Restorable Logical Topology in the Face of No or Partial Traffic Demand Knowledge

Reuven Cohen Gabi Nakibly
Technion – Israel Institute of Technology
Computer Science
Haifa, Israel

***Abstract*—The construction of a logical network on top of a physical (optical) infrastructure involves two intertwined tasks: logical link selection – deciding which pairs of routers will be connected by logical links (lightpaths), and logical link routing – deciding how to route each logical link across the optical network. The operator of such networks is often required to maximize the available throughput while guaranteeing its restorability. This paper is the first to combine these seemingly conflicting goals into one optimization criterion: maximizing the restorable throughput of the end-to-end paths. We address this problem in three cases: when the operator has no knowledge of the future bandwidth demands, when it has partial knowledge, and when it has full knowledge. We present efficient algorithms for each of these cases and use extensive simulations to compare their performance.**

I. Introduction

Modern communication networks consist of a logical topology overlaid on an optical physical infrastructure. Distinguishing between the logical and physical networks is crucial to flexibility and efficiency. However, this distinction gives rise to important cross-layer optimization issues, such as how to guarantee smooth restoration following a failure in the physical network. In this work we study the problem of designing a restorable logical network, which continues to operate efficiently after a physical failure. The input to this problem is a physical (optical) network, which consists of optical switches connected by fiber optic links. Only a subset of those switches has the capability to serve as routers. The logical network that we build consists of routers connected by lightpaths. Each lightpath is established over one or more optical fibers and the optical switches connecting these fibers. The constructed logical network should accommodate the traffic demands not only when all the physical components are operational, but also in the face of a physical failure.

The construction of a logical network is composed of two intertwined tasks: deciding which pairs of routers will be connected by logical links (lightpaths) and deciding how to route each logical link across the optical network. These two tasks are referred to as link selection and link routing, respectively. When setting up optical lightpaths as the links of the logical network, the dominating cost is of the transponders at the two ends of every lightpath, which convert optical to electronic signals and vice versa [11]. Therefore, building a logical network is always subject to a budget constraint, which is translated into an upper bound on the number of lightpaths (logical links) that can be established.

978–1–4799–3360–0/14/$31.00 ©2014 IEEE

Most past works on designing logical networks assume that the logical links are given, and focus on the link routing task. In contrast, we solve the two tasks together, because they have a tremendous impact on each other. To better understand the link selection and routing problem, consider Figure 1(a). This figure shows a physical network with 16 optical switches connected by 24 optical links. Assuming that only nodes a, b, c and d have the capability to serve as routers, logical links can be established only between pairs of these nodes. Figure 1(b) shows a possible logical network with 4 lightpaths (logical links). This logical network is not resilient because a failure of one node (i or c) or a failure of one logical link ($c-i$ or $i-d$) disconnects node d from the rest of the network. By selecting different logical links, we can get a more resilient network. In Figure 1(c) a failure of node c or link $c-i$ does not disconnect d from the rest of the logical network, but a failure of i or $i-d$ still does. By using the same logical links, but routing the logical link $c-d$ differently, we get the logical network in Figure 1(d), which is resilient to any single physical failure.

To deliver network services with guaranteed Service Level Agreement (SLA), it is not enough for the network operator to create a restorable topology. Very often, the operator should be able to provide "restorable throughput" [2], that is, end-to-end throughput whose availability is guaranteed also in the face of a failure. The desire to guarantee restorable throughput contradicts the desire to maximize throughput availability, because full restoration requires that some bandwidth must be reserved in the event of a failure. Nevertheless, these two requirements can be combined into a single optimization criterion: maximizing the *restorable throughput*. Since failures are often limited to a single network element, it is customary to guarantee restorable throughput under the assumption that a new failure may occur only after the network has recovered from all previous failures.

There are several possible schemes to guarantee end-to-end restorable throughput. These schemes are compared in [2], and the one called "Global Recovery" was shown the be the best. In this scheme, end-to-end logical paths are built over the logical network according to the users' requirements, where each logical path consists of one or more logical links. For each logical path, an end-to-end backup path is built in advance

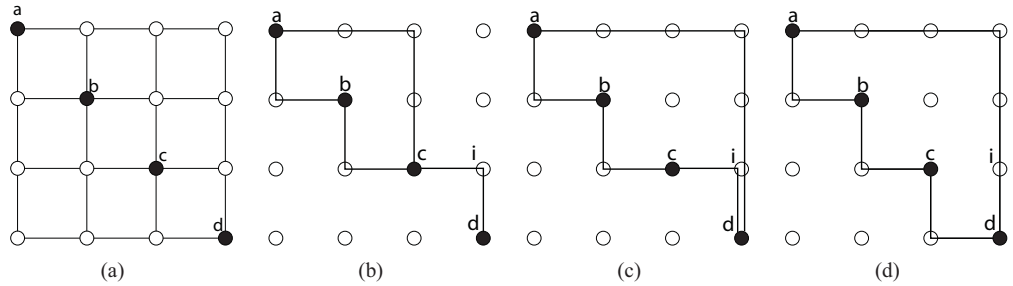


Fig. 1. An example of a physical network (a) and 3 logical networks (b-d) built over the physical network using the same number of links

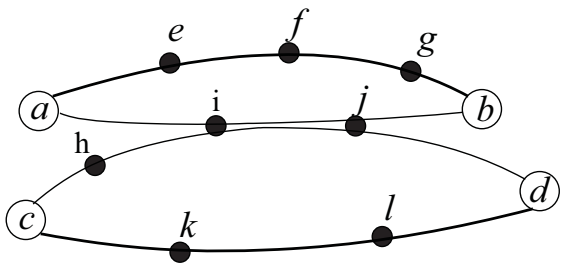


Fig. 2. The global recovery scheme

between the same pair of end nodes. The backup path protects against all physical link failures along the primary path, with which it shares no physical link.

Figure 2 shows an example of the Global Recovery scheme and the importance of constructing the logical network in such a way as to guarantee end-to-end restorable throughput. There are two primary end-to-end paths (thick lines) with guaranteed restorable throughput of 1Gb/s: $a-e-f-g-b$ and $c-k-l-d$. Note that the links of each path are logical links (lightpaths), built earlier on the optical topology. For each of these primary end-to-end paths, the following backups paths are built: $a-i-j-b$ is the backup path of $a-e-f-g-b$ and $c-h-i-j-d$ is the backup path for $c-k-l-d$. If no pair of the 13 lightpaths (links) shown in this figure shares a physical link, then a failure of one physical link will destroy only one logical link. This implies that a single failure cannot disconnect the two primary end-to-end paths ($a-e-f-g-b$ and $c-k-l-d$) at the same time. Thus, only 1Gb/s on the logical link $i-j$ must be reserved in order to guarantee the availability of 1Gb/s over $a-i-j-b$ if $a-e-f-g-b$ fails and the availability of 1Gb/s over $c-h-i-j-d$ if $c-k-l-d$ fails. In contrast, if the logical links $e-f$ and $k-l$ share an optical link, its failure will destroy both primary end-to-end paths. In such a case, 2Gb/s must be reserved on the logical link $i-j$ for post failure use.

In this paper we propose algorithms for building the logical topology on top of the optical network (i.e., selecting and routing logical links), using a new optimization criterion: maximizing the restorable throughput of the end-to-end paths. We address this problem in the following three cases:

- Case-1: no knowledge of bandwidth demands. Here the operator determines which logical node pairs will be connected by logical links and how each logical link will be routed, when future bandwidth demands are unknown. That is, the operator does not know which primary end-to-end paths will have to be admitted into the network, how many primary paths will be needed, and which nodes are more likely to serve as end points of such paths.
- Case-2: partial knowledge of bandwidth demands. Here, the operator does not know which primary end-to-end paths will have to be admitted into the network, but it knows each node's importance (weight). That is, it knows the relative proportion of traffic expected to originate from or be received by that node.
- Case-3: full knowledge. Here the operator knows which pairs of nodes need to be connected by primary end-to-end paths and the bandwidth demand for each path. While this case only rarely occurs in practice, it can serve us as a benchmark and as an "upper bound" for the other cases.

For lack of information, one cannot formulate an optimization problem that directly maximizes the restorable throughput in the first two cases. Therefore, in these cases we use a different optimization criterion: minimizing the network's *Shared Risk Link Groups* (SRLG) [22]. An SRLG of a physical link e is the set of all logical links routed over e. The cardinality of the SRLG associated with a physical link is known to be a good indicator of the damage to the logical network if this link fails. We seek to minimize the maximum cardinality of the SRLGs over all physical links.

The main contributions of this paper are as follows:

1) Introducing and formulating the problem of minimizing the maximum SRLG for the selection and routing of logical links.
2) Proposing a near-optimal approximation algorithm for the selection and routing of logical links while minimizing the maximum SLRG when no a-priori information about the users' bandwidth demands is given (case-1).
3) Proposing a near-optimal approximation algorithm for the selection and routing of logical links while minimizing the maximum SLRG when a-priori information of the nodes' weights is given (case-2).
4) Comparing the restorable throughput admitted in each of the above three cases.

The rest of the paper is organized as follows. Section II discusses related work. In Sections III, IV and V we propose algorithms to construct a logical network for case-1, case-2 and case-3, respectively. In section VI we evaluate the performance

of the different algorithms. Finally, Section VII concludes the paper.

II. Related Work

Many works deal with the construction of survivable logical DWDM/MPLS/IP overlay network on top of a physical optical network, e.g., [4], [5], [6], [13], [18]. Most of these works assume that the logical links are given in advance, and focus only on the routing of the logical links subject to some optimization criterion.

To ensure resiliency, of the logical network many network operators seek to minimize the maximum cardinality of the Shared Risk Link Groups (SRLGs) in the logical network [22]. In [18], the authors propose a routing algorithm that maximizes the connectivity of the logical network and propose an integer Linear Program to minimize the maximum SRLG. In [13], the authors study the impact of logical link routing on the amount of spare capacity that should be reserved on the logical links, in order to guarantee the required bandwidth demand also following a single failure. To address this problem, they assume that the bandwidth demand matrix is known in advance. Our work addresses a related problem, i.e., the maximum restorable throughput, while not making this assumption.

The design of a logical network on top of an optical network has been studied extensively for the case where network restoration in not important (e.g., [7], [10], [15] and references therein). Most of these works focus on the efficiency and quality of service of the logical network, without taking into account the possibility of failures.

There are some works that address both the link selection and link routing while taking into account the survivability of the logical network and the traffic it carries. In [12], the authors propose an algorithm to select and route logical links while ensuring that a given traffic demand matrix is satisfied by a set of node-disjoint paths between every pair of logical nodes. In [14], the authors address the problem of logical link selection and routing while minimizing the maximum traffic load on the logical links. Both works assume that the traffic demands are known in advance, in contrast to our work.

III. Case-1: No Prior Knowledge of Users' Bandwidth Demands

When no prior knowledge about the end-to-end paths to be established over the logical topology is available, the network operator can build a resilient logical network by seeking to minimize the maximum cardinality of the Shared Risk Link Groups (SRLGs) in the logical network [22]. As noted above, the cardinality of the SRLG associated with a physical link is known to be a good indicator to the damage caused to the logical network due to a failure of this link. In this section we address for the first time the link selection and routing tasks together, while minimizing the maximum SRLG size. One of our contributions is a near-optimal approximation algorithm that minimizes the maximum SRLG.

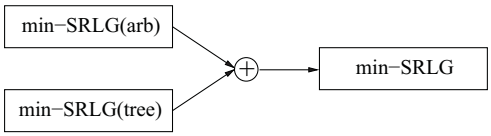


Fig. 3. The structure of the approximation algorithm for MM-SRLG

A. Problem Definition and Computational Complexity

Let $G_P = (V_P, E_P)$ be an undirected graph that represents the physical (optical) topology, where V_P is the set of optical switches and E_P is the set of optical links. Let $V_L \subseteq V_P$ be a subset of the optical nodes that can serve as routers. These routers are the nodes of the logical network, and they are the only physical nodes that can serve as end points of logical links. We assume that the budget allows to establish at most B logical links.

The problem we define is two-fold. First, deciding which pairs of logical nodes should be connected by a lightpath (logical link), while ensuring that the total number of lightpaths is B. Second, determining the path over which each of these B logical links should be established. The goal is to minimize the maximum number of logical links traversing a single physical link. We call this problem *MM-SRLG*. Our model allows more than one lightpath to connect two routers. This might be necessary in order to guarantee the connectivity between the two routers even when one lightpath fails, because usually the two lightpaths are routed over different physical routes.

We refer to the special case of MM-SRLG where the budget of the logical links is $B = |V_L| - 1$ as *MM-SRLG(tree)*. Using a reduction from the Directed Hamiltonian Cycle (DHC) problem [9] it can be shown that MM-SRLG(tree) is NP-Complete. Therefore the general MM-SRLG problem is also NP-Complete. We omit the proof for lack of space. It can be found in the full version of this paper [3].

B. An Approximation Algorithm for MM-SRLG

In this section we develop an approximation algorithm for MM-SRLG in three steps. We first present an approximation algorithm for the problem of constructing an arbitrary, not necessarily connected, logical network. Then, we propose another approximation algorithm for solving the problem when the logical network must be a connected tree (MM-SRLG(tree)). Finally, we combine the output of the two algorithms to produce a connected logical graph with the desired number of logical links. The final algorithm guarantees that SRLG $\leq$ OPT+3, where OPT is the optimal solution for MM-SRLG. Figure 3 depicts the structure of the final algorithm.

1) An Algorithm for MM-SRLG(arb): We start with the case were the constructed logical graph is not necessarily connected. We seek to build an arbitrary graph with a predetermined number of logical links, whose SRLG is minimum. We call this problem *MM-SRLG(arb)*. To solve MM-SRLG(arb), consider the reverse variant where the objective is to route the maximum number of logical links between the nodes of V_L such that the number of logical links traversing each physical link e does not exceed $c(e)$, where $c(e)$ is a capacity function on the physical

edges. Ref. [8] shows that if $c(e)$ is even for every $e \in E_P$, then there is an integral optimal solution. In [8], a polynomial time algorithm to find such a solution is presented. We refer to this algorithm as ALG, and propose the following algorithm for the MM-SRLG(arb) problem.

Algorithm 1: (An approximation algorithm for MM-SRLG(arb))

1) For $C = 1$ to $\lceil B/2 \rceil$ do (recall that B is the number of logical links)
   a) For every $e \in E_P$, set the capacity of e to be $2 \cdot C$ and call ALG (i.e., find the logical graph with the maximum number of logical links).
   b) If the number of logical links in the graph is $\geq B$, exit the loop.
2) From the set of logical links, choose an arbitrary subset of size B.

Each operation of Algorithm 1 runs in polynomial time. Since the algorithm loops at most $B/2$ times, a naive implementation would be pseudo-polynomial. However, the algorithm can be implemented in polynomial time by conducting a binary search on the values of C, instead of running on them sequentially.

Theorem 1: Algorithm 1 builds a logical network whose SRLG $\leq$ OPT+1, where OPT is the SRLG of an optimal solution for MM-SRLG(arb).

Proof: First, we note that there must be a value of C for which the number of logical links $\geq B$, because the capacity of each physical link may go as high as B. Assume that the algorithm produces a logical network whose maximum SRLG $S' >$ OPT+1. Let C' be the value of C in the final loop of Step 1. We have OPT $\leq S' - 2 \leq 2 \cdot C' - 2 = 2(C' - 1)$. On the other hand, we know that when $C = C' - 1$, the maximum number of logical links is $\geq B$. Hence, OPT $> 2(C' - 1)$ must hold, which yields a contradiction. ■

2) An Algorithm for MM-SRLG(tree): Our next step is to develop an algorithm for MM-SRLG(tree), which builds a logical tree whose maximum SRLG does not exceed 2. Since the minimum SRLG for any min-SLRG(tree) instance is 1, this algorithm can be viewed as a 2-approximation.

Algorithm 2: (A 2-approximation for MM-SRLG(tree))

1) Conduct a depth first search (DFS) tour on G_P starting at an arbitrary node. The result is a non-simple cycle that traverses all the nodes in V_P (including those that are also in V_L).
2) Break the cycle into sub-paths that are separated by the nodes of V_L.
3) Transform each sub-path into a logical link that connects the two end logical nodes. The routing of each logical link is according to the corresponding sub-path. Let E_L be the resulting set of logical links.
4) Remove from E_L as many logical links as possible while keeping $G_L = (V_L, E_L)$ connected.

Figure 4 depicts an example of Algorithm 2. Figure 4(a) shows an example of G_P with 3 logical nodes (denoted by rectangles). Figure 4(b) shows a cycle produced by a DFS tour. The cycle is divided into 4 sub-paths, each representing a logical link: 1-3, 3-2, 2-2 and 2-1. Figure 4(c) shows the final logical tree that contains a subset of 2 logical links: 1-3 and 3-2.

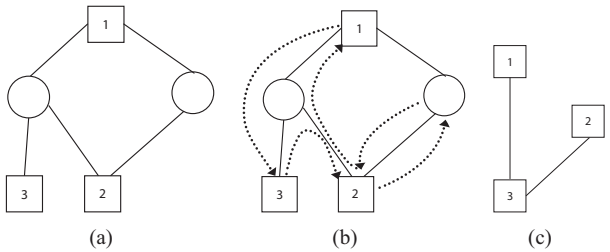


Fig. 4. An illustration of Algorithm 2: (a) a physical graph G_P (logical nodes are denoted by rectangles); (b) a DFS (dashed line) divided into sub-paths; (c) a logical tree after the removal of the redundant link $1 - 2$.

The Algorithm running time is obviously polynomial.

Theorem 2: Algorithm 2 produces a logical tree whose maximum SRLG ≤ 2.

Proof: Since the DFS traverses all the nodes, the set of logical links produced in Step 3 forms a connected logical graph. Every link in E_P is traversed by the DFS exactly twice, once in each direction. Thus, the maximum SRLG of E_L is 2. Finally, after the execution of Step 4, E_L forms a logical tree. ■

3) The Final Algorithm: We now solve the original min-SLRG problem, which requires the logical graph to be connected and contain no more than B links. To this end, we combine Algorithm 1 and Algorithm 2 in the following way. We first invoke Algorithm 2 to produce a logical tree. Then, we invoke Algorithm 1 to generate additional logical links, such that their total number will be B.

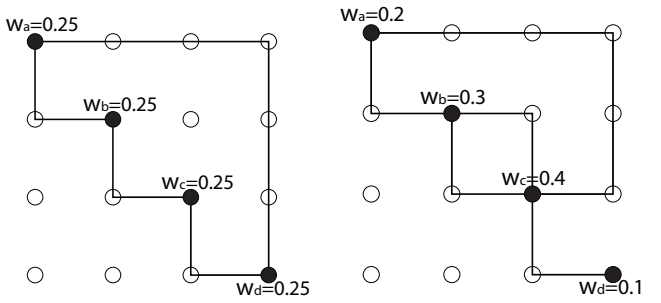
Algorithm 3: (An approximation for of MM-SRLG)

1) Execute Algorithm 2. Denote its output by E_L^{tree}.
2) Execute Algorithm 1 with a budget equals to $B' = B - (|V_L| - 1)$. Denote its output by E_L^{arb}.
3) Return $E_L^{tree} \cup E_L^{arb}$.

Note that the algorithm may output more than one logical link between a pair of logical nodes. As mentioned in the Introduction, we allow this in our model. This is necessary to accommodate cases in which the network operator wishes to provision between two routers a higher amount bandwidth than a single lightpath allows.

Theorem 3: Algorithm 3 produces a connected logical network whose maximum SRLG $\leq$ OPT+3, where OPT is the value of the optimal solution for MM-SRLG.

Proof: Since the output of Algorithm 2 is a connected logical graph, the output of Algorithm 3 is connected as well. It is easy to see that the number of logical links in the solution returned by the algorithm is B. Let S' denote the maximum SRLG of the solution produced by the algorithm. Consider an arbitrary subset of the logical links in an optimal solution, whose cardinality is $B - (|V_L| - 1)$. Denote the maximum SRLG of this subset by $S_{B-(|V_L|-1)}$. In addition, denote the maximum SRLG of the optimal solution of MM-SRLG(arb) with $B - (|V_L| - 1)$ logical links by $\text{OPT}_{B-(|V_L|-1)}$, and the



(a) a logical network with 4 lightpaths

(b) a logical network with 5 lightpaths

Fig. 5. Two logical networks over a physical network. Option (b) is less robust although it has more logical links

output of Algorithm 1 (with $B-(|V_L|-1)$ logical links) by S_{ALG-1}. Thus,

$$\begin{aligned}\text{OPT} &\geq S_{B-(|V_L|-1)} \geq \text{OPT}_{B-(|V_L|-1)} \geq S_{ALG-1}-1 \\ &\geq (S'-2)-1 = S'-3,\end{aligned}$$

The first inequality holds because $S_{B-(|V_L|-1)}$ is the maximum SRLG of a subset of the optimal solution. The second inequality holds because the maximum SRLG of an optimal solution must not be greater than an arbitrary solution, and the third inequality holds due to Theorem 1. Finally, the fourth inequality holds because the maximum SRLG of the tree produced by Algorithm 2 is 2, which implies that $S' \leq S_{ALG-1}+2$. ■

IV. Case-2: Using Information About the Weights of the Logical Nodes

As said earlier, when designing a logical network, the operator is unlikely to know the exact bandwidth demand matrix. However, in many cases each router (logical node) can be associated with a weight, which is proportional to the portion of traffic expected to originate from and received by it. This weight can be determined by the network operator before the logical network is set up, based on past experience, on the number of users/subnets expected to be connected through each node, on the importance of the node's geographical location, and so on. Herein we propose to use this weight as an indication of the degree this node should have in the logical network.

The idea that the number of lightpaths connected to every node should be proportional to a weight whose value is known in advance is one of the contributions of this paper. It allows us to build a logical network while taking into account the expected traffic load on each router although the actual traffic matrix is unknown in advance. We illustrate this idea in Figure 5, which shows two logical networks overlayed on the physical network of Figure 1(a).

In Figure 5(a) we assume that the 4 routers are equally important. Thus, each of them has an equal weight of 0.25. In this figure we also assume that the budget of the logical network allows to establish only $B=4$ lightpaths. Hence, each of the 4 routers should be an end point of $2 \cdot 0.25 \cdot 4 = 2$ lightpaths. Of course, there are other options to establish 4 lightpaths between the 4 nodes such that each node will be connected to one lightpath. In Figure 5(b), the same 4 nodes have different weights and the budget allows to establish $B=5$ lightpaths. Thus, we have 4 lightpaths connected to node c, 3 to node b, 2 to a and 1 to d. Although there are more lightpaths in Figure 5(b) than in Figure 5(a), the logical network is less robust, because it might be disconnected after a single failure.

We seek to maximize the restorable throughput while taking into account the weight of each logical node by proposing an approximation algorithm (Algorithm 6) for the minimization of the maximum SRLG while taking the weights of the nodes as constraints, and show that this algorithm produces logical networks that accommodate more restorable throughput than the networks produced by Algorithm 3.

A. The Weighted MM-SRLG Problem

In this section we address the weighted variant of MM-SRLG problem. This problem is similar to MM-SRLG, except that the degree of every logical node $v \in V_L$ must be proportional to a given weight w_v which is part of the network. The value of w_v is normalized such that $\sum_{v \in V_L} w_v = 1$, and each logical node v must have a degree of $2 \cdot B \cdot w_v$ in G_L. In addition, we assume that each logical node must have a degree ≥ 2; thus, $2 \cdot B \cdot w_v \geq 2$ must hold for every v.

We develop an approximation algorithm for the weighted MM-SRLG problem in a similar way to what we did in Section III for MM-SRLG. We first develop an algorithm for the weighted MM-SRLG(arb) problem, where the constructed graph is not necessarily connected. Then, we solve the MM-SRLG(cycle) problem where a logical cycle over the physical graph is produced. Here, we produce a cycle rather a tree in order to keep the values of the node weights even. This is necessary for the algorithm of MM-SRLG(arb). Finally, we combine the solutions of the two algorithms into a solution for the general problem.

1) An Algorithm for Weighted MM-SRLG(arb): In the weighted MM-SRLG(arb) problem we do not require the logical graph to be connected, but we take into account the weight constraints on the logical degrees. To take this into account, we transform the physical graph G_P into $G'_P = (V'_P, E'_P)$ as follows. First, for every logical node $v \in V_L$ we define a mirror node v'. Let V'_L be the set of these mirror nodes. Then $V'_P = V_P \cup V'_L$ and $E'_P = E_P \cup \{(v,v') | v \in V_L\}$. The weight of every mirror node $v' \in V'_L$ is $w_{v'} = w_v$.

Algorithm 4: (An approximation for weighted MM-SRLG(arb))

1) Construct G'_P and V'_L according to the above transformation.
2) For every $v \in V_L$, set the capacity of the link (v,v') to be $2 \cdot B \cdot w_v$.
3) For $C=1$ to $\lceil B/2 \rceil$ do
   a) Set the capacity of every $e \in E_P$ to be $2 \cdot C$.
   b) Find the maximum set of logical links in G'_P.
   c) If the total number of logical links equals B, exit the loop.

Since the output of the algorithm is B links, and since $\sum_v w_{v'} = 1$, then every logical node v is attached to exactly $2 \cdot B \cdot w_v$ logical links.

Theorem 4: Algorithm 4 builds a logical network whose SRLG $\leq$ OPT+1, where OPT is the SRLG of an optimal solution for weighted MM-SRLG(arb).

The proof of this theorem is similar to the one presented for Theorem 1.

2) An Algorithm for MM-SRLG(cycle): We now consider a variant of MM-SRLG, where a simple logical cycle that spans all logical nodes is produced. Such a cycle must have $|V_L|$ links. While this cycle is a simple cycle at the logical layer, it may be routed over a non-simple cycle in the physical layer. We adapt Algorithm 2 for the MM-SRLG(tree) to produce a solution for MM-SRLG(cycle) whose maximum SRLG ≤ 2.

Algorithm 5: (An algorithm for a cycle with a SRLG=2)

1) Find a tree on G_P that spans V_L.
2) Conduct a DFS tour on the tree, starting at an arbitrary node v, to produce a non-simple physical cycle C that traverses all nodes in V_L.
3) Break C into sub-paths, such that each sub-path must start and end at logical nodes and each logical node must be an end point of exactly two sub-paths.
4) Transform each sub-path into a logical link with the corresponding routing over the physical graph. Denote this set of logical links as E_L.

Theorem 5: The logical network generated by Algorithm 5 is a simple cycle whose maximum SRLG is not greater than 2.

Proof: Since the tour traverses all the nodes and returns to the starting node, the constructed logical network is a cycle. Since a logical node is an end point of exactly two links, then the cycle must be simple. The maximum SRLG is not greater than 2 because every link is traversed by the DFS exactly twice (once in each direction). ■

Note that in the constructed cycle above the degree of each logical node is 2.

3) The Final Algorithm: We are now ready to present the final approximation algorithm for the weighted MM-SRLG problem, which combines Algorithm 4 and Algorithm 5.

Algorithm 6: (an approximation Algorithm for weighted MM-SRLG)

1) Execute Algorithm 5. Denote its output by E_L^{cycle}.
2) For every logical node v reduce its weight by 1.
3) Execute Algorithm 4 with a budget that equals $B' = B - |V_L|$. Denote its output by E_L^{arb}.
4) Return $E_L^{cycle} \cup E_L^{arb}$.

Theorem 6: Algorithm 6 produces a logical network whose maximum SRLG $\leq$ OPT+3, where OPT is the value of the optimal solution of weighted MM-SRLG which is not greater than the optimal solution plus 3.

The proof is similar to the proof of Theorem 3. It is omitted due to lack of space. It can be found in the full version of this paper [3].

V. Case-3: Assuming Full Knowledge of The Users' Bandwidth Demands

We now address case-3, where the operator knows the average bandwidth demands between every pair of nodes on the logical network. In this case, we solve a joint optimization problem that:

1) Selects the logical links and routes them on the physical network.
2) Determines the end-to-end primary and backup paths for all traffic demands, and the bandwidth reserved on each path.

Case-3 only rarely occurs in practice, because the network operator only rarely has concrete knowledge of the future bandwidth demands when the logical network is constructed. Still, this case can serve as a benchmark for the maximum restorable throughput in case-1 and case-2. The joint optimization problem for this case can be formulated as an integer linear program (ILP), which we solve by relaxing its integrality constraints.

The linear program for the global recovery scheme requires as an input the primary path of each traffic flow. However, we do not have this information when the logical network is constructed. We solve this problem by building a backup path for every flow and every logical link the flow traverses, rather than building a single backup path for every flow. This strategy requires every backup path to start and end at the nodes of the failed logical link, and it is slightly inferior to the global recovery scheme [2].

The linear program is detailed in Appendix A.

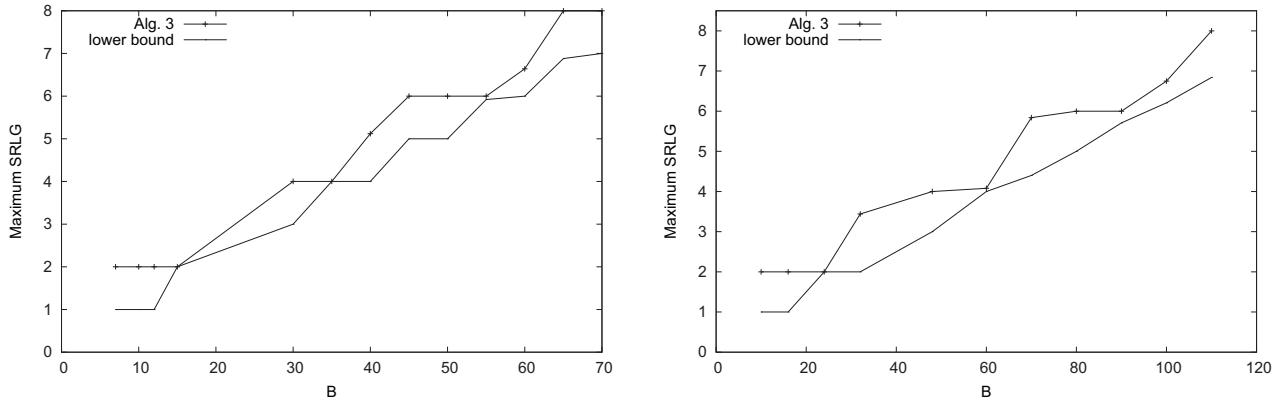
VI. Simulation Study

In this section we evaluate the performance of the proposed algorithms using extensive simulations. We first evaluate the SRLG performance of Algorithms 3 (case-1) and 6 (case-2). Then, we compare the restorable throughput in the three cases.

A. SRLG Performance

We first evaluate the SRLG performance of Algorithm 3. As in earlier related work [17], [18], [21], we use augmented real network topologies as our physical network: the NSFNET topology, which has 14 nodes and 29 links, and the USIP topology, which has 24 nodes and 53 links. For each topology we generate 125 random instances, each comprises of a subset of physical nodes, which serve as logical routers, and a budget B of logical links. For each instance we ran the following algorithms:

1) Algorithm 3: This is the MM-SRLG approximation algorithm presented in Section III-B3.
2) An algorithm that finds a lower bound (i.e. the best possible result) for MM-SRLG. This algorithm finds an optimal solution for the splittable variant of the problem, which can be represented as a linear problem and be solved in polynomial time. Obviously, an optimal solution to this variant is always better than or equal to the optimal solution for MM-SRLG. We then round up the fractional optimal solution returned by the algorithm. The rounded



(a) NSFNET physical topology, $|V_L|$=5 (b) USIP physical topology, $|V_L|$=8

Fig. 6. The SRLG performance of Alg. 3 compared to the theoretical lower bound

result is still a lower bound for MM-SRLG, because the optimal solution for MM-SRLG is also integral. The linear program is omitted due lack of space. It can be found in the full version of this paper [3].

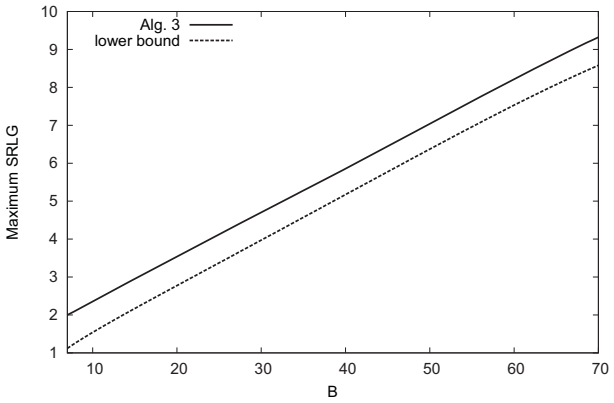
In each simulation instance, the locations of the logical routers are randomly chosen using a uniform distribution. The budget of logical links (B) varies between $|V_L|+2$ and $14\cdot|V_L|$.

Figure 6 shows the simulation results for the two physical topologies. In all the graphs the x-axis represents the budget B of logical links, while the y-axis represents the maximum SRLG size produced by each algorithm. We see in all graphs that Algorithm 3 performs very well, and it is very close to the theoretical lower bound. It is interesting to note that both algorithms exhibit a step-like behavior. The only distinction between different simulation instances with the same B is the locations of the logical nodes, which are determined based on uniform distribution. Thus, this step-like behavior indicates that the most dominant factor of the maximum SLRG size is the number of logical links (B), while the set of locations of the logical nodes has a negligible effect.

To validate the above results we depict in Figure 7 the performance of the above two algorithms while using artificially generated physical topologies. We randomly generate 20 physical topologies having 15 nodes and 30 links (equivalent in size to the NSFNET topology). To generate these topologies we use the BRITE simulator [19] while employing the Barabasi-Albert model [1]. This model captures two important characteristics of network topologies: incremental growth and preferential connectivity of a new node to well-connected existing nodes. These characteristics yield a power-law degree distribution of the nodes.

The results of Figure 7 closely resemble the ones for the NSFNET topology. Here also the difference between Algorithm 3 and the theoretical lower bound is less than 1. The curves are smoother than the curves for NSFNET, because we average the results over many instances of physical topologies.

We now evaluate the SRLG performance of Algorithm 6. The evaluation is done using the same simulation setup described above. We only show here the results for the NSFNET physical topology. Figure 8 shows the SRLG for Algorithm 6 and Algorithm 3. The x-axis represents the budget B of logical links, while the y-axis represents the maximum SRLG achieved by each algorithm. It is evident that Algorithm 3 obtains the lowest maximum SLRG size, which is not surprising since this algorithm is not restricted by the node weights.

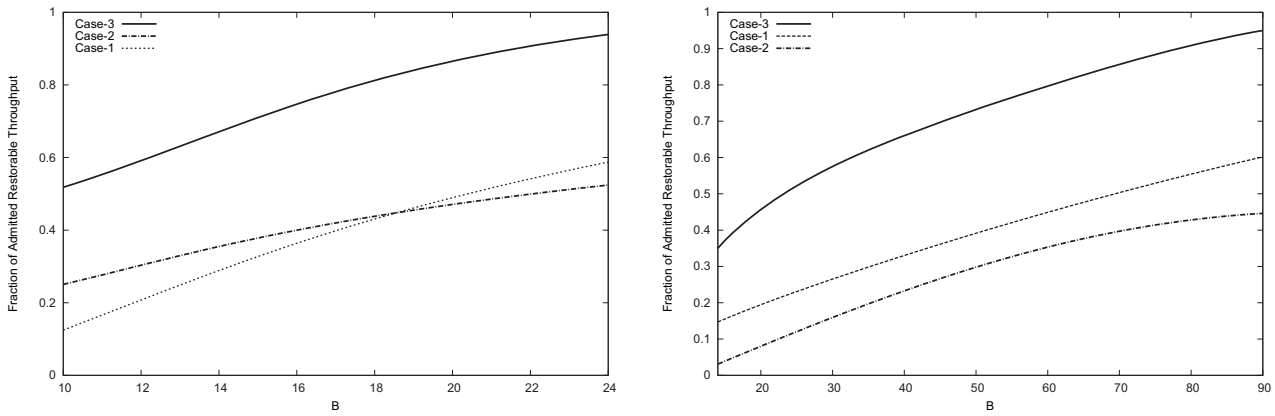


(a) $|V_L|$=5

Fig. 7. The SRLG performance of Alg. 3 compared to the theoretical lower bound averaged over 20 artificially generated physical topologies

B. Restorable Throughput Performance

We now evaluate the restorable throughput admitted by the logical networks constructed in each of the three cases. To measure this criterion, we generate traffic demands for the logical network based on the Gravity model [16], where the demand of every pair of logical nodes is proportional to the products of their weights. To measure the maximum restorable throughput that can be admitted into the logical network, we use the optimal algorithm presented in [2] for the Global recovery scheme [20]. As discussed earlier and depicted in Figure 2, in this scheme each primary logical path is associated with one backup logical path between the same end nodes. Each backup



(a) $|V_L|$=7 (b) $|V_L|$=12

Fig. 9. The restorable throughput performance of the algorithms given NSFNET as the physical topology

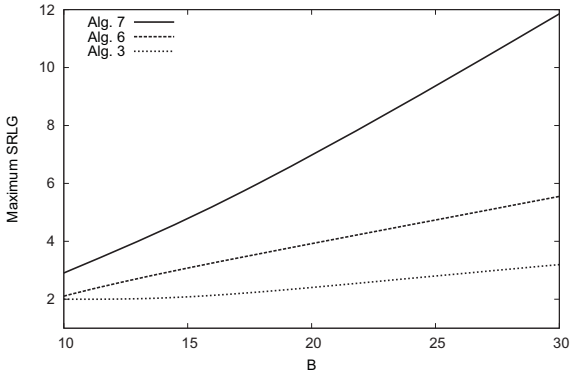


Fig. 8. The SRLG performance of the algorithms given NSFNET as the physical topology, and $|V_L| = 7$

path protects against the failure of any physical link in the primary path, and it therefore does not share any physical link with this path. This scheme is sometimes referred to as "path recovery scheme."

Figure 9 depicts the restorable throughput for the constructed logical network with 7 or 12 logical nodes. For this simulation, we use the NSFNET as the physical topology. The x-axis represents the number of logical links, denoted B, whereas the y-axis represents the fraction of traffic demand that can be admitted into the logical network while guaranteeing 100% restorability in the face of a single physical failure. As expected, the fraction of admitted throughput increases with B for all the algorithms. Furthermore, the algorithm for case-3 obtains 50-150% better throughput than the other two, with its advantage depending on the number of logical links. As this number increases, the advantage of the case-3 algorithm decreases due to the greater flexibility in the selection of the logical links, which in turn makes the full knowledge of future demands less important.

The difference between the case-1 and case-2 curves is interesting. We can see that with $|V_L| = 7$ (Figure 9(a)), case-2 has better restorable throughput than case-1 for small B values. However, when B increases, the throughput for case-3 improves, implying that for small budgets the network should be designed according to the expected traffic distribution. In networks with many logical links, this design criteria is less important, and the constraints on the link degrees are somewhat counterproductive. In this case, the SRLG plays a more dominant role, and it is better to design the network by minimizing the SRLG and ignoring the node weights.

In contrast, with $|V_L| = 12$, case-1 achieves a higher restorable throughput than that of case-2 for all B values (Figure 9(b)). In this scenario, the number of logical nodes is relatively high ($|V_L| = 12$ while $|V_P| = 14$). This means that for case-1 the logical links traverse a short path of only 1 or 2 physical links. For case-2, however, distribution constraints on the logical links means that a longer path with more physical links is required. This yields that the logical topology built in case-2 is more vulnerable to physical link failures; hence it can carry less restorable throughput.

VII. Conclusion

This paper is the first to propose algorithms for building the logical topology on top of the optical network while maximizing the restorable throughput of the end-to-end paths. We addressed this problem in three cases: when the operator has no knowledge of the future bandwidth demands, when it has partial knowledge, and when it has full knowledge. For the first case the most reasonable strategy is to minimize the maximum SRLG size. We showed that the resulting (new) problem is NP hard, and proposed an efficient approximation for solving it. For the second case we assumed that each router (logical node) can be associated with a weight, which is proportional to the portion of traffic expected to originate from and received by it. We used this weight as an indication of the degree this node should have in the logical network. In the third case, we assumed that the operator knows the average bandwidth demands between every pair of nodes on the logical network. We presented efficient algorithms for each of these cases, and ran extensive simulations to compare their performance.

REFERENCES

[1] A. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the World Wide Web. In *Physica A: Statistical Mechanics and Its Applications*, volume 281, pages 69–77, June 2006.

[2] R. Cohen and G. Nakibly. Maximizing restorable throughput in MPLS networks. *IEEE/ACM Transactions on Networking*, 18(2):568–581, 2010.

[3] R. Cohen and G. Nakibly. Restorable logical topology in the face of no or partial traffic demand knowledge. www.cs.technion.ac.il/~rcohen/PAPERS/rest-topology.pdf, 2013.

[4] O. Crochat and J.-Y. Le Boudec. Design protection for WDM optical networks. *IEEE Journal on Selected Areas in Communications*, 16(7):1158–1165, September 1998.

[5] O. Crochat, J.-Y. Le Boudec, and O. Gerstel. Protection interoperability for WDM optical networks. *IEEE/ACM Transactions on Networking*, 8(3):384–395, June 2000.

[6] Q. Deng, G. Sasaki, and C. Su. Survivable IP Over WDM: An efficient mathematical programming problem formulation. In *IEEE HSN*, 2002.

[7] R. Dutta and G. N. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks*, 1:73–89, 2000.

[8] A. Frank, A. V. Karzanov, and A. Sebo. On integer multiflow maximization. *SIAM J. Discret. Math.*, 10:158–170, 1997.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[10] A. Gençata and B. Mukherjee. Virtual-topology adaptation for WDM mesh networks under dynamic traffic. *IEEE/ACM Trans. Netw.*, 11(2):236–247, April 2003.

[11] O. Gerstel, R. Ramaswami, and G. H. Sasaki. Cost-effective traffic grooming in WDM rings. *IEEE/ACM Transactions on Networking*, 8:618–630, 2000.

[12] L. Gouveia, P. Pedro, and A. de Sousa. Hop-constrained node survivable network design: An application to MPLS over WDM. *Networks and Spatial Economics*, 8:3–21, 2008.

[13] D. Kan, A. Narula-Tam, and E. Modiano. Lightpath routing and capacity assignment for survivable IP-over-WDM networks. In *DRCN*, October 2009.

[14] G. R. Kiran and C. S. R. Murthy. QoS based survivable logical topology design in WDM optical networks. *Photonic Network Communications*, 7:193–206, 2004.

[15] V. R. Konda and T. Y. Chow. Algorithm for traffic grooming in optical networks to minimize the number of transceivers. In *IEEE Workshop on High Performance Switching and Routing*, pages 218–221, 2001.

[16] J. Kowalski and B. Warfield. Modeling traffic demand between nodes in a telecommunications network. In *Proceedings of ATNAC*, 1995.

[17] K. Lee, H.-W. Lee, and E. Modiano. Reliability in layered networks with random link failures. In *INFOCOM*, 2010.

[18] K. Lee and E. Modiano. Cross-layer survivability in WDM-based networks. *IEEE/ACM Transactions on Networking*, 2011.

[19] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An approach to universal topology generation. In *Proceedings of MASCOTS*, 2001.

[20] V. Sharma and F. Hellstrand. Framework for multi-protocol label switching (MPLS)-based recovery. IETF RFC 3469, February 2003.

[21] T. Venkatesh, T. Sujatha, and C. Murthy. A novel burst assembly algorithm for optical burst switched networks based on learning automata. *Optical Network Design and Modeling*, 4534:368–377, 2007.

[22] H. Zang and J. P. Jue. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine*, 1:47–60, 2000.

APPENDIX A
THE LINEAR PROGRAM FOR CASE-3

The linear program has the following parameters:

- F – the set of all logical node pairs. Each pair $f = (s_f, t_f) \in F$ constitute a flow.
- d_f – the bandwidth demand of flow $f \in F$.
- B – the total number of logical links that are budgeted to the network.

We define the following variables:

- $y^e_{f,(u,v)}$ – the fraction of d_f routed over the logical edge connecting the logical nodes u and v when physical edge $e \in E_P$ fails; when no edge fails, $\bar{e} = \phi$.
- x_f – the total routed fraction of d_f.
- $l_{(u,v)}$ – a binary variable that equals 1 if and only if there is a logical link connecting the two logical nodes u and v, otherwise 0.
- $r_e^{(u,v)}$ – a binary variable that equals 1 if and only if the logical link connecting the two logical nodes u and v traverses the physical link $e \in E_P$.

The target function is to maximize the total throughput $\sum_f d_f \cdot x_f$, subject to the following constraints:

$$(1)\quad \sum_{e=(j,i)\in E_P} r_e^{(u,v)} - \sum_{e=(i,j)\in E_P} r_e^{(u,v)} = \begin{cases} l_{(u,v)} & i = u \\ -l_{(u,v)} & i = v \\ 0 & \text{else} \end{cases}, \forall i \in V_P, \forall u, v \in V_L$$

$$(2)\quad \sum_{u,v\in V_L} l_{(u,v)} = B$$

$$(3)\quad \sum_{u,v\in V_L} y^e_{f,(u,v)} - \sum_{v,u\in V_L} y^e_{f,(u,v)} = \begin{cases} x_f & v = t_f \\ -x_f & v = s_f \\ 0 & \text{else} \end{cases} \quad \forall v \in V, \forall f \in F, \forall e \in \{E_P, \phi\}$$

$$(4)\quad \sum_{f\in F} d_f \cdot y^e_{f,(u,v)} \le l_{(u,v)} \forall u, v \in V_L, \forall e \in \{E_P, \phi\}$$

$$(5)\quad y^e_{f,(u,v)} \le 1 - r_e^{(u,v)}, \forall f \in F, \forall e \in E_P, \forall u, v \in V_L$$

$$(6)\quad r_e^{(u,v)} + y^e_{f,(u,v)} \ge y^{\phi}_{f,(u,v)} \quad \forall f \in F, \forall e \in E_P, \forall u, v \in V_L$$

$$(7)\quad 0 \le x_f \le 1,\ 0 \le y^e_{f,(u,v)} \le 1 \quad \forall f \in F, \forall u, v \in V_L, \forall e \in \{E_P, \phi\}$$

$$(8)\quad r_e^{(u,v)} \in \{0, 1\},\ l_{(u,v)} \in \{0, 1\} \quad \forall e \in E_P, \forall u, v \in V_L.$$

The set (1) of constraints ensures that every logical link is routed over the physical link. Constraint (2) ensures that there are the total number of logical links is B. The set (3) of constraints ensures traffic flow conservation over the logical network. The set (4) ensures that traffic flows only over the selected logical links. The set (5) ensures that traffic flow will be carried by logical links that traverse a failed physical link. The set (6) ensures a traffic flow be carried by the same logical links for all link failure events unless the logical link traverses a failed physical link. The set (7) of constraints ensures that the total routed bandwidth of each flow do not exceed flow demand. Finally, the set (8) of constraints ensures that each logical link is routed over a single physical path.

The above program is an integer linear program (ILP). To allow tractability in the simulation study we solve the fractional version of this ILP where we relax the integrality constraints.