Algorithms for network service guarantee under minimal link usage information

Sugwon Hong[1], Hwang-Kyu Lee[2], Seonuck Paek[3]

[1]Department of Computer Software, Myongji University
San 38-2 Namdong, Yongin, Gyeonggi-do, 449-728 Korea, swhong@mju.ac.kr
[2] Tecoware, 823-21 Yeoksam-dong, Kangnam-gu, Seoul, Korea, exarch@korea.com
[3] Division of Information Technology & Communication
Sangmyung University, paeksu@smu.ac.kr

Abstract. One way to guaranteeing service for an application flow even if a network happens to fail is to establish a restoration path with the bandwidth that amounts to the same of the flow. If the flows can share the bandwidth for their restoration paths with others, we can reduce bandwidth consumption required for restoration. It is also required that deciding sharable bandwidth among flows should be done using controllable link information at each node. This paper proposes an algorithm to determine the sharable bandwidth among application flows given local link usage information at each node, validates the results of the algorithm and analyze the conditions required to achieve the goal by simulation.

1 Introduction

Fast restoration time and service guarantee are the important goals to achieve the network reliability and survivability. One way to continuing traffic delivery when a network happens to fail is to establish an alternative path to detour to destination avoiding the point of failure. The network protection or recovery schemes using alternative paths and the classification are explained in [1]. Various rerouting methods using alternative paths have been extensively studied for network survivability [2].

For service guarantee of application flows, it is also required to guarantee traffic delivery without any service degradation. For this purpose not only do networks have to provide a restoration path, but also provide enough bandwidth to secure detoured traffic of an application flow. One simple way to guaranteeing service for an application flow when network failure happens is to establish a restoration path with the same bandwidth of a working path at the same time. However, it requires exact twice as much as the bandwidth for an application flow whenever a working path is established.

In order to reduce bandwidth consumption for restoration paths, those paths should share their bandwidths each other as long as the sharing does not degrade original service. One problem of sharing bandwidth between restoration paths is the complexity of keeping the necessary information at each node to determine which flows can share bandwidth at each link for this purpose. The information that each node should

maintain is enormous, and up-to-date information should be propagated to every node whenever new paths are established. Therefore, when we try to find a solution to determine sharable bandwidth, it is required that the method should be done using controllable information at each node.

In this paper we first define the problem in section 2. In section 3 we propose two algorithms to set up a bandwidth-guaranteed path for restoration with local link usage information. In section 4 and 5 we do simulation and analyze the results. In section 6 we conclude the paper.

2 Problem Definition

In this paper we assume that when a working path is established, its corresponding restoration path is set up simultaneously. The ingress node of the restoration path becomes the point, where traffic is transferred from the working path to the restoration path when links fail. In general this node is not a point of failure.

An example of establishing working and restoration paths is shown in figure 1. As shown in this figure, the traffic for session k1, k2, k3 flowing through the working paths will be transferred to the corresponding restoration paths respectively when any link or links on working paths happen to fail.

In order to guarantee traffic delivery without any service degradation when links fail on the working paths, we have to provide the equivalent bandwidth for their restoration paths. However, bandwidth consumption will become twofold.. Thus the problem to be raised is how much bandwidth should be assigned for a restoration path for guaranteeing service of an application flow, and at the same time minimizing bandwidth consumption as a whole.

To approach this problem we postulate the following two conditions.

Condition 1: The service must be guaranteed regardless of any link failure on a working path.

Condition 2: More than two working paths never fail simultaneously.

In order to satisfy the condition 1, a restoration path should take a disjoint path of the corresponding working path. The condition 2 is realistic assumption because more than two working path are unlikely to fail at the same time. If working paths do not have any shared link each other, they can use the same restoration path, consequently reducing the bandwidth consumption incurred due to the restoration path.

In figure 1, the bandwidth of the link (u, v) for the restoration path of the session k1 can be shared with the restoration paths of the other sessions if the sessions take disjoint links of working paths each other. However, suppose that session k1 and k2 take routes with at least one same link on their paths, say link (i, j). In this case if they have a shared link on their restoration paths, say link (u, v), the service will not be guaranteed when the link (i j) is broken. Thus the bandwidth for the session k1 and k2 at link (u, v) should be assigned as much as the sum of the bandwidths required for each working path.

In order to decide whether or not the bandwidths for restoration paths at each link can be shared with others, each node needs to have the following information:

$$\{\{k_1, k_2, ..., k_n\}, (L^{k1}, L^{k2}, ..., L^{kn}), (a^{k1}, a^{k2}, ..., a^{kn})\} \quad (1)$$

where $K=\{k_1,k_2,\ldots,k_n\}$ is the set of sessions for the restoration paths established at this moment passing through link (i, j). L^k is the set of links on the working path for the session k, and a^k is the bandwidth requested for the session k. To keep such information imposes enormous burden on each node. Thus not only do we need to find an algorithm to determine sharable bandwidths among restoration paths, the algorithm should be based on minimal link information as well.

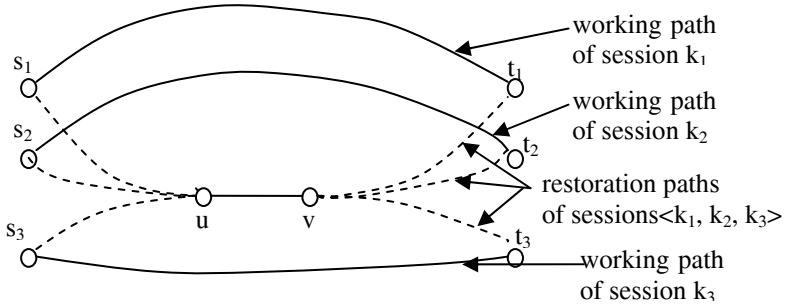

Fig. 1. The working paths and restoration paths of sessions $<k_1, k_2, k_3>$

3. Algorithms

3.1 Algorithm 1

One way to avoid keeping all information shown in equation (1) is to use the maximum bandwidth assigned to the links on a working path as the bandwidth required for its restoration path. This method was proposed and the effect of bandwidth reduction was shown in [3] and [4]. When a session establishes a working path and its restoration path, the source examines the bandwidths of all links assigned on the working path. Then it chooses the maximum bandwidth among them, and uses it as the bandwidth for its restoration path to be established. In this case the only information needed at each node is the total bandwidth assigned at this moment for working paths passing through the link.

This method satisfies the condition 1 if a restoration path takes disjoints routes of the corresponding working path. Since the bandwidth for a restoration path is the maximum bandwidth among those assigned to all links on the working path, any link failure on the working path will not degrade the service of a session whose traffic flows on the restoration path, which leads to satisfy the condition 2.

The following describes how we determine the bandwidth for a working path at link (i, j) and the bandwidth of its restoration path at link (u, v) when a new path request occurs.

a_{ij}^k : bandwidth for the working path of session k passing through link (i, j)

b_{uv}^k : bandwidth for the restoration path of session k passing through link (u, v)

m_{ij}^k : bandwidth requested by session k passing through link (i, j)

R_{uv} : bandwidth left at link (u, v)
G_{uv} : total bandwidth of all restoration paths to be established at link (u, v)
M : the maximum bandwidth among the links on a working path
F_{ij} : the total bandwidth assigned to all working paths passing link (i, j)
L(s,t) : a set of links consisting of a path from source *s* to destination *t*

$$M = \max_{all\ (i,j)\ \in\ L(s,t)} F_{ij} \tag{2}$$

$$a_{ij}^k = \begin{cases} m_{ij}^k & if\ R_{ij} > m_{ij}^k \\ \infty & if\ R_{ij} < m_{ij}^k \end{cases}$$

$$b_{uv}^k = \begin{cases} 0 & if\ M < G_{uv} \\ M - G_{uv} & if\ M > G_{uv} \quad and \quad R_{uv} \geq M - G_{uv} \\ \infty & if\ R_{uv} \leq M - G_{uv} \end{cases}$$

3.2 Algorithm 2

The goal of the algorithm is to find the bandwidth that can not be sharable among restoration paths since their working paths pass through the same link or links. The algorithm is also required to solve the problem using minimal link usage information. The algorithm 1 of section 3.1 uses the maximum bandwidth among the links which a working path passes through in order to determine the bandwidth for its restoration path.

We modify the algorithm 1 by allowing each node to have some information about links, which we call a link usage database, B(u, v) where (u, v) belongs to all links of a network. Every node keeps B(u, v) where the link (u, v) has the accumulated bandwidth of only the working paths whose restoration paths are passing through the link (u, v). So, when new restoration path is set up, we can look up B(u, v) to determine its bandwidth instead of finding M in equation (2) of the algorithm 1. This algorithm is described as follows:

a^k : bandwidth requested by session k

b_{ij}^k : bandwidth required for the restoration path of session k at link (*i, j*)

B(u,v) : the sum of bandwidths assigned for working paths which are assigned at link (*u,v*)

G(u,v): the sum of bandwidths assigned for the restoration paths which are established at link(*u,v*)

L : the set of links in a network
L^k : the set of links which consists of the working path of session k

1. initialize: $B(u,v) = 0 \quad for \quad all(u,v) \in L$
2. If a working path of session k is passing through link *(i,j)*,
   $B(u,v) = B(u,v) + a^k \quad for \quad all(u,v) \in L^k$
3: $b_{ij}^k = \max_{(u,v)\in L^k} B(u,v) - G(i,j)$

4 Experiment

In the experiment we simulate two kinds of network models. The model 1 consists of 5 nodes and 8 bi-directional links shown in figure 2. The model 2 is composed of 28 nodes and 45 bi-directional links shown in figure 3. The model 2 is a real transport network which has some backbone links to connect between local groups of nodes [5]. The model 1 is an artificial network where traffic is evenly distributed over all links. The reason we choose the model 1 is that this model highlights the sharing effect and might mislead results comparing to the model 2 which reflects real traffic distribution.

In this simulation, each node has the information about the bandwidths assigned to the links which are connected to the node. At each request, the simulation takes the following steps. First, two nodes are randomly selected as source and destination nodes, and random amount of bandwidth between 1 and 10 required for the path is determined.

Second, a working path and its restoration path are calculated based on the shortest pairs of disjoint path algorithm [6],[7]. The path is also chosen to satisfy the sharable bandwidth described by the algorithms. The simulation runs on requests varying from 20 to 700 at one batch run, and does 10 batch runs in total.

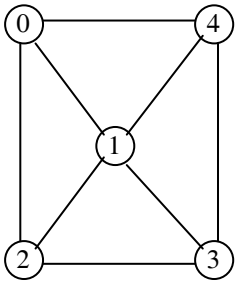

Fig. 2. The network model 1

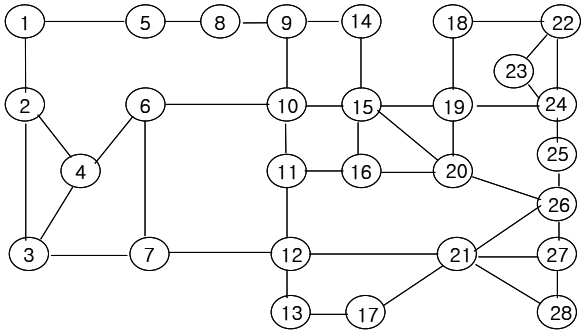

Fig. 3. The network model 2

5 Results and Analysis

The results of the algorithm 1 for the model 1 are shown in figure 4 and 5. In these figures, the bandwidth consumption means the sum of all bandwidths assigned to all links. This amounts to the total bandwidth requested for the paths multiplied by the average hops of links which each path consists of. NS denotes the case that no sharing is applied, while SS means that the sharing is applied. And AP and BP denote a working path and a restoration path respectively. In this figure, since the results of the algorithm 2 are very close to those of the algorithm 1, we show only the results of the algorithm 1. As shown in these figures we can save considerable bandwidth as we expected when the algorithm is applied.

The algorithm 1, however, does not always guarantee the reduction of bandwidth which is required for restoration. The figures 6 and 7 show such contradictory results. In this experiment of 200 requests, the total bandwidth for working paths is 3318.9 and the total bandwidth for restoration paths is 4955.9 when no sharing is applied. But when the algorithm 1 is applied for sharing, the restoration paths consume the total bandwidth of 6748.9. So sharing incurs more consumption.

In order to analyze this anomaly of sharing effect, let us assume the following case. Let us suppose that a session that has a working path from s to t passing the link (i j) and requires bandwidth n. Let us also suppose that bandwidth m is already assigned at link (i, j) and $m>n$. Then by the algorithm 1 the session requires the bandwidth at least m for its restoration path. If m is shared with very few other sessions which have the working paths passing through link (i, j), it incurs unnecessary bandwidth allocation on their restoration paths.

This kind of undesired effect happens when a network has several groups of nodes which have proximity each other within a group, and has hot spot links to connect between those groups. When the bandwidth is distributed over the links fairly evenly, the sharing algorithm gets a benefit. This happens in the case of the model 1. However, when the hot spot links to which more bandwidth are assigned than other links, the algorithm 1 might consume more bandwidth for restoration paths than when no sharing is applied.

To see this effect more clearly, we do another experiment which shows the bandwidth assigned to each link. Figure 8 shows the total bandwidth assigned at each link for the working paths and restoration paths respectively for the model 1 when the algorithm 1 is applied. The bandwidth distribution is fairly even among the links in this result. The result in figure 9, however, shows that the distribution is lopsided on a couple of links for the model 2. Therefore the working paths passing through these hot spot links require much more bandwidth for their restoration paths unnecessarily.

The algorithm 2 modifies the algorithm 1 by allowing each node to have a link data base which stores the accumulated bandwidth of only the working path whose restoration paths are passing through the link. In this way the algorithm 2 can calculate more accurate sharable bandwidth, consequently reducing the bandwidth consumption even to the cases when traffic distribution is heavily lopsided on some links.

We simulate the algorithm 2 for the network model 2 and obtain the results in figure 10. All assumptions and procedures are the same as done in the case of algorithm 1. The results show that the algorithm 2 saves the bandwidth for restoration paths comparing the result of figure 7.

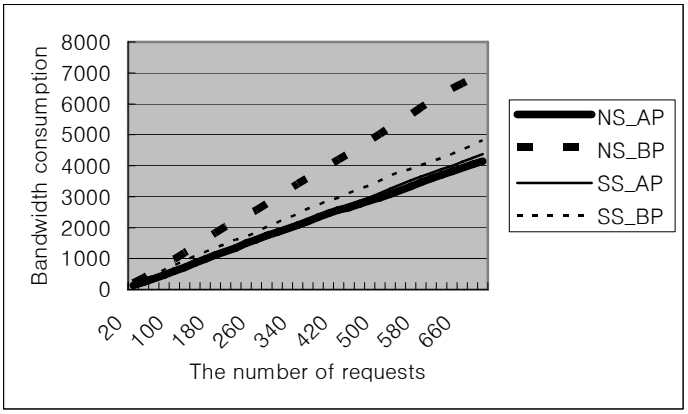

Fig. 4. Comparison of bandwidths for working and restoration path (model 1)

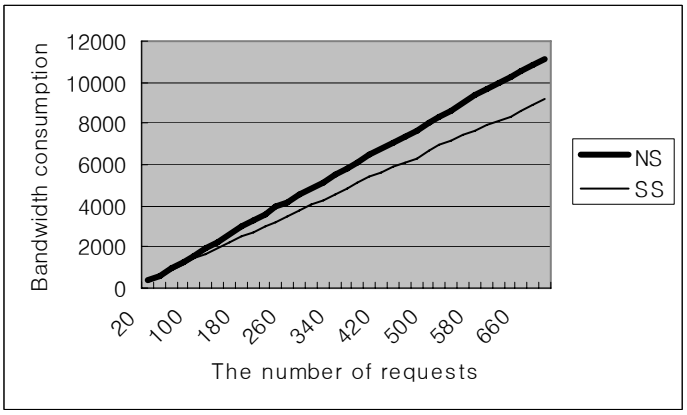

Fig. 5. Comparison of total bandwidths (model 1)

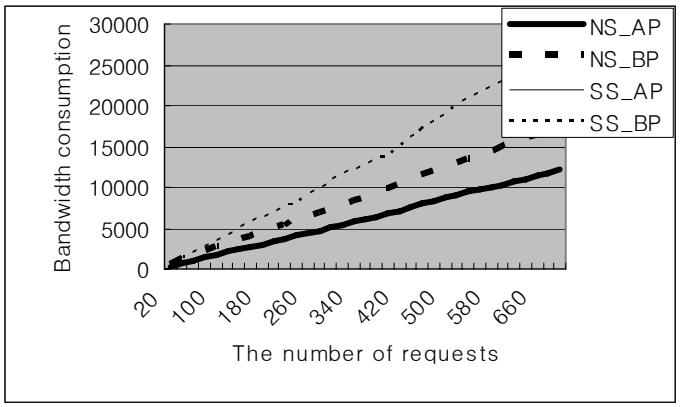

Fig.6. Comparison of bandwidths for working and restoration path by algorithm 1 (model 2)

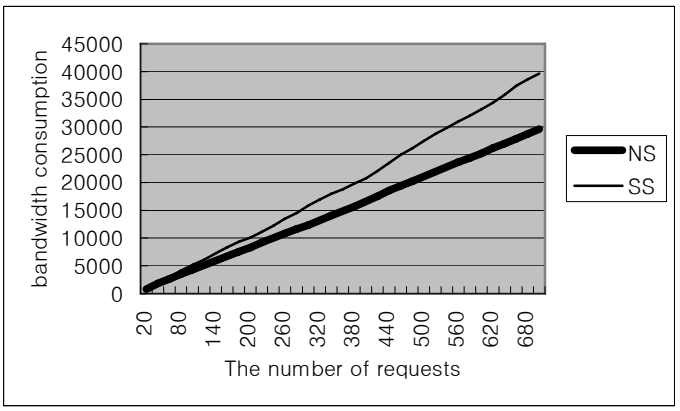

Fig. 7. Comparison of total bandwidths by algorithm 1 (model 2)

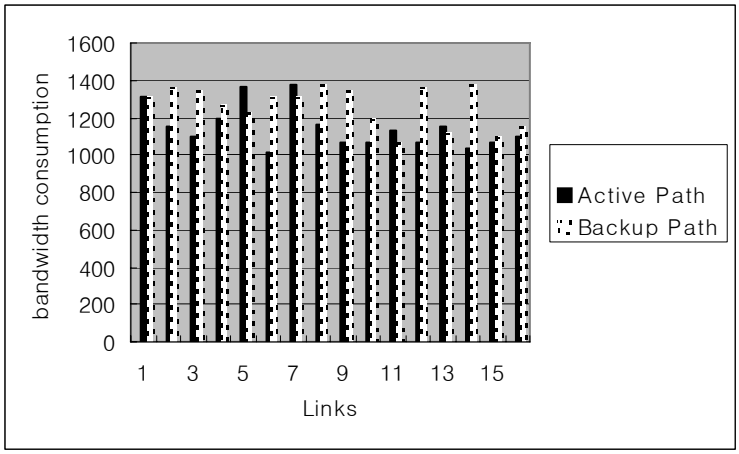

Fig. 8. The bandwidth assigned to each link by algorithm 1 (model 1)

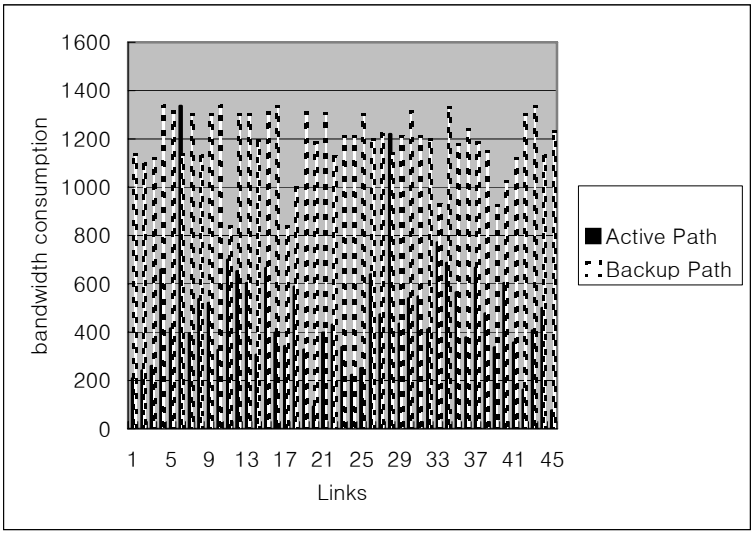

Fig. 9. The bandwidth assigned to each link by algorithm 1 (model 2)

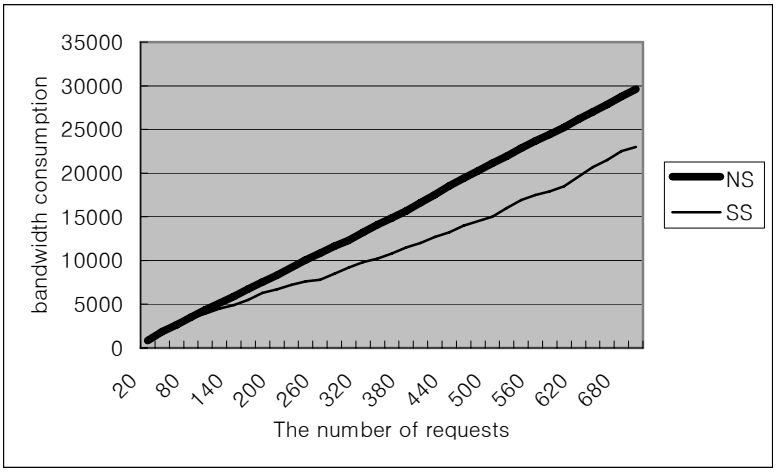

Fig. 10. Comparison of total bandwidths by algorithm 2 (model 2)

6. Conclusion

In this paper, we propose an algorithm to determine the bandwidth for restoration paths to guarantee service. For this purpose we approach the problem to meet two requirements. First, we should find the bandwidth required for restoration paths for assuring quality of service and at the same time use the least information of links at each node.

If we can use as much information as we need, it is not difficult to find the complete sharable bandwidth. However, to avoid heavy burden imposed on nodes to keep such information, we propose a way of finding the sharable bandwidth using minimal link usage information. The algorithm proposed here uses the link usage database at each node. The link usage database has the accumulated bandwidth of only the working paths whose restoration paths are passing through the link. The information is easily dealt with at each node. And we show that this algorithm works well under circumstances where traffic is unevenly distributed over links.

References

1. V. Sharma et al, " Framework for Multi-protocol Label Switching(MPLS)-based Recovery," RFC 3469, February 2003
2. R. Bhandari, " Survivable Networks: Algorithms for Diverse Routing," Kluwer Academic Publishers, 1999
3. M. Kodialam and T.V. Lakshman, " Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," INFOCOM 2000
4. K. Kar, M. Kodialam, and T. V. Laskshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," IEEE, J. of Selected Area of Com. Vol. 16(12), pp2566-2579, December 2000
5. K. Murakami, " Optical Capacity and Flow Assignment for Self-Healing ATM Networks Based on Line and End-to-End Restoration," IEEE/ACM Trans. On Networking 6(2), April 1998
6. J.W. Suurballe R.E. Tarjan "A Quick Method for Finding Shortest Pairs of Disjoint Paths," Networks, vol. 14, pp325-226, 1984
7. C.-L. Li, S. T. McCormick, and D. Simchi-Levi, "Finding Disjoint Paths with Different Path-Costs: Complexity and Algorithms," NETWORKS, vol. 22, pp653-667, 1992