Scalable Backup Configurations Creation for IP Fast Reroute

Shohei Kamamura, Takashi Miyamura, Cristel Pelsser, Ichiro Inoue, and Kohei Shiomoto
NTT Network Service Systems Laboratories, NTT Corporation
9-11, Midori-Cho 3-Chome Musashino-Shi, Tokyo 180-8585 Japan.
Email: {kamamura.shohei, miyamura.takashi, pelsser.cristel, inoue.ichiro, shiomoto.kohei}@lab.ntt.co.jp

***Abstract*—IP Fast Reroute techniques have been proposed to achieve fast failure recovery in just a few milliseconds. The basic idea of IP Fast Reroute is to reduce recovery time after failure by precomputing backup routes. A multiple routing configurations (MRC) algorithm has been proposed for obtaining IP Fast Reroute. MRC prepares backup configurations, which are used for finding a detour route after failures. However, this current algorithm requires too many backup configurations to recover from failures. We propose a new backup configuration computation algorithm for reducing configurations as much as possible. The basic idea is to construct a spanning tree excluding failure links in each backup configuration. We show that the effectiveness of our algorithm is especially high in large-scale power-law networks.**

I. Introduction

Link-state routing protocols, such as Open Shortest Path First (OSPF) [1], have been widely used for intra-domain routing resolution, but a few seconds are required to find alternate routes after failure occurrence [5]. This recovery time is too long to achieve robustness for the increasing number of multimedia applications. The failure recovery processes of link-state routing protocols mainly consist of 1) failure detection, 2) link-state flooding, and 3) backup routes computation. These processes have recently been improved. Bidirectional Forwarding Detection (BFD) can be used for fast failure detection [2]. Fast link-state flooding approach [3] and implementation of incremental SPF algorithm, which reduces the backup routes computation complexity, is proposed [4]. However, concerning 2 and 3, this is still not enough in large-scale networks because above approaches, whose processes are done after failures, depend on the network size. To achieve robustness in IP networks, fast failure recovery techniques which do not depend on the network size are required.

IP Fast Reroute techniques have been extensively studied for fast failure recovery in just a few milliseconds [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. The basic idea of IP Fast Reroute is to reduce recovery time after failure occurrence by precomputing the backup routes. The multiple routing configurations (MRC) method has been proposed for IP Fast Rerouting [5], [6], [7], [8]. The MRC method prepares backup configurations, which are precomputed and used for finding a detour route after failures. In a backup configuration, some links are assigned a higher metric value. Such links are called isolated links. The isolated links can be regarded as protected links. They are not used to forward the traffic when a resource fails. An arbitrary link is an isolated link in at least one backup configuration. Therefore, we can achieve fast recovery against any single failure using backup configurations.

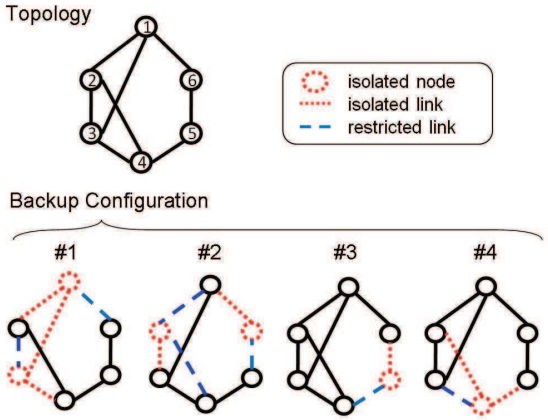

Fig. 1. Overview of backup configuration

In this paper, we define the new backup configuration-computation problem for minimizing the number of configurations. The number of the routing tables kept on a router is proportional to the number of backup configurations; therefore, minimization is needed to ensure scalability. We propose a new backup configuration-creation algorithm for reducing configurations as much as possible. Our proposed method is used to achieve robustness and scalability. The key point of our algorithm is that each of the backup configurations is made so that the topology, excluding the isolated links, becomes a spanning tree. Our evaluation results show that the effectiveness of our algorithm is especially high in large-scale power-law networks.

The rest of the paper is organized as follows. In Section II, we provide the characteristics of backup configurations. We describe IP Fast Rerouting using backup configurations, and then define the problem. In Section III, the issues of current algorithms and our new backup configurations-creation algorithm is presented. Our evaluation results are shown in Section IV, and related work is presented in Section V. Finally, we conclude our discussion in Section VI.

978-1-4244-5048-0/09/$26.00©2009 IEEE

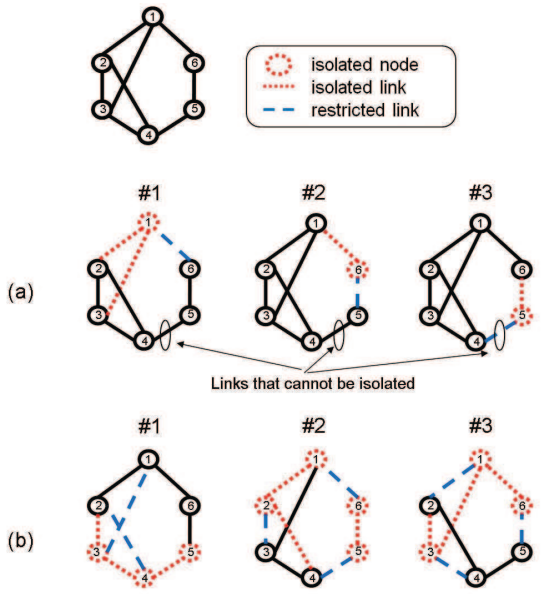

Fig. 2. (a) Example where conventional algorithm with parameter N=3 cannot provide backup configurations that protect all components. (b) Example where our proposed algorithm creates backup configurations for N=3.

II. Multiple Routing Configurations

In this section, we describe the characteristics of backup configurations used with the MRC method. We introduce IP Fast Rerouting using backup configurations. We then state our problem.

A. Overview of Backup Configurations

The characteristics of backup configurations defined by Kvalbein et al. [5] are as follows (Fig. 1). A backup configuration consists of normal links, isolated links, restricted links, and normal and isolated nodes. An isolated link is a link that may fail, and a restricted link is a link that can be used only as the last hop. The metric of an isolated link is set to the maximum value provided by the link-state routing protocol, and the metric of a restricted link is set to a large value, though it is not the maximum value [5]. An isolated node is a node that is only connected to isolated and restricted links. Therefore, backup configurations should satisfy the following characteristics.

1) Each backup configuration is a connected graph that does not contain isolated links.
2) The union of isolated nodes and links of all backup configurations correspond to the original topology.
3) Among the links that are connected to an isolated node, at least one is a restricted link, and the others are isolated links. The opposing node of a restricted link must not be an isolated node in the same backup configuration.

If backup configurations satisfy the above conditions, an arbitrary link is an isolated link in at least one backup configuration. In addition, if a link failure is caused by an opposing node failure, the links that are connected with the failed node are not used except when the failed node is the destination node. This is because the failed node is the isolated node. This node is only connected to isolated and restricted links. Therefore, the detour route determined by the backup configuration avoids the isolated node. Thus, for each single failure there is a backup configuration that avoids the failed components.

B. IP Fast Reroute using Backup Configurations

IP Fast Reroute can be achieved by using backup configurations [5]. Each backup configuration is precomputed and installed in all routers. Each router computes the shortest path and then creates the routing entries (relationship between destination IP address and next hop node) based on the original configuration, and for each backup configuration. If the router detects a link failure, it searches the backup configuration that isolates the link corresponding to the failed link. Next, the identifier of the selected backup configuration is marked in the type of service (ToS) field of the IP header. After marking, the failure-detecting router forwards the packets to the next hop node according to the routing entry of the selected backup configuration. Other routers can forward the IP packets according to the same backup configuration by referring to the ToS field.

C. Problem Statement

The problem we want to solve is to minimize the number of backup configurations while all nodes and links are isolated. The total number of backup configurations is an important factor in terms of scalability. Requiring too many backup configurations consumes a lot of router resources. This is because the number of the routing tables kept on a router is proportional to the number of backup configurations. If the number of routing entries exceeds the upper limit of the router, fast recovery cannot take place for certain arbitrary single failures because all backup configurations cannot be installed in the router. Therefore, minimizing the number of backup configurations to reduce routing entries is needed to ensure scalability.

III. Scalable Backup Configurations Creation (SBC^2)

A. The Issues of Existing Algorithms

Kvalbein et al. proposed a backup configuration computation algorithm that ensures failure recovery under a point of a single node or link failure [5]. First, the number of backup configurations N is determined, then each configuration is created as follows. The original set of metrics is copied from backup configuration #1 to #N, then the target node to be isolated in the target configuration (#1) is determined. More precisely, if node 1 is isolated in backup configuration #i, one link connected with node 1 is restricted, and the other links connected with it are isolated. Then, backup configuration #i+1 is selected as the target configuration. The restricted link and its opposing node, in backup configuration #i, are isolated in priority in backup configuration #i+1. If a node is isolated

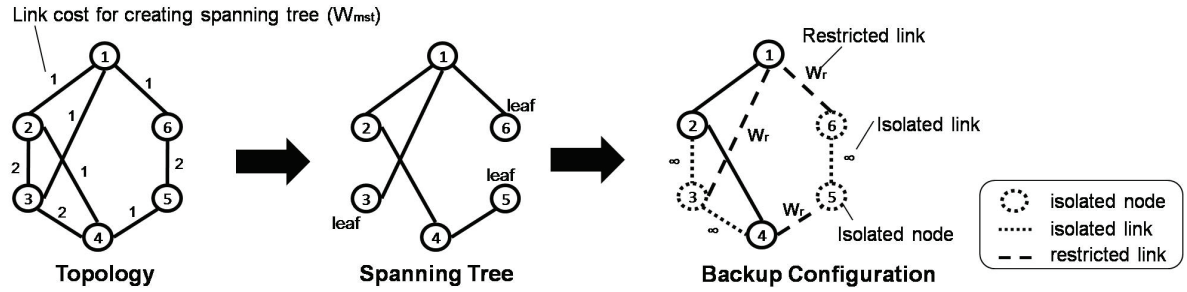

Fig. 3. Key idea of our algorithm. Spanning tree is created from topology. Backup configuration is created from spanning tree. The links, which are removed to create the spanning tree, become isolated links. The links, which are connected to the leaf node, become restricted links. The leaf nodes become isolated nodes.

in backup configuration #N, the next target configuration is backup configuration #1. This process is continued until every node is isolated in at least one backup configuration. This algorithm guarantees that all links are isolated when all nodes are isolated.

Even if this algorithm isolates all the components in at least one configuration, the following issues still remain.

1) This algorithm cannot handle multiple failures.
2) The length and thus the required link capacity of the detour route increases compared to the OSPF detour route. The detour route is not along the optimal (shortest) path.
3) This algorithm requires an extension of the standard protocol because the identifier of the backup configuration needs to be advertised to other routers.
4) The routers need to maintain a large number of backup configurations, and additional resources for these configurations are also needed.

There were a few approaches to solve issues 1- 3. Hansen et al. proposed a backup configuration computation algorithm considering multiple failures [6]. Kvalbein et al. reduced the length of the detour paths by optimizing the metrics in backup configurations [7]. Psenak et al. and Przygienda et al. are standardizing protocol extensions for IP Fast Reroute techniques using backup configurations [15], [16].

However, concerning issue 4, Kvalbein et al. do not take into account the minimization of the number of backup configurations [5]. Figure 2 illustrates this problem. In Fig. 2 (a), backup configurations are created according to the algorithm proposed by Kvalbein et al. [5] with parameter N=3. After node 5 and link 5-6 are isolated in configuration #3, link 4-5 has to be isolated. However, link 4-5 cannot be isolated in configuration #1 because the resulting topology, excluding the already isolated links and link 4-5, is not connected. Then, configurations #2 and #3 are selected; however, link 4-5 cannot be isolated in these configurations for the same reason. Therefore, more than four backup configurations are needed. (The case where parameter N=4 is illustrated in Fig. 1). In Fig. 2 (b), on the other hand, if isolated links and nodes are optimally placed, the number of backup configurations N can be reduced to 3. That is, the algorithm proposed by Kvalbein et al. is not optimal in terms of the number of backup configurations. Cicic et al. proposed an algorithm to reduce the number of backup configurations [8]. However, their algorithm does not isolate all links, so another information table for fast recovery is needed.

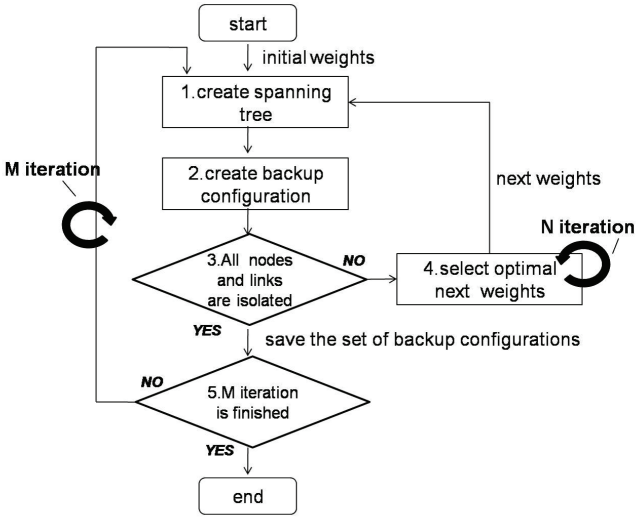

Fig. 4. Overview of our algorithm.

B. Overview of SBC^2

The algorithm proposed by Kvalbein et al. [5] has a simple computation process and ensures that all nodes and links are isolated by placing them to each backup configuration cyclically. However, this mechanism generates too many backup configurations. The above cyclical strategy renders the efficient placement of isolated nodes and links difficult. Our strategy, therefore, is to generate backup configurations by the unit. Each backup configuration maximizes the number of isolated nodes and links. Multiple backup configurations, which have this characteristic, are generated. The best combination, which minimizes the number of backup configurations, is selected.

The key point of our algorithm is that each backup configuration is made so that the topology, excluding the isolated links, becomes a spanning tree (Fig. 3). This process

TABLE I
NOTATION

p	Identifier of backup configurations
n	index number
G	graph
$V(G)$	set of all vertices in G
$E(G)$	set of all edges in G
$W_{org}(e)$	original metric of link e in graph G
$W(p,e)$	metric of link e in backup configuration p
$W^n_{mst}(p,e)$	weight of link e for creating spanning tree in p
W_r	metric of restricted links (high value)
$MST^n(p)$	minimum spanning tree of G based on $W^n_{mst}(p,e)$
RAND()	returns random value from W_{min} to W_{max}
COPY(X)	returns a copy of X
DEL_ ELEMENT(x, X)	delete element x from set X
NEW_ ISOLATED(T)	number of new isolated node(link) by tree T

TABLE II
SBC^2 ALGORITHM

```
1  p ⇐ 0, optid ⇐ 0 /* optid is ID of optimal set of weights */
2  for all e ∈ E(G) do
3     W(p,e) ⇐ W_org(e)
4     W^optid_mst(p,e) ⇐ RAND()
5  G'=COPY(G)
6  while (1) {
7     T ⇐ MST^optid(p)
8     for all v ∈ V(T) do
9        if (v is leafnode) then /* search the isolated node */
10          DEL_ ELEMENT(v, V(G'))
11    for all e ∈ E(G) ∩ E(T) (complement) do /* search the isolated link */
12       W(p,e) ⇐ ∞
13       DEL_ ELEMENT(e, E(G'))
14    for all e ∈ E(T) do /* search the restricted link */
15       if (e is connected with leaf node) then
16          W(p,e) ⇐ W_r
17    p++
18    for all e ∈ E(G) do /* initialize the next metrics */
19       W(p,e) ⇐ W_org(e)
20    if (G' == ∅) then
21       end
22    if (G' satisfy backup configuration condition) then
23       create W(p,e) based on G' (all member of G' are isolated)
24       end
25    else /* create N sets of weights, then select best set */
26       for (n = 0, newoptid = 0; n < N; n++) do
27          for all e ∈ E(G) do
28             W^n_mst(p,e) ⇐ W^optid_mst(p-1,e)
29             if (W(p-1,e) ≠ ∞) then
30                W^n_mst(p,e) += RAND()
31          for all e connected with arbitrary node v ∈ V(G') do
32             weight of one link is set to low value,
               and weight of the other links is set to a high value
33          T^n ⇐ MST^n(p)
34          if (n ≥ 1 and
            (NEW_ ISOLATED(T^n) ≥ NEW_ ISOLATED(T^newoptid))
35             newoptid ⇐ n
36       optid ⇐ newoptid
37 }
```

maximizes the number of isolated nodes and links on one configuration. We do not evaluate all the combinations of spanning tree in order to minimize the number of backup configurations, which is not efficient. Our algorithm is efficient because the search space is limited by changing the non-isolated nodes and links into the isolated ones in priority in the next spanning tree.

Figure 4 shows an overview of our algorithm. The processes in steps 1 and 2 are the key points shown in Fig. 3. The process in step 3 checks the end condition. If all nodes and links are isolated, a set of backup configurations is produced. The process in step 4 determines the set of weights to create the next optimal spanning tree. This process needs N iterations. For the performance gain, the M iteration can be applied to our algorithm in step 5. In the next section, we describe our algorithm in detail.

C. Algorithm

Our algorithm automatically creates backup configurations from an undirected weighted graph, G. The details of our algorithm are shown in Table 2, using the notation shown in Table 1. First, parameters are initialized (line 1-5), and a set of link weights are determined randomly (line 4). It should be noted that the set of weights to create the spanning tree is different from the set of metrics configured on the backup configurations. The following process is continued until the set of backup configurations that satisfy the characteristics shown in Section II-A are created (line 6-37).

The spanning tree is created based on a set of given weights, and then one backup configuration is created from this spanning tree (line 7-16). We create a minimum spanning tree by the Kruskal algorithm [18]. A minimum spanning tree is a spanning tree whose summation of the link weights is minimal. Therefore, the links with a higher weight tend to be removed at the time of creating the spanning tree. The leaf nodes in the spanning tree become isolated nodes (line 8-10). The links, which are removed to create the spanning tree, become the isolated links (line 11-13). The links, which are connected to the leaf nodes in the spanning tree, become the restricted links (line14-16).

The end of our algorithm is evaluated (line 20-24). Our algorithm maintains a graph, G', which consists of non-isolated links and nodes. If G' is empty, our algorithm finishes (line 21), or if a backup configuration whose isolated nodes and links are all elements in G' can be created, our algorithm creates such a backup configuration and finishes (line 24).

If the end is not satisfactory, our algorithm sets the weights of the spanning tree to produce the next backup configuration (line 26-36), and N sets of weights are created (line 26). With each new set of weights, we try to isolate the non-isolated nodes and links in priority; therefore, we apply the characteristics of the Kruskal algorithm to our algorithm. First

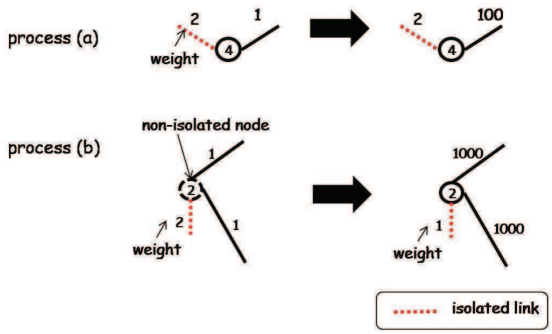

Fig. 5. Example of our algorithm. (a) shows the processes in line 29-30 of Table 2. (b) shows the processes in line 31-32 of Table 2.

the weights of the non-isolated links in the previous backup configuration are increased by a random value (line 30). This process isolates non-isolated links. Next, the weight of one link connected with a non-isolated node is randomly selected and is set to a lower value, and the weights of other links connected with it are set to a higher value (line 32). This process ensures that the non-isolated node becomes a leaf node in the spanning tree. Figure 5 shows the examples of these processes. The best set, which has the most new isolated nodes and links, is selected (line 34-36), and then the process starts again from the backup configuration creation (line 7).

In our algorithm, the form of the first spanning tree is determined by a random set of weights (line 4). This set of weights affects the generation of the following spanning trees that isolate the non-isolated components (links or nodes). However, sometimes this first set is not appropriate for generating the following set. Therefore, we iterate our algorithm with different initial sets of weights. Then, the best combination of backup configurations, whose number is minimal, is selected. Thus, the iteration avoids a fixed first spanning tree.

D. Termination

Our algorithm continues until all nodes and links are isolated in at least one backup configuration. As previously mentioned, the links, which are removed to create the spanning tree, become isolated links. Moreover, the leaf nodes in the spanning tree become isolated nodes. The process in line 30 ensures that the target links are isolated by increasing their weight value. Next the process in line 32 ensures that a target node is isolated by manipulating the link weights connected with it. Because of lines 30 and 32, at each step of a while loop, at least one link or node is isolated. Thus, the maximum number of while loops is $|V(G)| + |E(G)|$.

IV. Performance Evaluation

In this section we demonstrate the effectiveness of our algorithm through extensive simulation.

A. Aims and Simulation Conditions

We describe the aims of our evaluation and simulation conditions. First we show that our algorithm reduces the number of backup configurations compared to the the current algorithm [5] in various types of topologies. In addition, we discuss the available resources and computations of our algorithm.

In our evaluation, we use two different topology models in terms of node-degree. The node-degree expresses the number of links connected to a node. One model is the Waxman model whose node-degree is relatively uniform on each node. The other model is the Barabasi-Albert (BA) model whose node degree follows a power-law. In power-law network, most nodes have a small number of links while a small portion of nodes have a large number of links. We created 20 instances of topologies for each of these two models using the BRITE topology generation tool [19]. Although router-level topologies are with higher node degree [20], our evaluation conditions assume the POP (Point Of Presence) level networks. Therefore, the average node degree d_a is set to two (nodes have four links on average.), and calculated as follows:

$$d_a = |E|/|V| \tag{1}$$

$|V|$ is the total number of nodes and $|E|$ is the total number of edges. The number of nodes in these topologies ranges from 20 to 200 because maximum number of nodes in the intra-domain is assumed to be about 200 [21]. An evaluation index is the maximum number of backup configurations. The parameters of our algorithm was set such that there are 20 iterations (M=20) and N=5 (Table 2 line 26).

B. Effectiveness of SBC^2

Figure 6 shows the results for the Waxman model, and Fig. 7 shows the results for the BA model. These results suggest that the number of backup configurations of our algorithm never exceed the number of configurations obtained with the current algorithm [5]. There was a reduction of up to 56% for the 200 nodes with the BA model while there was a reduction of up to 28% for the 200 nodes with the Waxman model. The results mean that with the BA model, our algorithm is more effective. For example, 56% reduction of configurations means the 56% reduction of routing entries in each router.

Moreover, the amount of reduction increases as the number of nodes increase. This is due to the distribution of the node-degree in the topology. As described in Section III. A, the number of configurations that are generated by the current algorithm [5] increases when the topology has a lot of nodes with low node-degree. The current algorithm [5] ensures that the links and nodes, which could not be isolated in configuration #i, are isolated in configuration #i+1 in priority. That is, isolated links are placed at the neighboring position in backup configuration #i and #i+1 (Fig.2 (a)). Consequently, the topology excluding the isolated links in these backup configurations becomes the sparse topology in local range, and then a new isolated link cannot be placed in such a range. Therefore, another backup configuration is required. If the number of nodes increases in the BA model, the ratio of nodes with low node-degree also increases. Therefore, the number of configurations that are generated by the current algorithm

increases. In contrast, our algorithm, which generates the backup configuration by the unit of itself, absorbs the effect of the distribution of node-degree.

Next we discuss the available resources in backup configurations. Figure 8 shows the average ratio R_a of available links to all links in backup configurations. Although R_a decreases about 50%, the dependency to the network size is low. We explain this reason by the formulation. The set of links E_r, which are removed from the connected graph to create the spanning tree, is described as follows [22]:

$$|E_r| = |E| - |V| + 1 \tag{2}$$

Then, R_a is given by the following equation:

$$\begin{aligned} R_a &= \frac{|E| - |E_r|}{|E|} \\ &= \frac{|E| - (|E| - |V| + 1)}{|E|} \\ &= \frac{|V| - 1}{|E|} \\ &= \frac{1}{d_a} - \frac{1}{|E|} \end{aligned} \tag{3}$$

$$R_a \to \frac{1}{d_a}, \qquad (N \to \infty, E \to \infty) \tag{4}$$

By (3), we found that the value of d_a is dominant. By (4), if E becomes infinity, E_r converges on the reciprocal of d_a. Therefore, R_a mainly depends on the average node-degree in large networks. It is important that the backup routes, which are calculated by backup configuration, are used only traffic reaching failure. The traffic that does not use failure resources is forwarded according to the original configuration. Moreover, these routes are only used until backup routes by link-state protocols are calculated. Therefore, the impact of available resources reduction is low.

Finally we evaluated the computation time of our algorithm. The time for 20 iterations of our algorithm was about 133 seconds for 200 nodes using a C++ program on Xeon 2.9 GHz PC with 128 GBytes of memory. Although the computation time increases in proportion to the number of iteration times, the number of backup configurations converges in 20 iterations under all conditions. Our algorithm is relatively complex compared to the more simple current algorithm [5], but it is reasonable to suggest that this tendency of computation time, which should be computed offline, is acceptable.

V. Related Work

Some IP Fast Reroute techniques, which do not require backup configurations, have been proposed [9], [10], [11], [12], [13], [14]. In this section, we summarize these techniques.

Basic IP Fast Reroute techniques, which do not require extensions to existing link-state routing protocols to set the proper link metrics, have been proposed [9], [10]. Equal Cost Multi Paths (ECMP), paths whose length is based on the link

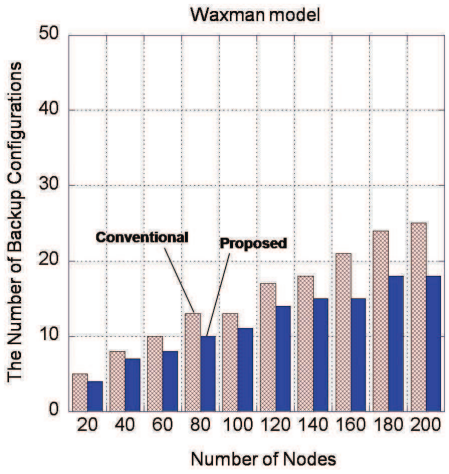

Fig. 6. Number of backup configurations in Waxman model .The average node degree is 2.

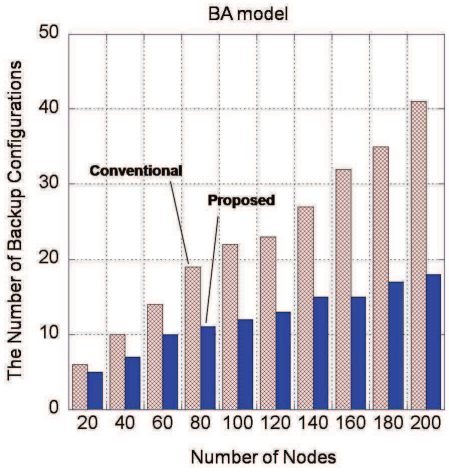

Fig. 7. Number of backup configurations in Barabasi-Albert(BA) model .The average node degree is 2.

metrics is the same, are used to forward the IP packets [9]. IP packets are distributed on each path according to a hash value based on their IP address or port number. Even if a link failure occurs on one route, the computation of the detour routes is not needed because other living routes can be used as detour routes. A loop free alternate (LFA) approach was proposed [10]. The node, which detects a failure, forwards the IP packets to the node called LFA, which is not the original next hop. The link metrics are set so that the route from LFA to the destination node does not include detecting the failure node. Therefore, routing loops never occur. The drawback of these techniques is that they cannot handle a single failure in an arbitrary form of the topology because the routes based on link metrics depend on the topology [17].

Nelakuditi et al. [11] proposed an IP Fast Reroute technique called failure insensitive routing (FIR). The key point of FIR is that a failure point is estimated from the relationship between the destination IP address and the incoming interface.

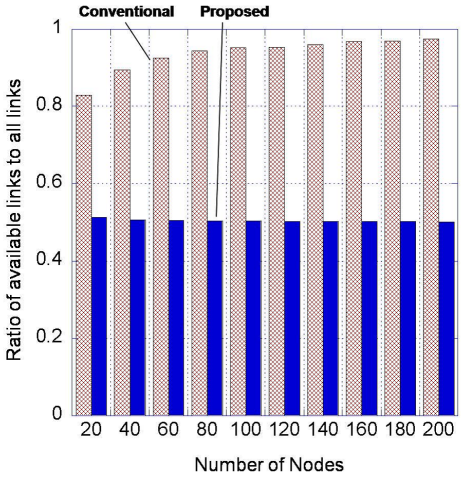

Fig. 8. The average ratio R_a of available links to all links in BA model.

If packets are received on an irregular interface, which is connected with the next hop node on the shortest path to the destination, the router interprets that a failure has occurred on the shortest path. The advantage of this estimating approach is that there is no need to advertise the failure point. However, FIR cannot handle the arbitrary multiple failures. FIR is extended to handle the node failure [12], and to work on the inter-domain environment [13].

A not-via address approach was proposed [14]. In this approach, IP packets are encapsulated, and a header is added to the packet. This header contains the not-via address. The encapsulated packets detour the failure points because the route for not-via address is precomputed to avoid the failure points. First, a failure detecting node encapsulates the IP packets to give the not-via address. All nodes forward the packets according to the not-via address. If these packets arrive at nodes located downstream of the failure points, the header with the not-via address is removed and then the IP packets are forwarded according to the original destination address. The detour route can be calculated flexibly with respect to each not-via address, but routers need to maintain the next hop according to the not-via addresses.

VI. Concluding Remarks

We argued that minimization of backup configurations to reduce the routing entries is needed to ensure scalability. We presented a new backup configurations computation algorithm that reduces the number of backup configurations. Our algorithm has a high feasibility on the existing IP Fast Reroute framework. We demonstrated its effectiveness through an extensive simulation study. The results showed that our algorithm reduced the number of backup configurations independent of the form of network topologies. For large power-law networks, our algorithm reduced the number of configurations by about 56% compared to the current algorithm. The effectiveness of our proposed algorithm increases as the number of nodes increases. Therefore, our algorithm is much more effective on large scale networks than the current algorithm.

Our spanning tree based algorithm has feasibility to isolate arbitrary links by link weights manipulation. Therefore, as further works, we will consider correlated failures recovery like SRLG (Share Risk Link Group) failures, and optimization of backup routes considering network load information as the weights to create the spanning tree.

References

[1] IETF RFC2328:"OSPF Version 2," April 1998.
[2] D. Katz and D. Ward. Bidirectional Forwarding Detection. Internet draft, draft-ietf-bfd-base-09.txt, work in progress, Feb 2009.
[3] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving subsecond IGP convergence in large IP networks," ACM SIGCOMM Computer Communication Review, vol. 35, no. 2, pp. 35 - 44, July 2005.
[4] Daniele Frigioni, Alberto Marchetti-Spaccamela, Umberto Nanni,"Incremental algorithms for the single-source shortest path problem ," Proceedings of the 14th Conference on Foundations of Software Technology and Theoretical Computer Science, Vol.880, pp. 113-124, 1994.
[5] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP Network Recovery using Multiple Routing Configurations," in Proceedings of INFOCOM, Apr.2006.
[6] A. F. Hansen, O. Lysne, T. Cicic, S. Gjessing, "Fast Proactive recovery from Concurrent Failures," In Proceedings IEEE International Conference on Communications, ICC 2007, June 2007.
[7] A. Kvalbein, T. Cicic and S. Gjessing, "Post-Failure Routing Performance with Multiple Routing Configurations," INFOCOM, May 2007.
[8] T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartmann, R. Martin, and M. Menth, "Relaxed Multiple Routing Configurations for IP Fast Reroute," In IEEE/IFIP Network Operations and Management Symposium 2008.
[9] IETF RFC2991:"Multipath Issues," Nov 2000.
[10] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-free alternates, " IETF internet Draft, 2005, draft-ietf-rtgwg-ipfrr-spec-base-04.txt
[11] S. Nelakuditi, et al., "Failure Insensitive Routing for Ensuring Service Availability," IW QoS', June 2003.
[12] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in Proceedings of IEEE Global Internet, vol. 4, Mar. 2005.
[13] J. Wang, and S. Nelakuditi, "IP Fast Reroute with Failure Inferencing," In Proceedings of INM'07, The 5-9's Workshop at ACM SIGCOMM, August 2007.
[14] draft-bryant-shand-IPFRR-notviaaddresses-02.txt.
[15] P. Psenak, S. Mirtorabi, A.Roy, L. Nguen, and P. Pillay-Esnault, "MT-OSPF: Multi topology (MT) routing in OSPF," IETF, RFC4915, June 2007.
[16] T. Przygienda, N. Shen, and N. Sheth, "M-ISIS: Multi topology (MT) routing in IS-IS," Internet Draft (work in progress), Oct. 2005, draft-ietf-isis-wg-multi-topology-11.txt.
[17] M. Gjoka, V. Ram, Y. Xiaowei, "Evaluation of IP Fast Reroute Proposals," COMSWARE Jan. 2007.
[18] Joseph. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," In Proceedings of the American Mathematical Society, Vol. 7, No. 1 (Feb, 1956), pp. 48-50
[19] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in Proceedings of IEEE MASCOTS, Aug. 2001, pp. 346-353.
[20] L.Li, D. Alderson, W. Willinger and J. Doyle, "A First Principles Approach to Understanding the Internet's Router-level Topology," ACM sigcomm 2004.
[21] Moy, J. T., "OSPF: Anatomy of an Internet Routing Protocol.," Addison Wesley 1998.
[22] C. Berge, The Theory of Graphs. New York: Dover Publications, 2001.