This article was originally published in a journal published by Elsevier, and the attached copy is provided by Elsevier for the author’s benefit and for the benefit of the author’s institution, for non-commercial research and educational use including without limitation use in instruction at your institution, sending it to specific colleagues that you know, and providing a copy to your institution’s administrator.

All other uses, reproduction and distribution, including without limitation commercial reprints, selling or licensing copies or access, or posting on open internet sites, your personal or institution’s website or repository, are prohibited. For exceptions, permission may be sought for such use through Elsevier’s permissions site at:

http://www.elsevier.com/locate/permissionusematerial
A heuristic algorithm for shared segment protection in mesh WDM networks with limited backup path/segments length

Hongbin Luo *, Hongfang Yu, Lemin Li

Key Lab of Broadband Optical Transmission and Communication Networks University of Electronic Science and Technology of China, Chengdu 610054, China

Received 25 May 2005; received in revised form 19 April 2006; accepted 27 April 2006
Available online 24 May 2006

Abstract

This paper investigates the problem of dynamic survivable lightpath provisioning against single-link failure in optical mesh networks employing wavelength-division multiplexing (WDM). We focus on the special problem of provisioning lightpath requests according to their differentiated protection-switching time, since lightpath may have different protection-switching time requirements. We assume that the protection-switching time requirements of connections can be transformed to the hop limits of backup path/segments by using techniques proposed in the literature such as [Y. Luo, N. Ansari, Survivable GMPLS networks with QoS guarantees, IEE Proc. Commun., vol. 152, (4) (2005) 427–431]. We propose a heuristic algorithm, namely Suurballe-based Heuristic Algorithm using Least number of segments for SSP with hop Limit (SHALL), to efficiently solve this problem. We inspect the effects of hop limit on various performance matrices and compare the SHALL approach with three other well-known protection approaches, namely shared path protection (SPP), shared link based protection (LBP) and cascaded diverse routing (CDR). Numerical results demonstrate that the SHALL approach outperforms its counterparts in blocking probability and protection-switching time with mirror decrease of spare capacity efficiency.

Keywords: Wavelength division multiplexing (WDM); Quality-of-Service (QoS); Survivability; Routing algorithms; Shared segment protection; Protection-switching time

1. Introduction

Wavelength-routed wavelength division multiplexing (WDM) networks have been considered to be a promising network infrastructure for future backbone transport networks. In such networks, each fiber link offers huge bandwidth capacity to carry user traffic. A single network failure may cause a large amount of data loss in the network, which would largely degrade network performance and even disrupt network services. Thus network survivability is of great importance in such networks. To guarantee network services, the network must incorporate effective protection and restoration capabilities to survive different kinds of network failures (e.g., a fiber cut or a node fault).

Although the higher layers (e.g., IP, ATM and SONET) may provide their own protection and restoration mechanisms, it is still attractive to provide protection and restoration capabilities at the optical layer because of a number of advantages, such as fast service recovery, efficient resource utilization, and protocol transparency.

Survivability is the ability of the network to withstand equipment and link failures. The main goals of survivable network design are to be able to perform rapid restoration at as small a cost as possible (i.e., using minimum resources). Node equipment failures are typically handled using redundant equipment within the node (including redundant switches). On the other hand, link failures, which are by far the most common failures in optical networks, occur due to backbone accidents. In this paper, we restrict ourselves to the case of link failures.

At the same time, there is an increasing importance for service providers to provide guaranteed service in recent
years. This entails that survivable routing schemes should not only be both capacity- and computational-efficient, but also minimize the possible restoration time for a specific connection, such that the maximum benefits can be gained in the operation of carrier networks [15].

Segment shared protection (SSP) [1–14] is one of the best approaches to meet the above design requirements, where a working path is divided into a set of possible overlapping active segments (AS), and provide protection for some or all of the links along each AS using a backup segment (BS), which is link/node-disjoint with the AS. Comparing with its counterpart-shared path protection (SPP) [15–21], SSP has been reported to achieve a better throughput by maximizing the extent of spare capacity resource sharing.

In this paper, we investigate the problem of how to efficiently derive backup segments with limited hop length for a given working path in a dynamic network environment, where connection requests arrive dynamically one after another. In the following, we present the art of state of SSP and motivate our study.

1.1. Literature review

Much work on SSP has been conducted in optical WDM networks. In [1] and [4], two similar dynamic algorithms are proposed for each link to switch over from its immediate upstream neighbor node and to merge back to the original path at the immediate downstream node and any of the downstream nodes, respectively. However, both of the studies do not impose any limitation on the length of the backup paths, and may not be able to guarantee the restoration time when a failure occurs. It is notable that the lengthy backup paths can degrade overall performance even if they share spare capacity with the other backup paths [3]. In [14], an algorithm is developed to find the working path first followed by its backup path segments. This study is characterized by the fact that the backup bandwidth sharing is not considered until the physical routes of the backup segments are defined, which may impair the total performance. The study in [5] provides an algorithm for computing QoS paths with restoration, which is characterized by considering multiple-link metrics in searching the working and protection segments. This study does not consider resource sharing and has adopted exhaustive searching for those backup segments for the working path. The study in [6] proposes an integer linear program ILP for performing SSP according to the working path given in advance. The algorithm is characterized by the fact that it has to inspect all the possible allocations for self-healing loops along the working path and iteratively try all the possible number of self-healing loops. The studies in [7,8] take a very similar approach to that in [6]. The algorithm finds a backup path segment for each link along the working path given in advance, in which a “backtrack” by hops is allowed, where can be an arbitrary positive integer or infinity. In [9], a novel approach is proposed for segment protection that makes use of a modified graph for facilitating the searching of the backup segments. This study, however, does not consider spare resource sharing, and may not be able to take advantages of the effort of segmenting the primary path. In [10], a simulation-based study is conducted to investigate the performance of path, sub-path, and link restoration. The same as that in [9], the study does not consider resource sharing, and does not clearly define the adopted survivable routing approach. It is notable that all of the above schemes deal with segment protection by having not considered the backup segment length constraint, or even attempt to deal with it. In [11], a framework known as short leap shared protection (SLSP) is proposed, which implements SSP by pre-assigning a series of switching/merging node pairs along a given working path. In [6] and [11], a dynamic survivable routing algorithm called cascaded diverse routing (CDR) is proposed. To improve the flexibility and performance in finding the link-disjoint working and protection path-pair in each self-healing loop, k-shortest paths ranking between switching and merging nodes of each self-healing loop is performed. CDR is reported to outperform the path shared protection solutions. However, due to the fact that the location of switching/merging nodes for each working path are predefined instead of dynamically computed, there exist opportunities for further improvement. In [12], a scheme is devised to partition the network into multiple sub-networks such that the protection is performed within each sub-network domain. Although better computation efficiency can be achieved, the scheme may fail short of being less dynamic to the traffic distribution variation. At last, an ILP is formulated in [13] to solve the working and protection path segments for a connection request in a single step. Although the optimal solution (least-cost) with the best resource sharing can be derived, the resultant huge computation complexity in solving the ILP avoids the approach from any practical application.

1.2. Motivation

As quality-of-service (QoS) getting more and more important, some lightpath requests may have differentiated protection-switching-time (PST) requirements. For example, lightpaths carrying voice traffic may require 50 ms PST while lightpaths carrying data traffic may have a wide range of PST requirements [24]. It is notable that, in shared protection, the PST required to restore a connection upon a link failure is mainly determined by the hop length of the backup path/segment. This is because, in order to restore a connection, one has to configure the cross-connects along the backup path/segment, which often consume a time greatly longer than the failure notification time [24]. Thus, we in this paper assume that the PST requirements of connections can be transformed to hop limits of backup path/segments by using techniques proposed in the literature such as [23]. Note that, although we focus on limiting the hop length of backup path/segments, the SHALL approach proposed in this paper can be easily extended to the case that the total hop length
of the active segment and its corresponding backup segment is limited.

Due to the pathwise node-/link-disjoint nature of path protection, SPP may not provision lightpath requests according to their PST requirements effectively in practical-sized networks [17,24]. As mentioned above, however, prior work on SSP does not impose any limit on the hop length of backup segments. Thus prior SSP schemes cannot guarantee to derive backup path/segments that can restore the traffic in a predefined PST threshold if any link fails in the working path, even there does exist such backup path/segments. Consider the network shown in Fig. 1 as an example, where the working path (with dark lines) is \( P = A \rightarrow B \rightarrow C \) and the number next to each link is the link hop length is 4. Under the SSP scheme, the two active segments are \( A \rightarrow B \) and \( B \rightarrow C \) whose backup segments are \( A \rightarrow D \rightarrow E \rightarrow F \rightarrow C \) with 3 hop and \( B \rightarrow E \rightarrow F \rightarrow C \) with 3 hop, respectively. If the connection requests a hop limit of 2, the above backup path/segments derived cannot meet this hop limit and will be rejected. However, one can easily derive a backup segment with exactly 2 hops for each active segment, i.e., backup segment \( A \rightarrow D \rightarrow B \) for active segment \( A \rightarrow B \) and backup segment \( B \rightarrow F \rightarrow C \) for active segment \( B \rightarrow C \).

Clearly, proper mechanisms are needed to provision such lightpath requests in a resource-efficient manner. However, as the spare capacity is dependent on the working capacity and the exponentially enlarged design space with the network size, it is difficult to get an optimal solution, especially when the hop length of a backup path/segment is taken into account. Similar to most prior studies, we adopt the Two-Step-Approach to explore the dependency. Specifically, we place our focus on how to efficiently derive the backup path/segments when the working path has been determined. We propose a heuristic algorithm, namely Suurballe-based Heuristic Algorithm using Least number of segments for SSP with hop Limit (SHALL), to solve this problem efficiently.

The rest of this paper is organized as follows. Section 2 presents the background of this work. Section 3 gives out the network model and formulates the problem studied here. The algorithm SHALL is described in detail in Section 4 and numerical results are presented in Section 5. At last, Section 6 concludes the paper.

2. Backgrounds

Survivability schemes are classified into proactive protection and reactive restoration. In reactive restoration scheme, the resources used for recovery are not reserved before the failure occurs. Therefore, when a working lightpath fails, a search is initiated to find a new backup lightpath. This method does not guarantee 100% recovery, since a backup lightpath may not be found because of the lack of spare capacity in the network. Moreover, the search for the backup lightpath after failure leads to longer restoration time than that of the proactive method. On the other hand, reactive methods are more efficient from the point of view of capacity usage. They also can naturally handle multiple simultaneous fiber failures, and provide low overheads in the absence of failure.

On the other hand, protection refers to operations that are performed in advance of failure to defend the network against any possible disruption and are typically dealt with by allocating redundant capacity on other network links and switching the affected traffic to the redundant capacity. This method yields a 100% restoration guarantee, since a backup lightpath will be always available to carry the disrupted traffic when a working lightpath fails.

Depending on where protection switching is done, mesh protection schemes can be classified into link protection, path protection, and segment protection. In link protection (also called loopback protection), alternate paths, called backup or protection paths, between the end points of each link are precomputed. Upon the link’s failure, all of the lightpaths using the link (called primary or working lightpaths) are switched to their corresponding backup segments at the end nodes of the link. This is illustrated in Fig. 2 (a), where the active links AC, CF and FH, are shown in bold lines. And their corresponding backup segments are A–B–C, C–D–G–F and F–E–H, respectively.

In contrast, path protection entails the end-to-end rerouting of all working lightpaths that use the failed link along
precomputed backup lightpaths. Here, the entire route of the working lightpaths may be changed. Path protection can be further classified into either failure-dependent or failure-independent. With failure independent path protection, the backup path has to be link (node) disjoint with the working path, in order to protect against any single link (node) failure. Fig. 2(b) and (c) show two examples, where backup paths (in dashed lines) are link disjoint and node disjoint with their corresponding working paths (in bold lines), respectively. With failure-dependent path protection, multiple detours that are not necessarily link/node disjoint with a given AP are selected and which detour to use depends on which active link has failed, but rerouted traffic always goes through the source node (either the rerouting takes place at the source node or rerouted traffic loopbacks to the source node). In a sense, link protection is similar to failure-dependent path in that in link protection, which detour to take also depends on which link has failed, except that link protection uses “local” recovery (rerouting).

In segment protection, as shown in Fig. 2(d), a working path is divided into several working segments and each working segment is then protected with a backup segment (instead of protecting the working path as a whole as in path protection schemes). In segment protection, the end node, which is the nearest one in the segment to the source node of the working path, of a segment is called the switching node of it. And the other end node of the segment is called the merging node of it. In the above example, A–C–F and C–F–H are two working segments and their corresponding backup segments are A–B–E–F and C–D–G–H, respectively. Segment-based protection schemes are also similar to failure-dependent path protection (or link protection), but rerouted traffic only needs to go through the node that starts a backup segment which protects the failed link, as opposed to the source as in path protection (or the immediate upstream node of the failed link as in link protection). In general, both link and path protection can, thus, be considered as a special case of segment-based protection.

To achieve bandwidth efficiency in link, path or segment protection schemes, backup bandwidth allocated to different backup segments or backup paths can be shared. However, a non-shared (or dedicated) protection scheme can pre-configure the cross-connects and, thus, results in faster failure recovery.

3. Network model and problem formulation

In this section, we describe the network model and the problem addressed in this paper.

3.1. The network model

We represent the network by an undirected graph \( G(V, E) \), where \( V \) is the set of nodes and \( E \) is the set of links. We denote by \( N \) and \( M \) the number of network nodes and links, respectively, i.e., \( N = |V| \) and \( M = |E| \). Each link in fact consists of two simplex links in opposite direction: one for the transmit path and the other for the receive path. An \((s, t, w, h)\)-walk is a finite sequence of nodes \( W = (s = v_0, v_1, \ldots, t = v_n) \), such that, for \( 0 \leq i \leq n - 1 \), \((v_i, v_{i+1}) \in E \). Here, \( n = |W| \) is the hop count of \( W \). Note that nodes and links may appear in a walk several times. An \((s, t)\)-path \( P \) is an \((s, t)\)-walk whose nodes are distinct.

3.2. Definition of link cost

In response to a request \( r = (s, t, w, h) \), the source node uses its knowledge of the network state to compute two things: (1) a working path \( P = (s = v_0, v_1, \ldots, t = v_n) \), where edge \( l_i = (v_i, v_{i+1}) \in E \) and (2) a backup path or a set of backup segments that can route traffic from \( s \) to \( t \) if any link \( l_i \) fails such that the required hop limit can be met.

The computation of \( P \) and backup path/segments is based on two kinds of link costs described next.

Cost of link \( l \) in the working path \( P \): if a link is used in the working path \( P \) of request \( r = (s, t, w, h) \), then \( w \) units of bandwidth has to be consumed on each link in \( P \). Therefore, the cost \( c_l \) of using a link \( l \) in the working path \( P \) is \( w + c_l \) if the residual bandwidth, \( R_l \), on link \( l \) is no less than \( w \), and \( \infty \) otherwise. That is, the cost function for finding a working path should be determined as follows:

\[
c_w(l) = \begin{cases} \infty, & \text{if } R_l \leq M \\ w + c_l, & \text{otherwise} \end{cases}
\]

where \( c_l \) is the cost for each unit of bandwidth taken by a working path along link \( l \).

Cost of link \( l \) in backup path/segments: For the links in the backup path/segments, however, there is no need to allocate \( w \) units of bandwidth due to backup bandwidth sharing. Generally, the backup bandwidth requirement on each link in the backup path can be determined only after the working path has been selected.

Let \( R_b \) be the total backup bandwidth allocated on link \( b \) to the set of connections whose backup paths traverse link \( b \), \( S_b^w \) be the total amount of bandwidth required by the set of connections whose working paths traverse link \( a \) and whose backup paths traverse link \( b \) and \( B_b^w \) be the additional backup bandwidth needed on link \( b \) in order to use it as a part of a backup path for a new connection whose working path traverses link \( a \). With these assumptions, the cost on link \( e \) along a new backup path can be determined as

\[
c(l) = \max_{k \in P} B_e^l,
\]

where \( B_e^l \) can be worked out by employing the following Eq. (3).

\[
B_e^l = \begin{cases} \infty, & \text{if } a = b \text{ or } R_e < w \text{ or } R_b < S_a^w + w - B_b, \\ e, & \text{if } S_a^w + w \leq R_b, \\ S_a^w + w - B_b, & \text{if } S_a^w + w > B_b \text{ and } S_a^w + w - R_b \leq B_b. \end{cases}
\]

where \( e \) is a small number that used to prevent a backup path from being too long.
Let $E(P)$ be the set of links in a path $P$, the total cost of the path should be the sum of the cost of its links, that is
\[ C(P) = \sum_{l \in E(P)} c(l). \] (4)

Similarly, let $E(W)$ be the set of links in a walk $W$, the total cost the walk should be the sum of the cost of its links, i.e.,
\[ C(W) = \sum_{l \in E(W)} c(l). \] (5)

### 3.3 Problem formulation

Given a connection request $(s, t, w, H)$, where $s$ and $t$ are the source and destination nodes of the connection request, respectively, $w$ is the bandwidth requirement and $H$ is the backup path hop limit. Assume the working path of the connection request has been derived and is denoted as $PH$. Let $Sl = (s, t, w, H)$. Find backup path/segments such that (1) the hop length of each backup path/segment is no longer than $H$, (2) the total costs of them are minimum and (3) the traffic can be restored if any link in the working path fails.

### 4. Heuristic algorithms

We are now in a position to present our proposed SHALL algorithm, whose pseudo-code is shown in Fig. 3, where $SH$ and $S^H$ are the set of backup segments whose hop length is within and not within the hop limit, respectively.

The SHALL algorithm first, for each link in the network, assign a link cost as defined in Eq. (2). It then computes a shortest path $Q$ from $s$ to $t$ in the graph using Suurballe’s algorithm [22]. By deleting the edges of $Q$ that are in $P$, one can get the optimal backup segments for the working path $P$ [7,22]. Evidently, the edges of $Q$ allow the edges on the working path $P$ to be restored if any link in $P$ fails. However, the maximum hop length of some backup segments may not be bounded by the hop limit. The SHALL algorithm then checks (lines 6–10), for each backup segment, whether or not the backup segment is within the hop limit. If it is the case, the backup segment is put into the set $SH$; otherwise, it is put into the set $S^H$.

Conside the example in Fig. 4(a), where it is assumed that the graph is the resulting graph after line 5 and the hop limit requirement is 4. Evidently, the hop lengths of backup segment $s-v_1-v_2-v_3-v_4$ and backup segment $u_4-v_8-v_9-v_{10}-v_7$ are exactly 4, which is bounded by the hop limit, respectively. However, the hop lengths of backup segment $u_6-v_6-v_7-v_8-v_9-v_{10}-v_{15}$ and backup segment $u_6-v_6-v_7-v_8-v_9-v_{10}-v_{15}$ are 5, which is not bounded by the hop limit. The SHALL algorithm thus put the two backup segments $s-v_1-v_2-v_3-v_4$ and $u_4-v_8-v_9-v_{10}-v_{15}$ into set $SH$, and put the two backup segments $u_2-v_4-v_5-v_6-v_7-v_8$ and $u_6-v_6-v_7-v_8-v_9-v_{10}-v_{15}$ into set $S^H$.

For each backup segment in set $S^H$, we need to replace it by one or more new backup segments that meet the hop limit requirement. For ease of presentation, we place our focus on how to replace a given backup segment by using one or more backup segments that meet the hop limit requirement. For this purpose, let the backup segment be represented as $ui = u_1, \cdots, u_k$ and its corresponding active segment is represented as $ui = u_1, \cdots, u_m, \cdots, u_p$, respectively. Since our objective is to use as less backup segments as possible, we first use a shortest path algorithm to derive the least hop path $P'$ from $ui$ to $uj$. If the hop length of path $P'$ is less than the hop limit, the algorithm then replaces the current backup segment with path $P'$ and continues to check the rest backup segments. Otherwise, the algorithm uses the sub-procedure PARTITION, which will be described in detail later, to partition the original backup segment into several new backup segments such that each new backup segment meets the hop limit requirement. If the PARTITION procedure succeeds, the algorithm SHALL then replace the original backup segment by the newly derived backup segments and then proceeds to the next backup segment. And otherwise, the algorithm SHALL will return fail and end. Having guaranteed the hop length of each backup segment is within the hop limit, the SHALL algorithm then checks whether these have some redundant backup segments (lines 18–19). If this is the case, the SHALL algorithm then deleting these redundant. And otherwise the algorithm will end by returning the derived backup segments (line 20).
Before presenting the PARTITION sub-procedure, we are glad to present the reason why we need to check whether there have some redundant backup segments. For this purpose, let us assume that the resulting graph after line 13 is shown in Fig. 4(b). One can see that, after partition, the backup segment \( u_3 - v_4 - v_5 - v_6 - u_3 \) is redundant since the only link it protects is link \( u_2 - u_3 \), which has been protected by another backup segment \( s - v_1 - v_2 - v_3 - u_3 \). Thus the connection can also be restored if backup segment \( u_2 - v_4 - v_5 - v_6 - u_3 \) is deleted. Furthermore, by deleting it, the total cost consumed for protection is also reduced. In the above example, the resulting graph by deleting the redundant backup segment \( u_2 - v_4 - v_5 - v_6 - u_3 \) is shown in Fig. 4(c).

We are now in a position to present the sub-procedure PARTITION (which is shown in Fig. 6). Let the backup segment to be partitioned, say \( B_u \), be represented as \( u_1 - \cdots - v_k - \cdots - u_l \) and its corresponding active segment be represented as \( u_1 - \cdots - u_m - \cdots - u_p \), respectively. The basic idea of the PARTITION sub-procedure is to calculate, out for every node, \( v_k \), in the backup segment, a shortest path (in hop count) from the node to the active segment (step 1) and then properly selects one or more paths from the newly derived shortest paths to partition the original backup segment into several new backup segments such that each of them satisfies the hop limit (steps 2–5).

We take Fig. 4(a) as an example to illustrate how the PARTITION sub-procedure works. As stated above, the backup segment \( u_6 - v_8 - v_13 = v_14 - u_{10} \), which has a hop length of 5, does not bounded by the hop limit, 4. We assume that the shortest path from node \( u_6 \) to \( u_{10} \) cannot meet the hop limit requirement. Thus, there is a need to partition the backup segment \( u_6 - v_8 - v_13 = v_14 - u_{10} \) into several new backup segments such that each of them meets the hop limit requirement. The resulting shortest paths (dashed lines) are shown in Fig. 5 and the number next to each path is its hop count. Note that each newly derived path \( (v_k - u_m) \) partitions the original backup segment into two new backup segments, i.e., \( (u_1 - v_k) + (v_k - u_m) \) and \( (u_m - v_k) + (v_k - u_l) \). For ease of presentation, we call the newly derived backup segment that is more close to the source node of the working path as the upstream backup segment and the other as the downstream backup segment. With this definition, we say that a newly derived path partitions the original backup segment into an upstream backup segment, \( B_u \), and a downstream backup segment, \( B_d \).

Having derived the shortest paths, the PARTITION procedure then selects the path \( (v_11, u_7) \) as the current path and it partitions the original backup segment into an upstream backup segment \( (u_6 - v_11) + (v_11 - u_7) \) with 2 hops and a downstream backup segment \( (u_7 - v_12) + (v_12 - u_{10}) \) with 5 hops. Similarly, the path \( (v_{13} - u_6) \) partitions the original backup segment \( u_6 - v_{11} - v_{12} - v_{13} = v_{14} - u_{10} \) into an upstream backup segment \( (u_6 - v_{13}) + (v_{13} - u_6) \) with 4 hops and a downstream segment \( (u_6 - v_{11}) + (v_{11} - u_6) \) with 5 hops. Since the length of the downstream backup segment is not smaller than that of the original, the procedure then let the path \( (v_11, u_7) \) as the candidate path and proceeds to select the path \( (v_{12}, u_7) \) with 2 hops as the current path. Similar to the above path, the newly derived downstream backup segment has a hop length 5 and is equal to that of the original one. Thus the procedure let the path \( (v_{12}, u_7) \) as the candidate path and selects the path \( (v_{13}, u_6) \) as the current path. At this time, the newly derived upstream backup segment and downstream backup segment have a hop length 4 and 3, respectively. Thus the procedure selects the path \( (v_{13}, u_6) \) as the candidate path.
and partitions the original backup segment into two new backup segments. At this time, the sub-procedure will return to the main procedure with SUCCESS. The resulting backup segments after partition for Fig. 4(a) is shown in Fig. 4(b).

It is notable that, while our focus in this paper is on limiting the hop count of the backup path/segment, the SHALL approach proposed in this paper can be easily extended to include the case that the total number of hop count of an active segment and its corresponding backup segment is limited. To achieve this, we only need to change lines 7 and 14 in the main procedure and the step 2 of the sub-procedure PARTITION such that the total number of hop count of an active segment and its corresponding backup segment is limited.

5. Numerical results

In this section, we give numerical results to illustrate the performance of the SHALL algorithm when compared with three well-known algorithms, namely CDR proposed in [6,11], shared path protection (SPP) and shared link based protection (LBP). We have simulated the above four approaches by injecting a large number (e.g., few hundreds) of randomly generated requests (one after another in an online fashion) into the network shown in Fig. 7. We remark that more results from different topologies also led to the same observations. Due to space limitation, however, those results are not shown here.
5.1. Traffic types

We consider two types of traffic: the incremental traffic and the fluctuating traffic. In the incremental traffic, connections arrive dynamically one after another and an established connection lasts sufficiently long and does not depart. Furthermore, it is assumed, as in [4], [6] and [9], that each link in the network has a sufficiently large capacity. Therefore, a connection will be blocked if and only if the hop length of the backup path/segments for the connection is not within the hop limit.

While in the fluctuating traffic, connection may terminate after certain duration. And it is assumed that each link has a finite capacity (16 units of bandwidth, wavelength). Thus in this case, a connection will be blocked if (1) there is no enough resource to set up the connection, or (2) the hop length of the backup path/segments for the connection is not within the hop limit. We assume that demands arrive at the network according to a Poisson process with an average rate \( \lambda \), and the holding times are exponentially distributed with mean \( 1/\mu \).

In both cases, the ingress and egress of a connection establishment are evenly distributed among all nodes and the bandwidth required by each connection is 1 U.

5.2. Performance metrics

We use the following performance metrics to evaluate the performance of the SHALL algorithm.

(1) Blocking probability: It is the ratio of the number of requests that are blocked over the total number of requests arrived during each simulation run. This performance metric is widely used in literature [9,10,14,15]. However, as pointed out in [21], it may unfairly favor a scheme that satisfies more “short” connections over another that satisfies fewer “long” connections.

(2) Components of protection-switching time: When comparing different protection schemes, the failure notification time, the length of backup segment, and the sum of the length of the active segment and the length of the backup segment can serve as good indications of the recovery time (other components of the recovery time such as failure detection time are almost independent of the protection scheme used) [6].

(3) The control and management complexity: It is defined as the average number of backup path/segments per lightpath.

(4) Resource overbuild: Since various approaches may choose different working paths for the same connection request, we first define the service bandwidth as the minimum amount of active bandwidth required to satisfy a given connection request (along a shortest path) without considering the need for connection. This service bandwidth is then independent of protection approaches. Resource overbuild is the percentage increase in the total bandwidth (active + backup) required by an approach over the above service bandwidth to satisfy all requests. Generally, it shows the efficiency of backup sharing of a protection approach and is a major optimization objective.

5.3. Performance results

5.3.1. Effects of \( \epsilon \)

We will first inspect the effects of \( \epsilon \) on the above presented performance metrics. Specifically, we present the case \( H = 6 \). Other values of \( H \) led to the same observations and are not shown here.

5.3.1.1. Effects on blocking probability. Fig. 8 shows the blocking probability of the four approaches under incremental traffic. As shown, \( \epsilon \) has great effect on the blocking probability. The blocking probabilities of the LBP, SPP and CDR approaches reduce rapidly with the increase of \( \epsilon \) while that of the SHALL approach remains 0. Fig. 8 also shows that even when the capacity of a network is sufficient, the LBP, SPP and CDR approaches cannot guarantee to derive backup path/segments, each of them meets the hop limit requirement, for a given connection request. This demonstrates that there is a need to derive appropriate algorithms to cope with the problem of routing connections with backup path length limit. And this in fact motivates our study in this paper.

5.3.1.2. Effects on the protection-switching time. Fig. 9–11 show the effects of \( \epsilon \) on average hop number of working path/segments, average hop number of backup path/segments and average total hop number of working/backup path/segments under incremental traffic, respectively. From the three figures, one can see that the average hop number of working path/segments, the average hop number of backup path/segments and average total hop number of working/backup path/segments reduce with the increase of \( \epsilon \). This is mainly due to the fact that, with a smaller value of \( \epsilon \), the backup paths are easier to share with each other and leads to longer backup paths while consuming less capacity, which in turn affects the hop length of working path/segments and the total hop number of the working/backup path/segments. From the above results, we can conclude that the protection-switching time reduces with the increase of \( \epsilon \).

5.3.1.3. Effects on the control and management complexity. Fig. 12 shows the effects of \( \epsilon \) on the control and management complexity for the four protection schemes under incremental traffic. One can see that with the increase of \( \epsilon \), the control and management complexity increases rapidly. As a matter of fact, the average number of backup path/segments per lightpath is increased from approximate 1.5 to 2.9 under the LBP approach when \( \epsilon \) increases from 0.01 to 0.99. This implies that we should use a smaller value...
for $\varepsilon$ in order to reduce the control and management complexity.

5.3.1.4. Effects on the resource overbuild. Fig. 13 shows that, as expected, the resource overbuild increases with the increment of $\varepsilon$. This implies that the efficiency of backup sharing of a protection approach decreases when $\varepsilon$ increases. Thus, in order to increase the efficiency of backup sharing, we should use a smaller value for $\varepsilon$.

5.3.2. Effects of hop limit

We now inspect the effect of hop limit, $H$, on the performance metrics stated in Section 5.2. As stated in Section 5.3.1, in order to increasing the efficiency of backup
sharing, we should use a smaller value for $e$. Thus in the following sections of this paper, we assume $e = 0.01$, if not clearly pointed out. It is notable that other values of $e$ led to the same observations and are not shown here.

5.3.2.1. Effects on blocking probability. Fig. 14 shows the effects of hop limit on the blocking probability under increasing traffic for each protection approach. Clearly, the blocking probability decreases rapidly with the increase of $H$ for the LBP, SPP and CDR approaches. This implies
that the hop limit has great impact on the blocking performance for the three approaches. Note that the blocking of a request under the increasing traffic model is due to the fact that the number of hops of a backup segment does not within the scope of hop limit, but not due to lack of capacity. This further implies that it is necessary to derive novel algorithms to guarantee the protection-switching time for a given connection request. As a matter of fact, the increase of H has little or no effect on the blocking probability for the SHALL approach.

5.3.2.2. Effects on the protection-switching time. Figs. 15–17 show the effects of H on average hop number of working path/segments, average hop number of backup path/segments and average total hop number of working/backup path/segments under incremental traffic, respectively. From the three figures, one can see that, as expected, the average hop number of working path/segments, the average hop number of backup path/segments and average total hop number of working/backup path/segments increase with the increase of H. Thus, the protection-switching time gets longer when the hop limit increases.

5.3.2.3. Effects on control and management complexity. The effects of H on the control and management complexity are shown in Fig. 18 for each protection approach under the increasing traffic model. Contrary to our intuition, the control and management complexity increases with the increase of H. The reason for this is that, with small value of H, only these connections with small backup segment length can be accepted by the network while these with larger one will be rejected, even if there is only one backup segment is out the scope of H. However, with larger H, one connection can be protected by more backup segments, each of them is within the hop limit. And thus leads to a higher control and management complexity.

5.3.2.4. Effects on the resource overbuild. Fig. 19 shows the effects of H on the resource overbuild under increasing traffic. One can see that, the hop limit, H, has little effect on the resource overbuild for the LBP, SPP and CDR approaches. This is because the fact that, under these protection approaches, whether a connection will be accepted by the network is determined by the length of its backup path/segments, which is randomly distributed. Thus the
increase of $H$ has little effect on the resource overbuild. However, as expected, for the SHALL approach, the resource overbuild reduces with the increase of $H$. The reason for this is that, with a larger value of $H$, the SHALL algorithm is more possible to derive backup path/segments with smaller cost for a given connection, and thus decreases the resource overbuild.

5.3.3. Comparison of the four protection approaches

Figs. 20–25 compare various aspects of the LBP, SPP and CDR approaches with our proposed SHALL approach under the increasing traffic model and we assume $H = 6$, $\varepsilon = 0.01$. Fig. 20 compares the blocking performance of the four approaches. From Fig. 20, one can see that the SHALL approach outperforms the other three
approaches. The reason for this is that, since the length of each backup segment must be within a given threshold, a connection request will be rejected even if there is one backup segment protecting it cannot meet the hop limit for the LBP, SPP and CDR approaches. On the other hand, the SHALL approach will derive a shorter backup segment for these that do not meet the hop limit by using more spare capacity and thus reducing the locking probability.

Fig. 21 compares the average hop number of working path/segments. As shown, the average hop number of working path/segments for the SHALL approach is smaller than that of the SPP approach while larger than that of the LBP and CDR approaches. This can be explained as...
follows. Since the LBP approach derives a backup segment for each link in the working path and the CDR approach restricts the hop length of each active segment at most 2 hops, the lengths of their active segments are thus must be 1 and smaller than two, respectively. For the SHALL approach, however, it may be the case that a working path can be protected by a short backup path, and thus the average hop number of working path/segments is larger than those of the LBP and CDR approaches. Fig. 22 compares the average hop number of backup path/segments of the four approaches. Fig. 22 shows that the average hop number of working path/segments of the SHALL approach is the largest while that of the SPP is the smallest and these of the CDR and SHALL approaches lie between with that of the CDR is slightly smaller than that of the SHALL approach. This is due to the fact that we only calculate the average number of segment per lightpath of the connections accepted by the network while does not take these rejected connections into account.

Fig. 24 compares the control and management complexity of the four approaches under increasing traffic. As shown, the control and management complexity of the LBP approach is the largest while that of the SPP is the smallest and these of the CDR and SHALL approaches lie between with that of the CDR is slightly smaller than that of the SHALL approach. This is due to the fact that we only calculate the average number of segment per lightpath of the connections accepted by the network while does not take these rejected connections into account.

Fig. 25 compares the resource overbuild of the four protection approaches. Fig. 24 shows that the resource overbuild of the LBP approach is the largest followed by that of the SHALL approach with that of the SPP approach is the smallest. The main reason for this is that the SHALL approach can efficiently derive backup path/segments for an active path/segment that does not meet the hop limit. And this will consume more spare capacity than the CDR approach.

From the above results, one can easily make the conclusion that the SHALL approach can efficiently derive backup path/segments with guaranteed hop limit for each connection, if such backup path/segments does exist. While the control and management complexity and the resource overbuild of it remain at a modest level.

5.3.4. Further study under fluctuating traffic

We simulated fluctuating traffic to further study the performance of the four approaches. In each simulation under fluctuating traffic, we simulated 10,00,000 connections. While most results from fluctuating traffic are same to that of the increasing traffic, we only present these differences. Fig. 26 shows the blocking probability of the four protection approaches under fluctuating traffic when \( H = 6 \) and \( \epsilon = 0.01 \). We remark that other values of \( H \) and \( \epsilon \) led to same results and are not shown here. From Fig. 25, one can see that, in most cases, the blocking probability increases with the increase of traffic load. Similar to that of the increasing traffic, the blocking probability of the LBP scheme is the largest among them while that of the SHALL algorithm is the smallest and these of the SPP approach and CDR approach lie between. However, there are some exceptions. The first is that the blocking probability of the SPP approach firstly reduces with the increase of the traffic load. However, when the traffic load reaches a threshold (approximately 100 – 120 Erlang), the blocking probability increases with the increase of the traffic load. The reason for this is that when the traffic load is light, the backup path derived by the SPP approach may be very long (due to backup capacity sharing) and cannot meet the hop limit requirement. Thus, the connection will be dropped and leads to higher blocking probability. When traffic load gets heavy, however, connections are blocked...
mainly due to lack of spare capacity. For ease of understand, we decompose the blocking probability into two parts: these blocked due to lack of capacity and these due to beyond the hop limit. The results are shown in Fig. 27. The second one is that the blocking probability of the SHALL approach is smaller than that of the CDR approach under light traffic load while larger than that of the CDR approach under heavy traffic load. The reason for this is mainly due to the facts that (1) the CDR approach has a smaller resource overbuild than the SHALL approach; (2) for the CDR approach, connections blocked under light traffic load are due to longer hop length but not due to lack of spare capacity; and (3) in either case, connections blocked under heavy traffic load are mainly due to lacking of spare capacity.

Fig. 28 shows the average hop number of backup path/segments under fluctuating traffic when $H = 6$ and $e = 0.01$. Similar to that of the increasing traffic, the average hop number of backup path/segments of the SPP approach is the largest with that of the CDR approach is larger than that of the LBP approach. However, the average hop number of backup path/segments of the SHALL approach is the smallest under modest to heavy traffic. The reason for this is just the effect of limiting each backup path/segment at most $H$ hops.

Another difference between the results from fluctuating traffic and these from increasing traffic is the average control and management complexity, which is shown in Fig. 29. One can see that the control and management complexity of the LBP approach is the greatest followed by that of the CDR approach while that of the SPP approach is the smallest. This implies that the control and management complexity of the SHALL approach is modest (approximately 1.5).

6. Conclusions

In this paper, we investigate the problem of dynamic survivable lightpath provisioning against single-link failures in optical mesh network employing wavelength-division multiplexing (WDM).

We focus on the special problem of provisioning lightpath requests according to their differentiated protection-switching time, since lightpath may have different protection-switching time requirements. We propose a heuristic algorithm, namely Suurballe-based Heuristic Algorithm using Least number of segments for SSP with hop Limit (SHALL), to solve this problem efficiently.
We inspect the effects of hop limit and $e$ on various performance matrices. We found that the hop limit and $e$ have great effects on each performance matrix. Specifically, the larger the value of $e$, the larger the blocking probability and the longer the average hop number of working path/segments. However, the average hop number of backup path/segments and the average total hop number of working/backup path/segments decrease when $e$ increases. We also found that the blocking probability decreases with the increase of $H$, which implies that larger value of $H$ is preferable.

We have compared the SHALL approach with three other well-known protection approaches, namely shared path protection (SPP), shared link based protection (LBP) and cascaded diverse routing (CDR). Numerical results demonstrate that the SHALL approach outperforms its counterparts in blocking probability and protection-switching time with mirror decrease of spare capacity efficiency.

Acknowledgements

This work was supported in part by the National Science Foundation of China (NSFC) under contract No. 60302010. The authors are glad to thanks the anonymous reviewers for their valuable comments.

References


Hongbin Luo received his B.S. degree in 1999 from Beihang University and his M.S. degree with honor in Communications and Information Science in 2004 from the University of Electro-Optical and Communication Networks, China. He is currently working toward his Ph.D. degree in Key Lab of Broadband Optical Fiber Transmission and Communication Network, UESTC. His research interests lie in the areas of QoS routing, optical networks including network survivability, IP over WDM, and optical burst switching.

Hongfang Yu received her B.S. degree in Electrical Engineering in 1996 and her M.S. degree in Communication and Information Engineering in 1999 from University of Electro-Optical and Communication Networks, China. She currently studies at the National Key Lab of Broadband Optical Fiber Transmission and Communication Networks, UESTC as a teacher. Currently, she is a Ph.D. student there. Her research interest includes optical network survivability and traffic engineering etc.
Lenin Li was born in Zhejiang, China. He graduated from Jiaotong University, Shanghai, China in 1952, majoring in electrical engineering. From 1952 to 1956, he was with the Department of Electrical Communications at Jiaotong University. Since 1956 he has been with the University of Electronic Science and Technology. He has been a Visiting Scholar in the Department of Electrical Engineering and Computer Science at the University of California at San Diego, USA, doing research on digital and spread spectrum communications. He is currently a professor of the School of Communications and Information Engineering, UESTC. He is a member of the Chinese Academy of Engineering. His present research interests are in the areas of communication networks including broadband networks and wireless networks.