Non-Weighted Interface Specific Routing for Load-Balanced Fast Local Protection in IP Networks

Steven S. W. Lee¹, Po-Kai Tseng², Alice Chen³, and Cheng-Shong Wu⁴

¹Department of Communications Engineering, National Chung Cheng University, Chiayi, Taiwan, R.O.C.
²Research Center for Information Technology Innovation, Academia Sinica, Taiwan, R.O.C.
³ITRI, Hsinchu, Taiwan, R.O.C.
⁴Department of Electrical Engineering, National Chung Cheng University, Chiayi, Taiwan, R.O.C.

Abstract—As a failure occurs, the affected traffic is quickly rerouted to backup paths for a network performing a fast protection scheme. Such a prompt reaction is aimed to reduce the damages caused by a failure. However, in some cases, the rerouted traffic may cause congestion along the backup paths which would lead to more packet losses than purely discarded affected flows. In this paper, we propose a load-balanced fast local protection scheme called Non-Weighted Interface Specific Routing (NISR) for determining the working and backup routing tables of IP routers. We jointly consider protection switching time, network survivability, and traffic load distribution together in the proposed scheme. In NISR, once a failure occurs, only the nodes adjacent to a failure divert affected traffic to backup paths. This local reaction process guarantees fast protection switching and reduces failure recovery time. Unlike the conventional IP routing, our approach relaxes the shortest path routing in computing working and backup routing tables. Most importantly, each interface in a router has its own routing tables. Combining the interface specific routing with the shortest path relaxation provides greater routing flexibility to enhance network survivability and load balancing. We formulate this as a mixed integer programming problem in which the traffic load on the most congested link is to be minimized. Since this problem is intractable by its NP-hard nature, we further decompose it into several sub-problems which are solved optimally and their solutions combined to provide a solution to the original problem. We perform experiments on some benchmark networks and compare the proposed scheme to several well-known schemes (including LFA, ECMP, OSPF, and NLB) on survivability ratio, link load distribution, and average path length for both normal and failure states. Through numerical results, we delineate that the proposed scheme achieves a sub-optimal solution, which is better for its high survivability and load balancing at the expense of slightly raising the average path hop count.

Index Terms—IP fast reroute, load balance, network optimization.

I. INTRODUCTION

The long packet delay and service interruption are unacceptable for QoS sensitive applications in IP networks. When a network failure occurs, the total recovery time for current link state routing protocols such as OSPF [1] may take several seconds to re-converge [2]. To mitigate the impact of failures, many IP fast local recovery schemes have been proposed in the literature. The main design principle is to pre-compute backup routes to bypass failure components and guarantee no transient loops during failure recovery. We classify those schemes into three categories: loop-free alternate (LFA) based schemes [3-5], tunneling-based schemes [6-7], and backup routing table (BRT) based schemes [8-10].

In LFA-based schemes, routers adjacent to a failure link forward the affected packets to neighbor nodes to bypass the failed link or node. Routers suppress flooding topology change information during the recovery duration to avoid frequent route flipping. The selections of new next hop neighbors have to ensure satisfying loop-free properties. ECMP [3], LFA [4], and U-Turns [5] belong to these kinds of schemes. However, these approaches may have lower survivability if the routers have no alternate loop-free neighbors to repair the failure.

In tunnel-based schemes, a tunnel is setup between the node that detects a failure and an intermediate node using the IP-in-IP technique. Once the intermediate router receives these encapsulated packets, it decapsulates and forwards them according to their original destinations via normal routes. This intermediate router must be selected carefully to avoid routing loops. Tunnels [6] and Not-via addresses [7] adopt a design concept like this. To encapsulate and decapsulate packets leads to an extra burden on routers in the network.

In BRT-based schemes, each router pre-computes backup routing tables (or backup configurations) before any failure occurs. In [8], an IP network is preplanned with multiple configurations and that configuration corresponds to a topology. Routing of a packet is determined jointly by its destination address and configuration mark. Thus packets can be rerouted to their backup paths by only changing the configuration marks on them. In [9-10], a protection procedure is triggered when a packet comes into a router through a port that is usually not used for that packet. However, load balancing was not considered, and the rerouted traffic would cause severe congestion on some heavily loaded links.

To the best of our knowledge, [11] is the first one to consider load balancing in IP fast failure recovery. Once a failure happens, the routers adjacent to a non-working device distribute the affected traffic over multiple backup paths through solving a Linear Programming (LP) problem incrementally. The rerouted traffic is distinguished from normal traffic by inserting a special mark on it. The diverted traffic will be allocated on the multiple backup paths and the allocation will be further refined with the subsequent LP
iterations. But, the subsequent LP iterations would lead to a succession of route migrations. The instable routes may cause other failures or even more packet losses during the recovery processes. To mark the affected packets also leads to an extra burden on routers.

In IP networks, a routing path is usually the shortest path according to the link metrics. However, one can relax the shortest path routing constraint if the routing tables can be constructed without creating loops. Such a concept has been applied in IP fast protection schemes [8,9,10]. We refer to those schemes belonging to non-weighted IP routing. In those schemes, only non-weighted IP routing is applied in construction backup paths. Working paths still follow the shortest path routing.

In [12], we propose a new approach called NLB, where we relax the shortest path routing constraints in computing working paths. Our scheme is the first one to relax shortest path constraints on both working and backup routings. It makes our system a purely non-weighted one. The scheme proposed in this paper applies the same concept. The major difference between the approach proposed in this paper and the one in [12] is that we introduce a non-weighted interface specific routing (NISR) concept in computing working and backup routing tables to enhance network survivability and load balancing.

In a conventional scheme, a router uses only one working routing table. A packet will be forwarded to a specific output interface no matter which input interface it comes from. In NISR, each input interface has its own routing table. A packet coming from a different input will be treated differently for routing. NISR provides more routing flexibility and enables us to design algorithms to achieve high survivability and load balancing over IP networks.

We use Fig. 1 to demonstrate the concept of NISR. Figure 1(a) is the example before allocating flow for od-pairs 2-D and 3-D. Each link in the figure is labeled by a pair of numbers: (link metric, traffic loading). After introducing 10 Mbps traffic demands for od-pairs 2-D and 3-D, the results are shown in Fig. 1(b), Fig. 1(c), and Fig. 1(d) for OSPF, NLB, and NISR respectively. Only traffic loading is labeled on each link in Fig. 1(c) and Fig. 1(d) since NLB and NISR are non-weighted systems. In Fig. 1(b) shortest paths 2-1-4-D and 3-1-4-D are used for packet delivering. Since the shortest path constraint is relaxed in NLB, in Fig. 1(c), the less congested path 1-5-D is used to replace the shortest path 1-4-D such that the maximum link loading can be reduced from 30 (link (1,4) of Fig. 1(b)) to 25 (link (1,5) of Fig. 1(c)). For NISR shown in Fig. 1(d), interfaces i1 and i2 take different routes even if the traffic is to the same destination D. In this example, by taking both 1-4-D and 1-5-D, the maximum link loading can be further reduced to 20 Mbps.

Another important strength of NISR is that it can enhance network survivability. For example, in NLB shown in Fig. 2(a), when link (3,D) fails, node 3 cannot recover the affected traffic. If node 3 reroutes the traffic carried on link (3,D) to node 1, it generates a loop among nodes 1, 2 and 3. This demonstrates the drawback of using only one routing table in a router. However, in NISR shown in Fig. 2(b), node 3 can successfully recover the traffic by sending the affected traffic to node 1. Since node 1 treats packets coming from different input interfaces in different ways, node 1 can forward the traffic from interface i5 to interface i6 to node 4 for failure recovery. By doing so, the affected traffic can be rerouted to destination D successfully.

In this paper, we formulate the NISR problem as a mixed integer programming problem in which the traffic load on the most congested link is to be minimized. Since this problem is NP-hard, we further decompose it into several sub-problems. Each sub-problem is solved optimally and the results for these are combined to provide a solution to the original problem.
The remainder of this paper is organized in four sections. We first give the problem formulation in Section II. In Section III, we present the problem decomposition technique and provide solution procedures to the problem. In Section IV, we show the experimental results and make performance comparisons to other well-known schemes. Finally, concluding remarks are made in Section V.

II. PROBLEM FORMULATION

In order to achieve interface specific routing, the given network topology is first transformed into an extended graph. For each node with $m$ input and $n$ output, it is replaced by a subgraph with $m$ artificial vertices connecting to input links, and another $n$ artificial vertices connecting to $n$ output links. Thus each artificial vertex is corresponding to an input interface or output interface. Fig. 3 depicts an example, where $m=n=2$. For simplicity, only node 1 is shown in its extension form. Besides, there are additional $m\cdot n$ artificial edges connecting the $m\cdot n$ artificial vertices (i.e., vertices i1, i2, i3, and i4 in Fig. 3). These artificial edges describe possible routing combinations inside a node. In order to facilitate local traffic add and drop, another two artificial vertices, vertices $a$ and $d$, are included. Vertex $a$ and vertex $d$ are the local add node and local drop node in this example.

We formulate the NISR problem as a mixed integer linear programming problem based on the transformed graph. Given network topology $G(N,L)$ and the demand volume for all origin-destination (od) pairs, the problem computes the working and backup routing tables of each node for normal state and failure states. The objective is to minimize the weighted sum of the maximum load on the most congested link.

The notations used in the formulation are shown as follows. Given input constant values:

- $N$: set of nodes in the network;
- $L$: set of links in the network;
- $V_{in}^{n}$: set of artificial vertices representing input interfaces in the node $n$;
- $W$: set of od-pairs;
- $P_{w}^{n}$: set of candidate paths for od-pair $w$;
- $S$: set of network states; Here we denote $s_{0}$ as the normal (non-failure) state and $s_{i}$ as the $i$-th failure state;
- $E_{in}^{n}$: set of artificial edges in the node $n$;
- $E$: set of artificial edges in the transformed graph;
- $E_{n}$: set of artificial edges connecting to vertex $v$ in node $n$;
- $E_{c}$: set of artificial edges which cannot be used in the state $s$;
- $N_{s}$: set of nodes which are affected by failure state $s$. We call node $i$ an affected node in state $s$ if node $i$ cannot deliver packets via its normal next hop in state $s$;
- $\bar{N}_{s}$: set of nodes which are unaffected in state $s$. $\bar{N}_{s} = N \setminus N_{s}$;
- $m_{s}$: a given weight for state $s$;
- $t_{i}^{d}$: demand volume for od-pair $w$;
- $W_{k}$: set of od-pairs whose destination is $k$. $W = \{W_{1}, W_{2}, ..., W_{N}, |N|\}$ is the amount of nodes in the network;
- $\delta_{p}$: = 1, if path $p$ uses artificial edge $c$; = 0, otherwise;
- $\Phi_{pl}$: = 1, if path $p$ uses link $l$; = 0, otherwise;

To help readers understand the notations, an illustration of sets $E_{s}$, $E_{s}^{n}$, $V_{in}^{n}$, $N_{s}$, and $E_{c}$ is shown in Fig. 4.

Decision variables:

- $x_{es}^{d}$: = 1, if artificial edge $e$ is used to transmit packets with destination $d$ in state $s$; = 0, otherwise. Routing tables are derived from this decision variable;
- $y_{ps}$: = 1, if path $p$ is used to transport packets at state $s$; = 0, otherwise;
- $u_{s}$: the maximum link load in state $s$;
- $z_{ls}$: the accumulated total used capacity before solving sub-problem $IP_{sub}\ l$;

The problem is formulated as follows.

Problem (IP):

$$\begin{align*}
\text{min} & \sum_{s=0}^{m} m_{s} u_{s} \\
\text{subject to:} & \\
\sum_{p \in P_{w}} y_{ps} & = 1 \quad \forall w \in W, s \in S \quad (1) \\
\sum_{e \in E_{c}} x_{es}^{d} & \leq 1 \quad \forall e \in E_{s}, s \in S, d \in N \quad (2) \\
x_{es}^{d} & = 0 \quad \forall e \in E_{s}, s \in S, d \in N \quad (3) \\
\sum_{p \in P_{w}} \delta_{pe} x_{es}^{d} & \leq x_{es}^{d} \quad \forall e \in E_{s}, s \in S, w \in W_{s}, d \in N \quad (4) \\
x_{es}^{d} & = x_{els}^{d} \quad \forall e \in E_{s}, n \in \bar{N}_{s}, s \in S, d \in N \quad (5)
\end{align*}$$
\[ y_{ps} = 0 \text{ or } 1 \quad \forall p \in P_w, w \in W, s \in S \quad (6) \]
\[ x^d_{es} = 0 \text{ or } 1 \quad \forall e \in E, s \in S, d \in N \quad (7) \]
\[ \sum_{w \in W} \sum_{p \in P_w} y_{ps} \phi_{ps} t_{wu} \leq u_k \quad \forall l \in L, s \in S \quad (8) \]

The objective function is to minimize the weighted sum of the most loaded link’s flow amount in each state. The weight of each state can be designated depending on the operator’s engineering purposes. If state \( k \) is considered to be the most important state, \( m_k \) is assigned a very large value to emphasize its importance.

Constraint (1) is the routing constraint to route one path for each od-pair under each state. Constraint (2) requires that only one artificial edge can be used for every input interface of each node. Constraint (3) limits \( x^d_{es} = 0 \) if artificial edge \( e \) belongs to \( E_s \), which means edge \( e \) cannot be used in state \( s \). Constraint (4) states that an artificial edge \( e \) can be used to carry flows for od-pair \( w \) in state \( s \) only if \( x^d_{es} = 1 \). Constraint (5) requires that a node still uses the same artificial edge as it used in the normal state to forward traffic in the failure state if the node is an unaffected node for that failure state. This constraint makes our scheme to be local recovery purpose. Only the router adjacent to a failure needs to react to a local failure. Constraints (6-7) require \( y_{ps} \) and \( x^d_{es} \) are binary variables, respectively. The last constraint, Constraint (8), is to find the most loaded link for each state \( s \).

Please note that by defining the failure set \( S \) properly the model can be applied to various kinds of failure protection scenarios, such as, link, node, or even SRLG failures by using the same model.

III. PROBLEM DECOMPOSITION AND GREEDY ALGORITHM

Even though this problem can be entirely formulated as the model shown in Sec. II, the large number of integer variables and constraints make it hard to obtain an optimal solution directly. Due to the intractable nature of this problem, it is unlikely to obtain an exact solution for realistic networks in reasonable computation time. For this reason, we present a low complexity Greedy Algorithm to find a sub-optimal solution.

**Greedy Algorithm:**

1. Sorting demand volume for each destination node \( k \) in descending order and putting their node ID in priority queue \( Q \).
2. Set \( i = 1; \quad z^i_L = 0 \quad \forall l \in L, s \in S \).
3. While \( i \leq |N| \)
4. \( k = Q[i]; \quad /* \text{DeQueue the highest order call from } Q*/ \)
5. Solve **Sub-problem** \( \text{IP}_{\text{sub} k} \).
6. Calculate \( z^i_L = z^i_L + \sum_{w \in W} \sum_{p \in P_w} y_{ps} \phi_{ps} t_{wu} \).
7. \( i = i + 1; \)
8. End

![Fig. 5. The Greedy Algorithm](image)

We decompose the above IP programming problem into \(|N|\) sub-problems, one for each destination node. We arrange the \(|N|\) sub-problems according to their demand volume (i.e., \( \sum_{w \in W} t_{wu} \); \( k \) is the destination of the sub-problem) in descending order, and call the first sub-problem as \( \text{IP}_{\text{sub} 1} \), the second one as \( \text{IP}_{\text{sub} 2} \),..., and the last one as \( \text{IP}_{\text{sub} |N|} \). The proposed Greedy Algorithm then solves all of the \(|N|\) sub-problems sequentially according to their order. The \( i \)-th sub-problem is as follows.

**Sub-problem** \( \text{IP}_{\text{sub} i} \) (assume the destination is node \( k \))

\[ \min \sum_{w \in W} m_{ik} u_s \]

subject to:

\[ \sum_{p \in P_w} y_{ps} = 1 \quad \forall w \in W_k, s \in S \quad (9) \]
\[ \sum_{e \in E_s} x^d_{es} \leq 1 \quad \forall e \in E_s, s \in S \quad (10) \]
\[ x^d_{es} = 0 \quad \forall e \in E_s, s \in S, w \in W_k \quad (11) \]
\[ \sum_{p \in P_w} \delta_{pe} \leq x^d_{es} \quad \forall e \in E_s, s \in S, w \in W_k \quad (12) \]
\[ x^k_{es} = x^{k}_{es} \quad \forall e \in E_s, s \in S, n \in N, n \neq k \quad (13) \]
\[ y_{ps} = 0 \text{ or } 1 \quad \forall p \in P_w, w \in W_k, s \in S \quad (14) \]
\[ x^k_{es} = 0 \text{ or } 1 \quad \forall e \in E_s, s \in S \quad (15) \]
\[ \sum_{w \in W_k} \sum_{p \in P_w} y_{ps} \phi_{ps} t_{wu} + z^i_L \leq u_k \quad \forall l \in L, s \in S \quad (16) \]

Figure 5 depicts the complete Greedy Algorithm. For the \( i \)-th sub-problem (assume its corresponding destination node is \( k \)), Constraints (9-15) are corresponding to Constraint (1-7) in Problem IP. In Constraint (16), \( z^i_L \) is the accumulated total...
used capacity before IP_{sub} i is computed. It is calculated on line 6 of the Greedy Algorithm. The algorithm terminates when all sub-problems are solved. We then obtain the overall working and backup routing tables for each node for every destination for all states.

In Fig. 6, we give an example and show the computational results. Graph extension of node 1 is depicted in Fig. 6(a) where node a is the artificial vertex for adding local traffic. The local drop port d is omitted to simplify the graph. We first make graph extensions for each node (we only show extensions of node 1 in Fig. 6(a)). The traffic matrix is shown in Fig. 6(b), entry (i,j) denotes the demand volume originated from node i to destination node j.

We applied the proposed Greedy Algorithm to obtain the solution. The traffic load of a normal state for each link is labeled on Fig. 6(a) and the obtained working routing table for node 1 is shown in Fig. 6(c). Figs. 6(d-f) display backup routing tables of node 1 for failure on link (1,2), (1,3) and (1,5) respectively. In these tables, the first column corresponds to the input interfaces and the first row represents the destination nodes. Each entry presents the determined output interface. For example, based on the working routing table (Fig. 6(c)), packets with destination 2 coming from interface i4 will be forwarded to interface i3. In the case of link (1,2) failing, as Fig. 6(d) indicates, interface i4 will result in using interface i2 to divert packets with destination 2.

IV. EXPERIMENTAL RESULTS

We perform the proposed Greedy Algorithm of Sec. III on COST239, NSF, and GTE networks shown in Fig. 7. In each network, two adjacent nodes are connected by two opposite directional links. The average node degrees of NSF, GTE, and COST239 are 3, 4.166, and 4.727 respectively. The weight of normal state u_{s_0} is 10 and each failure state’s weight u_{s_i} is 1 in our experiments. We assume the traffic demand between any two nodes is 10 Mbps. We consider a single link failure scenario that is the most often encountered case [13]. We compare some well-known existing approaches with proposed Greedy Algorithm to observe three performance metrics: network survivability, load distribution, and average path length.

A. Network Survivability

First, we examine the most important performance index—network survivability. We perform ECMP [3], LFA [4], LFA-SA, NLB [12], and NISR (i.e., the proposed Greedy Algorithm) and made performance comparisons among them. The computation of survivability is defined as the total number of successfully rerouted routes divided by the total number of affected routes. A route here is regarded as an IP packet stream for od-pair. For LFA, we randomly set a link weight to be an integer in the range [1,100]. For the LFA-SA scheme, we apply a simulated annealing algorithm to tune link weight until the highest survivability is reached. For ECMP, each link weight is set to be 1 to increase the opportunity of multiple equal cost paths between router pairs.

As shown in Fig. 8, NISR and NLB reach the highest 100% survivability. ECMP has the lowest survivability. LFA-SA strongly depends on the network topology. We discover that a network with a lower average degree would reduce the chance for ECMP and LFA to find loop-free alternate paths for traffic rerouting in failure states.

B. Link Load in Normal and Failure States

In the second set of experiments, we compare NISR, NLB, and OSPF to examine the link load distribution. We observe the performance by three kinds of statistics in the normal state and failure states: the carried flow on the most congested link, the average link load, and the variance of link load. The results are shown in Fig. 9, where “-n” and “-f” in the x-axis denotes the normal state and the failure states respectively.

Examination of Fig. 9(a) for performance on most congested link, we observe that NISR outperforms NLB in all cases. In particular, for the NSF network, NISR can reduce the load on the most loaded link 10% and 10.34% more than NLB for network in normal and failure state. The main reason is that NISR uses the interface specific routing to enhance the flexibility of routing. Such that NISR can achieve low link loading even though the mean degree of the NSF network is low.

The experimental results for the average link load are shown in Fig. 9(b). We discover that NISR has the highest values. That is because, in order to achieve our objective function (i.e., lower maximum link load), NISR needs to reroute affected traffic flows to longer paths. The more detailed comparison of path length is shown in next subsection. Longer rerouted path would imply larger average link load.

The variance of the link load is shown in Fig. 9(c). NISR receives the lowest variance in COST239 and GTE networks. Only in NSF, that is the smallest degree network, OSPF has better performance.

Fig. 8. Network survivability in the benchmark networks

Fig. 7. Benchmark networks for performance evaluation
Constraints are considered. Working and backup paths even though the load balance indicates that the proposed NISR scheme would not incur long routing scheme to offer working and backup paths. The results average path length than NLB because it uses a more flexible all states of the three benchmark networks. NISR has a longer the length gap between OSPF and NISR is within one hop in shortest path for each od-pair, it reaches the smallest value. In the final set of experiments, we perform OSPF, NLB, and NISR to evaluate the average path hop count. The results among survivability, load balancing, and path lengths can be achieved in the proposed NISR scheme.

**V. CONCLUSION**

In this paper, we proposed a Non-weighted Interface Specific Routing (NISR) scheme for load-balanced fast IP local protection. This scheme relaxes the conventional shortest path based routing and applies the interface specific routing to achieve load balancing in the normal and failure states. We have modelled the problem as a mixed integer programming (MIP) problem. The model can be applied to various kinds of failure protection scenarios including link, node, or even SRLG failures. Since the problem is an NP-hard problem, we decompose it into several sub-problems and propose a greedy based heuristic algorithm to obtain the sub-optimal solution. Experimental results indicate that the proposed scheme has higher survivability than ECMP, LFA, and LFA-SA. The results also indicates that the proposed scheme reaches more uniform load utilization on the network bandwidth usage. Compared to shortest path routing, NISR incurs only a small penalty of path length increasing (less than one hop count in our experiments). We conclude that the performance trade-off among survivability, load balancing, and path lengths can be balanced in the proposed NISR scheme.

**ACKNOWLEDGEMENT**

Part of this work is supported by the National Science Council, Taiwan, under grant number NSC 99-2221-E-194-022.

**REFERENCES**