Post-Failure Routing Performance with Multiple Routing Configurations

Amund Kvalbein, Tarik Čičić, Stein Gjessing
Simula Research Laboratory, Oslo, Norway

***Abstract*— The slow convergence of IGP routing protocols after a topology change has led to several proposals for proactive recovery schemes in IP networks. These proposals are limited to guaranteeing loop-free connectivity after a link or node failure, and do not take into account the resulting load distribution in the network. This can lead to congestion and packet drops. In this work, we show how a good load distribution can be achieved in pure IP networks immediately after a link failure, when Multiple Routing Configurations (MRC) is used as a fast recovery mechanism. This paper is the first attempt to improve the load balancing when a proactive recovery scheme is used. Unlike load balancing methods used with normal IP rerouting, our method does not compromise on the routing performance in the failure free case. Our method is evaluated using simulations on several real and synthetically generated network topologies. The evaluation shows that our method yields good routing performance, making it feasible to use MRC to handle transient network failures.**

I. Introduction

Traditional intradomain routing protocols, such as OSPF and IS/IS, recover from component failures by exchanging link state information and converging upon a new global view of the network state. This is a time-consuming process, that typically involves a period of instability and invalid routing in the network [1], [2], and is not sufficient for emerging time-critical internet applications.

Recently, the idea of *proactive* and *local* recovery at the IP layer has been proposed [3], [4], [5]. In these schemes, backup next-hops are prepared before a failure occurs, and the discovering router handles a component failure locally, without signalling to the rest of the network. The advantage of such solutions is that they allow an almost instantaneous response to a failure. Often, proactive recovery schemes are thought of as a first line of defense against component failures. They are used to maintain valid routing paths between the nodes in the network, until the routing protocol converges on a new global view of the topology. Such a strategy is particularly germane when facing transient failures, which are common in today's IP networks [6].

However, existing proactive IP recovery schemes are limited to guaranteeing loop-free connectivity in the network after a failure, and do not consider the post-failure load distribution. The shifting of traffic to alternate links after a failure can lead to congestion and packet loss in parts of the network [7]. This limits the time that the proactive recovery scheme can be used to forward traffic before the global routing protocol is informed about the failure, and hence reduces the chance that a transient failure can be handled without a full global routing re-convergence. Ideally, a proactive recovery scheme should not only guarantee connectivity after a failure, but also do so in a manner that does not cause an unacceptable load distribution. This requirement has been noted as being one of the principal challenges for precalculated IP recovery schemes [8]. We believe that a well engineered distribution of recovered traffic will be crucial for the adoption of any fast IP recovery method.

Important work on traffic engineering in OSPF/IS-IS networks focus on optimizing link weights, so that traffic is well distributed across the available links. The work in this area has focused either on the failure free case [9], [10], [11], or on finding link weights that work well both in the normal case and when the routing protocol has converged after a single link failure [12], [13], [14]. A major drawback of these solutions is that they compromise performance in the failure free case in order to give reasonable performance after a failure. Also, these schemes focus on the load distribution after the convergence of the IGP routing protocol, and are not designed to work with fast IP recovery schemes. Very little work has been done on the traffic engineering properties of proactive IP recovery methods.

We have previously proposed the use of Multiple Routing Configurations (MRC) to achieve fast recovery from link and node failures in IP networks [5]. MRC is a proactive recovery scheme, based on maintaining a small set of backup network configurations in the routers, which are used to reroute traffic locally in case of a failure. The local rerouting performed in MRC guarantees that a valid routing exists between any pair of nodes in an arbitrary biconnected network after a single link or node failure.

With MRC, the link weights are set individually in each backup configuration. This gives great flexibility with respect to how the recovered traffic is routed. The backup configuration used after a failure is selected based on the failure instance, and thus we can choose link weights in the backup configurations that are well suited for only a subset failure instances.

A. Our contributions

In this paper, we discuss how we can achieve a good load distribution in the network immediately after a link failure, when MRC is used as a fast recovery mechanism. We present an algorithm to create the MRC backup configurations in a way that takes the traffic distribution into account. Then, we present a heuristic aimed at finding a set of link weights for each backup configuration that distributes the load well in the network after any single link failure. Our scheme is strictly proactive; no link weights need to be changed after the discovery of a failure.

With MRC, all recovered traffic is routed in the backup configurations. This allows us, unlike previous proposals, to optimize for link failures without compromising performance in the failure free case. Also, our work is the first to address the issue of load balancing after a failure in the context of a proactive IP recovery scheme.

Our solution consists of three phases; first the link weights in the normal configuration are optimized while only taking the failure free situation into account, second we take advantage of the load distribution in the failure free case to construct the MRC backup configurations in an intelligent manner, and third we optimize the link weights in the backup configurations to get a good load distribution after any link failure.

Our method for link weight setting is based on perturbing link weights using a local search heuristic. The link weights in the backup configurations are optimized to give good performance after any link failure. Optimizing for all possible link failures does not scale well as network size increases, because of the number of evaluations needed. To overcome this problem, we assume that only a few link failures are *critical* with respect to the load distribution after failure, and optimize only over these failures [13].

We have evaluated our approach using simulations on several real and synthetically generated network topologies, and we find that we achieve a load distribution while using MRC that is better than after a full OSPF/IS-IS re-convergence with original link weights. Our results approach those of a method aimed at a good load distribution after the routing protocol has converged on the new topology [11], with the additional benefits that our method does not compromise on the performance in the failure free case.

The rest of this paper is structured as follows. We give a formal description of MRC in Sec. II. In Sec. III, we discuss what decides the post-failure load distribution under MRC, and present our algorithm for creating the backup configurations and our link weight optimization heuristic. Then we evaluate our method in Sec. IV, before we conclude and offer directions for further work in Sec. V.

II. Fast recovery using Multiple Routing Configurations

MRC is a method for fast recovery in arbitrary biconnected IP networks with shortest path routing. The method is based on creating a small set of backup routing configurations that are used in the case of a link or node failure. In the backup configurations, some links are given a weight much higher than the normal maximal link weight used in the network, thus restricting the routing in parts of the network.

The configurations are defined by the network topology, which is same in all configurations, and the associated link weights, which differ among configurations. We formally represent the network topology as a graph $G = (N, A)$, with a set of nodes N and a set of unidirectional links (arcs) A. A configuration is defined by this topology graph and the associated link weight function:

Definition. A *configuration* C_p is an ordered pair (G, w_p) of the graph G and a function $w_p : A \to \{1, \dots, w_{max}, |A| \cdot w_{max}, \infty\}$ that assigns an integer weight $w_p(a)$ to each link $a \in A$.

We distinguish between the normal configuration C_0 and the backup configurations $C_p, p > 0$. In the normal configuration C_0, all links have "normal" weights $w_0(a) \in \{1, \dots, w_{max}\}$. In the backup configurations, some links are given high weights to inhibit transit traffic:

Definition. A link $a \in A$ is *isolated* in C_p if $w_p(a) = \infty$.

Definition. A link $a \in A$ is *restricted* in C_p if $w_p(a) = |A| \cdot w_{max}$.

The link weight of restricted links is chosen so that any available path consisting of only "normal" links will be selected before one containing a restricted link by a shortest path routing algorithm. The isolated links are never used for data forwarding, while the restricted links are used only to access an isolated node:

Definition. A node $u \in N$ is *isolated* in C_p if

$$\begin{aligned} &\forall (u,v) \in A, w_p(u,v) \geq |A| \cdot w_{max} \\ \wedge \quad &\exists (u,v) \in A, w_p(u,v) < \infty \end{aligned} \tag{1}$$

In other words, nodes are isolated by assigning high weights to all their attached links. A link is always isolated in the same configuration as one of its attached nodes, but all links attached to a node can not be isolated in the same configuration, in order to make the node reachable in all configurations. The set of isolated nodes in C_p is denoted S_p, and the set of normal (non-isolated) nodes $\overline{S}_p = N \setminus S_p$.

Restricted and isolated links always have the same weight in both directions. All other links in C_p, that are neither isolated nor restricted, have weights in the normal weight range $1, \dots, w_{max}$, and may have asymmetric link weights. Figure 1 shows how all links and nodes in an example network graph can be isolated using three backup configurations.

The backup configurations have to be constructed so that, after the failure of a node or a link in the network, there will still exist a loop-free path with finite weight between every source and destination in the backup configuration where the failed element was isolated. Let $\pi_p(u,v)$ denote the shortest path from u to v in configuration C_p, and let $\mathcal{N}(\pi)$ denote the nodes on this path.

Definition. A configuration C_p is *valid* if and only if

$$\begin{aligned} \forall u, v \in N : \quad &\mathcal{N}(\pi_p(u,v)) \setminus (\overline{S}_p \cup \{u,v\}) = \emptyset \\ \wedge \quad &w_p(\pi_p(u,v)) < \infty \end{aligned} \tag{2}$$

In what follows, we assume all constructed configurations are valid. All valid backup configurations in MRC share a

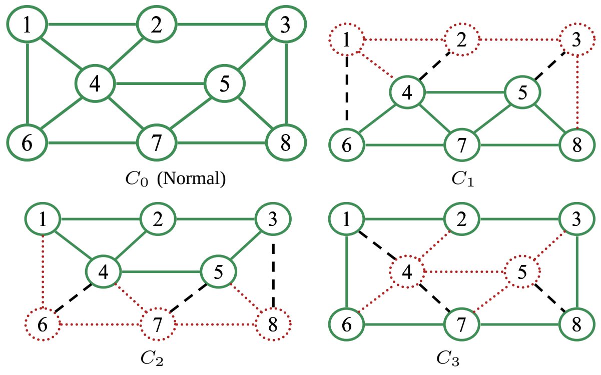


Fig. 1. An example network topology with three backup configurations. In the normal configuration, all links have weights in the normal link weight range. In the backup configurations, isolated nodes and links are depicted dotted, while restricted links are dashed.

characteristic internal structure, in that all isolated nodes are directly connected to a core of nodes connected by links with normal weights:

Definition. A configuration backbone $B_p = (\overline{S}_p, A_p), A_p \subseteq A$ consists of all non-isolated nodes in C_p and all links that are neither isolated nor restricted:

$$a \in A_p \Leftrightarrow w_p(a) \leq w_{\max} \tag{3}$$

A backbone is connected if all nodes in $\overline{S}_p$ are connected by paths containing links with normal weights only. Let $\mathcal{A}(\pi)$ denote the set of links on a path π.

Definition. A backbone B_p is *connected* if and only if

$$\forall u, v \in B_p : a \in \mathcal{A}(\pi_p(u,v)) \Rightarrow w_p(a) \leq w_{\max} \tag{4}$$

MRC constructs a set of valid backup configurations so that all links and nodes are isolated in a backup configuration. Let $\mathcal{C} = \{C_1, ...C_n\}$ be a set of backup configurations. We say that

Definition. A set, $\mathcal{C}$, of backup configurations is *complete* if

$$\begin{aligned} & \forall a \in A, \exists C_p \in \mathcal{C} : w_p(a) = \infty \\ \wedge \quad & \forall u \in N, \exists C_p \in \mathcal{C} : u \in S_p \end{aligned} \tag{5}$$

The number of backup configurations in a complete set for a given topology may vary depending on the construction model. In the construction algorithm used in this paper, each link and node is isolated in *exactly* one backup configuration. If more configurations are created, fewer links and nodes need to be isolated per configuration, giving a richer (more connected) backbone in each configuration.

For each configuration, a standard routing algorithm like OSPF or IS/IS is used to calculate configuration-specific shortest paths. Conceptually, we have a separate forwarding table for each configuration. In the normal, failure-free case, all traffic in the network is forwarded according to the normal configuration, where no links are restricted or isolated.

Let $C(u)$ denote the backup configuration where node u is isolated, i.e., $C(u) = C_p \Leftrightarrow u \in S_p$. Similarly, let $C(u,v)$ denote the backup configuration where the link (u,v) is isolated, i.e., $C(u,v) = C_p \Leftrightarrow w_p(u,v) = \infty$.

When a failure occurs, the discovering node locally diverts traffic that would normally go through the failed element to a backup configuration. The recovered packets are marked with a configuration identifier. The appropriate configuration is selected using locally available information only, and without knowing whether the loss of connectivity is due to a link or a node failure. Assume that traffic bound for egress node t can no longer be forwarded over link (u,v). Node u will select the correct backup configuration C_p as

$$C_p = \begin{cases} C(v) & \text{if } v \neq t \vee C(u,v) = C(v) \\ C(u) & \text{if } v = t \wedge C(u,v) \neq C(v) \end{cases} \tag{6}$$

For the details on how the backup configuration selection is performed, please refer to [5].

III. Routing optimization with MRC

MRC recovers from a link or node failure in the network by redirecting the affected traffic using predefined backup configurations. In this work, we restrict ourselves to only look at link failures. For a given traffic demand matrix, the load distribution in the network after a link failure depends on three factors:

1) The link weight assignment used in the normal configuration C_0.
2) The structure of the backup configurations, i.e. which links and nodes are isolated in each $C_p \in \{C_1, \ldots, C_n\}$.
3) The link weight assignments used in the backup configurations $C_1, \ldots, C_n$.

Given a network $G = (N, A)$ and a demand matrix D, let Φ be the cost of routing the traffic load through the network. Φ depends on how the load is distributed in the network, and the exact definition of Φ could depend on whether we want to minimize delay, avoid congestion etc. Our method is agnostic with respect to the choice of a particular function Φ, as long as it penalizes the use of heavily loaded links. The cost function we use in our evaluations is defined in Sec. IV.

With the shortest path routing used in OSPF/IS-IS, the cost Φ is determined by the network graph G, the demand matrix D, and the weight assignment w used in the network. Our goal is to minimize cost Φ in both the normal case and after any single link failure for a given G and D. Our strategy for achieving this is threefold. First, we use a heuristic to optimize the link weights in the normal configuration C_0. Second, we create the backup configurations $C_1, \ldots, C_n$. Third, we again use a heuristic to optimize the link weights in these backup configurations.

A. The failure free case

With MRC, all traffic is routed according to C_0 in the failure free case. When there is a failure, all recovered traffic is routed according to the appropriate backup configurations. This logical separation gives us great flexibility to distribute the recovered traffic across available links without sacrificing performance in the normal case. One of the attractive features of our solution, is that we can optimize the weights w_0 used in the normal configuration C_0 for the failure free case only, without taking the post-failure load distribution into account.

To optimize w_0, we adopt a modified version of the local search heuristic presented in [9]. We use this heuristic because it is well known and has been shown to give good performance with modest complexity, but in principle we could use any other weight search heuristic with the same objective of minimizing the cost function Φ.

The heuristic starts with a weight assignment w_0 where $w_0(a) = w_{max}/2$ for all $a \in A$, and calculates the load $l(a)$ on each link and the value of the cost function Φ resulting from w_0. Then a given number of iterations are performed. In each iteration, Φ is evaluated for a subset of the *neighborhood* of w_0. A neighbor of w_0 is a weight assignment obtained by changing the link weight of a single link. For each link in the network (one at a time), a new link weight from the range $\{1, \ldots, w_{max}\}$ is randomly picked, and Φ is evaluated after each change. The neighbor that gives the lowest value of Φ, is selected as the new w_0. To escape from local minima in the search space, the heuristic randomly changes the weight of a fraction of the links if there is no improvement after a given number of iterations. A hashing function is used to avoid looping between solutions. For a detailed explanation of the search heuristic, see [9].

B. Creating the backup configurations

The structure of the backup configurations is important for the load distribution after a failure. Traffic that is recovered in configuration C_p is forwarded only in the backbone B_p, except in the first and last hops. A configuration where many nodes and links are isolated gives a sparse (less connected) backbone. Such a configuration gives few options with regards to where recovered traffic should be routed. Conversely, a backup configuration with a *rich* backbone leaves more choices with respect to routing, and increases the possibilities to get a good distribution of load after a failure.

With MRC, the distribution of recovered traffic depends on the interaction between the structure of the backup configurations, and the weight assignments $w_1, \ldots, w_n$. Ideally, we would like to create the backup configurations and decide $w_1, \ldots, w_n$ at the same time in such a way that the cost Φ is minimized. However, such a solution would probably have to involve heavy computations, and in this work we instead settle for a solution where we first create the backup configurations, and then decide the link weight assignments. Joint optimization of the backup configuration structure and the link weight assignments $w_1, \ldots, w_n$ is left for future study.

The intuition behind our algorithm for creating backup configurations, is that we want the amount of traffic that is potentially recovered in each backup configuration to be approximately equal. We want to avoid that the failure of heavily loaded links results in large amounts of traffic being recovered in backup configurations with a sparse backbone. Instead, this traffic should be routed in a rich backbone, where we have a better chance of distributing it over less loaded links by setting appropriate link weights. The algorithm described here resembles the one we introduced in [5], with the major difference that while [5] tries to balance the number of isolated elements in each backup configuration, we here try to balance the amount of recovered traffic.

When we have decided the weight assignment w_0, the load on each link in the failure free case is given. We use this information to decide the *potential* of each node in the network and the potential of each backup configuration.

Definition. The potential $\gamma(u)$ of a node u is the sum of the load on all its incoming and outgoing links:

$$\gamma(u) = \sum_{v \in N} \left(l(u,v) + l(v,u) \right) \tag{7}$$

Definition. The potential γ_p of a backup configuration C_p is the sum of the potential of all nodes that are isolated in C_p:

$$\gamma_p = \sum_{u \in S_p} \gamma(u) \tag{8}$$

The input to our algorithm for generating backup configurations is the normal configuration C_0, and the number n of backup configurations we want to create. As we have shown before, n can be set surprisingly low; 3 or 4 backup configurations is usually sufficient to isolate all elements in a network [5]. In Sec. IV, we evaluate the effect the choice of n has on the post failure load distribution.

We start our layer generation algorithm by ordering all nodes with respect to their potential. Then each node is assigned to a tentative backup configuration, so that the potential γ_p of each backup configuration is approximately equal. The nodes with the smallest potential are assigned to C_1, those with somewhat higher potential to C_2, and so on with the nodes with the highest potential in C_n.

We then go through all nodes in the network, and isolate each node in its tentative backup configuration C_p. For some nodes, this might not be possible without breaking the definition of a valid configuration as defined by Eq. (2). This node is then attempted isolated in backup configuration C_{p+1}, C_{p+2} and so on, until all backup configurations are tried. If a node can not be isolated in any backup configuration, we give up and abort. We must then try again with a higher n. Note that when nodes can not be isolated in the backup configuration it was assigned to, this will disturb the desired property of equalizing γ_p among the backup configurations. However, in our experience this typically only happens for a very limited number of nodes, and the consequences are not severe.

The outcome of this algorithm is dependent on the network topology and the traffic demand matrix D. If the load is close to equally distributed on the links before a failure, we end up with approximately the same number of nodes isolated in each backup configuration. If the traffic distribution is more skewed (as is the case with the traffic model used in our evaluations), the algorithm typically ends up with isolating many nodes with a small potential in C_1, while only very few nodes, with a high potential, are isolated in backup configuration C_n. This is in accordance with the goal of having a rich backbone in which to reroute traffic after the failure of heavily loaded links.

C. Optimizing link weights in the backup configurations

When we have created the backup configurations $C_1, \ldots, C_n$, the next challenge is to decide the weight assignments $w_1, \ldots, w_n$. We use a similar search heuristic as in the failure free case. The straightforward way of doing this would be to evaluate the cost of the network for all possible link failures and for each candidate set of weight assignments. However, evaluating a candidate weight assignment is a rather expensive operation in terms of computing resources. The large number of evaluations needed to cover all failure instances makes this unfeasible for large networks. We therefore apply a strategy where we assume that a limited number of link failures are the most critical with respect to the load distribution, as introduced in [13]. Our method for deciding the weight assignments $w_1, \ldots, w_n$ in the backup configurations then consists of two subproblems. First we need to find the critical links, i.e the subset of links whose failure has the most grave impact on the load distribution in the network. Then we evaluate each candidate set of weight settings against the failure of the small set of critical links only. This gives a significant reduction in the number of cost evaluations needed, and makes our method feasible also for large networks.

1) Identifying critical links: Let Φ^a denote the cost of routing the demands through the network when link a has failed. We define the critical link set L_C as the k links that give the highest value of Φ^a upon failure, i.e. L_C is the set of links with cardinality k so that $\forall a \in L_C, b \notin L_C : \Phi^a \geq \Phi^b$. Note that the initial calculation of L_C is performed after we have optimized w_0, but before we have optimized $w_1, \ldots, w_n$.

There are two potential dangers with this choice of critical links. First, there might be links whose failure will give a high cost under *any* weight assignment, e.g. if there is only one possible backup path. Trying to optimize for the failure of such links is obviously futile. Second, the impact of a link failure on the network cost is a function of the current set of weight assignments. A failure that has little impact with one weight assignment, might have a grave impact with another weight assignment. We might thus end up with a situation where the failures that are in fact most damaging for the routing performance with the final weight assignment, are not included in the critical link set.

However, the independent routing of recovered traffic in the backup configurations greatly reduces the second point of criticism against our method for selecting critical links stated above. We only manipulate the weight assignments $w_1, \ldots, w_n$ used in the backup configurations in the second phase of our heuristic, and never change w_0. Hence, it is only the *recovered* traffic that is affected by the different weight settings evaluated. This makes L_C less dependent on the current weight assignments. To compensate for the dependency that

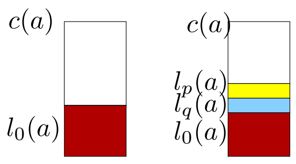


Fig. 2. Traffic on link a before and after a failure.

still exists, we recalculate L_C a few times during our search. While the first objection against our measure of criticality still holds (some failures give high cost independent of weight assignment), we will see that our selection of L_C gives good performance.

With MRC, there is a dependency between a particular link failure and the two backup configurations used to route the recovered traffic, as given by Eq. (6). Hence, the cost Φ^a after the failure of a link a in L_C is influenced only by the weight assignments w_p and w_q used in these two configurations, and not by the assignments used in the other backup configurations. For each backup configuration C_p, we define $L_p \subseteq L_C$ as the set of critical links whose failure results in recovered traffic being routed according to C_p:

Definition. The set of critical links L_p of a configuration C_p is

$$L_p = \{a \in L_C | a \notin B_p\} \tag{9}$$

2) Local search heuristic: When we have defined the critical links of each backup configuration, we perform a local search to optimize the weight assignments $w_1, \ldots, w_n$. Note first that according to Eq. (6), traffic is diverted to two different backup configurations after a failure, depending on the destination. After a failure, we will in general have traffic in two backup configurations C_p and C_q (in addition to the normal configuration). Letting $l_p(a)$ denote the load on link a that is routed according to configuration C_p, we have that $l(a) = l_0(a) + l_p(a) + l_q(a)$, as illustrated in Fig. 2.

The traffic distribution after a failure is thus dependent on the weight assignment in more than one backup configuration. Because of this, we can not optimize the weight assignments one at a time. Instead, we use an algorithm that tries to optimize all weight assignments $w_1, \ldots, w_n$ at the same time.

Like in the optimization of w_0 described above, we start with weight assignments where $w_p(a) = w_{max}/2, a \in A_p$. We then perform a given number of iterations, evaluating the cost function Φ over the critical link failures with different weight assignments. In our search heuristic, the aim is to minimize the sum Ψ of the cost of the network after the failure of each link in L_C:

$$\Psi = \sum_{a \in L_C} \Phi^a \tag{10}$$

In each iteration step, we perform the following operations:

1) First we select the next backup configuration C_p in a round robin fashion.
2) For each link a in the backbone B_p of this configuration (one link at a time), we choose a random link weight $w_p(a)$ from the interval $[1, \dots, w_{max}]$. This corresponds to evaluating $1/w_{max}$ of the neighborhood of w_p.
3) We evaluate Ψ for each of these candidate weight assignments.

Note that for the failure of the links in L_C that are not included in L_p for the current configuration, the evaluation performed in the third step will always yield the same Φ, irrespective of w_p. Hence, these values can be reused for all candidate weight assignments. We only have to recompute the cost of the network for the failures of the links in L_p. This significantly reduces the number of evaluations we have to perform in our heuristic.

If we do not see an improvement of Ψ after a given number of consecutive iterations, we jump to another area of the search space by randomly changing the link weight of a fraction of the links in the network.

3) Complexity: Optimizing n different weight assignments for a multitude of potential link failures is a complex task. An important goal in our approach has therefore been to create a heuristic that scales to networks of hundreds of nodes. This is achieved through the use of the critical link set L_C, and the further division of this into a set of critical links L_p for each backup configuration.

We can get an idea of the complexity of our heuristic by counting the number of evaluations of the network cost Φ^a we need to perform, compared to the methods in [13], [14]. These methods try to optimize a single link weight assignment only, and use the same strategy of only evaluating the most critical link failures. In each iteration, they need to calculate the value of Φ^a $|L_C|$ times for each candidate weight assignment. With our heuristic, we only need to evaluate Φ^a $|L_p|$ times for each candidate weight assignment. The number of links in L_p is dependent on the size of L_C and the number of backup configurations used to protect the network. The failure of a link (u, v) results in recovered traffic being diverted to one or two backup configurations according to Eq. (6), depending on whether u and v are isolated in the same configuration. The failure of a link can thus give traffic in at most 2 out of n backup configurations. If we assume that the number of isolated nodes are not very different between the configurations, we have that, on average, $|L_p|$ is roughly $|L_C| \cdot \frac{2}{n}$.

In each iteration, we only alter the link weights of links A_p in the backbone B_p of the current configuration. The weights of the isolated and restricted links that are not included in B_p are decided by MRC, and can not be changed. Obviously, the number of links $|A_p|$ in each backbone B_p is less than $|A|$.

To sum up our discussion so far; if we use i iterations with the methods described in [13], [14], evaluating the network cost Φ^a $|A|$ times with $|L_C|$ different link failures in each iteration, we will perform a total of $i \cdot |A| \cdot |L_C|$ evaluations of Φ^a. With our method, if we perform i iterations for *each* backup configuration, we end up with a total of

$$n \cdot i \cdot \overline{|A_p|} \cdot \overline{|L_p|} < 2 \cdot i \cdot |A| \cdot |L_C| \tag{11}$$

evaluations of Φ^a. This means that even if we let the number of backup configurations grow, we never need more than twice the number of evaluations needed by [13] and [14].

Evaluating Φ^a involves calculating a shortest path tree for each destination in the network. This can be done in a more efficient way by relying on incremental calculations [15] when evaluating Φ^a for different failures. Evaluating Φ^a is somewhat more expensive when using MRC, since we need to calculate shortest paths in one or two backup configurations in addition to the normal configuration. On the other hand, since we optimize for a smaller number of failures in each backup configuration, we have found that we can decrease the number of iterations used per configuration, and still achieve good results. All in all, our experience is that the running time of our heuristic is comparable to that of [14].

IV. Performance evaluation

We have evaluated our approach using simulations for a range of real and synthetically generated network topologies. We use the network cost and the maximum link load after failure as performance metrics.

A. Method

1) Topologies and traffic: We have tested our mechanism on topologies from four existing or planned real-world network topologies from the Rocketfuel [16] database: Sprint US (PoP level, 32 nodes, 64 links), COST239 (11 nodes, 26 links), Geant (19 nodes, 30 links) and German Telecom (10 nodes, 17 links). We have also performed tests on synthetically generated topologies. We generated topologies of four different classes - 32 nodes and 64 links, 32 nodes and 96 links, and 128 nodes and 256 links. The synthetic topologies were generated using the Waxman topology model [17]. For all the topologies, both real and synthetic, all links have an equal abstract link capacity of 1 in our tests.

To evaluate the link load changes after the failure, it is necessary to know the traffic demands between all network origins and destinations. Even for real networks, this data is generally unavailable, due to its confidentiality and difficulties in collecting it. We chose to synthesize the origin-destination (OD) flow data by drawing flow values from a probability distribution, and matching the values with the OD pairs using the heuristic described in [18]. In short, we sorted the OD pairs according to their node degree and the likelihood of one of them being used as the backup node in the case of a single link failure. Then, we matched the sorted OD pair list with the sorted list of flow intensities generated using the gravity model, which is suited for this purpose [19].

Once the OD matrix is generated, it needs to be scaled to the link capacities so that it can provide a meaningful evaluation of the effect of link failures on the flows. It has proven hard to find a general parameter setting that achieves this for all networks. We chose to tune the load so that the maximum link load after the worst case failure is about 100%. In most cases, this corresponds to a maximum link load in the failure-free case of approximately 2/3 of the link capacity.

2) Routing and cost function: We used shortest path routing in all calculations. When multiple equal cost paths toward a destination were available, the load was split equally among them.

To evaluate a given weight assignment, we must define the cost Φ of routing a given traffic demand through the network. In this work we choose to adopt the commonly used cost function introduced in [9]. Using this cost function, each link a is given a cost ϕ_a dependent on its load $l(a)$ and its capacity $c(a)$. The total network cost $\Phi = \sum_{a \in A} \phi_a$ is then the sum of the cost of each link. The cost $\phi_a(l(a))$ of a link is defined as the continuous function with $\phi_a(0) = 0$ and derivative:

$$\phi'_a(x) = \begin{cases} 1 & for \quad 0 \leq x/c(a) < 1/3, \\ 3 & for \quad 1/3 \leq x/c(a) < 2/3, \\ 10 & for \quad 2/3 \leq x/c(a) < 9/10, \\ 70 & for \quad 9/10 \leq x/c(a) < 1, \\ 500 & for \quad 1 \leq x/c(a) < 11/10, \\ 5000 & for \quad 11/10 \leq x/c(a) < \infty \end{cases} \tag{12}$$

The cost function $\phi_a(l(a))$ is defined so that it is cheap to send traffic over lightly-loaded links, while adding traffic to a link a that is already overloaded gives a very high value of ϕ_a.

3) Evaluation setup: In our experiments, we optimized C_0 for the failure free case using the heuristic described in Sec. III-A with 1000 iterations. We jumped to another area of the search space by randomly perturbing weights if we saw 200 iterations without an improvement. When optimizing the link weights in the backup configurations $C_1, \ldots, C_n$, we used as little as 20 iterations per backup configuration, and did a random perturbation after 10 non-improving iterations. We used a critical link set size $|L_C| = 20$.

As an evaluation benchmark in our experiments with the GEANT network, we compare our method to an unrealistic full rerouting approach where link weights are optimized to fit the new topology after each specific link failure. This optimization is done in the same way as the optimization of the failure free C_0. Performing this operation for every link failure takes much computing resources, and is only feasible in our experiments for small networks. To test the performance of our weight setting heuristic, we also compare to an idealized MRC approach where link weights in the backup configurations are optimized to fit a single link failure only. We use the same heuristic as before, but in each iteration we evaluate Φ^a for a single link only, instead of taking all critical links into account.

In our evaluation of real and synthetic networks shown in Tab. I, we show the performance of MRC using 5 and 10 backup configurations. We compare this to the results given by a complete OSPF/IS-IS re-convergence on the normal configuration. Also, in lack of other proactive recovery mechanisms that try to optimize the routing after a failure, we compare MRC performance against the method for robust routing described in [14]. This method constructs a single set of link weights that performs well in both the failure free case and with a single link failure. It is not designed to work with any fast reroute mechanism, and the load distribution is hence only achieved after a full shortest path re-convergence on the new topology. A drawback with this method is that its performance can not be optimized for failure free operation only. In our experiments, parameters are set so that we allow a cost increase of up to 20% in the failure free case with this method.

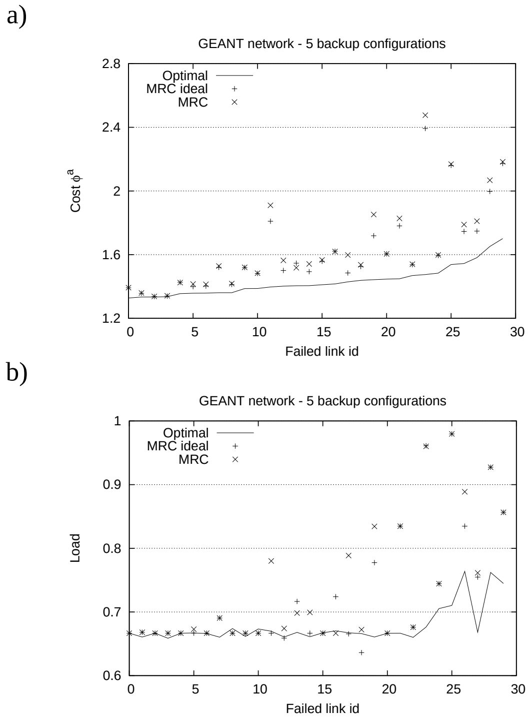


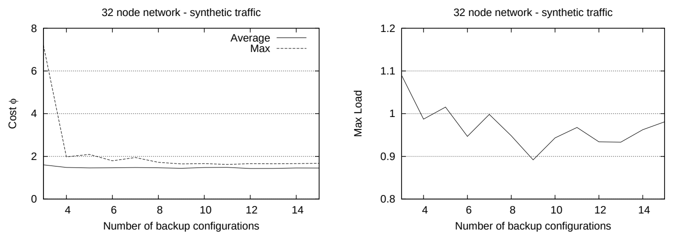
Fig. 3. Cost Φ^a and maximum link load in the network after each link failure.

We use the cost Φ and the load on the most loaded link in the network as our evaluation parameters. To be able to compare networks of different size, we normalize Φ with the cost of routing the demand through the same network with unlimited link capacities, i.e. a network where $\phi_a = l(a)/c(a)$ according to Eq. (12).

B. Results and discussion

1) Cost and link loads in a single network: Figure 3a shows the network cost Φ^a after the failure of each link in the GEANT network topology. The cost is shown for the unrealistic optimal shortest path rerouting, idealized MRC, and our MRC approach. The link failures are sorted on the x-axis after increasing cost in the optimal case. The traffic demand is scaled so that the cost Φ is 1.33 in the failure free case, giving a maximum link load of 0.67. Figure 3b shows the maximum link load in the network after the same link failures.

The graphs show that for most failures, MRC performance is close to that of the unrealistic optimal rerouting. For a few link failures, our MRC approach diverts more from the



(a) (b)

Fig. 4. Cost Φ^a and maximum link load in the network after the worst case link failure. Φ^a is median over 20 runs, maximum load is mean over 20 runs.

optimal. In these cases, the MRC backup configurations are constructed so that recovered traffic is routed over links that are already somewhat loaded. We see that when this happens, the performance of our heuristic is close to that of the idealized MRC. This indicates that if we want to further improve the performance of MRC, we could expect the best results by improving the backup configuration construction algorithm, instead of creating a better weight search heuristic. Note that MRC sometimes gives a lower maximum link load than the optimal shortest path rerouting. This happens when MRC is forced to create longer recovery paths (giving a higher Φ) due to the restrictions in the backup configurations, but this happens to avoid the most heavily loaded link that would otherwise be used.

2) Varying the number of backup configurations: Figure 4a shows the network cost Φ^a after the worst case link failure for a synthetically generated network with 32 nodes and 64 links, using a varying number of backup configurations. Since our weight setting search contains an element of randomness, we sometimes experience cost values that deviate significantly from what is expected. To mitigate this effect, the values shown are the median value obtained by running our algorithm 20 times with a different seed.

As expected, we see that the cost is highest when the minimum number of backup configurations (3 for this network) is used. The load balancing improves when we increase the number of backup configurations used. Since each node in the network is isolated in exactly one backup configuration, increasing the number of backup configurations gives richer backbones to route the recovered traffic in. We see that increasing the number of configurations used beyond 8 gives a very limited effect for this network. We have observed similar trends for other networks. This indicates that it is possible to achieve a good load balancing using a modest number of backup configurations. As seen in Fig. 4b, the maximum link load after the worst case failure shows more variation than the maximum Φ^a. This is a result of the piecewise linear nature of the cost function in Eq. (12), which does not prefer two links with load 0.95 to one link with load 0.90 and one with load 1.00.

3) Evaluation over different networks: We have evaluated the network cost and the maximum link load after the worst case link failure for a range of real-world and synthetically generated network topologies, as shown in Tab. I. Results are shown for MRC using 5 and 10 backup configurations, a normal full SPF re-convergence, and the method described in [14], denoted S/G. We have run experiments for 5 different topologies of each type of synthetic topologies. For each recovery method we show the average cost Φ^a_{avg} after each link failure, the cost Φ^a_{max} after the worst case link failure, and the load l_{max} of the most heavily loaded link after the worst case link failure. We also show the cost Φ and the maximum link load in the failure free case. The values shown in the table are median values over 3 runs with different seed.

The general trend is that MRC performs better than the normal shortest path rerouting after the worst case link failure, with respect to both cost and maximum link load. MRC performance is improved if we increase the number of backup configurations used. Using 10 backup configurations, MRC performance gets close to that of the S/G method, and for networks of moderate size and connectivity (T32-64), MRC performance is as good as that of S/G. The cost Φ in the failure free case is up to 20% higher with the S/G method than with MRC - in our experiments we typically saw values that were 3-15% higher.

Normal SPF re-convergence performs better for larger networks (T128-256) than for small networks. We believe this is partly a result of the traffic model used. With larger networks, the chance that a heavily loaded link is selected in a backup path decreases, and a normal shortest path re-convergence is closer to the optimal solution.

V. Conclusion and Future Work

In this paper, we have argued that the post-failure load distribution should be taken into account when designing a proactive recovery scheme for IP networks. We think this is imperative for the adoption of any such scheme. We presented an algorithm for creation of backup configurations and a link weight assignment heuristic that reduces the chance of congestion after a link failure when MRC is used for recovery. Our method does not compromise performance in the failure-free case, and it is strictly pre-configured; no calculations are necessary after the failure.

We have evaluated our method using both real and synthetic network topologies. Our results show that by using our scheme, MRC offers better post-failure load distribution in the network than what is achieved by a full global rerouting using the original link weights. In particular, our heuristic reduces the load on the most loaded links in the network after a worst-case link failure compared to a normal shortest path rerouting. The performance of our method is about the same as that of the method described in [14], which is not designed to be used with a proactive IP recovery scheme and that reduces performance in the failure free case.

There are several possible directions for future work related to the present study. While in this paper the backup configuration construction and the link weight optimization are two separate steps, we believe that even better results can

TABLE I

COST AND MAXIMUM LINK LOAD FOR SELECTED REAL AND SYNTHETIC NETWORK TOPOLOGIES

			Proactive recovery						**Reactive recovery**					
	Failure free		**MRC n=5**			**MRC n=10**			**S/G**			**Normal SPF**		
Network	Φ	l_{max}	Φ^a_{avg}	Φ^a_{max}	l_{max}	Φ^a_{avg}	Φ^a_{max}	l_{max}	Φ^a_{avg}	Φ^a_{max}	l_{max}	Φ^a_{avg}	Φ^a_{max}	l_{max}
German Tel	1.40	66%	1.91	4.85	102%	1.95	5.00	102%	1.63	2.05	81%	14.60	86.53	117%
Geant	1.36	68%	1.65	2.39	101%	1.69	4.94	108%	1.58	1.90	90%	2.54	31.91	120%
Sprint US	1.18	64%	1.40	6.05	110%	1.39	6.00	110%	1.40	5.58	110%	1.35	5.53	110%
Cost239	1.39	66%	1.57	2.62	99%	1.56	2.62	99%	1.51	1.94	79%	1.55	2.61	99%
T32-64-0	1.33	66%	1.48	2.20	103%	1.45	1.59	82%	1.42	1.60	87%	1.41	1.63	98%
T32-64-1	1.26	59%	1.39	1.73	95%	1.38	1.54	75%	1.36	1.54	94%	1.34	1.91	102%
T32-64-2	1.33	67%	1.48	2.21	100%	1.48	2.21	100%	1.42	1.52	89%	1.42	2.15	104%
T32-64-3	1.30	67%	1.46	2.65	105%	1.46	2.65	105%	1.61	3.04	109%	1.47	5.17	111%
T32-64-4	1.29	66%	1.42	1.91	96%	1.41	1.79	90%	1.35	2.04	102%	1.36	2.28	103%
T32-96-0	1.35	67%	1.43	1.99	104%	1.42	1.62	99%	1.39	1.47	92%	1.41	2.23	109%
T32-96-1	1.34	78%	1.46	3.60	110%	1.45	3.22	111%	1.39	1.85	101%	1.50	10.86	114%
T32-96-2	1.36	72%	1.59	7.60	117%	1.46	1.85	103%	1.43	3.05	111%	1.56	6.88	114%
T32-96-3	1.35	65%	1.44	2.27	108%	1.42	1.63	100%	1.41	1.57	98%	1.39	1.69	101%
T32-96-4	1.36	76%	1.48	5.05	113%	1.48	4.02	111%	1.40	1.53	97%	1.46	4.89	112%
T128-256-0	1.23	67%	1.27	1.34	91%	1.26	1.32	95%	1.25	1.28	86%	1.25	1.28	90%
T128-256-1	1.21	66%	1.24	1.30	83%	1.24	1.30	85%	1.23	1.25	73%	1.22	1.24	70%
T128-256-2	1.18	67%	1.21	1.31	92%	1.21	1.33	93%	1.19	1.22	72%	1.19	1.22	72%
T128-256-3	1.20	66%	1.23	1.31	84%	1.23	1.31	90%	1.22	1.23	82%	1.21	1.24	82%
T128-256-4	1.20	66%	1.23	1.31	84%	1.23	1.31	90%	1.21	1.24	76%	1.21	1.23	76%

be achieved by unifying these two processes. We also think that the idea of multiple parallel network configurations can be used to give dynamic load balancing in the failure-free case, by diverting traffic away from heavily-loaded links using an alternative configuration.

In the final stages of the work on this paper, we discovered the technical report [20]. This report describes a method for creating multiple topologies to achieve fast rerouting in IP networks, and a heuristic to set link weights in the topologies. They compare their post-failure load distribution to that achieved by using the not-via approach [4], and find that their multi-topology strategy performs better for the tested networks according to their metrics. Both the method for creating backup topologies and for setting link weights is substantially different from ours. As future work, we plan to compare the performance of this method to that of our own.

ACKNOWLEDGEMENTS

We would like to thank our colleague Olav Lysne for help and insightful comments during the work with this paper.

REFERENCES

[1] D. Watson, F. Jahanian, and C. Labovitz, "Experiences with monitoring OSPF on a regional service provider network," in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. IEEE Computer Society, 2003, pp. 204–213.

[2] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet Routing Convergence," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 293–306, June 2001.

[3] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs. reactive approaches to failure resilient routing," in *Proceedings INFOCOM*, Mar. 2004.

[4] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using not-via addresses," Internet Draft (work in progress), Oct. 2005, draft-bryant-shand-IPFRR-notvia-addresses-01.txt.

[5] A. Kvalbein, A. F. Hansen, T. Čičić, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *Proceedings INFOCOM*, Apr. 2006.

[6] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone network," in *Proceedings INFOCOM*, Mar. 2004.

[7] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proceedings INFOCOM*, Mar. 2003, pp. 406–416.

[8] S. Rai, B. Mukherjee, and O. Deshpande, "IP resilience within an autonomous system: Current approaches, challenges, and future directions," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 142–149, Oct. 2005.

[9] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights." in *Proceedings INFOCOM*, 2000, pp. 519–528.

[10] Y. Wang, Z. Wang, and L. Zhang, "Internet traffic engineering without full mesh overlaying," in *Proceedings INFOCOM*, April 2001, pp. 565–571.

[11] A. Sridharan, R. Guirin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 234–247, April 2005.

[12] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, "IGP Link Weight Assignment for Transient Link Failures," in *18th International Teletraffic Congress*, Berlin, Germany, Aug. 2003.

[13] B. Fortz and M. Thorup, "Robust optimization of OSPF/IS-IS weights," in *INOC*, oct 2003, pp. 225–230.

[14] A. Sridharan and R. Guerin, "Making IGP routing robust to link failures," in *Proceedings of Networking*, Waterloo, Canada, 2005.

[15] G. Ramalingam and T. Reps, "An incremental algorithm for a generalization of the shortest-path problem," *J. Algorithms*, vol. 21, no. 2, pp. 267–305, 1996.

[16] "Rocketfuel topology mapping," WWW, http://www.cs.washington.edu.

[17] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[18] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating IP traffic matrices: Initial recommendations," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 19–32, July 2005.

[19] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 93–96, Oct. 2005.

[20] G. Apostolopolous, "Using multiple topologies for IP-only protection against network failures: A routing performance perspective," ICS FORTH, Crete, Greece, Tech. Rep., apr 2006.