Would Diversity Really Increase the Robustness of the Routing Infrastructure against Software Defects?

Juan Caballero*, Theocharis Kampouris*, Dawn Song‡*, Jia Wang†
*Carnegie Mellon University ‡UC Berkeley †AT&T Labs-Research

Abstract

Today's Internet routing infrastructure exhibits high homogeneity. This constitutes a serious threat to the resilience of the network, since a bug or security vulnerability in an implementation could make all routers running that implementation become simultaneously unusable. This situation could arise as a result of a defective software upgrade or a denial-of-service attack.

Diversity has been proposed as a solution to increase resilience to software defects, but the benefits have not been clearly studied. In this paper, we use a graph theoretic approach to study the benefits of diversity for the robustness of a network, where robustness is the property of a network staying connected under a software failure. We address three fundamental questions: 1) How do we measure the robustness of a network under such failures? 2) How much diversity is needed to guarantee a certain degree of robustness? 3) Is there enough diversity already in the network or do we need to introduce more?

We find that a small degree of diversity can provide good robustness. In particular, for a Tier-1 ISP network, five implementations suffice: two for the backbone routers and three for the access routers. We learn that some networks may already have enough diversity, but the diversity is not adequately used for robustness. We observe that the best way to apply diversity is to partition the network into contiguous regions using the same implementation, separating backbone and access routers and taking into account if a router is replicated. We evaluate our approach on multiple real ISP topologies, including the topology of a Tier-1 ISP.

*This material is based upon work partially supported by the National Science Foundation under Grants No. 0311808, No. 0433540, No. 0448452, No. 0627511, and CCF-0424422. Partial support was also provided by the U.S. Army Research Office under the Cyber-TA Research Grant No. W911NF-06-1-0316, and under grant DAAD19-02-1-0389 through CyLab at Carnegie Mellon. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ARO, NSF, or the U.S. Government or any of its agencies. This work was also supported in part by the Korean Ministry of Information and Communication and the Korean Institute for Information Technology Advancement under program 2005-S-606-02.

1 Introduction

Today's Internet routing infrastructure exhibits high homogeneity. A few router vendors dominate the market and each ISP relies on a very limited number of vendors to build their infrastructure. Simplified network operation, uniform operator training, reduced complexity, and interoperability are common reasons behind this homogeneity. Such homogeneity, however, raises serious questions about the resilience of the routing infrastructure against software defects. Computer systems are notoriously known for being laden with bugs and vulnerabilities, and routers are no exception.

Software defects in routers are not uncommon. Markopoulou et al. [23] found that 12% of all failures in a Tier-1 ISP network were router related, and a NIST-funded survey [15] in 2002 found that in financial services approximately 16% of reported major bugs were attributed to router and switch problems. Most vendors acknowledge these problems and provide search tools or toolkits to find the flaws in specific software versions. Vulnerabilities in router software are also frequent and can allow denial-of-service attacks [2, 4, 5], remote execution of system-level commands with no authentication [3], unauthorized privileged access [1], or possible remote shell execution [7].

Thus, a real threat lurks. A serious bug in a router software implementation could make all routers running that implementation simultaneously unusable. Due to the router homogeneity in ISPs and enterprise networks, this would effectively disconnect their customers and sites, halting their operation and dealing a major blow to their business and their perceived quality. This dramatic scenario could appear as a result of a defective software upgrade, or a security vulnerability that opens the door for a denial-of-service attack on the routing infrastructure.

Diversity, i.e., using different software implementations from different code bases on different routers, could increase the overall resiliency of a network against some software defects but it would also increase the complexity and cost associated with network deployment and management. Thus, we need to understand the benefits of diversity. To the best of our knowledge, this is the first paper that systematically studies the effectiveness of using software diversity to increase the resiliency of the router infrastructure against simultaneous router failures, which might happen due to software defects in routers.

In this paper, when we refer to *network robustness* or simply to *robustness*, we mean the property of the network staying connected under a software failure[1] that simultaneously disables all routers running a specific implementation. We address the following three fundamental questions: 1) How do we measure the robustness of a network under such failures? 2) How much diversity is needed to guarantee a certain degree of robustness? 3) Is there enough diversity already in the network or do we need to introduce more?

There are many factors contributing to the resiliency of a network. As a first step, in this paper we consider the most fundamental property to guarantee the resiliency of a network, the *connectivity*. Without connectivity no routing is possible and no quality-of-service can be guaranteed. A good example was the Sprint network incident on January 9, 2006, when failure of two links led to a network partition [9]. Other properties such as network capacity, which will be reduced due to the failure and might lead to an increase in delay and packet loss, are left as future work.

To study whether the network topology will stay largely connected or will be partitioned into small unconnected components, when all routers running a certain implementation become unusable, we propose a graph theoretic approach and convert the problem into a version of a graph coloring problem, where routers are colored based on their implementations and connectivity needs to be maximized when any color is removed from the graph. A color failure (i.e., removing a color) is equivalent to disconnecting all nodes using that color and has two impacts: the disconnection of the nodes themselves, which cannot be avoided, and the disconnection of other nodes who relied on those nodes to connect to the rest of the graph. Our goal is to minimize this second factor so that a network is as robust as a fully connected topology (i.e., full mesh), since removing any number of nodes from a fully connected topology does not affect the connectivity of the remaining nodes.

In this paper, we show how to measure the robustness of the network against software failures, propose a family of coloring algorithms to best apply the diversity, and evaluate those algorithms over real topologies, including the topologies obtained from a large Tier-1 ISP and from the Rocketfuel project [28]. Our results show that a small degree of diversity can provide good robustness against simultaneous router failures. A large Tier-1 ISP network can obtain good robustness when using a total of five implementations: two for the backbone routers and three for the access routers. In addition, we show that large networks might already have enough diversity, since the homogeneity is due to a large fraction of the nodes using a small number of implementations, even when the total number of implementations present might be large. Thus, the existing diversity is not being adequately used for robustness. To benefit from the available implementations, ISPs would need to change the relative number of nodes per implementation and to geographically distribute the diversity. Furthermore, we show that our algorithms can apply the available diversity in a way such that a network achieves robustness near the optimal one that can be achieved with a fully connected topology.

Scope and deployment cost: Diversity can protect against software vulnerabilities that affect one or multiple implementations from the same vendor. But, some software vulnerabilities might be shared across implementations from multiple vendors. This can happen when there is code reuse between vendors, e.g., by using a third party library containing a vulnerability. Past examples include the ASN.1 parser vulnerabilities found in OpenSSL that affected many different platforms using this library [6].

This paper does neither claim that diversity can protect against all software defects nor that we should redesign all networks to accommodate for diversity. Rather, we show that diversity greatly helps with simultaneous router failures and that many networks already have a significant amount of diversity, due to legacy routers, changes of provider, or budget restrictions. Our results show that the number of implementations already present in some large networks could, if properly used, provide good robustness against simultaneous router failures. However, the existing diversity is not used adequately for robustness. An adequate use would require changing the relative number of nodes using each implementation, and a better geographical distribution. We argue that a large part of the cost usually assigned to diversity, such as interoperability problems, network management or operator training is already being paid in networks where multiple implementations are already present. Thus, the additional cost introduced by shifting the relative number of nodes from each implementation and by changing the geographical distribution of the diversity could be relatively small in these networks. Our goal is not to evaluate the cost of redistributing or introducing more diversity in networks since we lack important data for such evaluation, only available to ISPs, but to understand the benefits of diversity and how should such diversity be applied. Our contributions are summarized below.

Measure the robustness against specific failures: There has been previous work on metrics that describe the global robustness of a graph. Rather, we show how to measure

[1]A router is a combination of hardware and software. In this paper we focus on software failures, which are more likely to simultaneously affect all routers running an implementation, but our approach can also protect against similar hardware failures.

the robustness of a network against specific failure types, modeling the failure's effect on connectivity. We focus on one failure type, i.e., simultaneous router failures, which we convert into a version of a graph coloring problem, but we believe that our approach can be easily applied to many other failure types.

Design and implement algorithms that properly apply diversity: We learn that the best way of applying diversity is to partition the network into contiguous regions that use the same implementation, taking into account the node roles and possibly replicated nodes. Our results show that our algorithms can apply the diversity in a way that a network achieves robustness near the optimal that can be achieved with a fully connected topology.

Demonstrate that a small degree of diversity can achieve good robustness: We show that using a good coloring algorithm the amount of diversity needed to provide good robustness is small. A large Tier-1 ISP network can obtain good robustness when using a total of five implementations: two for the backbone routers and three for the access routers.

Expose that the existing diversity could already provide good robustness: Our data shows that the homogeneity in Tier-1 ISP networks comes from a large fraction of the nodes using a few implementations, even when the total number of implementations present is large. Thus, the existing diversity is not being adequately used for robustness. To benefit from the existing diversity, ISPs would need to change the relative number of nodes per implementation and to geographically distribute the diversity.

The remainder of the paper is organized as follows. Section 2 defines our graph theoretic approach. In Section 3 we present the metrics needed to evaluate the robustness of the network. Then, in Section 4 we present graph coloring algorithms that properly apply the diversity, and in Section 5 we evaluate them over different topologies. Section 6 presents the related work. Finally, we discuss extensions and future work in Section 7 and conclude in Section 8.

2 Problem Statement

In this section, we convert the problem of using router software diversity to increase network robustness into a graph coloring algorithm. We represent a given network topology as a graph $G = (V, E)$, where V is the set of nodes representing routers in the network, and E is the set of undirected edges each corresponding to a link between two routers in the network. Let $n = |V|$ be the number of nodes, and $m = |E|$ be the number of edges in G. Let $C_k = \{c_1, c_2, \ldots, c_k\}$ be a set of k distinct colors representing the available implementations that any router can utilize. We say a *vertex-coloring algorithm* takes as input the graph G and the color set C_k, and outputs a *colored graph* $G_k = (V, E)$ where each node in V has been tagged with a color from C_k. Thus, the coloring algorithm determines which implementation should be run by each router, by assigning a color to each router in the network.

A *color failure* is the simultaneous failure of all the routers running a specific implementation, which makes those routers unusable. This color failure can be caused by a defective router software upgrade, or a denial of service attack on the routing infrastructure. A color failure is represented as a *color removal*, which takes as input the colored graph G_k and a color $c_i \in C_k$ and returns a *color-removed subgraph* $G_k^i = (V, E^*)$ with $E^* \subseteq E$, which is the subgraph of G_k generated by *disconnecting* all nodes of color c_i from G_k, where disconnecting a node means removing all edges connected to that node while leaving the node in the graph. Intuitively, the color-removed subgraph represents the remaining network topology after a color failure.

Problem definition: Given a graph $G = (V, E)$ with n nodes and m edges, a set C_k with k distinct colors, and a robustness function $\phi(\cdot)$, find a k-color vertex-coloring of graph G that maximizes $\phi(\cdot)$. Our problem is different from the most common graph coloring problem, which tries to find a *proper coloring*, meaning no two adjacent nodes are assigned the same color [16]. We do not require adjacent nodes to have different colors. We try to find a coloring of G that maximizes a robustness function ϕ, where ϕ is computed on the subgraphs of G obtained by removing the colors, one at a time.

Note that the probability of finding a software defect in one implementation should be independent of the probability of finding a defect in any other implementation. Thus, the implementations used to add diversity to the network should come from different code bases. To summarize, our problem is composed of two main parts. First, how to define the function ϕ, such that it measures the robustness of the graph upon a color removal. We address this in Section 3. Second, how to design an effective vertex-coloring algorithm, that achieves high robustness for a given graph with a limited number of colors. We address this in Section 4.

3 Measuring Network Robustness

In this section we introduce the metrics used to describe the robustness of a graph under color failures. First, we describe connectivity metrics that can be used on any graph, and then we define the robustness metric on a colored graph, as a function of a connectivity metric.

3.1 Connectivity Metrics for a Graph

We use the connectivity of the graph as a measure of the robustness of the network topology. There have been numerous previously proposed connectivity metrics for a graph such as: number of components in the graph[2], normalized size of the largest component [10], pair connectivity [27], average size of the components (excluding the largest component) [10], average distance [10], number of biconnected components [32], minimum cut-set size for a balanced bi-partition of the graph [29], and effective diameter [26].

But we are not aware of any in-depth comparison of these connectivity metrics that allows to select one of them. Thus, a significant part of our work has been to identify the important characteristics that a connectivity metric should possess. In this paper, we require the following characteristics to be satisfied by a connectivity metric: 1) it has to be non-negative; 2) it should monotonically decrease as more nodes are disconnected; 3) it needs to be normalized; 4) it should represent only binary connectivity, that is, if two nodes are connected or not, rather than try to measure the degree of connectivity between them, and 5) it should be as intuitive as possible.

Note that some of the above metrics, such as those using shortest path computations or hop counts, attempt to measure not only if two nodes are connected, but also the degree of connectivity between them, but in doing so introduce an assumption about shortest path routing being used in the network, which is not desirable. We believe the proper approach is to first guarantee the physical connectivity of the network before investigating the impact on routing and end-to-end performance. We select the following two connectivity metrics because they satisfy the above requirements. Note that, the robustness function is independent of the connectivity metrics used and other metrics can thus be selected.

The *normalized size of the largest component (NSLC)* [10] is the size of the largest component over the total number of nodes. Ideally, when removing a color from the graph, the remaining nodes should all form a single component and thus be all reachable to each other.

The *pair connectivity (PC)* [27] is the fraction of connected node pairs in a graph over the total number of distinct pairs of nodes. It measures for each node, how many other nodes it can reach. This metric takes into account all the components in the graph rather than just the largest one and it is useful when the graph is partitioned into multiple large components where each component connects a significant number of nodes and should not be ignored.

$$PC = \frac{\sum_{i=1}^{comp} \frac{1}{2}|L_i|(|L_i|-1)}{\binom{n}{2}}$$

where $comp$ is the number of components in the graph and $|L_i|$ is the number of nodes in the ith component. Note that we need to first extract the graph's components before calculating any of the two metrics, which takes $O(n+m)$.

[2]In graph theory, a component is a set of connected nodes, i.e., two nodes belong to the same component if and only if there is a path between them.

Modeling node importance: So far, our connectivity metrics consider each node in the graph to be equally important. However, for a given graph, some nodes can be more important than others. For example, a node carrying more traffic would be considered more important than those carrying less traffic. Alternatively, a node which connects to more customers or has more capacity can be considered important. In this paper, we assign different weights to different nodes in the graph to reflect their importance.

In our setting, we do not need to use link weights because we only deal with node failures, rather than individual link failures, and in this case node weights allow dealing with link weights in a straightforward way. Since a node failure implies the failure of all the links connected to the node, then a node can be assigned a weight proportional to the sum of the weights of all the links it connects to. We now extend our connectivity metrics to the case where nodes are associated with weights.

The *weighted normalized size of the largest component (W-NSLC)* is the sum of the weights of all nodes that belong to the largest component, over the total weight of all nodes in the graph.

The *weighted pair connectivity (W-PC)* takes into account the sum of the products of all the node weights that belong to the same component.

$$\textit{W-PC} = \frac{\sum_{i=1}^{comp} \sum_{j \in L_i} w_j (\sum_{k \neq j; k \in L_i} w_k)}{\sum_{i=1}^{n} w_i (\sum_{j \neq i} w_j)}$$

where $comp$ is the number of components in the graph and L_i is the ith component of the graph.

Disconnecting a node: To generate G_k^i from G_k, we disconnect all nodes with color i by removing its edges but we do not remove the nodes themselves from the graph. The reason is that this allows a fair comparison between the connectivity of G_k and G_k^i. Figure 1 illustrates an example. There are two alternative ways to generate G_2^1 from G_2, disconnecting the black node (case A) or removing it (case B). Table 1 shows the connectivity metrics for both cases. Note that the connectivity for case B is the same as the connectivity for the original colored graph, which is misleading. However, in case A the connectivity reflects the node removal, and as a result, the normalized size of the largest component is 0.66.

3.2 Robustness Metrics for a Colored Graph

Intuitively, the robustness of a colored graph measures the remaining connectivity of the colored graph when a color is removed. Given a colored graph and a connectivity

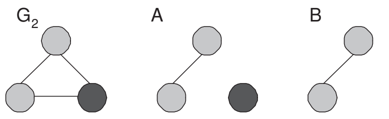


Figure 1. Two alternatives to generate G_2^1 from G_2: disconnecting the black node (A) or removing it (B)

Metric	G$_2$	A	B
Pair Connectivity	1	0.33	1
Normalized Size of Largest Component	1	0.66	1

Table 1. Connectivity metrics for both possible alternatives to generate G_2^1 from G_2

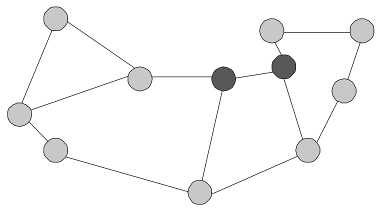


Figure 2. An unbalanced coloring of the Abilene backbone

metric, we define two robustness metrics: 1) the average of the connectivity of the subgraphs created by disconnecting all nodes of a specific color, and 2) the minimum connectivity among all the subgraphs created by disconnecting all nodes of a specific color. Thus, given the colored graph G_k and a connectivity metric $f(\cdot)$, we first obtain all color-removed subgraphs G_k^i by removing each color c_i in turn, where $i \in [1, k]$. Then, we compute the robustness of G according to the connectivity measure f as follows.

$$\phi_{G,f}^{avg} = \frac{1}{k} \sum_{1 \leq i \leq k} f(G_k^i) \qquad \phi_{G,f}^{min} = \min_{1 \leq i \leq k} f(G_k^i)$$

The average robustness is the most straightforward measure of robustness, representing the expected robustness under a color failure. But, if taken alone, it can sometimes be misleading because some colored graphs can present good average robustness but bad minimum robustness. Since, a priori we do not know what the probability of failure is for each of the implementations, we need to try to protect against failures of *any* of the implementations. That is precisely what the minimum robustness gives us: the worst case.

One scenario in which we might get good average robustness and bad minimum robustness is when the coloring is unbalanced, that is, most nodes in the graph use one color and the rest split the remaining colors. Figure 2 shows that scenario in the Abilene backbone network with two colors. The average robustness is quite good, 0.5 for the normalized size of the largest component, but the minimum robustness is only 0.18. The average is good because the removal of the black color gives a very high connectivity that compensates the poor connectivity when the gray color is removed. Such an unbalanced coloring creates a situation where if the grey color fails the graph becomes almost unusable. This situation is clearly undesirable and we will favor coloring algorithms that produce balanced colorings.

4 Coloring Algorithms

In this section, we propose coloring algorithms that assign a color, from a set of k available colors, to each node in the graph. Finding an optimal coloring of a graph is known to be NP-complete. We have devised a total of 9 approximation algorithms which can be classified into 4 different families. Due to space constrains, we present only the results for the *region coloring* family because the two algorithms from this family clearly outperform all others in our evaluation. We refer the reader to our extended paper for more information on the other algorithms [12].

This section is structured as follows. In Section 4.1 we present the intuition behind region coloring and the two algorithms that belong to this family. Then, in Section 4.2 we extend the region coloring algorithms to handle two common cases found in ISP networks: 1) different node roles, and 2) replicated nodes.

4.1 Region Coloring

Region coloring tries to divide the network into contiguous regions and color each region with the same color. This is somewhat similar to a geographic coloring but with the difference that the regions are found automatically, i.e., no geographical information needs to be provided with the graph. Thus, the intuition behind region coloring is to try to isolate the failure to a contiguous region of the graph. Since networks are designed with geography in mind, then removing a particular region of the graph has little additional effect on the rest of the graph, other than the disconnection of the nodes themselves.

This problem is related to the *graph clustering* and *graph partitioning* problems. Graph clustering algorithms try to find peninsulas of connectivity, i.e., regions with a high density of intra-regional links and a few inter-regional links connecting them with other regions. Graph partitioning algorithms employ a similar approach, but the goal is to split the network into balanced partitions, i.e., partitions with a similar number of nodes.

Thus, the region coloring family contains two different types of algorithms: Cluster and Partition algorithms. Our results, presented in Section 5, indicate that graph partitioning algorithms attain better robustness than graph clustering algorithms on the same graph, because clustering algorithms may produce unbalanced partitions which result in bad minimum robustness. In addition, we learn that the colored graph is most robust when the partitions are all contiguous. When faced with the decision of providing either perfectly balanced but discontiguous partitions, or slightly unbalanced but contiguous partitions, the latter usually achieves better robustness. Our results show that a graph colored with a partitioning algorithm that provides contiguous and balanced partitions, achieves robustness near the maximum provided by a fully connected topology with the same number of nodes.

4.2 Extending Region Coloring to Handle Roles and Replicated Nodes

In the previous section, we assumed that all nodes in the network can use any of the k colors in C_k. However, the role that a router plays in a network will further constrain the implementation the router may use. For example, backbone routers and access routers might be built with different design goals and using very different technologies. Thus, a backbone router might not be able to use the implementation of an access router and vice versa. With roles, we require that each node in the graph is appended with a role tag. We consider two main roles: *access routers* and *backbone routers*. Each role has a different color set: *access color set* $C^A_{k_a}$ and *backbone color set* $C^B_{k_b}$. and a node can only be assigned a color from the color set corresponding to its role.

Also, ISPs often replicate important nodes to increase the robustness against node failures. The node and its replica usually connect to the same neighbors so that the replica can take the place of the replicated node in case of node failure, or that the load can be split between both nodes. Typically, backbone routers might be replicated in pairs. Intuitively, to achieve best robustness the replicated node and the replica should be colored differently. To identify replicated nodes, we use a similarity metric employed in social networks: *Structural Equivalence* [21]. Two nodes are said to be structurally equivalent if they connect to the exact same set of neighbor nodes. When we identify such equivalent nodes, we color them differently. In our extended paper we extend this metric to handle nodes that are similar but not identical [12].

When taking into account roles and replicated nodes, the region coloring algorithms work as follows: 1) color all backbone routers, since they provide connectivity to the access routers, and 2) color the access routers, which provide connectivity to the customers. These extensions can be used with either clustering or partitioning algorithms. Next, we present them in conjunction with the Partition algorithm since our experiments have shown this to be the best combination.

Coloring backbone routers: We create the *backbone graph* by removing all access routers and their corresponding links from the original graph. Then, we consider three cases in coloring the backbone graph: 1) if every backbone router has a structurally equivalent neighbor, then color each router in a structurally equivalent pair differently, 2) if no backbone router has a structurally equivalent neighbor, then use region coloring, and 3) if only some backbone routers have structurally equivalent neighbors, then color each node in a structurally equivalent pair differently and use region coloring but imposing the constraint that two nodes previously colored differently need to belong to different regions.

Coloring access routers: We create the *access graph* by collapsing all the backbone nodes into a single node, that connects to all access routers. We consider two cases: 1) if the probability of a failure in $C^A_{k_a}$ is independent from the probability of a failure in $C^B_{k_b}$, for example because they come from different vendors or code bases, then we evenly split the colors in $C^A_{k_a}$ among all access routers to create a balanced colored graph, and 2) if failures in access and backbone colors cannot be assumed independent, then for each access router, if all backbone routers connecting to it have the same color, then the access router is assigned a different color, if available. Otherwise, there is no constraint on the color assigned to the access router.

5 Evaluation

In this section, we evaluate our coloring algorithms on a number of network topologies. The section is structured as follows. In Section 5.1 we describe the network topologies used in the evaluation. Then, in Section 5.2 we compare the different coloring algorithms on a Tier-1 ISP topology. Next, in Section 5.3 we compare the different topologies when colored with the same algorithm. In Section 5.4 we extend our results to node weights. Finally, in Section 5.5 we provide data on the available diversity in a Tier-1 ISP network and show how our algorithms can be used to better apply such diversity.

5.1 Experimental Setup

The network topologies used in the evaluation are presented in Table 2. All the topologies are router-level topologies, where each node represents a router and each edge

Topology	Date	Nodes	Edges
Tier-1 ISP	Oct 2006	A few hundreds	A couple of thousands
Cenic	Aug 2006	51	91
Abilene	Sep 2006	12	15
Exodus	Jan 2002	201	434
Sprint	Jan 2002	604	2268
Verio	Jan 2002	960	2821
Mesh	N/A	100	4950

Table 2. Network topologies used in the evaluation.

represents a link between routers. The table is divided into three parts. The top part shows real ISP topologies that we have access to, including the topology of a Tier-1 ISP; the middle shows Rocketfuel topologies [28] also used in our experiments; and at the bottom, as a base case, we present a synthetic topology which is a full mesh of 100 nodes. All topologies have a single connected component. If there are parallel edges between a pair of nodes, we collapse them into a single edge.

The Rocketfuel topologies were obtained by the authors through external probing. Though they are known to contain some inaccuracies, we consider them a good approximation of a router-level ISP topology and accurate enough to test our algorithms. We use the Mesh topology as a best case since disconnecting a node in a full mesh topology has no effect on the connectivity of the rest of the graph.

Implementation details: We implement the metrics and coloring algorithms using the JUNG graph library [8]. We use the graph clustering algorithm from Wu et al. [31] since it scales linearly with the topology size (i.e., $O(n + m)$) and allows to predefine the number of clusters. We use the graph partition algorithms from Karypis et al. [19], which employ multilevel recursive-bisection and scale well while providing good flexibility. Any other clustering or partitioning algorithms could be used, though as we will show, algorithms that generate balanced and contiguous partitions work best.

5.2 Coloring Algorithms

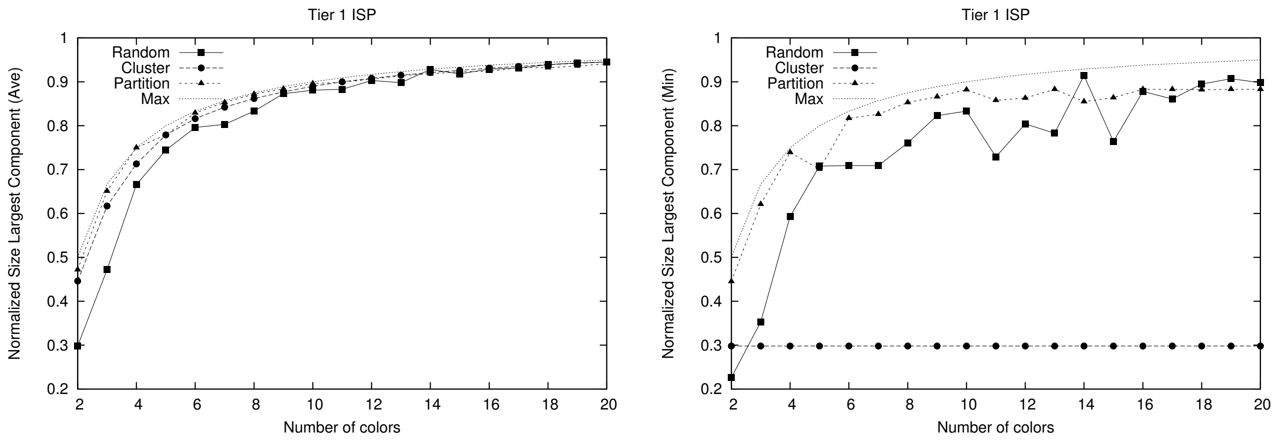
In this section, we compare the robustness achieved by the Partition and Cluster coloring algorithms without extensions, that is, assuming no distinction between backbone and router nodes, and no replicated nodes. These assumptions usually hold in medium size networks such as Abilene or Cenic, but might not hold for large Tier-1 ISP networks. We present results for the extended algorithms in Section 5.5. In addition to the Cluster and Partition algorithms, we also present results for the *Random* algorithm which randomly assigns a color from C_k to each node. The Random algorithm is used as a baseline to compare the other algorithms. The evaluation of the remaining algorithms can be found in our extended paper [12] and their performance is usually somewhere between the Random and Partition algorithms.

Comparing to the optimal coloring: We can provide a guideline on how close to optimal our algorithms work. The idea is that no topology can be more robust than a fully connected topology with the same number of nodes. Thus, in Figure 3 the dotted line labeled *Max* shows the robustness achieved in a full mesh topology with the same number of nodes and balanced coloring. That is, for two colors, $\frac{n}{2}$ nodes are colored using one color and the other $\frac{n}{2}$ are colored with the other color. Then, we can compare the robustness achieved by our approximation algorithms to this guideline to know how well our algorithms are performing.

Figure 3 shows the robustness achieved on the Tier-1 topology by the different algorithms measured using the Normalized Size of the Largest Component (NSLC). Similar results for the Pair Connectivity (PC) on the Sprint topology are provided in Appendix A. We observe from Figure 3(a) that the Cluster and Partition algorithms clearly outperform the Random algorithm with respect to the average robustness for any number of colors less than fourteen.

These results indicate that coloring nodes at random locations in the graph is not a good strategy, since region coloring, which uses the opposite approach, exhibits better performance. Recall that disconnecting all nodes from one color creates two different impacts: the disconnection of the nodes themselves, which cannot be avoided, and the disconnection of any other node who relied on those nodes to connect to the rest of the graph. The second factor can be reduced and sometimes eliminated with a good coloring. This is where region coloring seems to excel, i.e., the simultaneous removal of nodes that are located nearby, in the same region of the graph, has little impact on other nodes located further away in different regions.

Unbalanced partitions: Figures 3(a) and 3(b) show that the Cluster algorithm performs close to the Partition algorithm in terms of average robustness, but significantly worse in terms of minimum robustness. This is due to the unbalanced coloring performed by the Cluster algorithm. For example, when using two colors on the Tier-1 ISP topology, the Cluster algorithm assigns one color to more than $\frac{2}{3}$ of the nodes and the other color to less than $\frac{1}{3}$ of the nodes. On the other hand, the Partition algorithm assigns each color to roughly half of the nodes. This is expected since a graph partitioning algorithm tries to balance the number of nodes in each partition. Using the Cluster algorithm, when the least common color fails, the remaining connectivity is quite good and on average compensates the poor connectivity created by the opposite case. Appendix A shows a more detailed example of the same effect.



(a) Average Normalized Size Largest Component (b) Minimum Normalized Size Largest Component

Figure 3. Robustness achieved by the different coloring algorithms on the Tier-1 ISP topology when using the Normalized Size of the Largest Component as connectivity metric.

Figure 3(b) shows that the Partition algorithm regularly outperforms the others in terms of minimum robustness. We also observe that the minimum robustness for the Cluster algorithm stays flat. This is because the Cluster algorithm we use is deterministic, i.e., it always finds the same partitions on different runs over the same graph. Once a bad partition is found, e.g., an unbalanced or discontiguous partition, it is likely to be kept and adding more colors will not improve the minimum robustness.

Since we do not have prior knowledge on which color might fail, we need to optimize for both average and minimum robustness. Otherwise, a failure of the color with the largest number of nodes would have a major impact on the network. As we have seen, unbalanced partitions can generate good average robustness, but they will introduce bad minimum robustness which is not desired. Thus, we prefer the Partition algorithm over the Cluster algorithm, since the Partition algorithm tends to generate balanced partitions.

Non-contiguous partitions: In Figure 3(b), the minimum robustness for the Partition algorithm takes a dip at $k = 5$. This is also observed on other topologies. We found that one of the partitions in the split is non-contiguous, that is, the same partition contains two groups of nodes that have intra-group connectivity but no inter-group connectivity. The main problem with non-contiguous partitions is that they might disconnect other partitions that depend on the non-contiguous partition for connectivity to the rest of the graph.

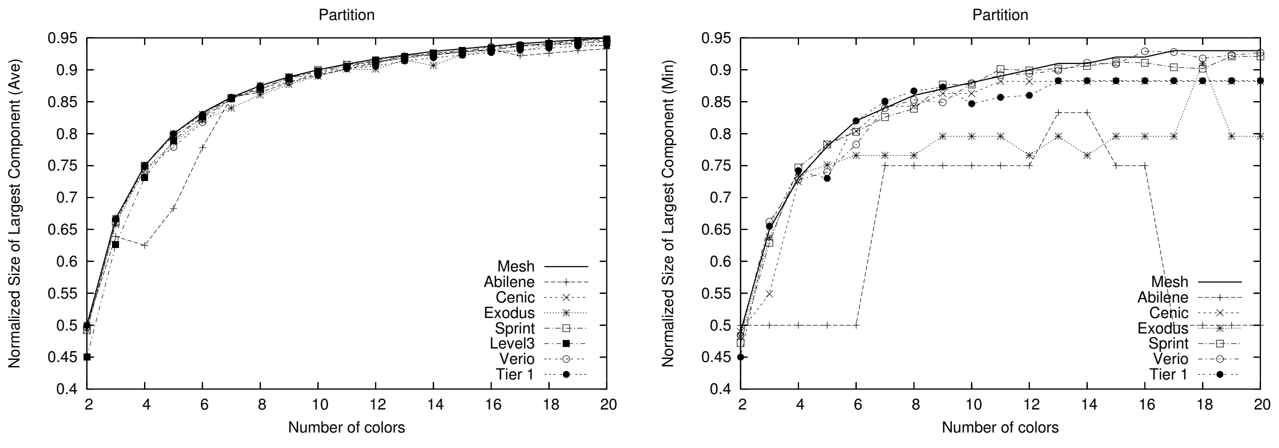
Good graph partitioning algorithms tend to avoid non-contiguous partitions but this is not always easy given that the number of nodes in each partition needs to be balanced. Some algorithms such as the one we use [19] have a parameter to control how unbalanced the partitions can be. Our solution for non-contiguous partitions is to rerun the partitioning algorithm with an increased value of this parameter if any of the partitions is found to be non-contiguous. This increases the final robustness but increases the runtime. Since the Partition algorithm is fast, we can generate and color multiple partitions and then select the best coloring, without significantly impacting the runtime. Appendix C shows the runtime for the different algorithms.

In summary, we observe that the best approach to color the network is to use the same color for each region of the graph, since the failure of one node impacts closer nodes more than nodes located further away (i.e., fate-sharing). Among the two algorithms that use this approach, Cluster and Partition, the Partition algorithm works best because it produces a more balanced coloring.

In addition, the selected partitioning algorithm should provide contiguous partitions as these improve the overall robustness of the network. Note that for a coloring algorithm to provide a robust coloring, it is not enough to produce a balanced coloring. For example, a perfectly balanced random coloring, produces colored graphs with smaller robustness than the less balanced ones produced by the Partition algorithm.

5.3 Network Topologies

Next, we compare the robustness of the various topologies for a given coloring algorithm. We use the Partition algorithm which outperformed the others in Section 5.2. As a baseline we use the synthetic Mesh topology since it is the most resilient topology. That is, after removing a color, the remaining nodes are still fully connected; there is no additional impact from disconnecting a node.



(a) Average Normalized Size Largest Component

(b) Minimum Normalized Size Largest Component

Figure 4. Robustness of the different topologies when colored using the Partition algorithm.

Figure 4 shows the robustness of different topologies using the Partition algorithm and the NSLC connectivity metric. Similar results for the PC metric are provided in Appendix A. For this experiment, we first force the graph partitioning algorithm to try to find very balanced partitions. If any of the generated partitions is non-contiguous, we relax the balancing constraint. We find that we rarely have to relax the balancing constraint more than once for any color set size. We observe that, using the Partition algorithm, most topologies achieve average and minimum robustness close to the Mesh topology. Thus, when any router can run any implementation, we can achieve robustness close to the optimal obtained with a full mesh, by coloring the graph using the Partition algorithm.

It is not surprising to find that the Abilene topology often performs worse than others. This can be explained by its simple topology with few nodes and minimal link redundancy between nodes. Being a non-commercial network, it is unlikely that robustness was one of Abilene's design goals. However, we also find that the Exodus topology also performs worse than the other commercial network topologies, specially in the worst case. This indicates that the Exodus topology has less geographical redundancy.

5.4 Node Importance

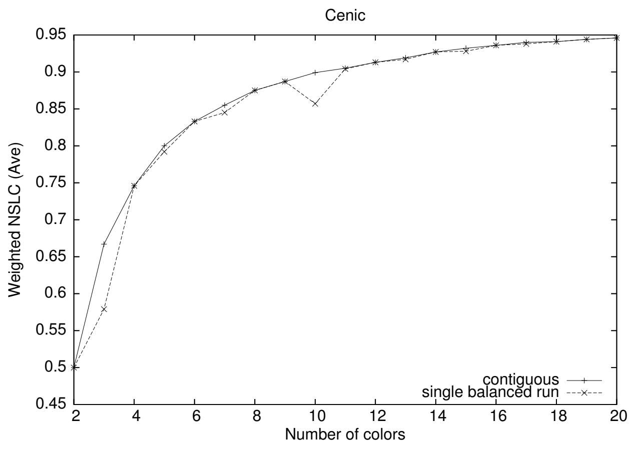
Nodes in a topology may not be equally important. This can be modeled by assigning a weight to each node. Some partitioning algorithms, such as the one we use [19], allow node weights to be specified and compute the partitions to achieve similar total weight in every partition, rather than a similar number of nodes in each partition. Thus, in this case a balanced partition, has approximately the same sum of node weights in each partition.

As an example, Figure 5 shows the robustness of the weighted Cenic topology using the Partition algorithm, where each node is assigned a weight equal to the number of customers that connect to that node. We select the Cenic topology because we have such information available. Note that the significance of the weights can be arbitrary and we use this specific weight just as an illustrative example. The dashed line shows the result when we run the Partition algorithm forcing balanced partitions, while the solid line shows the final result where if one of the partitions was non-contiguous, we relax the balancing constraint. Compared with Figure 4, we observe that similar robustness can be achieved in the weighted topology as that in the unweighed topology. There are two points (i.e., $k = 3$ and $k = 10$) where the partitioning algorithm is not able to find contiguous partitions if constrained to very balanced partitions. When we relax this constraint, we obtain partitions that are more unbalanced, but contiguous, which improves the final result specially for minimum robustness.

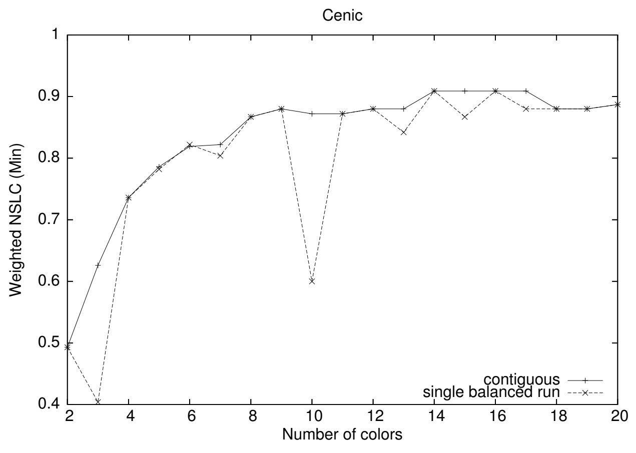
5.5 Optimizing the Use of Existing Diversity

In this section we first present the diversity that currently exists in the Tier-1 ISP topology, which we call *original coloring*. Then, we compare the robustness of the original coloring to the robustness in the colored graphs obtained by the Partition algorithm without extensions (Basic Partition), and the Partition algorithm using the role and replicated node extensions (Extended Partition).

The Tier-1 ISP topology uses 8 implementations with two of the implementations being used by over 90% of the nodes. The fact that there are two dominating implementations supports the hypothesis that current networks are highly homogeneous, but the surprising fact is the large number of implementations already available in the net-



(a) Average Weighted Normalized Size Largest Component



(b) Minimum Weighted Normalized Size Largest Component

Figure 5. Robustness of the Cenic topology, with node weights assigned based on the number of customers connected to the router, and selecting the split that gives no discontiguous partitions.

work. Table 3 compares the robustness of the Tier-1 ISP using the original coloring, with the robustness achieved using the Basic Partition and Extended Partition algorithms. Since there are eight colors in the original coloring, we set the number of available colors for the Basic Partition algorithm to eight and allow any router to select any of the colors. For the Extended Partition algorithm we need two sets of colors: one for the backbone nodes and one for the access nodes. We observe that, among the eight implementations in the original coloring, only two are used for backbone routers. Therefore, we allow the Extended Partition algorithm to use two colors for backbone routers and six colors for access routers.

We observe that the original coloring has bad minimum robustness when the color failure affects one of the colors used by the backbone routers. Both Partition algorithms achieve improved average and minimum robustness by distributing the available diversity better throughout the network. This is specially significant for minimum robustness, achieving values much closer to the average robustness. The Extended Partition algorithm shows worse minimum robustness than the Basic Partition algorithm, since it has an additional constraint that routers can only use implementations adapted to their role. But this constraint is more realistic for Tier-1 ISP networks, where the role separation of backbone and access routers becomes evident.

Figure 6 shows the degree of diversity needed to achieve a certain level of robustness in the Tier-1 ISP topology using the Extended Partition algorithm. From the total number of colors shown in the x-axis, two colors are always used for the backbone routers and the rest used for the access routers. The results show that two colors are enough to guarantee the robustness of the backbone, given the existing amount of redundancy. This is shown as a good average robustness with any number of access colors. It also shows that we start to achieve good overall robustness (e.g., minimum robustness ≥ 0.5 and average robustness ≥ 0.75) when we use at least 3 colors for the access routers, that is with a total of 5 colors.

To conclude, we observe that the original Tier-1 network has a significant amount of diversity, but the diversity is not best used to maximize robustness against color failures. Using a realistic coloring algorithm such as Extended Partition, we can achieve a significant increase in the robustness of the topology against color failures, by redistributing the diversity, without increasing the amount of implementations used in the network.

6 Related Work

Zhang et al. [33] first proposed the use of diversity to increase the survivability of a network, inserting diversity at each level of the networking stack. More recently, O'Donnell et al. [25] studied the problem of how to use diversity to limit the spread of malware on a network topology. O'Donnell et al. assume that each node runs multiple implementations and takes an online decision, in a distributed fashion, of which implementation to use at any given time. In our case, nodes run a single implementation and network operators select in a centralized and offline manner, using our coloring algorithms, which implementation each node should run from the set of available implementations. Junqueira et al. [17,18] propose to use diversity in a cooperative backup system to protect against correlated host failures due to software vulnerabilities. Their results on distributed systems are analogous to ours on routing in-

Metric	Original coloring	Basic Partition	Extended Partition
Average robustness (NSLC)	0.713	0.875	0.855
Minimum robustness (NSLC)	0.055	0.867	0.760
Average robustness (PC)	0.647	0.765	0.739
Minimum robustness (PC)	0.016	0.752	0.578

Table 3. Comparison of the robustness of the Tier-1 topology using the original coloring and the coloring obtained by the Basic Partition and Extended Partition algorithms. Each coloring uses 8 colors.

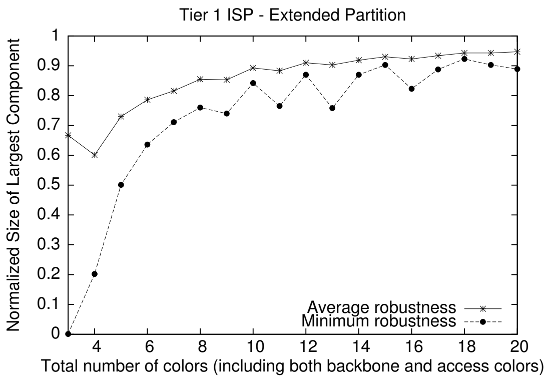


Figure 6. Tier-1 robustness using the Extended Partition algorithm, which includes the role and replicated node extensions. Two colors are used for backbone routers.

frastructure in that they show that diversity is common and can be exploited for increased resiliency.

Previous work on network robustness has focused on the Internet topology. Albert et al. [10] showed that scale-free networks are resilient to random faults but vulnerable to attacks that target the most connected nodes. Park et al. [27] studied the susceptibility of the Internet to a mixture of random faults and attacks. Magoni [22] studied the Internet robustness to attacks finding that the removal of a few nodes could significantly damage the Internet's connectivity but such an attack would be costly to mount. Li et al. [20] raised concerns about the application of power laws to the router-level Internet topology. Markopoulou et al. [23] classified the different network faults from a Tier-1 ISP network and found that 12% of all faults were router-related problems.

There has been extensive work on the connectivity of the Internet and generally in identifying graph connectivity metrics [10,26,27,29,32]. Several node importance metrics have also been proposed [14, 20, 24, 30]. Finally, there is also research work on graph partitioning. Wu and Huberman [31] use a voltage ranker to find clusters while Karypis and Kumar [19] use a k-way multi-level approach to partition hypergraphs. We use both algorithms in this paper to compare the differences between clustering and partitioning algorithms.

7 Discussion

In this section we present extensions to our work that we are currently testing, other foreseeable applications for our approach and some future work.

External connections: ISPs need to route their customer's traffic to the rest of the Internet. For all destinations not directly connected to the ISP network, this means forwarding the traffic to another ISP, either an upstream provider or a peer. In order for the network to remain functioning upon a color failure, we need to ensure that there is at least one external connection in each partition when a color fails. With the Partition algorithm, we can apriori fix nodes in different regions, and the partitioning will try to keep those nodes in the specified regions. We use this property to fix border routers in different partitions.

Robustness beyond connectivity: In this paper we have limited our analysis to the connectivity of the network, because connectivity is the fundamental property that needs to be guaranteed for the network to be functional. Once we address connectivity, we can study the impact on higher layers. Clearly, the disconnection of a large number of routers impacts the routing and the end-to-end quality of service and might introduce higher CPU load on the remaining routers, packet loss increments due to congestion on the remaining links, or an increase in end-to-end delay due to longer paths. We leave the study of these effects on the higher layers as future work.

Robustness beyond color failures: There is a need to develop a general framework to measure network robustness, that encompasses any type of failure and any kind of impact. Such a framework would allow us to predict the impact of failures and to understand the effect of a proposed local change (e.g., add or remove a router or a link) on the global network robustness. In this paper we have presented a first step towards that framework by studying the network robustness under simultaneous router failures. This is a worst case scenario because it involves simultaneous and geographically disperse node failures. This type of failures are infrequent but have great impact. In our future work we plan to extend our approach to handle more frequent failures with more limited impact.

8 Conclusion

In this paper, we have shown that diversity does increase the robustness of the routing infrastructure against simultaneous router failures, by answering three fundamental questions. First, we have presented how to measure the robustness of the network when faced with a color failure, using connectivity and robustness metrics that capture the impact of a node failure and its effect on the rest of the topology. Second, we have shown that a small degree of diversity in the network can provide good robustness against simultaneous router failures. A large Tier-1 ISP network can obtain good robustness when using a total of five implementations: two for the backbone routers and three for the access routers. The amount of diversity needed is small because our proposed coloring algorithms maximize the benefits of the available diversity. We observe that the best way of applying diversity is to partition the network into contiguous regions that use the same implementation, taking into account the node roles and possibly replicated nodes. Finally, we expose that large networks might already have enough diversity, since the homogeneity is due to a large fraction of the nodes using a small number of implementations, even when the total number of implementations present might be large. Thus, the existing diversity is not being adequately used for robustness. To benefit from the available implementations, ISPs would need to change the relative number of nodes per implementation and to geographically distribute the diversity. When using the same number of implementations currently available in a Tier-1 ISP network we can increase the number of nodes in the largest component from 5% to at least 76% in the worst case.

9 Acknowledgements

We would like to thank Abraham Flaxman, Alan Frieze, Min Gyung Kang, Franck Le, Sihyung Lee, and Bruce Maggs for their feedback. We would also like to thank Geoff Voelker for providing us with information about the Phoenix project. Finally, we would like to express our gratitude to the Crosstex, JUNG, and Metis developers for making available great software.

References

[1] Cert Advisory CA-2002-03 (February 2002). http://www.cert.org/advisories/CA-2002-03.html.

[2] Cert Cisco Information for VU#139491 (April 2002). http://www.kb.cert.org/vuls/id/JPLA-53D2H9.

[3] Cert Vulnerability Note VU#205225 (July 2006). http://www.kb.cert.org/vuls/id/205225.

[4] Cert Vulnerability Note VU#409555 (May 2006). http://www.kb.cert.org/vuls/id/409555.

[5] Cert Vulnerability Note VU#583638 (January 2005). http://www.kb.cert.org/vuls/id/583638.

[6] Cert Vulnerability Note VU#748355 (August 2002). http://www.kb.cert.org/vuls/id/748355.

[7] Cert Vulnerability Note VU#959203 (August 2005). http://www.kb.cert.org/vuls/id/959203.

[8] JUNG: Java Universal Network/Graph Framework. http://jung.sourceforge.net.

[9] The Backhoe: A Real Cyberthreat. http://www.wired.com/news/technology/1,70040-0.html.

[10] R. Albert, H. Jeong, and A. L. Barabasi. Error and Attack Tolerance in Complex Networks. *Nature,* 406:378 – 382, 2000.

[11] B. Bollobás and O. Riordan. Slow Emergence of the Giant Component in the Growing M-Out Graph. *Random Structures Algorithms,* 27(1):1–24, 2005.

[12] J. Caballero, T. Kampouris, D. Song, and J. Wang. Would Diversity Really Increase the Robustness of the Routing Infrastructure Against Software Defects? Carnegie Mellon University, Technical Report CMU-CyLab-07-002, Pittsburgh, Pennsylvania, February 2007.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. *Proceedings of the SIGCOMM Conference,* Cambridge, Massachusetts, August 1999.

[14] L. C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry,* 40(1):35 – 41, 1977.

[15] G. Tassey, Editor. The Economic Impacts of Inadequate Infrastructure for Software Testing. National Institute of Standards and Technology, Technical Report NIST PR 02-3, Gaithersburg, Maryland, May 2002.

[16] T. R. Jensen and B. Toft. *Graph Coloring Problems.* Wiley-Interscience, 1994.

[17] F. Junqueira, R. Bhagwan, A. Hevia, K. Marzullo, and G. Voelker. Surviving Internet Catastrophes. *Proceedings of the USENIX Annual Technical Conference,* Anaheim, California, May 2005.

[18] F. Junqueira, R. Bhagwan, K. Marzullo, S. Savage, and G. Voelker. The Phoenix Recovery System: Rebuilding from the Ashes of an Internet Catastrophe. *Proceedings of the Workshop on Hot Topics in Operating Systems,* Lihue, Hawaii, May 2003.

[19] G. Karypis and V. Kumar. Multilevel K-Way Hypergraph Partitioning. *VLSI Design,* 11(3):285 – 300, 2000.

[20] L. Li, D. Alderson, W. Willinger, and J. Doyle. A First Principles Approach to Understanding the Internet's Router Technology. *Proceedings of the SIGCOMM Conference,* Portland, Oregon, August 2004.

[21] F. Lorrain and H. C. White. Structural Equivalence of Individuals in Social Networks. *Journal of Mathematical Sociology,* 1:49 – 80, 1971.

[22] D. Magoni. Tearing Down the Internet. *IEEE Journal on Selected Areas in Communications,* 21(6):949 – 960, 2003.

[23] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of Failures in an IP Backbone. *Proceedings of the IEEE International Conference on Computer Communications,* Hong Kong, China, March 2004.

[24] M. E. J. Newman. Assortative Mixing in Networks. *Physical Review Letters,* 89(20):208701, Oct 2002.

[25] A. J. O'Donnell and H. Sethu. On Achieving Software Diversity for Improved Network Security Using Distributed Coloring Algorithms. *Proceedings of the ACM Conference on Computer and Communications Security,* Washington, D.C. October 2004.

[26] C. Palmer, G. Siganos, M. Faloutsos, C. Faloutsos, and P. B. Gibbons. The Connectivity and Fault-Tolerance of the Internet Topology. *Proceedings of the Workshop on Network Related Data Management,* Santa Barbara, California, May 2001.

[27] S. Park, A. Khrabrov, D. M. Pennock, S. Lawrence, C. L. Giles, and L. H. Ungar. Static and Dynamic Analysis of the Internet's Susceptibility to Faults and Attacks. *Proceedings of the IEEE International Conference on Computer Communications,* San Francisco, California, March 2003.

[28] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP Topologies with Rocketfuel. *Proceedings of the SIGCOMM Conference,* Pittsburgh, Pennsylvania, August 2002.

[29] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network Topology Generators: Degree-Based vs. Structural. *Proceedings of the SIGCOMM Conference,* Pittsburgh, Pennsylvania, August 2002.

[30] S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A Simple Conceptual Model for the Internet Topology. *Proceedings of the IEEE Global Communications Conference,* San Antonio, Texas, November 2001.

[31] F. Wu and B. A. Huberman. Finding Communities in Linear Time: A Physics Approach. *European Physical Journal B,* 38:331-338, 2004.

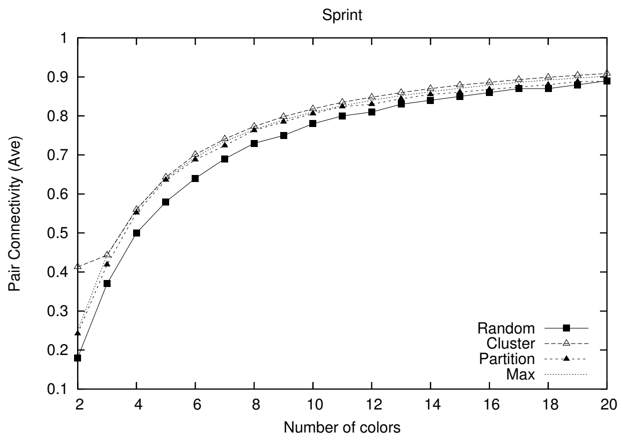
[32] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A Quantitative Comparison of Graph-Based Models for Internet Topology. *IEEE/ACM Transactions on Networking,* 5(6):770–783, 1997.

[33] Y. Zhang, S. K. Dao, H. Vin, L. Alvisi, and W. Lee. Heterogeneous Networking: A New Survivability Paradigm. *Proceedings of the New Security Paradigms Workshop,* Cloudcroft, New Mexico, September 2001.

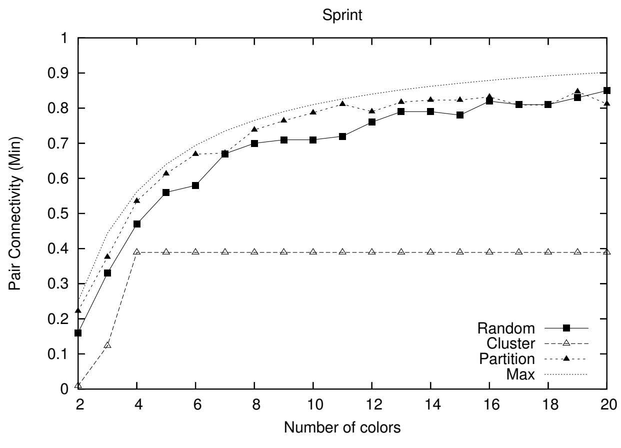
A Pair Connectivity Results

This section presents additional results using the Pair Connectivity metric. Figure 7 shows the performance of the different algorithm on the Sprint topology and Figure 8 shows the robustness of the different topologies when colored using the Partition algorithm. The results are analogous to those presented in Sections 5.2 and 5.3 for the Normalized Size of the Largest Component. In addition, we present a more detailed example on how unbalanced partitions decrease the robustness of a network.

Figure 7(a) shows that the Cluster algorithm performs better than the Partition algorithm in terms of average robustness. This is due to the unbalanced coloring performed by the Cluster algorithm. With two colors, the Cluster algorithm assigns one color to 58 nodes and the other color to the remaining 546 nodes. Removing the first color yields a PC of 0.817 and removing the second one yields a PC of 0.009. Thus, the average PC is 0.413. On the other hand, the Partition algorithm assigns the first color on 309 nodes and the other color on 295 nodes. This is expected since a graph partitioning algorithm tries to balance the number of nodes in each partition. Thus, in both cases the PC is close to 0.25. As a result, the minimum PC yielded by the Cluster algorithm is 0.009, while the corresponding figure for the Partition algorithm is 0.25. Note that the above example highlights some interesting observation on the pair connectivity. If the colored graph has $\frac{n}{2}$ nodes using each color (most balanced coloring with 2 colors), the maximum value for PC is close to 0.25, while that for NSLC is 0.5. It might surprise the reader that in the above example the average robustness is 0.413, larger than 0.25. Since the PC metric is quadratic in the number of nodes, a case with very



(a) Average Pair Connectivity



(b) Minimum Pair Connectivity

Figure 7. Pair Connectivity of the coloring obtained using the different coloring algorithms on the Sprint topology

good connectivity overcompensates a case with small connectivity. This explains why in Figure 7(a) the average PC for the Cluster algorithm is above the Max line.

B Randomized Coloring Analysis

Here, we conduct theoretical analysis for the effectiveness of the randomized coloring on large power-law graphs, such as the Internet AS-level topology [13]. Note that the power-law model does not apply to router-level topologies [20] and here we deal with the AS-level Internet topology where each AS randomly selects a router implementation to use on all its routers.

Let $H_m^{(n)}$ represent a random graph consisting of n nodes formed in the following way (a.k.a. preferential attachment): nodes are added one at a time, by joining each new node to an independently chosen set of m earlier nodes where the probability for choosing a given node is proportional to its degree. We then construct a subgraph $H_m^n(p)$ by deleting each node independently with probability $1-p$. Bollobás and Riordan [11] have shown that there is a "phase transition": there is a certain critical value p_c such that a component of order $\Theta(n)$ remains with high probability as $n \to \infty$ if and only if $p > p_c$.

Let $L_1(G)$ represent the size of the largest component of G, $L_2(G)$ represent the size of the second largest component of G. More formally, we have the following theorem [11]:

Theorem 1 *Let $m \geq 2$ and $0 < p < 1$ be fixed, and set*

$$p_c = \frac{1}{2}\left(1 - \sqrt{\frac{m-1}{m}}\right) \tag{1}$$

If $p \leq p_c$, then

$$L_1(H_m^{(n)}(p)) = o(n)$$

holds with high probability as $n \to \infty$. If $p = p_c + \epsilon$, $\epsilon > 0$, then

$$L_1(H_m^{(n)}(p)) = f_m(\epsilon)n + o(n) \tag{2}$$

and

$$L_2(H_m^{(n)}(p)) = o(n)$$

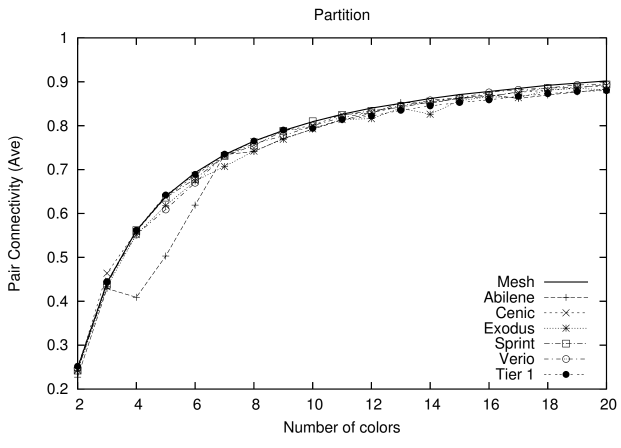
holds with high probability as $n \to \infty$, where

$$f_m(\epsilon) = exp(-\Theta(1/\sqrt{\epsilon})).$$

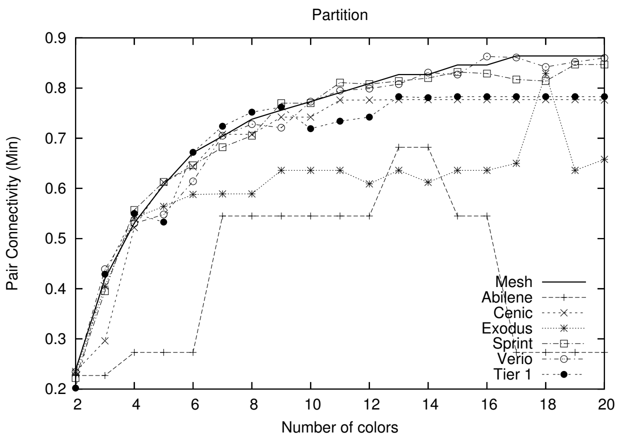
As m gets higher the critical value p_c gets lower. Thus, with $m = 2$, which is the lowest possible value of m, we have $p_c = 0.146$. Thus, for every graph with $m \geq 2$, if we retain edges with probability slightly higher than 0.146, then we expect to observe a very large component, linear to n.

Note that Theorem 1 states the necessary condition in order to retain a giant component, which is a component that has size linear to the size of the initial graph, when deleting nodes at random in a preferential-attachment random graph. With randomized coloring using k colors, when we remove all nodes of the same color, we can easily see that it is essentially equivalent to deleting nodes at random in the same graph with probability $1/k$.

Therefore, we can conclude that for a large graph following the preferential attachment model (where $n \to \infty$), even the minimum number of two colors is sufficient in order to have a giant component, because in this case we are retaining nodes with probability $p = 0.5 > p_c$. When we have 3 colors we are retaining nodes with probability $p = 0.666$.



(a) Average Pair Connectivity



(b) Minimum Pair Connectivity

Figure 8. Robustness of the different topologies when colored using the Partition algorithm.

Indeed, even though in our graphs the number of nodes does not really approach infinity, as required by the above theorem, we have performed experiments that show that when using 3 colors we achieve a high robustness.

C Algorithm Runtime

Table 4 shows the algorithms runtime on the Verio topology, which has the largest number of nodes. The algorithms

Algorithm	Run Time (19 runs)	Run Time (1 run average)
Dynamic Load	711.8	37.5
Cluster	100.1	5.3
Static Load	42.8	2.2
Dynamic Degree	31.6	1.7
Redundancy-Next	29.9	1.6
Static Degree	29.6	1.6
Random	29.5	1.6
Redundancy-Random	29.3	1.5
Partition	2.2	0.1

Table 4. Algorithm runtime (in minutes) on the Verio topology. The middle column shows the total runtime for coloring the graph 19 times, from 2 to 20 colors. The rightmost column shows the average time per color set

are run on a Pentium IV desktop computer with 1.8 GHz CPU and 1 GB of RAM. The runtime includes the time to color the graph and the time to compute the robustness of the colored graph.

The Dynamic Load algorithm is the most expensive, taking 37 minutes per color set. On the other hand, the Partition algorithm takes only a few seconds per color set, which includes both generating the partitions and coloring them. Thus, it is quite feasible to run the Partition algorithm multiple times, e.g., by varying the degree of unbalance in the graph, and select the partition with highest robustness.