Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Communications 29 (2005) 59–68

computer communications

www.elsevier.com/locate/comcom

Improving signaling recovery in shared mesh optical networks

Chadi Assi[a,*], Wei Huo[a], Abdallah Shami[b], Nasir Ghani[c]

[a]*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada*
[b]*Department of Electrical and Computer Engineering, the University of Western Ontario, London, ON, Canada*
[c]*Department of Electrical and Computer Engineering, Tennessee Tech. University, USA*

Received 29 September 2004; revised 9 April 2005; accepted 11 April 2005
Available online 11 May 2005

Abstract

Current restoration signaling is a two-phase messaging procedure between the sender and the receiver and depends heavily on the message propagation delays during the recovery process and the cross-connect switching times. Offset time (OT) based restoration was proposed in [C. Assi, Y. Ye, S. Dixit, M. Ali. Control and management protocols for survivable optical mesh networks, IEEE Journal of Lightware Technology, November 2003.] to address the impact of these parameters and upper bound (OT-UB) expressions were derived. However, our closer investigation showed that as the network conditions change (e.g. increasing the number of wave-length channels per link) the benefits this restoration paradigm offer rapidly diminish, thereby rendering the argument of avoiding conventional restoration signaling inappropriate. In this paper, we propose a more accurate model to estimate the OT of each failed connection using a time-driven scheduling procedure. We also study the applicability of the proposed recovery protocol under double link failures assumption and we propose extensions to the protocol. We evaluate our proposal through simulation experiments and we show that substantial restoration gain can be achieved under varying network conditions.

Keywords: Optical networks; Shared mesh restoration; Protocols; Dual failures; Simulations

1. Introduction

Traditionally, ring based SONET networks offered 50 ms restoration time (RT) using pre-allocated protection capacity along pre-planned protection paths [1]. Recently, mesh based networks [2] have received much attention due to the increased flexibility they provide. Now, one of the key benefits of optical mesh networks is the improved bandwidth utilization due to the sharing of restoration resources across multiple failure-independent connections [3–6]. However, unlike automatic protection switching (APS) of SONET rings and dedicated protection of mesh networks where rapid network recovery upon failures is achieved, shared restoration exhibits increased recovery latencies [4,6]. Shared network restoration typically involves a common set of steps, including (1) backup path selection, (2) failure detection, (3) notification, and (4) signaling recovery protocols. Upon the detection of a failure and the receipt of a failure notification, the nodes responsible for initiating the recovery commence a signaling procedure to configure appropriate protection resources (e.g. wavelength and XC switches) for each of the failed connections [7–9]. Therefore, efficient network restoration schemes are required to eliminate or minimize the associated switching and signaling latencies.

Currently, a commonly used signaling mechanism is a two-phase process [7,10] (also called Round-Trip restoration); when the source node of a failed connection is notified of a failure, it sends a recovery message towards the destination along its designated backup route to configure the associated protection resources. The destination upon receiving this message will prepare an acknowledgement (ACK) and forwards back to notify the source of the successful setup of the backup route and finally the source node resumes its transmission. Obviously, the Round-Trip propagation delays of these recovery messages will have

* Corresponding author
E-mail addresses: assi@ciise.concordia.ca (C. Assi), w_huo@ciise.concordia.ca (W. Huo), achami@ieee.org (A. Shami), ghanin@tntech.edu (N. Ghani).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.04.005

a significant adverse impact on the network restoration time. Moreover when a fiber link is cut, a large number of connections may fail and their recovery is initiated simultaneously; therefore a surge of restoration messages may arrive at a node, each requesting the configuration of a particular cross-connect switch. The node receiving such requests, after processing these messages, will issue the appropriate commands to configure its switch fabric. Depending on the switch architecture, these commands may be handled either sequentially, in patch or parallel [9]. Clearly, significant queuing (processing) delays of recovery messages and switching waiting times along with message propagation delays may lead to longer recovery times.

The authors of [11] highlighted this problem and proposed restoration message aggregation to reduce the impact of message queuing delays and they also showed that a switch fabric with parallel command execution could substantially cope with the switching delays. It is important, however, to note that message aggregation may have some limitations due to the constraints imposed in aggregating messages. Moreover, the network parameters that severely impact the network recovery times the most are the restoration message propagation delays and the switch configuration waiting times when deploying switches with sequential configuration, which in essence message aggregation is not meant to resolve.

Since propagation delays are expected to have larger impact on service restoration, they could yield substantial gain in achieving fast network recovery if eliminated. Therefore, in this current work we propose to modify the two-phase signaling to eliminate the impact of the message propagation delays. A source node upon notification, it starts its recovery procedure and subsequently schedules (after some offset time, OT) the transmission of its data. Unlike the offset time proposed in [8], where only upper bound is derived for each connection, here we propose a time-driven scheduling procedure to accurately estimate the offset time of each failed connection. We demonstrate that the new restoration framework presents considerable improvement over its predecessor under varying network setting parameters and regardless of the switch fabric architecture. The paper is organized as follows. Section 2 gives an overview of OT-UB restoration. Subsequently in Section 3 we introduce the mechanism of proposed scheduling restoration scheme. In Section 4 we discuss the applicability of the proposed approach to situations with dual failures. Sections 5 and 6 are performance evaluation and conclusion.

2. Offset-time based upper bound restoration

In shared mesh networks, restoration resources are reserved at the time a connection is provisioned and they are configured upon the occurrence of a failure along the path of the connection. Because these resources are already reserved, contention between two (or more) connections attempting to restore is unlikely to occur. Therefore, unlike the Round-Trip restoration [10], the source does not need to wait for the receiver to acknowledge the successful configuration of the backup route. Instead, the source can compute the time it takes for the restoration path to be ready upon commencing the recovery process and offsets its transmission accordingly. This offset time must be selected such that any intermediate switch along the backup path must have its cross-connect configured prior to the arrival of data. Note that when a failure occurs, a large number of connections may fail; as the restoration procedure initiates, a large number of recovery messages may simultaneously arrive at some node(s) to configure protection resources. Therefore, a message may experience major queuing delays before it is processed. To successfully select the offset times, we proposed in [8] that the source node of each affected connection be informed by the total number of failed connections (N) and accordingly it assumes a worst-case delay in computing its OT (i.e. the message is always the last to be processed at each node along the restoration path). This will result in upper bound for the offset time (OT-UB); the OT-UB can be expressed by [8]:

$$T_{\mathrm{g}}^{1} = \sum_{i=0}^{n-1} (N \times T_{\mathrm{P}}) + T_{\mathrm{P}} + N \times T_{\mathrm{SC}}; \tag{1}$$

where T_{P} is the message processing time, T_{SC} is the switch configuration time. Here, $\sum_{i=0}^{n-1}(N \times T_{\mathrm{P}})$ represents the upper bound queuing delays a recovery message will experience along all nodes, except the destination, on the restoration path and $T_{\mathrm{P}}+N\times T_{\mathrm{SC}}$ represents the message processing time at the destination plus the upper bound waiting time (i.e. worst case) for the particular switch to be configured. For further details, please refer to [8]. The restoration time (*RT*), therefore becomes

$$\mathrm{RT} = T_{\mathrm{d}} + T_{\mathrm{N}} + T_{\mathrm{g}}^{1}; \tag{2}$$

where T_{d} is the failure detection time, and T_{N} is the failure notification time; n is the number of hops between source and destination. Clearly, this approach eliminates the impact of the message propagation delays from the overall RT. However, due to the fact that no exact information is available on the size of the surge of restoration messages arriving simultaneously at a node (herein, such a node is termed as a *conflict node*) other than N, the *RT* could become increasingly large as the network condition changes (e.g. increasing number of wavelengths or deploying switches with longer T_{SC} [7]) since

$$\frac{\partial T_{\mathrm{g}}^{1}}{T_{\mathrm{SC}}} \propto N; \tag{3}$$

this eventually could deteriorate the performance of OT-UB scheme. This behavior stems from the fact that the source node of a failed connection does not have enough information to accurately estimate the size of the surge of restoration messages arriving simultaneously at intermediate nodes(s) along its backup route.

3. A scheduling approach for rapid restoration

Due to the fact that no exact information is available on the size of the surge of restoration messages arriving at a node and competing for configuring their protection resources other than the total number of failed connections (N), larger offset times will be selected by each node; this will result in additional unnecessary delays that could significantly increase the network RT. Also note that some of the restoration messages sharing a conflict node may not necessarily be in conflict, depending on the inter-arrival time(s) of their arrivals at the conflict node (as explained later).

Therefore, we need some mechanism that can accurately model the sequence of arrival of restoration messages at conflicting nodes and decide upon that how much delay each message experiences before its switch is configured in other words, if we know how many recovery messages (and their order of their arrivals) may arrive simultaneously at any node along the backup route of a failed connection, then we can estimate the switching delays a connection may experience before its protection resources are configured. Accordingly, one can estimate a more accurate offset time than the upper bound. If there are more than one conflicting nodes for a failed connection, then the largest waiting time a message experiences at a conflicting node is used to compute the OT for the corresponding failed connection. Hence, the problem is reduced into a *scheduling problem* where the inputs are (1) the network topology; (2) the set of failed connections along with (3) their restoration routes; the output will be a *timetable* where each failed connection is associated with its OT such that no conflict can occur when they start the recovery. This algorithm is 'centralized' and will be executed by the node detecting the failure (i.e. upstream node of a failed link). We illustrate our approach through an example in Fig. 1. It shows a sample network with 4 backup connections (P_1-P_4) that need to be configured upon the failure of a link f (that is not shown in the figure). The numbers on the links represent the link propagation delays in milliseconds.

Given the delays on each link in the network, Ω_f can virtually trace the propagation of each recovery message for each failed connection throughout the network and determine the sequence of arrival of recovery messages at each intermediate node as well as the likelihood of conflict a message may have with other recovery messages for other connections. One requirement, however, is that Ω_f must maintain a knowledge of the protection routes of all connections route through link f; this information is made available to the node during the provisioning phase. We note here that maintaining this information does not pose any scalability problem since only information about connections routed through this node are to be maintained and not all connections currently in the network. The amount of this information can at most be $O(W\cdot R)$ where W is the number of wavelengths per link and R is the nodal degree (number of edges leaving this node) of this current node.

Fig. 2 shows a time sequence of the flow of the configuration messages, as determined by Ω_f, where $P(i, j)$ represents the arrival of a recovery message for connection i at time j. We assume in this example that the notification times T_N of P_1-P_4 are 0, 0.5, 3, and 1 ms, respectively.

Upon following the flow of these messages, it becomes easy to identify which other recovery messages a particular message (of a particular failed connection) is contending with. For example, the recovery message for P_1 is the first message arriving at node A, and it is the second message arriving at nodes C and E. Therefore under worst-case assumption, it will at most be the second message to be processed along any node on its protection route and there will at most be one message waiting ahead before it is processed. Hence, its offset time parameter should be 2 whereas using the upper bound [8] scheme, the offset time is computed based on $N=4$ (i.e. 4 failed connections).

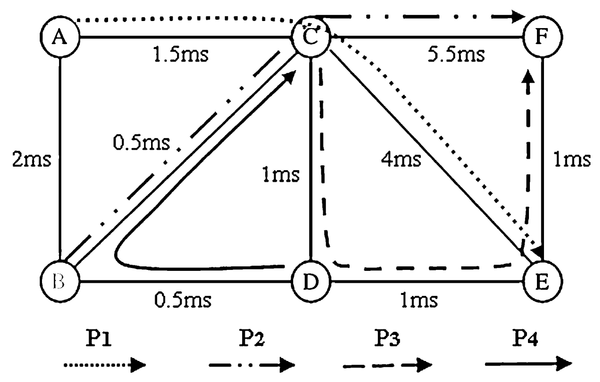

Fig. 1. Illustrative example—network topology.

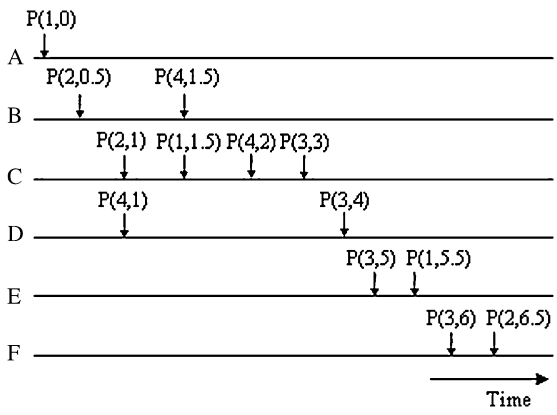

Fig. 2. Illustrative example—sequence of messages.

We now formally explain the scheduling procedure, but first we make the following observations. The main delays we are trying to minimize in this framework are the propagation delays and the switching times as the message processing delays are considered smaller in comparison with the switching times ($T_P \ll T_{SC}$). We start by giving some variable definitions:

N Total number of working connections going through f;
$W_f = \{w_1, w_2, \ldots, w_N\}$ The set of failed working connections going through failed link f;
s_{w_i} Source node of connection w_i;
$T_N^{w_i}$ Time it takes to notify the source of connection;
L_{w_i} The restoration parameter of connection;
$L_{w_i}^j$ The restoration parameter of connection w_i estimated at node j along its backup path;
$t_{w_i}^j$ Arrival time of recovery message for connection w_i at node j;
$\Delta(j-1, j)$ Propagation delay between node j and its upstream node. So, $t_{w_i}^j = t_{w_i}^{j-1} + N \times T_P + \Delta(j-1, j)$;
T_d Failure detection time;
$C_{id}^{w_i}$ Unique identification for connection;
Φ_j Timetable of node j, which is initially empty. Each item of the timetable has two elements: $\{C_{id}^{w_i}, t_{w_i}^j\}$;
n_{w_i} The number of nodes along the restoration route for connection;
T_{g,w_i}^2 Offset time for connection.

Algorithm 1. Pseudo code of the Scheduling Procedure

```
for each failed connection $w_i$; $(i=1,\ldots,N)$ do
    Compute its failure notification time $T_N^{w_i}$;
    for each node $j$ along the backup path for $w_i$ do
        Compute $t_{w_i}^j = t_{w_i}^{j-1} + N \times T_P + \Delta(j-1, j)$;
        Insert $\{C_{id}^{w_i}, t_{w_i}^j\}$ into $\Phi_j$;
        Sort $\Phi_j$ based on $t_{w_i}^j$;
        Compute $L_{w_i}^j$ (which is also the position of $t_{w_i}^j$ in $\Phi_j$)
    end for
end for
for each $w_i$ do
    Compute $L_{w_i} = \max_{k=1,2,\ldots,n_{w_i}}\{L_{w_i}^k\}$
    Compute
    $T_{g,w_i}^2 = (n_{w_i} - 1) \times L_{w_i} \times T_P + T_P + L_{w_i} \times T_{SC}$
end for
```

The computational complexity of Algorithm 1 is $O(L \cdot N^2 \cdot \log N)$, where N is total number of working connections going through failed link f, L is the average length of backup paths.

Upon the failure of link f, Ω_f (the upstream node of the failed link) will run a scheduling algorithm (Algorithm 1) to evaluate the OT for each failed connection. Subsequently, a notification message is sent to the source node of each connection along with the computed OT to start its recovery procedure and accordingly it schedules (offsets) the restoration of its data.

Note that, here the notification time ($T_N^{w_i}$) for a connection w_i is computed as follows:

$$T_N^{w_i} = \sum_{j=\Omega_f}^{j-1=S_{w_i}} \Delta(j-1, j) + N \times T_P \times m_{w_i}; \qquad (4)$$

where the first component in the equation accounts for the propagation delays of the notification message between Ω_f and S_{w_i}, and the second part of the expression represents the total processing time of a message at a node along the notification path times the total number of hops along the notification path, m_{w_i}. Here, worst case processing delays ($N \times T_P$) are considered for a notification message as it propagates along the notification path; the reason is that notification messages for different failed connections may also contend for processing along notification route. Since T_P is typically small (few μ seconds), this upper bound processing delays will not affect the performance of the recovery protocol. Finally, since the computed notification time is an upper bound expression that is used in the OT computation, it will be slightly larger than the actual notification time. Therefore, the source node upon receiving the notification message (at $T_{N,\text{actural}}^{w_i}$) it will wait ($T_N^{w_i} - T_{N,\text{actural}}^{w_i}$) before it commences its recovery, in order to avoid causing any perturbation to the computed offset times.

We note here that this scheduling algorithm provides a good approximation on the waiting time of XC switch request by estimating the number of requests waiting ahead of a recovery message. However, it ignores the inter-arrival time between two consecutive requests. After identifying the sequence of arrivals at some node, one can further check whether the inter-arrival time of two consecutive messages is larger than the T_{SC}. If so, the switch of the second request can be directly configured without any delay, since the configuration of the previous request has already terminated. This means that some messages may arrive consecutively at a conflicting node, but actually they may or may not be in conflict depending on their inter-arrival time. Otherwise, the second message will need to wait for some period of time until all requests waiting ahead are processed. As before, all conflicting nodes need to be considered, and to determine the offset time we choose the node with maximal waiting time for the request to be processed. As an example, let $\delta 1$ and $\delta 2$ be the inter-arrival time of recovery messages between (P_1, P_2) and (P_1, P_3) at nodes C and E, respectively (Fig. 2). If $(\delta 1 < T_{SC}) \wedge (\delta 2 < T_{SC})$, then the recovery message for connection P_1 does not contend with any of the other messages and therefore smaller OT can be achieved. Obviously, this optimized approach will further complicate the scheduling procedure, so we do not discuss it any further here but we present its evaluation in Section 5.

4. Double link failure recovery

As single link failures are common failure scenarios, normally recovering from these failures is completed within

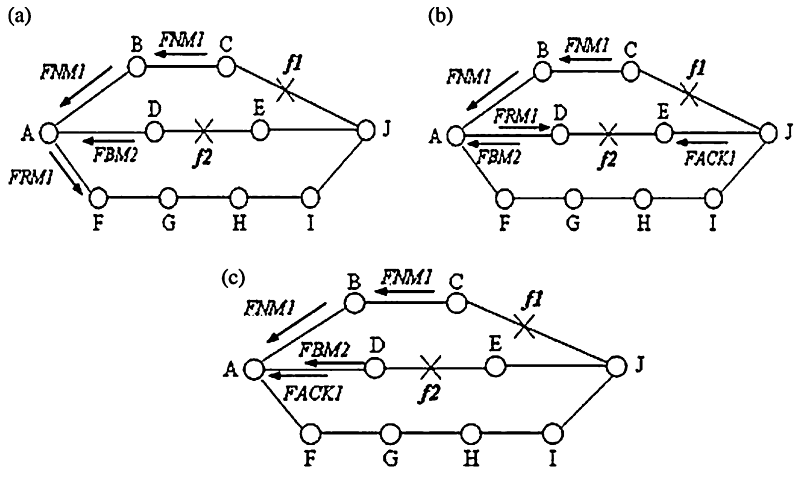

Fig. 3. Recovery from double-link failures.

few milliseconds to few seconds. However the time it takes to repair a link may be few hours to few days [1] and it is likely that a second failure occurs during this period, causing two links to be down simultaneously. Previous research has addressed the problem of routing connections under dual failure assumptions [12,13] where extra protection capacity is pre-planned in the network. In this section we are only interested in the applicability of the proposed restoration framework under double failure assumptions. We assume, as in [12], each connection is protected by two link-disjoint backup (primary and secondary) paths. We further assume that both failures occur near-simultaneously, that is the second occurs while the network is recovering from the first failure; otherwise, recovery from both failures is treated independently; i.e. similar to single link failure.

Upon detecting a failure, the upstream node (Ω_{f_k}) of the failed link (f_k, $k=1$, 2) will send a notification message (*FNM*(f_k)) to the source node of each failed connection and simultaneously it broadcasts a failure message (*FBM*(f_k)) to all nodes in the network. We assume two links can fail simultaneously in any arbitrary order. We let f_1 and f_2 be the first and second failure, respectively and G_1 and G_2 be the two groups of working connections to be restored upon the failures of f_1 and f_2. We also define: $b_1^{i,1}$, and $b_1^{i,2}$: the primary and secondary backup paths, respectively for a connection $w_i^{G_1}$ in G_1; $b_2^{i,1}$, and $b_2^{i,2}$: the primary and secondary backup paths, respectively for a connection $w_i^{G_2}$ in G_2.

4.1. Conventional recovery procedure, case I

We start by first studying the conventional (Round-Trip) restoration scheme under dual failures. There are two possible scenarios to be considered:

A. If the second failure does not impact the first primary backup ($b_1^{i,1}$) of the failed connection ($w_i^{G_1}$) then the recovery of $w_i^{G_1}$ proceeds on $b_1^{i,1}$ (regardless of the order of arrival of *FBM*(f_2) to the source node of $w_i^{G_1}$).

B. Otherwise (i.e. the second failure affects $b_1^{i,1}$), three different cases should be considered according to the arrival time of *FBM*(f_2) at the source node of $w_i^{G_1}$ (we assume $w_i^{G_1}$, $b_1^{i,1}$ and $b_1^{i,2}$ routed through [A-B-C-J], [A-D-E-J] and [A-F-G-H-I-J] accordingly, Fig. 3.):
   (i) Node A receives *FNM*(f_1) after *FBM*(f_2): the source node restores the connection by sending a failure recovery message (*FRM*$_1$) along $b_1^{i,2}$ (see Fig. 3a).
   (ii) The second failure (f_2) occurs while attempting a recovery along $b_1^{i,1}$: the source node will receive *FBM*(f_2) shortly after sending FRM1 along $b_1^{i,1}$. In this case, the source upon receiving *FBM*(f_2) attempts a new recovery along $b_1^{i,2}$ and any reserved resources along $b_1^{i,1}$ will be released (Fig. 3b).
   (iii) The second failure occurs shortly after setting up $b_1^{i,1}$: $b_1^{i,1}$ will be considered as a new connection and its source node is notified and restoration takes place along $b_1^{i,2}$ (Fig. 3c).

4.2. Scheduled recovery procedure, case II

By contrast, the proposed scheduling-based restoration scheme exhibits different behavior. Here when a link fails, its upstream node will compute the offset time ($T^2_{g,w_i^{G_k}}$, Algorithm 1) for each failed connection $w_i^{G_k}$ in the corresponding failed group (G_k, $k=1$, 2). It will also determine the time at which all connections in this group are expected to complete recovery (that is the maximal restoration time, $RT_{\mathrm{MAX}}^{G_k}$, of the group).

4.2.1. Derivation of $RT_{MAX}^{G_k}$

Let ψ be the number of ordered recovery messages arriving at node j for connections $w_i^{G_k}, w_{i+1}^{G_k}, \ldots, w_{i+\psi}^{G_k}$ and let $t_{i+\psi}^{j}$ the arrival time of the recovery message of the last connection, that is $w_{i+\psi}^{G_k}$. The time at which the switch for this last connection is configured is t^j_{finish}:

$$t^j_{\text{finish}} = t^j_{i+\psi} + L^j_{w^{G_k}_{i+\psi}} *(T_P + T_{SC}); \tag{5}$$

where $L^j_{w^{G_k}_{i+\psi}}$ is the restoration parameter of connection $w^{G_k}_{i+\psi}$ at node j, and it is derived in Section 3. The time at which group G_k completes its recovery can then be computed as follows:

$$RT^{G_k}_{\text{MAX}} = \max_{x\in\Delta}(t^j_{\text{finish}}); \tag{6}$$

where Δ is the set of all nodes involved in the recovery (see Algorithm 1)

A notification message (*FNM*($T^2_{g,w^{G_k}_{i+\psi}}$, $RT^{G_k}_{\text{MAX}}$, N_{G_k}, $T^{G_k}_{\text{d}}$, f_k)) will be then sent to the source node of a failed connection where N_{G_k} is the total number of failed connections in group G_k, $k=1, 2$, $T^{G_k}_d$ is the failure detection time of that same group and f_k is the identity of the failed link. Also, the upstream node of the failed link will broadcast a failure message *FBM*($RT^{G_k}_{\text{MAX}}$, N_{G_k}, $T^{G_k}_{\text{d}}$, f_k), $k=$ 1,2 in the network to all nodes.

As stated previously in Section 3, the scheduling-based restoration procedure achieves successful recovery by accurately estimating the size of the surge of restoration messages arriving simultaneously at a *conflict* node(s) along the backup paths and computing the offset times accordingly (Algorithm 1). Now if a second failure occurs, recovery of one group of failed connections will interfere with the recovery of the second group, potentially creating new conflict nodes and/or increasing (changing) the size of the surge of restoration messages arriving at conflict nodes. As a result, the offset times computed for connections in both groups will yield erroneous recovery. One simple solution is to allow only one group (G_1) at a time to recover while the second group (G_2) will have the restoration of its connections shifted until connections in G_2 entirely complete their recovery.

However, some connection(s) in G_2 may have started their recovery[1] as soon as the source node(s) receive failure notification (*FNM*(f_2)) given that *FBM*(f_1) from G_1 has not yet been received. In this case, a node along a protection route may receive recovery messages from connections in both groups and accordingly these messages must be segregated by that node (based on the detection time of the corresponding failure, which is transmitted along with the message) so that recovery messages from G_2 are processed upon the completion of all recovery messages for connections in G_2 at that node.

Therefore, to ensure a proper contention-free restoration of connections $w^{G_2}_i$, their data recovery is *shifted* (see the derivation in the appendix of the shift time) accordingly. This shift is intended only for the scheduling of transmission of backup data; i.e. while recovery is triggered immediately by the source node of a failed connection in G_2, the backup traffic is transmitted only when resources are successfully configured.

4.2.2. Derivation of $t^{G_2}_{shift,w^{G_2}_i}$

There are two possible different scenarios depending on which message (*FBM* or *FNM*) is received first at the source node of $w^{G_2}_i$:

A. If *FBM*(f_1) is received first then *FNM*(f_2) and if $t^{w^{G_2}_i}_{f_2}$ is the arrival time of *FNM*(f_2) at the source node of $w^{G_2}_i$:

$$t^{G_2}_{\text{shift},w^{G_2}_i} = \begin{cases} RT^{G_1}_{\text{MAX}} - t^{w^{G_2}_i}_{f_2} & \text{if } RT^{G_1}_{MAX} > t^{w^{G_2}_i}_{f_2}; \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

B. If *FBM*(f_1) is received after receiving *FNM*(f_2) and $t^{w^{G_2}_i}_{f_2}$ is the arrival time of *FBM*(f_1) at the source node of $w^{G_2}_i$:

$$t^{G_2}_{\text{shift},w^{G_2}_i} = \begin{cases} RT^{G_1}_{\text{MAX}} - t^{w^{G_1}_i}_{f_1} & \text{if} RT^{G_1}_{\text{MAX}} > t^{w^{G_1}_i}_{f_1}; \\ 0 & \text{if} RT^{G_1}_{\text{MAX}} < t^{w^{G_1}_i}_{f_1} \text{ or} RT^{G_1}_{\text{MAX}} < t^{w^{G_2}_i}_{f_2}. \end{cases} \tag{8}$$

Now, because of the random nature of the failures and their occurrence in the network, notifications of the first failure may arrive *after* the broadcast failure messages of the second; moreover, some connections in G_1 may have their primary backup routes affected by the second failure. In this case, those connections will be restored on their second backup routes. However, the source nodes of these failed connections (connections in G_1 whose backup paths are affected by the second failure) have no further information to appropriately schedule their restoration. Moreover, the restoration of these connections should not affect or cause any perturbation to the already scheduled connections by creating new conflict nodes or adding new messages (altering the size of the surge of messages previously estimated) to existing conflict nodes.

For this reason, we propose that connections unaffected by the second failure to be restored using the computed offset times (i.e. using the method of Section 3). However, failed connections in G_1 whose primary backup routes are affected by the second failure will be restored only upon complete recovery of all other connections in G_1 and G_2. The new offset times for these connections are adjusted locally at their source nodes to $T^1_{\text{g}}(N_{G_1}) + T^1_{\text{g}}(N_{G_2})^2$ (after a source node had received both *FNM*(f_1) and *FBM*(f_2)). This upper bound offset time will guarantee a contention free and a successful recovery for these connections.

[1] By recovery here we mean the configuration of restoration resources, not the actual transmission of backup data.

[2] $T^1_{\text{g}}(N_{G_k}) = \sum_{i=0}^{n-1}(N_{G_k} * T^i_{\text{p}}) + T_{\text{p}} + N_{G_k} * T_{\text{SC}}$

Below we describe the complete simple steps used for double failure recovery:

1. For $w_i^{G_1}$:
   A. If the source node receives both messages, but *FBM*(f_2) prior to *FNM*(f_1):
      (i) If $b_1^{i,1}$ is affected by f_2, then the source node cannot use $T^2_{\mathrm{g},w_i^{G_1}}$ as offset time. Rather, the backup traffic is restored on $b_1^{i,1}$ and the offset time to be used is $T^1_{\mathrm{g}}(N_{G_1}) + T^1_{\mathrm{g}}(N_{G_2})$.
      (ii) Otherwise, $T^2_{\mathrm{g},w_i^{G_1}}$ is used to offset the retransmission of backup data along $b_1^{i,1}$.
   B. Otherwise, (i.e. the source node of $w_i^{G_1}$ has received *FNM*(f_1) but not *FBM*(f_2)), therefore it triggers the recovery procedure of its failed connection(s) using the computed offset time(s), irrespective of whether another failure had occurred or not (since no *FBM* message had been received). Later, when *FBM*(f_2) is received:
      (i) If $b_1^{i,1}$ is affected by f_2, then the source node initiates a new recovery along $b_1^{i,1}$ and it uses a new offset time $T^1_{\mathrm{g}}(N_{G_1}) + T^1_{\mathrm{g}}(N_{G_2})$ to restore its traffic. Any reserved resources along $b_1^{i,1}$ will be released.
      (ii) Otherwise, ignore the received message.

2 For $w_i^{G_2}$:
   A. If the source node receives both messages, but *FBM*(f_1) prior to *FNM*(f_2):
      (i) If $b_1^{i,1}$ is affected by f_1, the source node commences recovery on $b_1^{i,1}$ and schedules its backup retransmission using the upper bound offset time, T^1_g (NOTE: here, the computed offset time ($T^2_{\mathrm{g},w_i^{G_2}}$) cannot be used along $b_2^{i,2}$ since it was computed for restoration along $b_2^{i,2}$, however, the upper bound will guarantee successful recovery as it assumes worst case delays among connections in G_2) after shifting by $t^{G_2}_{\mathrm{shift},w_i^{G_2}}$ (to guarantee that G_1 has completed its recovery). Therefore, the new offset time for $w_i^{G_2}$ is $t^{G_2}_{\mathrm{shift},w_i^{G_2}} + T^1_{\mathrm{g}}(N_{G_2})$.
      (ii) otherwise, $w_i^{G,2}$ is restored on $b_2^{i,1}$ and the new offset time is: $t^{G_2}_{\mathrm{shift},w_i^{G_2}} + T^2_{\mathrm{g},w_i^{G_2}}$.
   B. Otherwise, (i.e. the source node of $w_i^{G,2}$ receives *FNM*(f_2)), it immediately starts its recovery (as no information is available about the first failure) along its primary backup path $b_2^{i,1}$. Upon receiving *FBM*(f_1):
      (i) If $b_2^{i,1}$ is not affected by f_1, it schedules its backup data retransmission using a new offset time: $t^{G_2}_{\mathrm{shift},w_i^{G_2}} + T^1_{\mathrm{g}}(N_{G_2})$.
      (ii) Otherwise, it will release any reserved resources along $b_2^{i,1}$ and restarts its recovery along $b_2^{i,2}$ and reschedules its backup data retransmission using a new offset time: $t^{G_2}_{\mathrm{shift},w_i^{G_2}} + T^1_{\mathrm{g}}(N_{G_2})$.

5. Performance evaluation

We evaluate the performance of the proposed recovery scheme and we compare it with the conventional two-way messaging restoration procedure. The 16-node NSF network [2] is used in our simulation study. The metric for the performance evaluation is the network restoration time. An event-driven simulation tool is developed to model the distributed provisioning and recovery protocol. We assume that each link consists of two unidirectional fibers. Connections requests arrive as a Poisson process and the connection-holding time follows a negative exponential distribution with mean $1/\mu = 100$ ms. The Random Wavelength Selection Algorithm is used to select the candidate wavelength for setting up connections. We simulate the failure of one (two) unidirectional link(s) after running the simulation for some period and satisfying some traffic demand.

5.1. Performance under one single link failure

We start by comparing the two-phase conventional restoration and the OT-UB schemes; Fig. 4 shows the performance of the two restoration protocols under different setting parameters. Clearly, when the number of wavelength is smaller and/or the switch configuration time is shorter, the OT-UB scheme outperforms the two-phase messaging approach (as the impact of *N*, number of failed connections, is minimal). But as the switch configuration time increases the performance of the OT-UB degrades as more switching delays start to have greater impact on the restoration times. On the other hand, the conventional two-way messaging is slightly affected since the round trip propagation delays are the major factor affecting its performance. The reason that the OT-UB degrades faster (i.e. larger slope) is due to the fact that the scheme considers all failed connections (which may not necessarily be in contention state) and results in

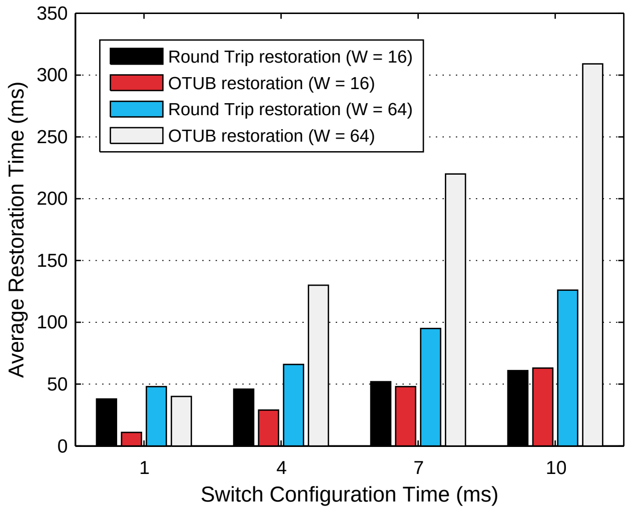

Fig. 4. Network *RT* vs. T_{SC}.

overestimating the offset times and hence the network recovery time. As T_{SC} increases, the overestimated value grows linearly and unnecessarily. The impact is clear when we increase the number of wavelengths, as potentially more connections will fail (N), and again the overestimation further diminishes the performance of OT-UB rendering the argument of avoiding the conventional two-phase signaling inappropriate.

Clearly, overestimating the number of restoration messages arriving at conflict nodes results in performance degradation. Therefore, we proposed a scheduling scheme to provide a closer approximation of the size of the contending messages and modeled the sequence of their arrival at intermediate nodes along the restoration routes. Fig. 5 shows a comparison between the conventional two-phase restoration and the scheduling-based restoration for 64 wavelengths.

Unlike the upper bound scheme, by deploying scheduling the node detecting the failure can accurately approximate the OT of each failed connection upon identifying their arrival sequence at conflict nodes. As the figure shows, substantial improvement can be achieved over the OT-UB by avoiding the use of upper bounds for the OT, whereas a moderate improvement (30–15 ms) can be achieved over the conventional two-way messaging approach. The figure also shows that as T_{SC} increases and/or the number of wavelengths increases, the new scheduling recovery framework is not affected and its slope is consistent with that of the two-way messaging. This suggests that we modeled quite accurately the size of the surge of messages contending at conflict nodes while also eliminating the round trip delays.

As we mentioned earlier, some messages may arrive consecutively at a conflicting node and depending on their inter-arrival time, they may or may not be in conflict. We proposed a more optimal scheduling scheme to account for these cases, and here we show its performance. Fig. 6 shows the comparison between the conventional signaling and the scheduling based restoration. Clearly, by tracing the inter-arrival time of restoration messages we are able to better and more accurately estimate the number of conflicting messages and therefore determine exactly the OT a connection needs to wait before it restores. Here, an improvement of 35–20 ms is achieved over the two-phase signaling.

We can also study the impact of the number of wavelengths on network recovery times. Fig. 7 shows the network restoration times for different wavelengths when $T_{SC}=2$ ms. It is interesting to see that the multiplier of T_{SC}

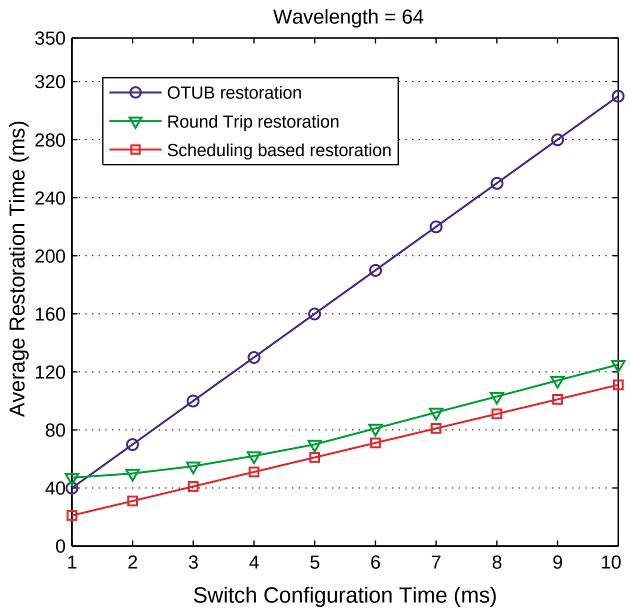

Fig. 5. Network RT vs. T_{SC}, $W=64$.

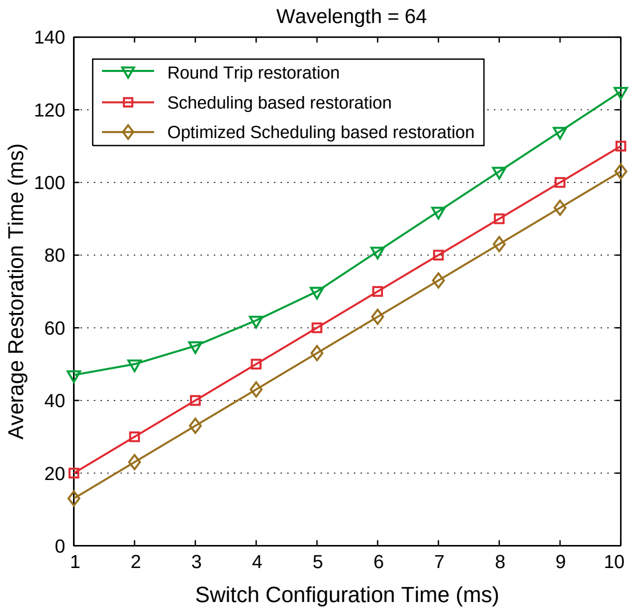

Fig. 6. Network RT vs. T_{SC}, $W=64$.

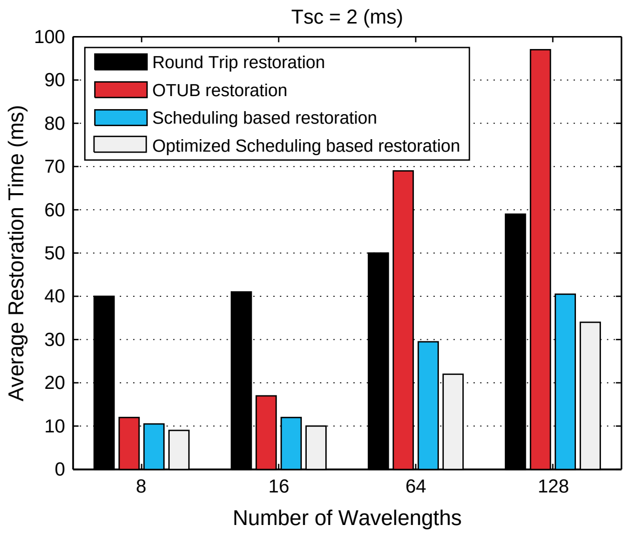

Fig. 7. RT vs. number of wavelengths, $T_{SC}=2$ ms.

in the expression of the OT for the OT-UB will have larger impact on the network restoration time especially as W increases. This reveals again the negative impact of the overestimation that OT-UB yields over the other schemes. Whereas the proposed scheduling scheme shows linear increase that is consistent with the conventional messaging protocol, which backs our argument in saying that the proposed restoration parameter accurately models the surge of restoration messages flow. The figure shows that when W is small, even the OT-UB scheme performs better than the conventional two-way messaging. However, its performance (i.e. the OT-UB) greatly depends on the network parameters and the figures show that as W increases its actual negative behavior becomes apparent. Note that the new proposed approach will exhibit good perform under varying network conditions.

We also study the impact of the switch architecture on the network recovery times. Namely, we compare the two-way messaging procedure and the scheduling-based restoration for both the consecutive and the parallel switching fabric [9]. Fig. 8 shows the simulation comparison. Clearly, parallel switch architecture eliminates the switching delays and therefore results in better recovery times. The only delays this architecture incurs are the fault notification delays and the round-trip delays of the two-way messaging recovery. The latter delay is clearly eliminated when scheduling based restoration is deployed and in comparison with the two-way recovery scheme, restoration times of 10–20 ms can be achieved.

Finally, we study the impact of the propagation delays on the restoration procedure. It is clear now that the conventional scheme relies on the two-way messaging between the sender and the receiver to ensure that the restoration path has successfully been setup. As such, it suffers from the network propagation delays, whereas the other proposed schemes eliminate completely the impact of these delays. Now to study the impact of the propagation delays we use the NSF network with virtual link distances; a new multiplier coefficient is introduced to linearly increase the distances between two adjacent nodes; e.g. a coefficient of 2 will double the links distances. Fig. 9 shows the simulation results when the number of wavelengths is 64 and $T_{SC}=3$ ms. Clearly, as the distances increase, the propagation delays increase and the restoration time of the conventional two-way signaling substantially increases

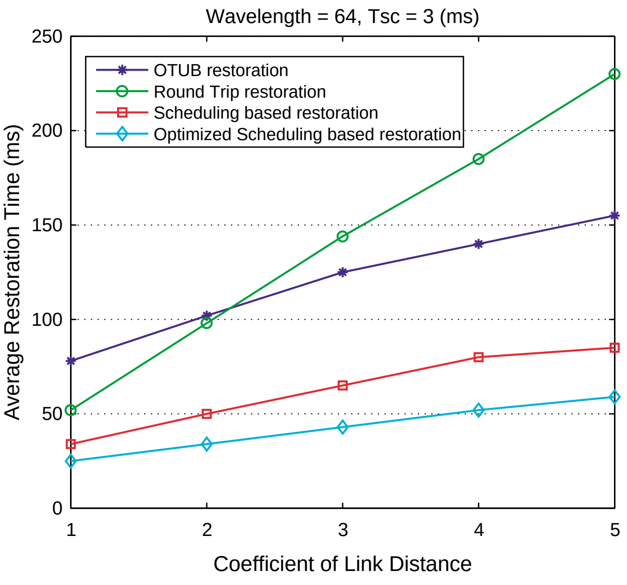

Fig. 9. *RT* vs. link distance coefficient (consecutive).

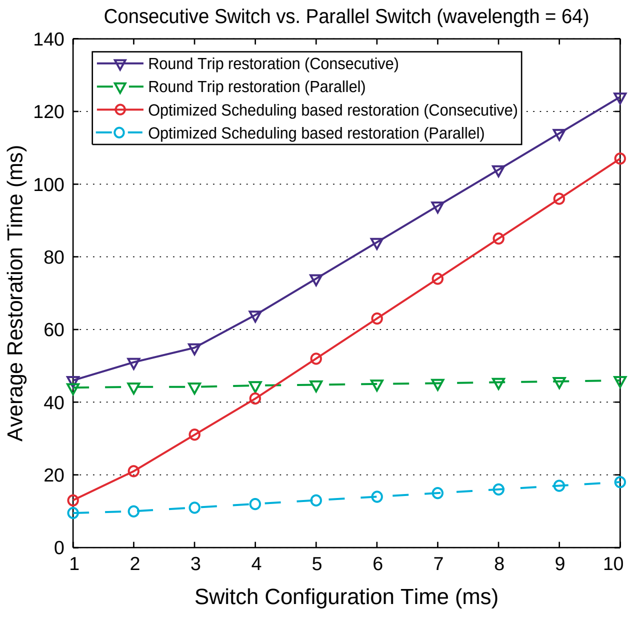

Fig. 8. Consecutive vs. parallel comparison.

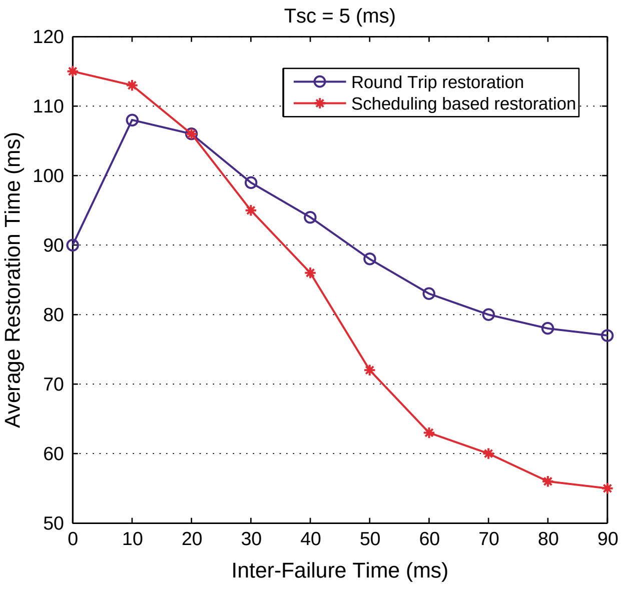

Fig. 10. *RT* vs. inter-failure time (β).

whereas this effect on the scheduling restoration scheme is negligibly small.

5.2. Recovery under double link failures

In this section we evaluate and compare the effectiveness of the Round-Trip restoration and the optimized scheduling based restoration under the double-link failure assumptions. We let be the inter-failure time of the two consecutive failures. There is an interesting behavior for the Round-Trip scheme (Fig. 10). When β is small, the *RT* increases; this is justified by the fact that while connections in G_1 are being restored, the second failure occurs and affects some of those connections before completing their recovery. Therefore, they stop their recovery on their first backup routes and schedule a new recovery on their second backup routes. Now as β slowly increases, some of the connections in G_1 will be given more time to do their recovery but not enough to complete it; when the second failure occurs, these connections will be restored along their second backup paths and hence there is an increase in the *RT*. When the inter-arrival of both failures is large, some of the connections from G_1 affected by the second failure will have enough time to completely finish the recovery and therefore, when the second failure occurs, they will be recovered as new connections in G_2. Hence, we notice a decay in the restoration time as β increases (note that the *RT* now does not account for the attempt being made along the first primary backup).

Alternatively, scheduling recovery exhibits slightly different behavior. First, some connections in G_1 are affected by f_2 and they are recovered upon the completion of all connections in G_1 and G_2. Second, when β is small, $RT^{G_1}_{\mathrm{MAX}} - t_{f_2}^{w_i^{G_2}}$ is large and therefore, the shift time for $w_i^{G_2}$ is large; the *RT* is large. As β increases, subsequently, connections in G_2 will be shifted only for a small period. Therefore, the restoration time decreases as the inter-failure arrival time increases. Note that the Round-Trip scheme exhibits better performance than the scheduling scheme when is small; that is due to the fact that when both failures occur near-simultaneously, the scheduling scheme (1) recovers using OT-UB and moreover, (2) the shift period for connections in G_2 is bigger.

6. Conclusion

In this paper, we addressed a fundamental problem for designing a survivable optical transport networks; that is the capability to quickly recover from element failures. By using the framework of Offset-Time and a time-driven scheduling procedure, we proposed an accurate model to estimate the offset time of each failed connection. Comparing with the Round-Trip restoration and OT-UB restoration, we showed that our proposal eliminates the propagation delays and the accumulation of switching delays, and it achieves the best recovery performance upon the single link failure. We also studied the applicability of the proposed recovery protocol under double link failures assumption. Through detailed simulation experiments, we showed that by deploying this scheduling scheme, substantial restoration gain can be achieved under varying network conditions.

References

[1] W. Grover, Mesh-based Survivable Networks: Optical and Strategies for Optical, MPLS, SONET and ATM Networking, Prentice Hall, 2003.
[2] B. Mukherjee, Optical Communication Networks, McGraw-Hill, 1997.
[3] S. Ramamurthy, B. Mukherjee, Survivable WDM Mesh Networks, Part II—Restoration, IEEE ICC 1999.
[4] J.-F. Labourdette, Shared Mesh Restoration in Optical Networks, Proc. OFC’04 2004.
[5] G. Li, J. Yates, D. Wang, C. Kalmanek, Control plane design for reliable optical networks, IEEE Commun. Mag. 40 (2) (2002) 90–96.
[6] J. Yates, G. Li, Challenges in Intelligent Transport Network Restoration, Optical Fiber Communications Conference, invited paper, March 2003.
[7] M. Goyal, G. Li, J. Yates, Shared mesh restoration: a simulation study, IEEE OFC’02 2002.
[8] C. Assi, Y. Ye, S. Dixit, M. Ali, Control and management protocols for survivable optical mesh networks, IEEE J. Lightware Technol. 2003.
[9] R. Doverspike, G. Sahin, J.L. Strand, R.W. Tkach, Fast restoration in a mesh network of optical cross-connects, Proc. OFC’99 1999.
[10] J.P. Lang, et al., Generalized MPLS recovery functional specification, Internet Draft-Work in Progress, 2002.
[11] M. Goyal, J. Yates, G. Li, W. Feng, Benefit of restoration signaling message aggregation, Proc. OFC’03 2003.
[12] W. He, A. Somani, Path-based protection for survivable double-link failures in mesh-restorable optical networks, Proc. IEEE GlobeCom 2003.
[13] H. Choi, S. Subramaniam, H.A. Choi, On double-link failure recovery in WDM optical networks, IEEE INFOCOM 2002; 808–816.