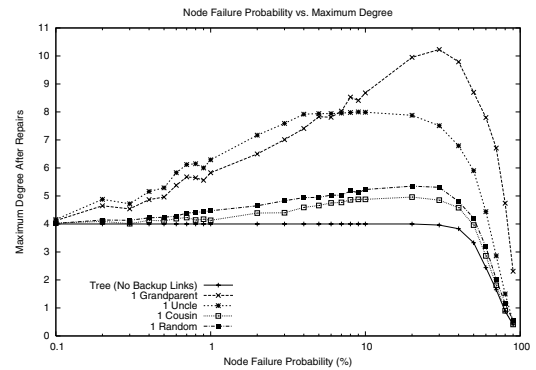


Fig. 7. Maximum node degree over all nodes, averaged over 100 trials, after repairs according to the different augmentation policies. The original tree has 4 children per internal node and depth of 5.

It certainly seems unlikely that the relative cost of random links would be any lower.

C. Backup Load Distribution Analysis

To quantify the extent to which different backup node selection policies distribute backup load unevenly, we measured the degree of each node after failures and repairs had occurred. Figure 7 shows the maximum number of children supported by a single node after repairs according to each backup policy. These results reflect our expectations of backup load distribution properties of the different policies. The simple tree has no repair process and therefore the maximum degree never rises past the initial degree (four). The grandparent policy, which we expected to distribute backup load unfairly, indeed shows a high maximum degree after repairs, especially for node failure rates around 20%. In an overlay using this policy, some nodes that usually forward to 4 children will receive backup requests for 6 more, requiring 250% of the bandwidth used in the original tree.

The uncle policy distributes backup load more evenly than the grandparent policy, but still considerably worse than the cousin and random policies. The ordering among all these policies may be a direct result of the extent to which each relies on nodes higher in the tree as backup parents. The grandparent policy goes up two levels in the tree to find a backup, and so none of the backup load is borne by the nodes at the bottom two levels (which constitute the vast majority of all nodes). The uncle policy goes up one level, and so distributes load more evenly than the grandparent policy. The cousin policy finds backup parents only on the same level, and the random policy selects backup parents without level bias, so both distribute load very fairly.

The differences between the maximum degree measured using the cousin versus the random policies is much smaller, with cousin consistently lower than random. We believe that this difference in the maximum degree is attributable to the difference in degree variance; the random policy's independently random choices result in a binomial distribution of backup degrees, while the cousin policy distributes the backup links

uniformly. As such, the random policy consistently has some nodes with slightly more than their fair share of load.

We also analyzed backup load distribution for our 3280- and 265,720-node networks, and found identical orderings in the maximum degree measure, with very similarly-shaped curves. These graphs are omitted for brevity.

D. Results Summary

No single backup policy emerged as a clear winner in every possible situation. Given approximations of the network size, node failure rates, and the application's prioritization of the performance metrics, our results can suggest an informed choice of backup policy. This study produced the following useful results:

1) A single backup link per node can make a dramatic difference in the resiliency of an overlay to node failure. When up to 10% of nodes fail simultaneously, almost all of the remaining nodes remain connected. Since the remaining nodes are still connected, they can be reorganized into a new tree at relatively low cost. In contrast, we observe that an overlay without backup links would lose connection to 30-65% of nodes under the same circumstances.

2) Deterministically selected backup links are significantly shorter than randomly selected backup links. We believe that the backup path length will be important to most overlay networks, as it has implications for repair speed, transmission latency, and efficient network resource use. Looking at the two policies which spread backup load most equitably, we find that the cousin policy shows vastly (42-52%) shorter average backup path lengths than the random policy. As such, we believe that for most applications the cousin policy is the best choice when selecting a single backup link.

3) Height-biased backup node selection policies result in uneven backup load distribution. When using a grandparent or uncle policy, each node selects backup nodes higher up in the tree. This bias, combined with the exponential growth of each tree level, creates significant risk of overloading individual nodes with backup requests. The cousin and random policies can provide similar connectivity with up to 20% node failure, while distributing load much more uniformly.

4) The random policy offers the best resiliency in very large networks and when using more than one backup link per node. Though random links are more costly than cousins, they offer better connectivity in very large graphs. We also noted in experiments that additional cousin links after the first produce rapidly declining marginal benefit. This suggests a compromise: if the first backup link is selected according to the cousin policy, and any additional links are selected randomly, the short cousin link will be used when possible, and longer random links will provide additional resiliency.

VI. CONCLUSIONS

We find that trees augmented with local, deterministic backup paths are able to meet or exceed the resiliency to failure achieved with global, random backup paths in moderately-sized networks. We observe similar resiliency results in graphs

ranging from hundreds to hundreds of thousands of nodes. Moreover, the cousin backup policy is able to match the resiliency of the random backup policy while reducing the average backup path length and maintaining a fair distribution of backup load. These results suggest that the protocol overhead, complexity, and reliance on central nodes that result from the use of randomized backup links may be unnecessary; similar resiliency and performance may be attainable with simple deterministic backup selection policies.

Further experiments are underway to evaluate the performance of backup selection policies when node failures are not independently random. This includes cases where different nodes have different failure probabilities as well as non-random node failure processes like geographically localized failures, adversarial failures, and load-induced failures.

REFERENCES

- [1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *ACM Symposium on Theory of Computing*, pages 171–180, 2000.
- [2] Fred S. Annexstein and Kenneth A. Berman. Distributed models and algorithms for survivability in network routing. In *IPDPS*, 2000.
- [3] F.S. Annexstein, Ken Berman, Tsan sheng Hsu, and Ram Swaminathan. A multi-tree generating routing scheme using acyclic orientations. In *Theoretical Computer Science*, 240: (2) 487-494, 2000.
- [4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application level multicast. In *ACM Sigcomm*, 2002.
- [5] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *ACM Sigmetrics*, 2003.
- [6] A. Barabasi, E. Ravasz, and T. Vicsek. Deterministic scale-free networks. In *Physica A* 299, (3-4), pp. 559-564, 2001.
- [7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles*, 2003.
- [8] Yang-Hua Chu, S. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS 2000*, pages 1–12, Santa Clara, CA, June 2000.
- [9] Bram Cohen. Incentives build robustness in bittorrent.
- [10] Dejan Kostic, Adolfo Rodriguez, Jeannie Albrechtnd, and Amin Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of the 19th ACM Symposium on Operating System Principles*, 2003.
- [11] Planet Lab. www.planet-lab.org.
- [12] John C. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *INFOCOM*, pages 1414–1424, San Francisco, CA, March 1996.
- [13] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *ACM/IEEE NOSSDAV*, 2002.
- [14] Rob Sherwood, Ryan Braud, and Bobby Bhattacharjee. A cooperative bulk transfer protocol. In *IEEE Infocom*, 2004.
- [15] Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. University Press, 1999.
- [16] www.pdos.lcs.mit.edu/strib/plapp.
- [17] X. Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications, 1997.
- [18] Rajendra Yavatkar, James Griffioen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, pages 333–344, 1995.