Fast Proactive Recovery from Concurrent Failures

Audun Fosselie Hansen†, Olav Lysne*, Tarik Čičić* and Stein Gjessing*
*Simula Research Laboratory, Oslo, Norway
† Telenor R&I, Oslo, Norway
Email: {audunh, olavly, tarikc, steing}@simula.no

Abstract—Recovery of traffic in connectionless pure IP networks has traditionally been handled by a full re-convergence of the network state. This process operates in a time scale that is not compatible with new real time and highly dependable services. Recently, schemes for fast local and proactive recovery in connectionless IP networks have been proposed. All these schemes are designed to guarantee recovery of the failure of one component. As IP protocols are used to carry more highly dependable services and new wireless infrastructures are approaching, guaranteed failure coverage of more than one failure becomes necessary. In this paper we present and evaluate a scheme that guarantees to handle any two concurrent failures in a network. We are not aware of any other schemes that addresses such guarantees. We evaluate and compare it with other known recovery schemes, and we show how it gives substantially better recovery success rates than the schemes designed for one fault tolerance, also for more than two failures.

I. INTRODUCTION AND BACKGROUND

The dependability requirements on IP and its related protocols are continuously increasing due to three main reasons. First, the IP protocols are carrying more and more services that heavily influence our daily lives, with respect to both personal and business tasks. Second, the IP protocols are used in special purpose networks like emergency networks and industry automation networks where the dependability requirements are very high. Third, the IP protocols are increasingly adopted to new wireless infrastructures that are more failure intensive than traditional fixed networks. Examples of those are ad hoc networks [1] and wireless mesh networks [2].

Recovery of failures in IP networks is currently handled by a full re-convergence of the network state. This process of IP re-convergence does not operate in a time-scale compatible with the new services offered today. Improvements have been proposed for all the steps associated with a full IP re-convergence. The interval of the hello-messages has been decreased [3], the update dissemination has been more effective [4], [5], [6] and the shortest path three calculation is running faster [7]. Despite all these improvements, the convergence time is still close to a second [8]. The reason for this is that the timers associated with the routing protocols cannot be tuned below a certain level, due to problems with route flapping and instability [9], [8].

The overall reason for the long time-scale is the fact that IP re-convergence is a reactive and global process. To speed up the process of IP recovery the only option is to design a proactive and local approach, which means that the node detecting a failure initiates the rerouting and the alternative next hops have been pre-calculated. MPLS fast reroute is a typical example of how this can be solved in a connection-oriented network [10]. However, in a connectionless network where forwarding decisions are taken independently hop by hop, such an approach is not straightforward due to potential looping of packets. Recently, several schemes have been proposed to solve this challenge for connectionless networks. [11] gives an overview and a conceptual comparison of some of them. IETF has also started a standardization process of IP fast reroute, and they point at three different candidates to solve fast reroute [12]. These are Failure Inferencing based Fast Rerouting (FIFR) [13], IP fast reroute using not-via addresses (not-via) [14] and a method proposed by the authors, named Multiple Routing Configurations (MRC) [15]. These methods are designed to cover any single link or node failure in a network.

A lot of studies have been conducted to explore the failure characteristics in fixed IP networks [16], [17], [18]. They show that multiple simultaneous failures must be considered. Such failures can be characterized as either correlated (failing due to same cause) or uncorrelated. We then define concurrent failures as both correlated and uncorrelated failures that affect the traffic in the network at the same time. The arguments for handling concurrent failures are continuously growing stronger. For very dependable services, both on the Internet and in special purpose networks, even rare concurrent failures should be recovered if possible. Catastrophes like flooding, terror attacks and power outages are other examples of extreme events where some important services must survive. The needs for communication increase while the infrastructure capabilities decreases [19], [20]. The slow re-convergence of IP is another important argument for handling more than one failure. During this re-convergence process there is a certain probability that a new failure may occur before the process has completed. Such failures are not guaranteed to be covered by the current proactive methods. In [17], such failures have been proven and named “overlapping” failures.

In wireless networks, the needs to address concurrent failures are implicitly given due to vulnerability to signal loss and degradation. In addition, nodes constituting an ad hoc network will to some extent be mobile, and a node losing the signal of its next hop would regard that as a failure. Some ad hoc networks run link state routing protocols (e.g. OLSR [21]) like connectionless fixed IP networks, and hence they share the challenges regarding proactive recovery of concurrent failures [12], [2].

In this paper, we will present 2DMRC as a method that
guarantees fast recovery from any two concurrent failures in a connectionless IP network. 2DMRC is built on the same basic principles as MRC [15] and uses multi-topology routing (MTR) [22] to build backup topologies that isolate a certain number of components. We also evaluate 2DMRC and compare it with the three schemes mentioned in [12], i.e. not-via, FIFR and MRC. To compare the relevant properties of our methods, we adjust MRC to use the same forwarding and weight properties as 2DMRC (sec. IV). This adjusted method will be referred to as DMRC.

The outline of this paper is as follows. Section II will introduce assumptions and problem statements regarding local and proactive methods for two fault tolerance. Section III will present the most relevant schemes for proactive local recovery. Section IV will present our scheme that guarantees recovery from any two concurrent failures (2DMRC). Section V will present evaluations of the scheme with respect to scalability, recovery success rates and backup path lengths. Finally, section VI will give a short discussion, conclusion and prospects of future work.

II. PROBLEM STATEMENT

Addressing local, proactive recovery of multiple failures in a network is a challenging task. We make several assumptions to target the problem area. First, we have that if a method shall be able to protect against any situation where \( k \) components in a network fail, the network must itself be at least \( (k + 1) \)-connected. The main goal of this paper is to devise and evaluate a local, proactive method that guarantees recovery of two concurrent failures in three-connected networks.

A considerable challenge regarding multiple failures in networks is related to packet looping. Since the recovery actions are taken locally by the node detecting a failure, this node has no knowledge about other failures in the network when it decides where to reroute a packet. Hence, a packet might reach a new failure or loop back to a previous failure. Also the connectionless nature of pure IP networks can result in loops since all forwarding decisions are taken hop-by-hop. Moreover, if a network has more failures than the recovery method can handle, packets can also potentially reach an infinite loop.

Another challenge considered is the failure of a link towards an egress node that is the only exit of the network. This challenge is denoted the “last hop problem”. In most cases a scheme that is designed to handle node failure recovery implicitly addresses link failures too, as the adjacent links of the failed node can be avoided. This is true for intermediate nodes, but in the last hop case such a scheme will drop packets unnecessarily if it is only a link that has failed and not the entire node. When there is more than one failure in the network, it is considered very hard to provide guarantees for the last hop case since every pair of nodes, links and their combination must be isolated in some way. We are not aware of any approach that can offer a scalable solution to this, and hence we address this problem separately.

2DMRC, that we present in this paper, guarantees recovery of two regular concurrent failures in three-connected networks. It also guarantees that infinite packet looping will not occur.

III. RELATED WORK

There exist some proposals for proactive recovery of single component failures in connectionless IP networks, but we are not aware of any schemes that addresses guarantees for more than one failure. Most of the schemes for IP recovery are presented and conceptually compared in [11]. Some of the schemes presented are not fully proactive, and hence do not operate in the required time-scale. Others do not operate in pure IP networks or put some restrictions on the network layout. We focus on the schemes that are fully proactive (fast recovery), connectionless, guarantee single failure recovery in two-connected networks and that are pointed out in the IETF framework for IP fast reroute [12]. Those are MRC, FIFR and not-via. These methods will also be compared to 2DMRC in section V.

A. Multiple Routing Configuration (MRC)

In [15], the authors presented Multiple Routing Configuration as a method to proactively recover any single node or link failure in connectionless IP networks.

Basically, MRC uses the network graph and the associated link weights to produce a set of backup network configurations. A backup configuration is defined as the original topology with manipulated link weights. The link weights in these backup configurations are manipulated so that for each link and node failure, and regardless of whether it is a link or node failure, the node that detects the failure can safely forward the incoming packets towards the destination by selecting the appropriate backup configuration. This approach can be implemented using multi-topology routing, standardized within IETF [22], [23], as one of the multiple topologies can represent a backup configuration in MRC. Since IETF refers to an MRC configuration as a topology, we will throughout this paper use the term backup topology instead of backup configuration. Packet marking will determine according to what topology a node should route the packet. The node that detects the failure will then mark the packet with the appropriate topology ID.

MRC, as presented in [15], solves the last hop problem (sec. II) for one fault tolerance by assigning three levels of link weights to the backup topologies. The first level of link weights is the original link weights from the original topology. A link assigned to the second level is refereed to as a restricted link and is used to isolate nodes, i.e. ensure that no traffic will be routed through that node in a backup topology. The third level is referred to as isolated links, and the goal of this level is to explicitly isolate a link. This third level is used to solve the last hop problem (sec. II) when there is one failure in the network.

The restricted links are assigned a link weight so high that no traffic will be routed over the link unless there are no other options. The isolated links are assigned a link weight of infinity, which means that the link will never be used in that topology.

Figure 1 gives an example of how every node and link from the original topology \( T_0 \) can be isolated using three backup topologies \( (T_1, T_2, T_3) \). In topology \( T_3 \) for instance, node 4 and 5 are isolated and will never be used for transit traffic.
Moreover, links 4-6, 2-4, 4-5, 3-5 and 5-7 are isolated and will never carry traffic at all. Links 1-4, 4-7 and 5-8 are restricted and will only be used as first hop from an ingress node or as a last hop towards an egress node. As an example, let's look at traffic that in $T_0$ would follow the path 1-4-5 with node 1 as ingress and node 5 as egress. If node 4 or the link 1-4 failed, node 1 would forward the packet in topology $T_3$ to node 2, which would then forward to node 3 in topology $T_3$ etc. The path would then be 1-2-3-8-5.

**B. Failure Inferencing based Fast Rerouting (FIFR)**

Failure Inferencing based Fast Rerouting (FIFR) [13] has been proposed as an extension of [24] to also cover node failures. The main ideas behind FIFR is that a router will maintain interface-specific forwarding tables, and that a router should be able to infer whether a packet has experienced a failure and is not routed according to the default forwarding tables. When a packet arrives at a node through an unusual interface (through which it would never arrive had there been no failure), the corresponding potential component failures can be inferred, and a next hop avoiding those components is chosen. These interface-specific forwarding tables can be precomputed. Thus under FIFR, when a component fails, only nodes adjacent to it locally reroute packets to the affected destinations, and all other nodes forward packets according to their precomputed interface-specific forwarding tables without relying on network-wide link-state advertisements.

Evaluations of FIFR for single fault tolerance can be found in [13].

**C. Tunneling using not-via addresses (not-via)**

The main idea behind the not-via approach [14] is to precalculate tunnels to a not-via address that detours the packets around the failure in the same manner as the connection-oriented counterpart MPLS local detours [10]. Figure 2 illustrates the basic principles.

One must ensure that the packets affected by the failure of router E are delivered to router M that according to the primary route to destination D is downstream of E (figure 2). Routers advertise not-via addresses for all their neighbor components. A not-via address is used by other routers when the corresponding component has failed. Each router in the network must calculate the best path to each not-via address or group of addresses. The path is calculated on a network topology that does not contain the component the not-via address is meant to protect. The router S that detects the failure will then encapsulate the packets and address them to the not-via address that router M has advertised for the particular failure (Me in figure 2). The routing table of router S and all other routers in the network will have a destination address Me (fig. 2) which have been calculated on a topology without router E.

For evaluations of not-via with respect to one failure tolerance, we refer to [25], [26].

**IV. 2DMRC - HANDLING TWO CONCURRENT FAILURES**

The recovery schemes presented in section III will in some cases be able to recover the traffic from more than one failure, however, they provide no guarantees. The recovery success depends on the probability of reaching the second failure and the probability of looping back to the first failure. For highly dependable services such probabilities will not suffice, and a method providing guarantees is necessary. 2DMRC is designed to provide these guarantees when a network contains two concurrent failures. 2DMRC will of course also guarantee recovery if there is only one failure in the network.

2DMRC shares some of the basic properties with MRC [15] in terms of building backup topologies and using multi-topology routing [22], [23]. However, while MRC [15] uses three levels of link weights, 2DMRC only uses two levels: the original weights and the weight for restricted links. The main reasons for this are to enable a simple algorithm for building backup topologies and to enable a simple forwarding scheme. In addition, we expect an approach using only two levels of link weights to offer more routing flexibility and also a higher probability to provide a modest number of backup topologies. Although 2DMRC uses only two levels of link weights, the properties offered by MRC can still be met by doing some adjustments to the forwarding process (sec. IV-C).

**A. Basic design constraints**

Before we dive into the details of how to construct the backup topologies, we introduce some basic requirements that must be fulfilled to build a complete and valid set of backup topologies. These requirements are necessary to ensure that every pair of failures can be recovered, and to ensure that the packets are routed correctly also when there exist two concurrent failures in the network.

**Req 1:** A node must not carry any transit traffic in the backup topology where it is isolated. Still, traffic must be able to be forwarded from an isolated ingress node and reach an isolated egress node.

**Req 2:** In each backup topology, all node pairs must be connected by a path that does not pass through an isolated node.
Algorithm 1 presents an algorithm that builds backup topologies for graphs that are at least two-fault tolerant, i.e., networks that are not disconnected when any two components fail. The main building blocks of the algorithm and at the same time define some invariants. Requirement 1 determines the weight \( W \) of restricted links. In addition, the requirements lead to the following invariants which will be explained below.

**Inv 1:** A pair of nodes can be isolated in a backup topology only if all other nodes in the network can reach each other without passing through any of the nodes from this pair, i.e., their removal will not disconnect the current topology (from Req 1 and Req 2).

**Inv 2:** A backup topology can only be deleted from the set of backup topologies if the nodes that it isolates can all be isolated in another existing backup topology (from Req 3).

Algorithm 1 presents an algorithm that builds backup topologies according to these invariants. We use the notation from Table I. It should be noted that we have optimized some of the steps in the real implementation to improve the execution time. However, to make the algorithm more readable and understandable, we present here this simplified version.

The main building blocks of the algorithm can be divided into an initial part (line 2 - 20) and a merge part (line 25 - 46). The initial part creates one initial backup topology for every pair of nodes in the network. For a network with \( n \) nodes, the number of initial backup topologies would be \( \frac{n(n-1)}{2} \).

The merge part then attempts to merge these initial backup topologies without violating the invariants presented above. The merge part starts with one of the \( k \) initial backup topologies \((G_k)\). Then it successively tests if this backup topology also can isolate any of the pairs \((v_i, v_j)\) from the other \( k - 1 \) initial backup topologies. This can be done only if isolating \((v_i, v_j)\) in \((G_k)\) will not disconnect the other nodes in \((G_k)\). If this can be done successfully, the initial backup topology that isolates \((v_i, v_j)\) can be deleted from the set of initial backup topologies.

\( isolated(G_k) \) finds the two isolated nodes in the initial backup topology \( G_k \).

\( div(v_i, v_i, G_k) \) tests whether both \( v_i \) and \( v_j \) can be removed from topology \( G_k \) without disconnecting the topology, and hence explicitly tests that invariant 1 holds and indirectly that invariant 2 holds. If \( div() \) returns false, the pair \((v_i, v_j)\) will not be deleted from the set of initial backup topologies and not be added to the merged topologies, and hence invariant 2 is not violated.

### Algorithm 1: Creating Backup Topologies

```plaintext
2 b = 0
3 G0 = G
4 S = ∅
6 forall v_i ∈ V do
8     forall v_j ∈ V : v_j ≠ v_i do
9         if div(v_i, v_j, G_k) then
11             forall e_{i,n} ∈ E_i do
12                 w_{i,n}^b = W
13         end
15             forall e_{j,n} ∈ E_j do
16                 w_{j,n}^b = W
17         end
18     S = S ∪ G_k
20     b = b + 1
22 end
23 end
25 b = 0
26 M = ∅
27 while S ≠ ∅ do
28     G_k = G_k ∈ S
29     S = S \ G_k
30     forall G_k ∈ S do
31         (v_i, v_i) = isolated(G_k)
32         if div(v_i, v_i, G_k) then
34             forall e_{i,n} ∈ E_i do
35                 w_{i,n}^b = W
36         end
38             forall e_{j,n} ∈ E_j do
39                 w_{j,n}^b = W
40         end
42     S = S \ G_k
44 end
46 M = M ∪ G_k
48 b = b + 1
49 end
```

### C. Forwarding of packets with 2DMRC

The algorithm from the previous section builds backup topologies that can be regarded as additional routing tables in each node. In this section we describe a forwarding scheme that utilizes these backup topologies. Like multi-topology routing ([22], [23]), we assume that a packet header contains a
topology ID pointing at what “routing table” the packet should be forwarded according to. When a node receives a packet, it then looks up the destination address and the topology ID to identify the routing table and the corresponding next hop to forward the packet.

This becomes more complicated when the node that receives a packet has detected a failure. This node must then change the topology ID in the packet header to a backup topology that will not return the failed interface as the next hop. If there are at most two failures (non last hop) in the network, 2DMRC guarantees that there exist a backup topology that will avoid both failures.

Algorithm 2 gives a pseudo code overview of the forwarding decisions for a node when it has detected a failure. The basic idea is to successively try the backup topologies from the lowest topology ID to the highest topology ID (maxID). If there are more than two failures in the network, 2DMRC does not guarantee to recover the traffic and care must be taken to avoid that packets will loop until TTL expires. Since a node can never change the topology ID to a lower topology ID than the current ID, we can safely drop the packets when the topology ID has reached the maxID (line 2 and 4). If the incoming topology ID is not the maxID, lines 6 to 20 try successively backup topologies until the maxID has been reached or a valid next hop has been found. If a valid next hop has been found the packet is forwarded to that next hop, marked with the corresponding topology ID (line 23).

If, however, the maxID has been reached without having a valid next hop, there is a possibility that the failed interface is towards an egress node, i.e. a last hop problem is encountered. In that case, all backup topologies will return the failed interface as next hop. Lines 13 to 18 then introduce deflection to a random neighbor (not the failed) in the backup topology with ID one higher than the incoming topology ID. Deflection adds a portion of randomness in the forwarding process. However, studies have shown that deflection can give a positive effect with respect to link overload [27]. This mechanism will solve all last hop problems if there is one failure in the network and almost all when there are two failures in the network. The only case where 2DMRC does not solve the last hop problem, is when there are two link failures to the egress node, the random node for deflection is the node detecting the second link failure, and the backup topology chosen is the topology where both the nodes detecting these link failures are isolated.

V. EVALUATION

We evaluate 2DMRC with respect to the number of backup topologies needed, recovery success rates with increasing number of faults and backup path lengths. The number of backup topologies is an important measure with respect to the scalability of the method since these extra topologies must be stored in a node. The recovery success rate will measure the amount of data streams that can be recovered and will therefore indicate what scheme that can offer the best recovery guarantees. Backup path lengths will primarily indicate to what extent the packets will experience longer delay. In addition, path lengths can also give an indication on the increased load in the network after a failure, and hence the potential of congestion can be predicted.

The results for 2DMRC will, when relevant, be compared with results for its single failure counterpart DMRC, not-via, FIFR and IP full re-convergence (OSPF).

A. Method

We have developed a tool in Java that implements the different schemes together with an evaluation framework that calculates the properties regarding success rate and backup path lengths. We have used shortest path routing and for simplicity the original weight on all links in the networks has been 1. We have generated synthetic topologies with the Brite topology tool [28] using the Waxman model [29]. The number of nodes has been between 16 and 128, with two and three times the number of links. In addition, we have used the Cost239 [30] and DFN3 networks to confirm that the results from the synthetic networks also apply to real network layouts.

For the recovery success rates and the backup path lengths we have calculated results for 2, 3, 4 or 5 concurrent node failures that are not articulation points, i.e. their removal does not disconnect the rest of the network. As mentioned before, these results are valid also for link failures, except for the last hop case. For this reason, last hop failures have been evaluated separately.

We have calculated results for all possible source and destination pairs, for all combinations of either 2, 3, 4 or 5 failures. One sample is defined as source i, destination j, failure k, ..., failure l, where i, j, k, ... and l are always disjoint.

The following will present our implementation choices for the different recovery schemes.

1) 2DMRC: We have implemented 2DMRC as described in section IV. For the evaluations of recovery success rates and backup path lengths, the number of backup topologies has been determined by the lowest number returned from the algorithm presented in section IV-B. For recovery success rates we count an unsuccessful sample when a packet is dropped.

<table>
<thead>
<tr>
<th>Algorithm 2: Forwarding decisions in nodes that detect a failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ( \text{if}) topologyID = maxID ( \text{then})</td>
</tr>
<tr>
<td>4 ( \text{dropPacket()})</td>
</tr>
<tr>
<td>6 ( \text{else})</td>
</tr>
<tr>
<td>7 ( \text{tmpID} \leftarrow \text{topologyID})</td>
</tr>
<tr>
<td>8 ( \text{while}) tmpID &lt; maxID AND nextHop = failure ( \text{do})</td>
</tr>
<tr>
<td>9 ( \text{tmpID} \leftarrow \text{tmpID} + 1)</td>
</tr>
<tr>
<td>10 nextHop ( \leftarrow \text{findNextHop(destAdr, tmpID)})</td>
</tr>
<tr>
<td>11 ( \text{end})</td>
</tr>
<tr>
<td>13 ( \text{if}) nextHop = failure ( \text{then})</td>
</tr>
<tr>
<td>14 ( \text{tmpID} \leftarrow \text{topologyID} + 1)</td>
</tr>
<tr>
<td>16 nextHop ( \leftarrow \text{randomNeighbor})</td>
</tr>
<tr>
<td>18 forward(nextHop, tmpID)</td>
</tr>
<tr>
<td>20 ( \text{else})</td>
</tr>
<tr>
<td>22 forward(nextHop, tmpID)</td>
</tr>
</tbody>
</table>

3www.dfn.de
due to reaching the maximum topology ID as described in section IV-C.

2) DMRC: DMRC has been implemented along the same principles as 2DMRC, with the exception that it isolates single nodes instead of every pair of nodes. An unsuccessful sample is counted like for 2DMRC.

3) Not-via: Not-via has been implemented as described in [14]. It should be noted that the results for not-via will also count for MPLS local detours [10]. For recovery success rates we count an unsuccessful sample when traffic that is already in a not-via tunnel experiences another failure. If not-via should handle this second failure, one must build not-via tunnels for not-via tunnels. We do not consider that as a scalable solution, due to a numerous number of addresses and SPT calculations.

4) FIFR: We have implemented FIFR as described in [13]. Since FIFR has no built-in mechanisms to prevent loops when experiencing multiple failures, we have chosen to count an unsuccessful sample when it experiences a loop.

5) OSPF: As a reference point we have implemented IP full re-convergence (represented by OSPF). The results on OSPF will show what would be the optimal case when OSPF has performed a full re-convergence from all the failures. However, a full re-convergence from many concurrent simultaneous failures might take a long time due to problems with route flapping and instability. Hence, the network will probably operate for a while without being fully recovered. To represent this case, we also evaluate a case where OSPF has only recovered from one of the n failures (the first detected) in the network (denoted OSPF 1). OSPF with full re-convergence will recover all traffic. For OSPF 1 we count an unsuccessful sample when the traffic experiences a second failure.

B. Scalability - number of backup topologies for 2DMRC

There is no doubt that isolating all pairs of nodes (2DMRC) in the network will increase the number of backup topologies compared to isolating only single nodes (DMRC). The question is, however, will this number still be within acceptable bounds?

Table II presents the results from calculations on various kinds of networks, both real network layouts (Cost239 and DFN) and synthetically generated ones. For each setting (number of nodes - number of links) of the synthetic networks, we have used the algorithm on 100 random networks.

Table II shows the percentage of the topologies that needed from 3 to 12 backup topologies. The right most column shows the average number of backup topologies. The real network layouts (Cost239 and DFN) required 5 and 4 backup topologies, respectively. We also observe that the average number of backup topologies is indeed within acceptable bounds even for large networks. However, we also observe that there exist some cases where the number of backup topologies reach some higher values. The algorithm from section IV-B contains some randomized steps that sometimes result in unfavorable selections, and hence the number of backup topologies can get high. Effacing these unfavorable selections will be a task for future work.

Table III compares the number of backup topologies for the single failure approach DMRC and the double failure approach 2DMRC.

C. Recovery success rates

Figure 3 presents the recovery success rates for the different methods. The x-axis denotes 2, 3, 4 and 5 node failures and the y-axis denotes the percentage of all samples that successfully reached the destination (egress in the network). The figure shows the numbers from a random topology with 32 nodes and 96 links. The red thick line named “No fail” denotes the number of samples that never experienced any failures. The difference between “No fail” and the line representing a scheme then denotes the amount of successfully recovered samples. We see that 2DMRC have a higher success rate than the other methods. The other methods give almost the same success rate, however a little difference has been observed as a repeated tendency. The order, starting with the method with highest success rate, seems to be DMRC, not-via, FIFR and OSPF 1.

We have done the calculations on several topologies with the same characteristics and the variance in results between them is negligible, e.g. 0.007 percentage points for three failures with 2DMRC. This variance is slightly increasing with the number of failures. We also observed the tendency that the schemes with the highest recovery success rates also had the lowest variance.

The same results and differences as presented in figure 3 were also observed for other network sizes and for the real network layouts.

For networks with lower average node degrees, we found that the success rates decreased. This is due to the fact that more links give higher probability to avoid the failures.
As mentioned before, we have separated the tests of last hop (egress) failures from the intermediate node failures. 2DMRC also shows good results for the last hop cases. With two failures in the network, 2DMRC gives a recovery success rate of 99.88 %, compared to 100 % when there is no last hop failure (fig. 3). With three failures in the network, the success rate is 99.84 % compared to 99.96 % in the no last hop case.

D. Backup path lengths - network load

We have chosen to present backup path lengths as an indication on both packet delay and load in the network after failures. In [15], we saw a strong correlation between backup path lengths and the overall load in the network. In addition to backup path lengths, also the recovery success rate will influence the overall load in the network. The higher success rate, the more load will be maintained in the network. A drawback of using backup path lengths and success rates as measures on network load is that we are not able to point out the particular links that will carry the peak load. This means that we are not able to give figures on packet loss due to congestion. However, our experience with such measurements on packet loss are that the results are so dependent on parameters like type of network, traffic matrix, link weight settings etc. that it is as good as impossible to state any general conclusions from them. [31] exemplifies this problem by studying 65 papers from MOBICOM. They found that the papers give different assumptions on parameters like type of networks and traffic load, resulting in diverse results.

As a consequence of these experiences we have chosen to use backup path lengths as a parameter-independent indicator for both packet delay and overall network load. Higher success rates and longer backup paths indicate higher network load.

Figure 4 presents the distribution of backup path lengths for the different recovery schemes. The calculations have been performed on two random Waxman networks with 32 nodes and 96 links. The plot represents samples that have experienced at least one failure, and where the network is stricken with 3 concurrent node failures. We observe that the schemes designed for one failure tolerance (DMRC, FIFR and not-via) give almost the same backup path lengths. 2DMRC gives some longer paths in general while OSPF gives some longer maximum values compared to the schemes for one-fault tolerance. 2DMRC isolates many nodes in one backup topology which reduces the number of alternative links to route on, and hence the paths will be longer. Another aspect is that both 2DMRC and OSPF recover a larger amount of traffic than the other schemes. This improved recovery comes with the cost of longer paths, due to the fact that this extra recovered traffic often experience more than one failure and must be rerouted several times.

We have also calculated the path lengths in networks with fewer and more failures. The tendencies seen in figure 4 are also valid for those cases.

We have also seen that the backup path lengths increase with the size of the networks and decrease with the average node degree. However, the difference between the schemes is the same. This complies with the findings in [15]. We have also observed that increasing the number of backup topologies decreases the lengths of the backup paths.

We have seen how 2DMRC outperforms the other methods with respect to success rates. However, we have also seen how that comes with the cost of longer backup paths and generally more load in the network. This means that there might be a certain probability of congestion on one or more links, and hence packet loss may be the case. This would of course reduce the experienced success rate. However, the amount of congestion will be heavily dependent on the overall load in the network before the failures, the traffic matrix and the load of the nodes or links that fail. Anyhow, the probability of congestion can be decreased by only recover the traffic with highest priority. Congestion avoidance can also be handled by post-failure load balancing as presented in [32].

VI. DISCUSSION, CONCLUSION AND FUTURE WORK

In this paper we have argued that resistance to concurrent failures in IP networks becomes more and more important due to more dependable services and more unstable infrastructures like wireless networks. We have presented 2DMRC that guarantees recovery from any single failure and any two concurrent failures in connectionless pure IP networks. In addition it improves the general recovery success rate.
considerably compared to other methods that solve proactive local recovery in IP networks.

2DMRC requires more backup topologies and hence more state in each router than its single failure counterpart DMRC. Still, we are confident that the increased state requirements are acceptable for the types of networks and nodes that are supposed to implement this method.

If we look only at the methods designed for one fault tolerance, we can give some guidelines with respect to employment. As they give almost the same performance with respect to recovery success rate and backup path lengths in networks with concurrent failures, we can state that resilience against multiple concurrent failures is not the pros or cons that should determine what single failure method to implement in a network. FIFR, however, has a drawback compared to DMRC and not-via with respect to preventing loops after an unsuccessful recovery action. FIFR has no mechanism to decide when to drop a packet, and consequently, packets have a potential to loop until the TTL has expired. FIFR also lack a solution to the last hop problem (sec. II). Other pros and cons for these methods can be found in [33].

Since this paper has presented an extension of DMRC to handle two concurrent failures, one should also ask if the other methods (FIFR and not-via) also could be extended with such a feature? According to our best knowledge, we are not able to foresee how that could be done. We do not believe that the interface-specific forwarding in FIFR would guarantee loop-free routing in such a case. We also have problems foreseeing a scalable solution for not-via.

As further work on multiple failures, we are currently studying DMRC with respect to correlated failures, e.g. shared risk groups. We will also combine 2DMRC with a method studying DMRC with respect to correlated failures, e.g. shared link failures in an IP backbone,” in Proceedings of INFOCOM 2004, Mar. 2004.


REFERENCES


