Journal of Information Processing Vol.22 No.2 97–105 (Apr. 2014)

[DOI: 10.2197/ipsjjip.22.97]

Regular Paper

Adaptive Load Balancing Based on IP Fast Reroute to Avoid Congestion Hot-spots

Masaki Hara[1] Takuya Yoshihiro[2,a)]

Received: April 29, 2013, Accepted: October 9, 2013

Abstract: Several load balancing techniques for IP routing scheme have appeared in the literature. However, they require optimization process to compute optimal paths to meet traffic demand so that it requires a mechanism to measure traffic demand and to share them among all routes in order to follow dynamics of traffic. It naturally results in communication overhead and losing sensitivity to follow traffic dynamics. In this paper, we investigate a load balancing mechanism from another approach, i.e., based on IP fast reroute mechanisms. The main idea is simply to forward packets into detour paths supplied by IP fast reroute mechanisms only when packets meet congestion. This strategy enables us to use vacant resources adaptively as soon as they are required to avoid and dissolve the congestion. Through traffic simulation we show that IP fast reroute based load balancing mechanisms improve the capacity of networks.

Keywords: IPFRR, load balancing, traffic engineering, congestion control

1. Introduction

Recent growth of the Internet and the rapid increase of user traffic require more bandwidth and communication quality for infrastructure networks. To make the most of existing network resources, several traffic engineering (TE) techniques have been proposed so far. Currently TE techniques for MPLS scheme have become important [1] and several techniques are working in practice mainly in backbone networks. However, since MPLS routers are expensive and complicated, IP routing is still given significant attention, and many TE tecniques in IP networks are continuously studied even now.

As an early-stage TE technique in IP networks, Forts et al. [2] presented a method to optimize link metrics to achieve good load-balancing performance. They formulated the problem to optimize link metrics by means of linear programming, which computes the link metrics that minimize the maximum link usage from the traffic demand among every pair of nodes. As a result, they showed that IP based TE potentially achieves a comparable level of load-balancing performance compared to MPLS based TE techniques.

However, in IP based shortest-path schemes, traffic tends to be concentrated on particular links or nodes because the shortest-paths tend to use the common low-cost links. From this point of view, Dasgupta et al. [3] presented a result of performance comparison between the metric-optimized IP shortest path scheme and the MPLS based traffic engineering scheme. Their simulation demonstrated that, under the traffic variation coming from link failure, IP based TE often brings severe congestion hotspots caused by concentration of shortest paths, while MPLS based TE leads to much less congestion. Their result showed one aspect of the vulnerability in IP based TE techniques, and offers a point to improve.

As a more efficient load balancing approach than metric optimization that possibly improves the vulnerability above, several multi-path load-balancing schemes have been proposed. Mishra et al. presented an OSPF-based TE technique S-OSPF (Smart-OSPF) [4], in which source nodes distribute traffic to their neighbor nodes to try load balancing among the shortest paths from neighbor nodes. In S-OSPF, each node selects a set of neighbor nodes to distribute traffic among them without creating loops, and determines the ratio of traffic to distribute to these neighbors using a linear-programming based optimization process. Antić et al. presented TPR (Two Phase Routing) [5], [27] that once forwards packets to some intermediate nodes using IP tunnels and then forwards them to their destinations using normal shortest paths. They are capable of more efficient load balancing than metric-optimization based traffic engineering. They both, however, require the optimization process that computes the paths for traffic distribution from a traffic demand matrix. In practice, the optimization process requires mechanisms to measure the traffic demand, to share the measured information among all nodes, and to re-compute the new distribution paths. This process would results in losing sensitivity for dynamic transition of traffic.

In this paper, as another approach of load balancing in IP routing schemes, we propose a load balancing mechanism that is based on the IP fast reroute technique. IP fast reroute is the technique to prevent packet loss in case of link/node or component failure using pre-computed alternative paths [9], [12], [15]. IP fast reroute techniques achieve immediate recovery against failure (usually in 50 msec), but they require considerable overhead

1 Graduate School of Systems Engineering, Wakayama University, Wakayama 640–8510, Japan
2 Faculty of Systems Engineering, Wakayama University, Wakayama 640–8510, Japan
a) tac@sys.wakayama-u.ac.jp

such as additional routing table entries. Unfortunately, this overhead for pre-computed backup paths is dedicated only for the case of failure. The load balancing mechanism that we propose in this paper utilizes these unused paths even under normal state where no failure is present to reduce congestion hot spots of networks.

The approach of load balancing using IP fast reroute has several good characteristics as follows:

1. Packet forwarding is based on local decision so that no additional communication overhead (especially to measure and advertise information of traffic demand) is required for load balancing.
2. No optimization process is required to follow dynamics of traffic so that sensitive response is achieved for traffic transition. This property is effective especially in case of failure protected by IP fast reroute mechanisms.
3. It is possible to work seamlessly with IP fast reroute mechanisms which provide failure protection function within the same framework.
4. Since only the packets faced to be dropped are detoured, communication paths are basically the same as the shortest-path routing scheme. Namely, the current operational experience over traffic estimation is available.

Note that there are several studies on load balancing that incorporate IP fast reroute schemes such as Ref. [7]. However, they are based on the IP fast reroute schemes that use alternative hops for failure protection, such as LFA (Loop-Free Alternate) [8]. Thus, not only they cannot provide backup paths for every single link/node failulre, but also the load-balancing performance would be limited because every backup path is 1-hop long. To the best of our knowledge, this paper is the first proposal to provide load-balancing fuctionality over full-coverage IP fast reroute schemes.

The remainder of the paper is organized as follows: In Section 2 we describe the base IP fast reroute mechanism called SBR (Single Backup-table Rerouting) and present our load balancing technique over SBR. In Section 3, we will have a quick look at the property of SBR detour paths that is used for load balancing. In Section 4 we give the results of traffic simulation to show the performance of our mechanisms. In Section 5 we describe several IP fast rerouting schemes proposed so far and discuss how we can apply the proposed load balancing technique to them. Finally, we conclude the work in Section 6.

2. Load Balancing Mechanisms Using IP Fast Reroute Scheme SBR

2.1 SBR Mechanisms

In this section we explain the base IP fast reroute mechanism SBR (Single Backup-table Rerouting) [19], [20] and describe how packets are forwarded into backup paths in it. SBR is an IP fast reroute scheme that recovers every single link failure with low overhead, i.e., two 1-bit flags on the packet header and one additional routing table. Note that there are several major IP fast reroute mechanisms studied so far, but we use SBR as the base scheme of our load balancing method because the mechanism is convenient to design load balancing functions; SBR itself uses several 1-bit flags on packet header. Naturally, our load

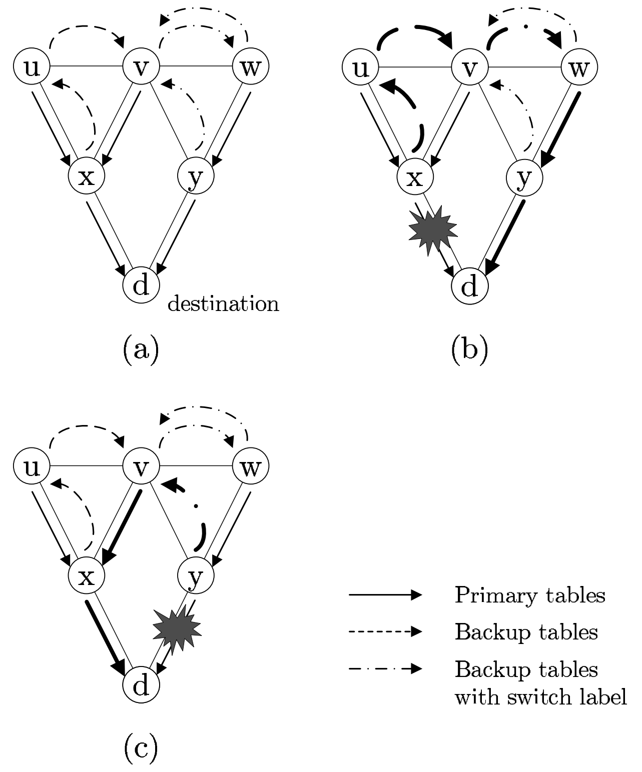

Fig. 1 The SBR mechanism.

balancing method can be implemented with other major IP fast reroute schemes such as NotVia [12] and FIFR [15]. How to apply the load balancing method to these IP fast reroute schemes is described in Section 5.

SBR is first described in a theoretical manner [19] and later its practical protocol that extends OSPF framework is presented in [19]. Note that the algorithm to compute a backup routing table is similar to Ref. [22], which presents a time-efficient algorithm to compute the secondary routing table for FIFR. Specifically, the main difference as a link protection framework is that SBR considers link metrics to compute backup paths, whereas [22] computes backup paths based on hop-count.

The mechanism of SBR is illustrated in **Fig. 1**. Since we describe SBR as an extension to the link-state routing scheme, it is assumed that every node has its *primary routing table* that represents the shortest-path tree for each destination. In SBR, every router has a secondary routing table, called a *backup table*, to protect any single link failure. To guarantee at every node a backup path that bypasses the next-hop link, backup table entries are classified into two types, “backup” and “switch,” and we distinguish the type of a backup table entry by the extra 1-bit label attached to each backup table entry. For an example to show the SBR mechanism, see Fig. 1 (a) that shows the situation that every node (except *d*) in the network has its primary and backup next-hops for a destination *d*. Primary next-hops are indicated by solid arrows and backup next-hops by two sorts of broken arrows. Here, you see that backup next-hops are classified into two types “backup” or “switch,” where “switch” type plays a special role in forwarding packets.

Packets are forwarded in combination with these two routing tables. See Fig. 1 (b) for an example. Once a link (*x*, *d*) fails,

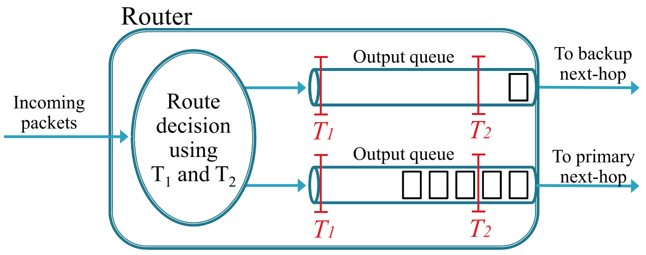

Fig. 2 The proposed load balancing mechanisms.

the node x detects it and immediately forwards packets to u instead of d, using its backup next-hop. Then the packets travel along backup next-hops for a while, and after going through that of switch type, i.e., (v,w) in this figure, they return to the primary route to reach the destination. Namely, packets sent from u destined to d travel along the path $u \to x \to u \to v \to w \to y \to d$.

Such forwarding is realized by a 1-bit *b-flag* on the packet header; keep b-flag 0 when a packet uses primary tables, change to 1 when the packet is firstly forwarded using backup tables (at x in Fig. 1 (b)), then return to 0 when the packet passes a backup next-hop of type "switch" (at v in Fig. 1 (b)). Namely, this flag represents which table to be used to forward the packet. SBR is able to protect every single link failure. Figure 1 (c) is the example when link (y, d) failed. The packets sent from w destined to d travel along the path $w \to y \to v \to x \to d$.

Additionally in SBR, we introduce another 1-bit flag called *r-flag* on packet header, which prevents loops in case of multiple link failure. The r-flag is set when a packet first returned to the shortest path (i.e., when the b-flag is first reset), and the packet with r-flag is never forwarded into backup next-hop. Namely, r-flag limits the number of detours in order to prevent loops caused by repeated rerouting.

2.2 Load Balancing Mechanisms Based on SBR

Based on SBR, we propose a load balancing technique to utilize vacant resources adaptively to prevent packet loss. Our strategy is simply to rescue the packets which are to be dropped as they have met congestion, by means of forwarding them into the backup path instead of the primary path. Our method detects congestion using output queue length. See **Fig. 2**. There is a router with three links, and we assume that packets come from the left side and are forwarded to a link in the right side. When a packet without b-flag arrives at the router, the router makes route decision for it, i.e., decides the output interface according to the queue length of its primary next-hop. Specifically, when the sum of queue length and the packet size is longer than threshold T_1 (T_1 is usually set as 100% of the queue length), the packet is enqueued into the queue of the backup interface with the packet's b-flag set, and otherwise the packet is enqueued into the queue of the primary interface.

This simple rule, however, causes the detour chain problem. The detour chain problem is the problem where detouring packets that bypass a congestion cause another congestion to generate detouring packets again, and the repeated detouring continues like a chain. This problem must be avoided since the chain generates lots of detour packets which heavily consume network resources, and in the worst case they may form a loop, resulting in significant amplification of congestion.

To prevent the problem from occurring, we introduce another threshold T_2 on the output queue to suppress detouring packets before they cause a new congestion. When a packet with b-flag or r-flag arrives at the router, the router checks the queue length of the next-hop interface (i.e., the queue length of the backup next-hop is checked for b-flag packets, and that of the primary next-hop is checked for r-flag packets) to decide the route to forward it. If the queue length is longer than T_2, the router drops the detouring packet, and otherwise, the router forwards it into the corresponding next-hop according to the forwarding rule of SBR.

Note that we detour packets only if the packets are to be dropped under the conventional single shortest-path scheme. Our strategy is to use vacant resources to rescue packets as long as they do not cause any bad effect on (non-detouring) shortest-path traffic. By protecting the shortest-path traffic from the harmful influence of detouring traffic, we try to have the shortest-path traffic behave the same as in the non-load-balanced scheme, even in our load balancing environment. Also note that to reduce the bad effect (such as delay) on shortest-path traffic, it is important that T_2 is sufficiently small, but at the same time, T_2 should hold the minimum necessary value to afford dynamics of the traffic in the network.

2.3 Enabling Multiple Detour

We further consider detouring packets more than once to improve the performance in larger networks. In a larger network, a packet meets congestion more than once with higher probability so that multiple detouring is essential from the viewpoint of scalability. To enable this, we introduce *r-count* instead of r-flag on the packet header which indicates the number of detours that the packet experienced. By introducing the maximum allowed number of detours for each packet, we allow several detours while preventing packet loops. If we allow detouring three times, a 2-bit field on packet header is required for r-count.

The change on router behavior is simple: if a packet without b-flag is received and the primary next-hop is congested (i.e., the queue length is more than T_1), the router forwards the packet into backup next-hop and increments the r-count of the packet only if its r-count is lower than the maximum allowed value, and the packet is dropped if the r-count is larger than or equal to the maximum allowed value. Note that only the packets which are using the primary route (i.e., the packet without b-flag) can be detoured, because SBR does not provide further "backup paths" for the packets that are currently using backup paths.

Table 1 summarizes the whole behavior of routers in our proposal. This table shows the process of the forwarding decision and flag manipulation at each router when a packet arrives. The process is different according to the condition on the state of flags (b, r) on the packet header and the queue length of the next-hop interface. Here, let (b, r) be the b-flag and the r-counter of the packet, let *max* be the number of maximum allowed detouring, and let $q^{(p)}$ and $q^{(b)}$ be the queue length of the output queue for the primary and backup next-hop, respectively.

Table 1 Formal forwarding process at each router in the proposed method.

Conditions		Packet Manipulation	
Flags and Counters (b,r)	Conditions on Queue	Forwarding Decision	Flag Manipulation
(0,0)	$q^{(p)} < T_1$	Forwarding into primary next-hop	
	$q^{(p)} \geq T_1$ and $q^{(b)} < T_2$	Forwarding into backup next-hop	Increment r-counter, and set b-flag if the backup table entry is not the switch type.
	$q^{(p)} \geq T_1$ and $q^{(b)} \geq T_2$	Drop	
(0,n)	$q^{(p)} < T_2$	Forwarding into primary next-hop	
	$q^{(p)} \geq T_2$ and $q^{(b)} < T_2$	Forwarding into backup next-hop	Increment r-counter, and set b-flag if the backup table entry is not the switch type.
$1 \leq n <$ max	$q^{(p)} \geq T_2$ and $q^{(b)} \geq T_2$	Drop	
(0,max)	$q^{(p)} < T_2$	Forwarding into primary next-hop	
	$q^{(p)} \geq T_2$	Drop	
(1,n)	$q^{(b)} < T_2$	Forwarding into backup next-hop	Reset b-flag if Backup Table entry is the switch type.
$1 \leq n \leq$ max	$q^{(b)} \geq T_2$	drop	

3. Optimality Analysis of SBR Paths

The characteristic of backup paths is important because it directly effects on the load-balancing performance. Thus, in advance of traffic simulation, we just have a quick look at the optimality of the backup paths of SBR, and see that the backup paths of SBR is near optimal.

To achieve better load balancing performance in the presence of congestion hot spots, it is desirable that the backup paths are as short as possible. Thus, we define the *best backup path* from node *s* to *d* as the shortest path in the network where we omit the link expected to fail, (i.e., the link connected to primary nexthop from *s*), and we analyze how the length of the backup paths are different from the best backup paths.

The topologies used in our analysis are generated by BRITE [23] with the Waxman [24] and the Barabasi-Albert [25] models, which model the Internet topology. To see the influence of scalability, the number of nodes is varied between 10 and 400. Also to see the effect of network density (i.e., link-to-node ratio), we also vary its range between 1.2 and 2.0. In every case, we tried two patterns of link-cost distribution: one is randomly given costs within certain range (10–100), and the other is constant value. We executed the algorithm of SBR to create backup tables [21] for these networks and obtained the results shown in **Figs. 3** and **4**. Here, all results are the average of 10 times repetition.

In Fig. 3, we show the ratio of the best backup paths among all computed backup paths (i.e., the ratio of the pair of nodes for which the computed backup path is the best backup path), with the variation of network scale. It shows that the performance is very good where all values are over 99%, meaning that 99% of the backup paths computed are the best backup paths. Especially, the case of constant link costs marks a fine score compared with random cost distribution. Figure 4 is the result where we vary the density (i.e., link-to-node ratio) of networks. We create sparse networks by generating the network whose link-to-node ratio is 2 using BRITE, then randomly removed edges one by one as long as the network is 2-link-connected. Here, 2-link-connected means that there are at least two link-disjoint paths between every pair of nodes. The result is also good; all values are over 97%.

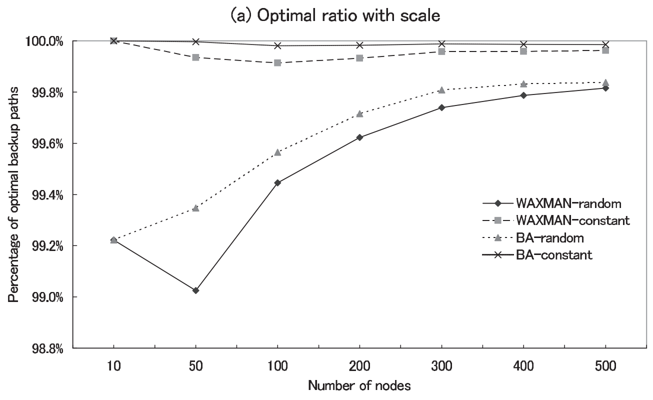

Fig. 3 Optimal ratio of backup paths (with number of nodes).

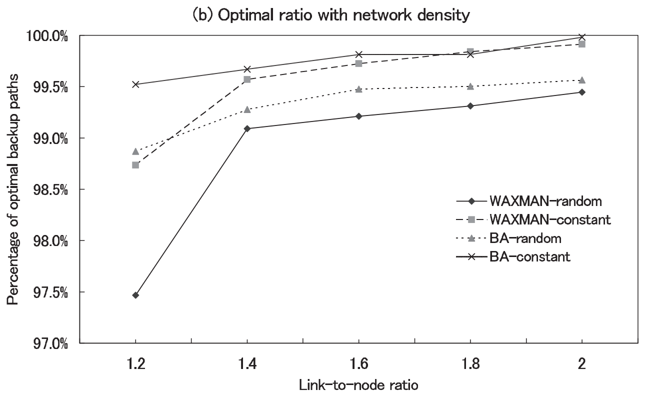

Fig. 4 Optimal ratio of backup paths (with link density).

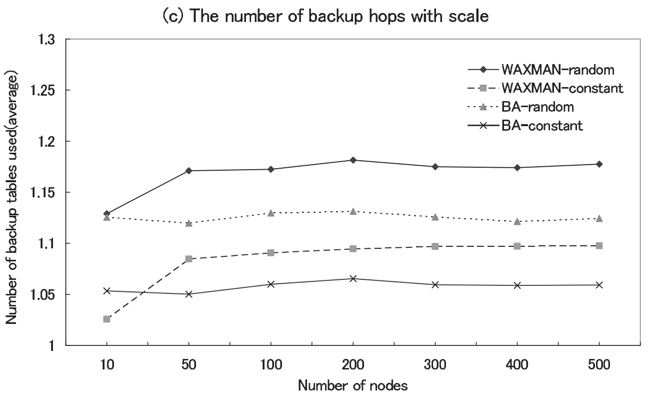

Fig. 5 Number of backup hops (with number of nodes).

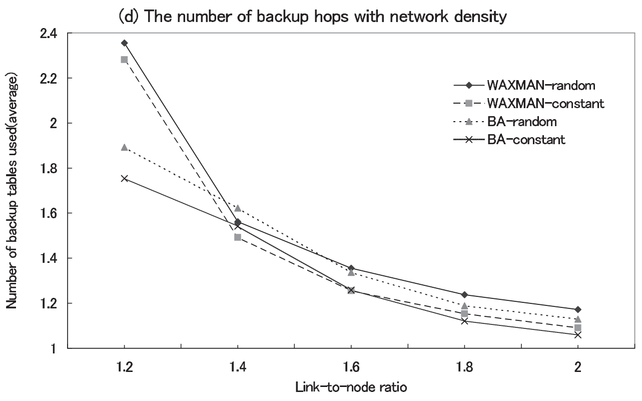

Fig. 6 Number of backup hops (with link density).

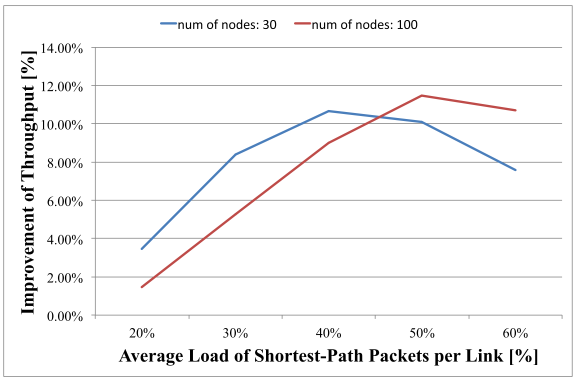

Fig. 7 Improvement of throughput with variation of traffic load.

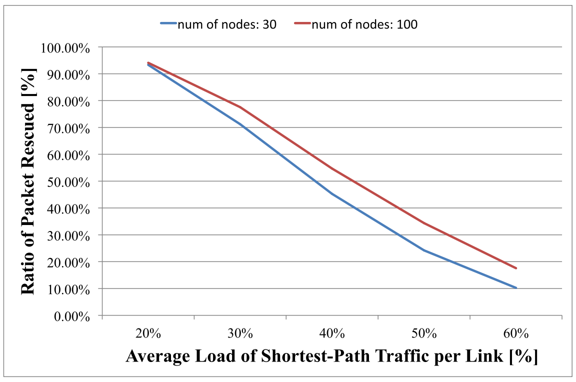

Fig. 8 Ratio of packets rescued by detouring.

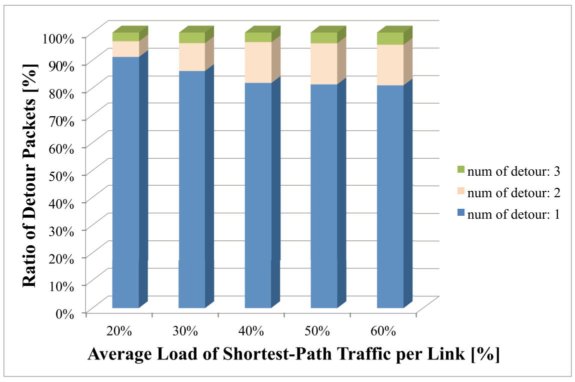

Fig. 9 The number of detours to reach destination (case of 30 nodes).

The performance improves slightly as the density increases.

In **Fig. 5**, we show the average number of hops in backup paths. From the result, the average backup hops are low and in every case the average is under 1.2 hops. Although this test is done with dense networks in which the link-to-node ratio is 2, it is shown that the length of the backup paths is mostly small, regardless of network scale. **Figure 6** shows the same value under variation of link-to-node ratio. It is natural that the number of backup hops increases as networks become sparse.

4. Traffic Simulation

4.1 Simulation Setup

We evaluate the proposed load balancing method through traffic simulation using ns-2 [26] simulator with random topologies that models the Internet. To evaluate the performance of the case where traffic is partially concentrated on particular links, our scenario is designed to locate several narrower links to cause congestion. Our simulation settings are shown as follows: We generate the network topology of Waxman model [24] with 30 and 100 nodes, respectively, using topology generator BRITE [23]. We select 10% of the links randomly as "bottleneck" links which bandwidth is 50 Mbps while normal link bandwidth is 100 Mbps. In this network, we generate several 10 Mbps CBR flows with 1 Kbyte packets in which the source and the destination nodes are also selected randomly. We increase the CBR flows as time passes to raise the load of the network. Output queue length is as long as 50 packets. Threshold T_1 is 100% and T_2 is 10% of the output queue length. The number of maximum allowed detouring is set to 3. Simulation is done with the randomly generated 10 topologies, and for each of them, we select two distinct sets of bottleneck links and traffic patterns, and then we use the average values among them to show the results.

4.2 Throughput Performance

We first describe the throughput performance of our load balancing method. **Figure 7** shows the improvement of total network throughput of the proposed method compared to conventional single-path shortest-path routing, with the variation of network load. To perform fair comparison of the proposed and the conventional methods, we use as x-axis the average link usage with the shortest-path packets, where the shortest-path packets means the packets that have not detoured by the proposed method. Namely, the network load in x-axis is determined from the number of flows generated at that time. The result shows that, in both cases of 30 and 100 nodes, the proposed method improved network throughput, and the improvement reaches 10% at 40–50% of the link load. There is a small difference between the cases of 30 and 100 nodes, but generally they indicate the similar performance. **Figure 8** shows the ratio of packets rescued out of all the detoured packets. Both cases indicate similar performances, where the rescued ratio decreases as the load increases.

Figure 9 shows the classification result of the rescued packets by the number of detours experienced. Most part of them is rescued with single detour, while two and three times detoured packets occupy a non-negligible part of them. This result shows that the multiple detouring contributes to the improvement of throughput. **Figure 10** shows the same classification result in the case of 100 nodes. The similar tendency is seen here, but the ratio of

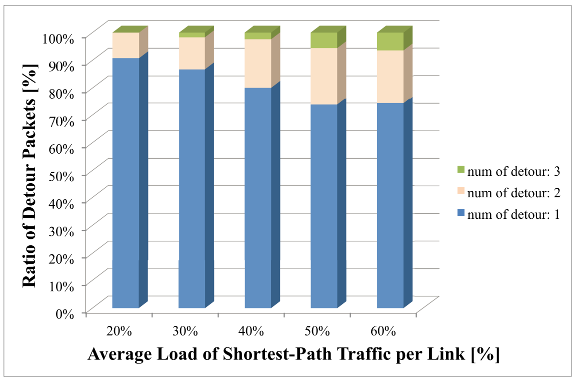

Fig. 10 The number of detours to reach destination (case of 100 nodes).

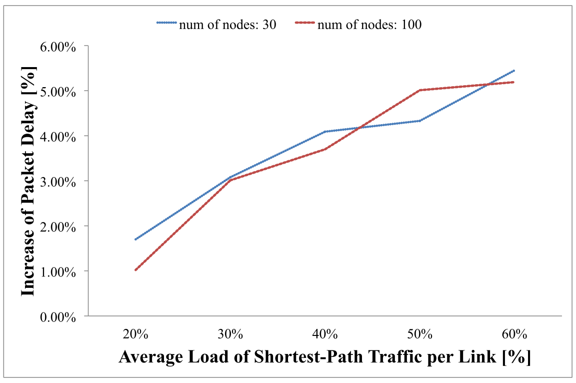

Fig. 11 Increase of packet delay of shortest-path traffic.

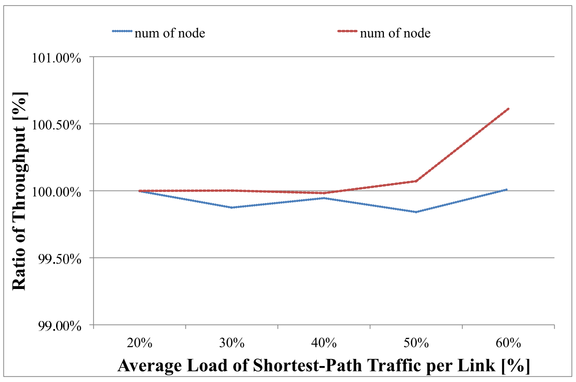

Fig. 12 Ratio of throughput of shortest-path traffic.

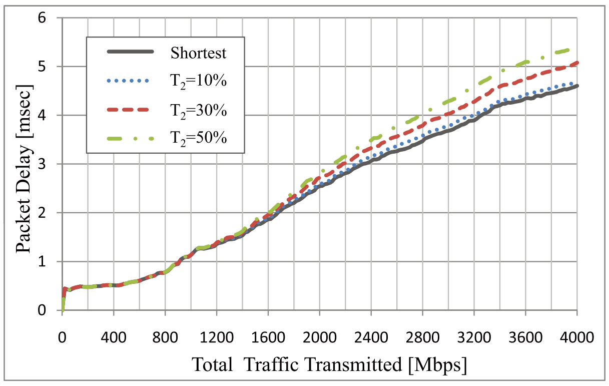

Fig. 13 Averaged packet delay to reach destinations.

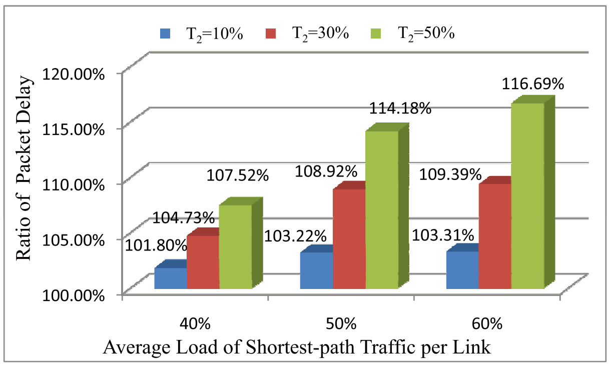

Fig. 14 Increase of packet delay compared to the shortest-path routing.

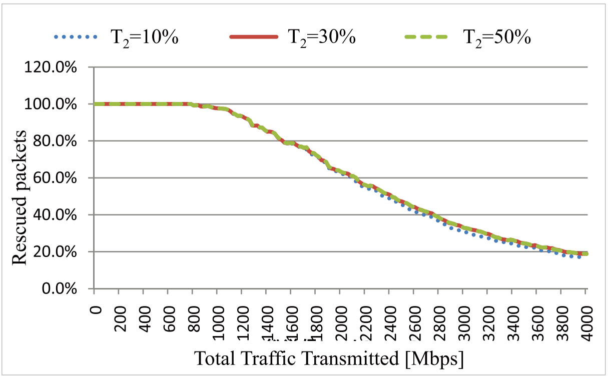

Fig. 15 Ratio of rescued packets in detoured packets.

more than one detour grows larger as the network size gets larger.

4.3 Effect on the Original Traffic

Our method intends to utilize vacant capacity using backup paths, without effecting on the behavior of the original non-detoured traffic. In this section, we show the results to clarify the influence of the detoured traffic over the original shortest-path traffic.

Figure 11 shows the increased delay of the original traffic, compared to the case of the shortest-paths without the proposed method. The packet delay slightly increased; at most 5% when the load is as high as 60%. **Figure 12** shows the effect over the throughput of the original traffic in ratio compared to the case of the shortest paths. In both cases of 30 and 100 nodes, the throughput of the original traffic is almost the same as the case of the shortest paths.

Note that we can decrease the delay if we use a smaller T_2 value; there is a trade-off between the delay and the resilience against the bursty traffic. Therefore, we conducted another simulation to estimate the effect of T_2 value. The simulation is done using the same scenario and parameters, except that the value T_2 is varied.

Figure 13 shows the packet delay to reach destinations in the 30 node scenario under variation of total transmitted rate. Naturally, packet delay gets larger as the transmitted rate increases. Also, the packet delay gets larger as T_2 goes larger. **Figure 14** is the same result with a different x-axis, the network load used in Fig. 7. The increase of packet delay is within 5% when $T_2 = 10\%$, whereas it increases rapidly when we increase T_2. **Figure 15** shows the ratio of rescued packets among all the detoured packets. The rescued ratio is almost the same regardless of T_2 values, although the case $T_2 = 10\%$ is slightly small. This implies that, the value $T_2 = 10\%$ is sufficiently large to afford fluctuation of traffic in this scenario.

From the whole results above, it is clarified that the effect of the detoured traffic over the original traffic is sufficiently small.

5. Discussion

5.1 Consideration of Base IP Fast Reroute Mechanisms

In this paper, we investigated the performance of the proposed load balancing method, which is based on an IP fast reroute mechanism SBR. There are, however, several well studied IP fast reroute mechanisms [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] other than SBR, and we can also use them as the base IP fast reroute schemes of the proposed method. In this section, we first describe the literature of IP fast reroute schemes concisely, and then present how to modify the existing IP fast reroute schemes to provide the load balancing functionality.

In designing IP fast reroute mechanisms, there is a tradeoff between the overhead required to extend routing protocols and the capability of schemes. LFA (Loop-Free Alternate) [8] would be the least overhead IP fast reroute scheme, which only uses the pre-computed alternative hop if a node cannot use the primary next-hop due to failure. However, they cannot guarantee to protect even single link failure because LFA can use the backup paths whose length is limited to 1-hop.

To cover every single link/node failure, further overhead is required. Lee et al. proposed a FIR (Failure Inferencing based Rerouting) [14], which covers single link failure. FIR requires an additional routing table, which is same as LFA, but watches the in-coming interface of packets to distinguish the packets that have met failure and need more to use the secondary routing table to avoid the failed link. Later, they proposed FIFR [15], [16] that recovers from any single node failure with the same overhead. SBR [19], the IP fast reroute scheme used in this paper, is classified into this two-table approach. The authors further extended SBR to support protection of any single node failure [9]. With a 2-bit field on packet header, SBR prevents inefficient packet bouncing in backup paths as well as harmful packet loops in case of multiple failure.

As another major approach for IP fast reroute, Shand et al. proposed the mechanism based on IP tunneling, which is called NotVia [12]. It protects any single node/link failure by forwarding packets first to some intermediate nodes using IP tunnels, and then forwarding them to their destinations using shortest paths. NotVia requires the overhead of IP encapsulation and the table of the intermediate nodes for each destination.

As a scheme that is possibly more capable, Kvalbein et al. proposed a multi-tree approach for node protection called MRC [17], [18], which computes multiple spanning trees computed from the topologies in each of which several nodes and links are "isolated" from the original topology. MRC is potentially capable to protect multiple failure, but it requires heavy overhead; it not only requires multiple routing tables corresponding to each spanning tree, but also packet marking on packet header to indicate the tree that the packets are forwarded with.

To apply our load balancing mechanism to these IP fast reroute schemes in the literature, we have to implement (1) the r-counter

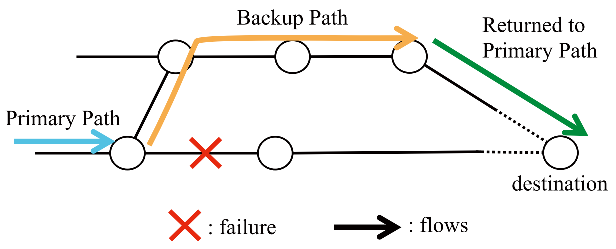

Fig. 16 Three parts of forwarding paths in IP fast reroute schemes.

to count the number of rerouting that the packet experienced, and (2) the flag to judge whether a packet is in the backup path or not. Fortunately, in most of the IP fast reroute schemes proposed so far, including all the referred ones in this paper, this extension is possible; they normally use the primary path, switch to the backup path in face of failure, and then return back to the primary path (see **Fig. 16**). Because most IP fast reroute schemes distinguish these three parts of the forwarding path, the same extension that we present for SBR is possible for these IP fast reroute schemes using a few-bit field on packet header.

5.2 Practical Issues

In this paper, we evaluated the proposed method in comparison to the naive IPFRR scheme, rather than the TE based load-balancing methods such as Refs. [2], [4], [5], [6], [28]. We note that it is difficult to implement them as real-time load balancing schemes due to the complexity of their optimization process.

The link-cost optimization approach [2] and S-OSPS [4] incorporate the linear programing as its optimization process that requires a demand matrix as its input. Because the demand matrix consists of the requested bandwidth between all pairs of nodes, it requires cost to measure the traffic amount among all pairs of nodes in a network. Note that, in practice, a naive observation would not be able to measure the pairwise traffic amount within the network. To avoid this difficulty, Kodialam et al. [28] proposed an optimization process that requires only the in-coming and the out-going traffic amount at all nodes. Although they reduced the measurement cost in the network, their linear-programming-based optimization process includes $O(MN^2)$ variables and constraints, where M is the number of links and N is that of nodes; the complexity of the problem is too large and critical for practice use. Reference [27] reported that it takes more than five hours even for a small real topology.

Antic et al. [27] proposed a more time-efficient optimization process for load balancing, which includes $O(N)$ variables and $O(M)$ constraints. They reported that it takes at most 60 seconds even for a large real topology stored in Rocketfuel topology database [29]. Although this process might be used in a small topology, it would be hard to pursue the rapid transition of the recent Internet traffic in a large topology. In the current Internet, the traffic transition is getting harder to predict due to several sorts of unpredictable high-rate traffic such as P2P and on-demand flows. Flexible and immediate control of traffic would be valuable in the current and the future Internet.

The proposed method, in contrast, provides immediate reaction against real-time transition of traffic by omitting neither the complex traffic measurement within a network and the timeconsuming optimization processes. This not only allows us to implement a load-balancing function inexpensively, but also enables us to pursue any rapid transition of the Internet traffic, which would provide a new benefit in load-balanced traffic controlling in IP networks.

As another issue that should be considered in practice, we would consider the few-bit field on packet header. Because the standard of IP format hardly affords extra-bits on packet header, even a few-bit extra field may be a problem in practice. For IPv4, TOS (Type Of Service) field would be the candidate for the few-bit field. Although TOS field is used in several other standards such as DiffServ [30], it is still possibly available in the network that does not deploy such mechanisms. For IPv6, the option header may be a candidate for it. Note that this issue is the matter of standardization. Although discussions for it would be required towards practical use, we now just mention that the space for the few-bit field is possibly available.

6. Concluding Remarks

In this paper we proposed a new load balancing technique based on IP fast reroute mechanisms. Our method detours packets when packets meet congestion to make the most of vacant resources of networks. Also, to suppress the bad influence of detoured traffic on the original shortest-path traffic, our method gives priority to the original traffic over the detoured traffic. Through traffic simulation with various network load, we confirmed that the proposed method improves network capacity at most 10% under the scenario with congestion hot-spots. We also confirmed that the bad influence on the original traffic is suppressed to a level acceptable in practical use.

There are several tasks to be done in the future. First, it is desirable to perform evaluation on the real topology with realistic traffic patterns. Although the result in this paper using the modeled random topology shows a general performance of the proposed method, case studies using specific real topologies is valuable to understand the behavior of the proposed method in practice. Second, evaluation of the performance when we apply the proposed load-balancing method to other IP fast reroute schemes is one of the major interests for the future. Note that, several IP fast reroute schemes have been proposed that protect single node failure with the same overhead as single link protection. Especially, the performance over node-protection IP fast reroute schemes would be one of the tasks to be done for the next step. Finally, we point out that it is essential to take TCP traffic into account. TCP performance is degraded due to packet reordering. We have mainly two choices to deal with TCP traffic, i.e., (i) applying hash-based load balancing where packets in the same flow use the same path, or (ii) detouring only UDP packets to balance the network load. To find the better choice is one of the important issues for the future.

References

[1] Awduche, D.O., Malcolm, J., Agogbua, J., O'Dell, M. and McManus, J.: Requirements for Traffic Engineering Over MPLS, RFC2702 IETF (1999).

[2] Forts, B. and Thorup, M.: Internet Traffic Engineering by Optimizing OSPF Weights, *Proc. INFOCOM 2000*, pp.519–528 (2000).

[3] Dasgupta, S., de Oliveira, J.C. and Vasseur, J.P.: A Performance Study of IP and MPLS Traffic Engineering Techniques under Traffic Variations, *Proc. IEEE Globecom 2007*, pp.2757–2762 (2007).

[4] Mishra, A.K. and Sahoo, A.: S-OSPF: A Traffic Engineering Solution for OSPF based Best Effort Networks, *Proc. IEEE Globecom 2007*, pp.1845–1849 (2007).

[5] Antić, M. and Smiljanić, A.: Oblivious Routing Scheme Using Load Balancing Over Shortest Paths, *Proc. IEEE ICC 2008*, pp.5783–5737 (2008).

[6] Oki, E. and Iwaki, A.: F-TPR: Fine Two-Phase IP Rerouting Scheme over Shortest Paths for Hose Model, *IEEE Communications Letters*, Vol.13, No.4, pp.272–279 (2009).

[7] Zhang, M. and Liu, B.: Traffic Engineering for Proactive Failure Recovery of IP Networks, *Tsinghua Science & Technology*, Vol.16, No.1, pp.55–61 (2011).

[8] Atlas, A. and Zinin, A.: Basic Specification for IP Fast Reroute: Loop-free Alternate, RFC5286 IETF (2008).

[9] Yoshihiro, T. and Jibiki, M.: Single Node Protection without Bouncing in IP Networks, *Proc. IEEE 13th International Conference on High Performance Switching and Routing* (*HPSR*), pp.88–95 (2012).

[10] Rai, S. and Mukherjee, B.: IP Resilience within and Autonomous System: Current Approachs, Challenges, and Future Directions, *IEEE Communications Magazine*, Vol.43, No.10, pp.142–149 (Oct. 2005).

[11] Bryand, S. and Shand, M.: IP fast-reroute framework, IETF Internet-draft draft-ietf-rtgwg-ipfrr-framework-09.txt (2008).

[12] Shand, M., Bryand, S. and Previdi, S.: IP Fast Reroute Using Not-via Addresses, draft-ietf-rtgwg-ipfrr-notvia-addresses-04.txt (2009).

[13] Reichert, C., Glickmann, Y. and Magedanz, T.: Two Routing Algorithms for Failure Protection in IP Networks, *Proc. 10th IEEE Symposium on Computers and Communications* (*ISCC 2005*), pp.97–102 (2005).

[14] Lee, S., Yu, Y., Nelakuditi, S., Zhang, Z.L. and Chuah, C.N.: Proactive vs. Reactive Approaches to Failure Resilient Routing, *Proc. IEEE INFOCOM '04* (Mar. 2004).

[15] Zhong, Z., Nelakuditi, S., Yu, Y., Lee, S., Wang, J. and Chuah, C.N.: Failure inferencing based fast rerouting for handling transient link and node failures, *Proc. IEEE Global Internet* (Mar. 2005).

[16] Wang, J. and Nelakuditi, S.: IP Fast Reroute with Failure Inferencing, *Proc. SIGCOMM Workshop* (*INM 2007*), pp.268–273 (2007).

[17] Kvalbein, A., Hansen, A.F., Čičić, T., Gjessing, S. and Lysne, O.: Multiple routing configurations for fast IP network recovery, *IEEE/ACM Trans. Networking*, Vol.17, No.2, pp.473–486 (2009).

[18] Čičić, T., Hansen, A.F., Kvalbein, A., Hartmann, M., Martin, R. and Menth, M.: Relaxed multiple routing configurations for IP fast reroute, *Proc. NOMS2008*, pp.457–464 (2008).

[19] Yoshihiro, T.: A Single Backup-Table Rerouting Scheme for Fast Failure Protection in OSPF, *IEICE Trans. Comm.*, Vol.E91-B, No.9, pp.2838–2847 (2008).

[20] Ito, H., Iwama, K., Okabe, Y. and Yoshihiro, T.: Single backup table schemes for shortest-path routing, *Theoretical Computer Science*, Vol.333, No.3, pp.347–353 (2005).

[21] Ito, H., Iwama, K., Okabe, Y. and Yoshihiro, T.: Polynomial-Time Computable Backup Tables for Shortest-Path Routing, *Proc. SIROCCO 2003*, pp.163–177 (June 2003).

[22] Xi, K. and Chao, J.: IP Fast Rerouting for Single-Link/Node Failure Recovery, *Proc. IEEE BROADNETS2007*, pp.142–151 (2007).

[23] Medina, A., Lakhina, A., Matta, I. and Byers, J.: BRITE: An approach to universal topology generation, *Proc. IEEE MASCOTS*, pp.346–353 (Aug. 2001).

[24] Waxman, B.: Routing of Multipoint Connections, *IEEE J. Select. Areas Commun.* (Dec. 1988).

[25] Barábasi, A.L. and Albert, R.: Emergence of Scaling in Random Networks, *Science*, Vol.286, pp.509–512 (Oct. 1999).

[26] The network Simulator NS-2, available from ⟨http://www.isi.edu/nsnam/ns/⟩ (accessed 2013-08-12).

[27] Antić, M., Maksić, N., Knežević, P. and Smiljanić, A.: Two Phase Load Balanced Routing using OSPF, *IEEE Journal on Selected Areas in Communications*, Vol.28, No.1, pp.51–59 (2010).

[28] Kodialam, M., Lakshman, T.V. and Sengupta, S.: Maximum Throughput Routing of Traffic in the Hose Model, *IEEE Proc. INFOCOM 2006* (2006).

[29] Rocketfuel, available from ⟨http://www.cs.washington.edu/research/networking/rocketfuel/⟩ (accessed 2013-08-12).

[30] Nichols, K., Blake, S., Baker, F. and Black, D.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC2474 IETF (1998).

Masaki Hara received his B.E. and M.E. degrees from Wakayama University in 2008 and 2010, respectively. He is currently working with Nintendo Co., Ltd.

Takuya Yoshihiro received his B.E., M.I. and Ph.D. degrees from Kyoto University in 1998, 2000 and 2003, respectively. He was an assistant professor in Wakayama University from 2003 to 2009. He has been an associate professor in Wakayama University from 2009. He is currently interested in the graph theory, distributed algorithms, computer networks, wireless networks, medical applications, bioinformatics, etc. He is a member of IEEE, IEICE, and IPSJ.