Survivable Routing of Mesh Topologies in IP-over-WDM Networks by Recursive Graph Contraction

Maciej Kurant and Patrick Thiran

***Abstract*— Failure restoration at the IP layer in IP-over-WDM networks requires to map the IP topology on the WDM topology in such a way that a failure at the WDM layer leaves the IP topology connected. Such a mapping is called *survivable*. Finding a survivable mapping is known to be NP-complete, making it impossible in practice to assess the existence or absence of such a mapping for large networks. (i) We first introduce a new concept of *piecewise survivability*, which makes the problem much easier in practice (although still NP-complete), and allows us to formally prove that a given survivable mapping does or does not exist. (ii) Secondly, we show how to trace the vulnerable areas in the topology, and how to strengthen them to enable a survivable mapping. (iii) Thirdly, we give an efficient and scalable algorithm that finds a survivable mapping. In contrast to the heuristics proposed in the literature to date, our algorithm exhibits a number of provable properties (e.g., it guarantees the piecewise survivability) that are crucial for (i) and (ii).**

***Index Terms*— Optical communication, wavelength division multiplexing, survivability, graph theory**

I. Introduction

GENERALLY, there are two approaches for providing survivability of IP-over-WDM networks: protection and restoration [1]. Protection uses pre–computed backup paths applied in the case of a failure. Restoration finds dynamically a new path, once a failure has occurred. Protection is less resource efficient (the resources are committed without prior knowledge of the next failure) but fast, whereas restoration is more resource efficient and slower. Protection and restoration mechanisms can be provided at different layers. *IP layer* (or *logical* layer) survivability mechanisms can handle failures that occur at both layers, contrary to *WDM layer* (or *physical* layer) mechanisms that are transparent to the IP topology. It is not obvious which combination (mechanism/layer) is the best; each has pros and cons [2]. IP restoration, however, deployed in some real networks, was shown to be an effective and cost–efficient approach (see e.g., Sprint network [3]). In this paper we will consider exclusively the *IP restoration approach*.

Each logical (IP) link is mapped on the physical (WDM) topology as a *lightpath*. Usually a fiber is used by more than one lightpath (e.g., in Sprint the maximum number is 25 [4]). Therefore a single physical link failure usually brings down a number of IP links. With the IP restoration mechanism, these IP link failures are detected by IP routers, and alternative routes in the IP topology are found. In order to enable this, the IP topology should remain *connected* after a failure of a physical link; this in turn may be guaranteed by an appropriate mapping of IP links on the physical topology. We call such a mapping a *survivable mapping*.

By lack of space, in this paper we consider only link failures. All the results are easily extended to node failures (see [5]) and multiple failures (see [6]). First, we are interested in the *existence* of a survivable mapping for a given pair of logical and physical topologies. There is some work on the topic in the literature, but it assumes *ring topologies* at the physical [7], [8] or the logical [9], [10] layer. We study the existence of a survivable mapping for general mesh topologies at both layers, which is foreseen to be the main future topology. To date, the only general method verifying the existence of a survivable mapping is an exhaustive search (or equivalent) run for the *entire* topology. Due to NP-completeness of the survivable mapping problem [9], the exhaustive approach is not realizable in practice for the topologies larger than a few nodes. To bypass this difficulty, we introduce a new type of mapping, which preserves the survivability of some subgraphs ('pieces') of the logical topology; we call it a *piecewise survivable mapping*. The formal analysis of the piecewise survivable mapping shows that a survivable mapping of the logical topology on the physical topology exists if and only if there exists a survivable mapping for a *contracted* logical topology, that is, a logical topology where a specified subset of edges is contracted (contraction of an edge amounts to removing it and merging its end-nodes). This new result substantially simplifies the verification of the existence of a survivable mapping. Of course, the problem remains NP-complete, but this simplification allows us, for the first time, to solve many instances of moderate and large topology size, which makes it applicable in practice.

A second application of a piecewise survivable mapping is tracing the vulnerable areas in the network and pointing where new link(s) should be added to enable a survivable mapping. To the best of our knowledge, this is also a novel functionality.

Third, the formal analysis reveals an easy way to incrementally expand the survivable pieces in a piecewise survivable mapping. This leads us to an efficient and scalable algorithm that searches for a survivable mapping, which

Manuscript received April 13, 2006; revised Nov. 29, 2006. This work is financially supported by grant ManCom 2110 of the Hasler Foundation, Bern, Switzerland.

The authors are with LCA - School of Communications and Computer Science, EPFL, CH-1015 Lausanne, Switzerland (e-mail: {maciej.kurant, patrick.thiran}@epfl.ch).

Digital Object Identifier 10.1109/JSAC.2007.070606.

we call SMART ("Survivable Mapping Algorithm by Ring Trimming"). SMART is different from the algorithms that solve this problem proposed in the literature. These algorithms can be divided into two groups: (i) greedy search based on Integer Linear Programming (ILP), and (ii) heuristics. The ILP solutions can be found for example in [9], [11], but they lead to an unacceptably high complexity, even for networks of small size (few tens of nodes). The second approach uses various heuristics, such as Tabu Search [11], [12], [13], Simulated Annealing [2] and others [1], [14]. If a heuristic fails, nothing can be claimed about the existence of a survivable mapping. We introduced the SMART algorithm in [15] as such a heuristic, without any formal analysis. Simulations in [15] showed that SMART is efficient and scales very well. The concept of piecewise survivability introduced in the present paper makes the formal analysis of SMART possible. It revealed that the SMART algorithm actually opens a third group (iii) in the family of algorithms that search for a survivable mapping. One of our key results is that, contrary to the heuristics (ii), SMART never misses a solution if there is one. This is because, even if SMART does not fully converge, the mapping it returns is piecewise survivable. This mapping is defined for a subset of logical links and leaves the remaining logical links unmapped. We prove that if a survivable solution exists, the remaining unmapped logical links can be still mapped in a way ensuring the survivability of the resulting full mapping.

This paper extends the results of our conference paper [5] in two directions. First, we extend and restructure the theoretical part to make it more complete and self-contained. In particular, we added some results on the convergence of the SMART algorithm (see Theorems 2 and 5). Second, the entire analysis in [5] assumes unlimited capacities of optical fibers. In this paper we first present the general results without capacity constraints, and then we discuss the applicability of our theorems and algorithms in the presence of limited capacities; the results are grouped in Section VIII.

The organization of this paper is the following. Section II introduces notations and formalizes the problem in the absence of capacity constraints. Section III gives three fundamental theorems. For better readability all proofs are moved to the Appendix. Section IV introduces the SMART algorithm and discusses its properties. Section V describes our implementation of SMART. Section VI discusses a number of possible applications of SMART. Section VII presents the simulation results. Section VIII introduces the capacity constraints to the analysis. Finally, Section IX concludes the paper.

II. Notation and problem formulation

A. Generalities

We use the formal notation of graph theory, mainly based on [16]. However, we also introduce the stack of our definitions well suited to the problems we tackle. The following general notation is used:

- ϕ corresponds to the *physical* topology,
- L corresponds to the *logical* topology,
- C corresponds to the *contracted* topology (introduced later in Section II-C),
- $a, b, c, d, e \ldots$ are used to denote edges/links,[1]
- $u, v, w \ldots$ are used to denote vertices/nodes,[2]
- p is used to denote a path, i.e., a sequence of edges, where two consecutive edges have a common end-node. We say that a node u is in a path p, $u \in p$, if u is an end-node of at least one edge in p. A path p from vertex v to vertex u will be denoted by $p_{v,u}$.

Physical and logical topologies are represented by undirected simple graphs: $G^\phi = (V, E^\phi)$ and $G^L = (V, E^L)$, respectively. V is the set of vertices, E^ϕ and E^L are the sets of undirected edges. In reality, not every physical node (i.e., optical switch) has an IP routing capability, which would imply $V^\phi \supseteq V^L$. All the the results in this paper hold for $V^\phi \supsetneq V^L$, but for the sake of simplicity we have chosen to keep V^ϕ and V^L identical ($V^\phi \equiv V^L \equiv V$).

B. Lightpath and Mapping

Definition 1 (Lightpath): A logical link e^L is mapped on a physical topology as a physical path p^ϕ in such a way that p^ϕ connects the same two vertices in G^ϕ as e^L connects in G^L.

In optical networking terminology, such a physical path p^ϕ is called a *lightpath*. The failure of any physical link in p^ϕ breaks the lightpath and consequently brings down the logical link e^L. Note that, since we release the capacity constraints in Sections II-VII, we do not have to consider the wavelengths assigned to lightpaths and wavelength converters placement.

Definition 2 (Mapping): Let P^ϕ be the set of all possible physical paths in the physical topology and $A \subset E^L$ be a set of logical links. A *mapping* M_A is a function $M_A : A \to P^\phi$ associating each logical link from the set A with a corresponding lightpath in the physical topology.

For some particular logical edge $e^L \in A$, M_A returns a physical path $p^\phi = M_A(e^L)$, $p^\phi \in P^\phi$. For arguments beyond A, M_A is not defined. We also allow putting a set of logical links $A_{sub} \subset A$ as an argument, which results in a set of lightpaths $M_A(A_{sub}) \subset P^\phi$. Similarly, taking as an argument a logical path p^L whose edges are in A, we obtain a set of lightpaths $M_A(p^L) \subset P^\phi$ associated with the edges of p^L.

Example 1: Fig. 1 illustrates the definitions given above. In Fig. 1a the mapping M_A is defined for the subset A of logical links (marked in bold in the logical topology). For example, we have $M_A(f^L) = \langle d^\phi, b^\phi, g^\phi \rangle$, which means that the lightpath assigned for the logical edge f^L consists of three physical links. Fig. 1b presents a mapping defined for the subset B, whereas the mapping M_{E^L} in Fig. 1c is defined for all links of the logical topology $E^L = A \cup B$.

We will often deal with mappings of different subsets of logical edges. Let $A_1, A_2 \subset E^L$. For consistency, we always require that:

$$\text{for every } e^L \in A_1 \cap A_2 : \; M_{A_1}(e^L) = M_{A_2}(e^L). \quad (1)$$

The mappings M_{A_1} and M_{A_2} can be merged, resulting in a mapping M_{A_3} defined as follows

$$A_3 = A_1 \cup A_2 \quad (2)$$

$$M_{A_3}(A_3) = M_{A_1}(A_1) \cup M_{A_2}(A_2). \quad (3)$$

[1]The terms *edge* and *link* are used interchangeably

[2]The terms *vertex* and *node* are used interchangeably

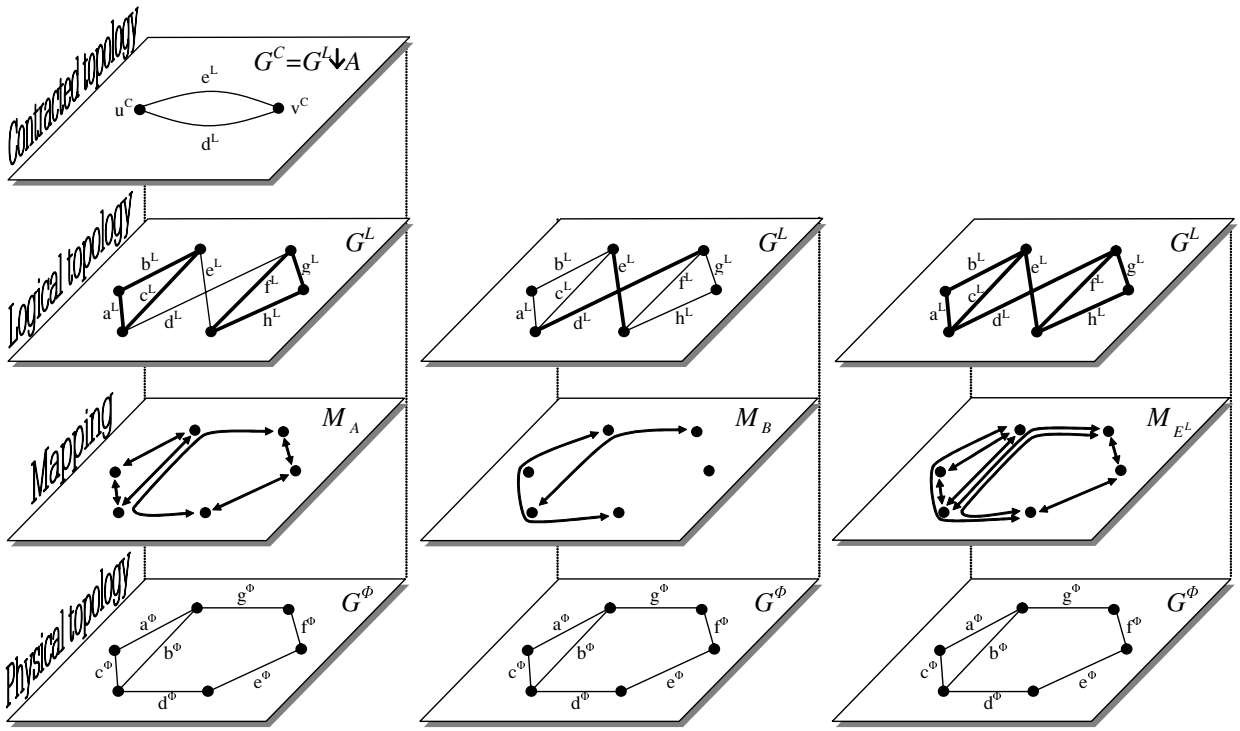


Fig. 1. Three mapping examples. We have four layers, from bottom to top: the physical topology G^ϕ, the mapping M, the logical topology G^L and the contracted logical topology G^C (only in (a)). In (a) the pairs $\left[G^L_{\{a^L,b^L,c^L\}}, M_A\right]$ and $\left[G^L_{\{f^L,g^L,h^L\}}, M_A\right]$ are survivable, and therefore the pair $\left[G^L, M_A\right]$ is piecewise survivable. In (b) the mapping M_B maps edge-disjointly the set $B = \{d^L, e^L\}$ of two logical links. The contracted topology G^C in (a) is composed of these two links. Taking G^C and M_B together, we obtain the pair $\left[G^C, M_B\right]$, which is survivable. In (c) the pair $\left[G^L, M_{E^L}\right]$ is survivable, that is M_{E^L} is a survivable mapping of the entire logical topology.

For convenience of notation, we will write (2) and (3) as $M_{A_3} = M_{A_1} \cup M_{A_2}$.

C. Contraction and Origin

In the paper we will often use the graph operator of *contraction*, which is illustrated in Fig. 2 and is defined as follows:

Definition 3 (Contraction [16]): Contracting an edge $e \in E$ of a graph $G = (V, E)$ consists in deleting that edge and merging its end–nodes into a single node. The result is called the *contraction of a graph G on an edge e* (or simply a *contracted graph*), and is denoted by $G^C = G \downarrow e$.

By extension, we also allow contracting a set of edges $A \subset E$, resulting in a contracted graph $G^C = G \downarrow A$, obtained by successively contracting the graph G on every edge of A. It is easy to show that the order in which the edges of A are taken to contraction, does not affect the final result.

Let $G = (V, E)$, $A \subset E$ and $G^C = (V^C, E^C) = G \downarrow A$. Note that by construction $E^C = E \setminus A$. Therefore each edge of G^C can be found in G, as depicted in Fig. 2. This is not always true for vertices. A vertex of V^C may either 'originate' from a single vertex in G (like w^C in Fig. 2), or from a connected subgraph of G (like v^C and u^C). We call this relation an $Origin(\cdot)$.

Definition 4 (Origin): Let $G^C = G \downarrow A$. Now take a subgraph $G^C_{sub} \subseteq G^C$. We say that $G_{sub} = Origin(G^C_{sub})$, if G_{sub} is the maximal subgraph of G that was transformed into G^C_{sub} by the contraction of A in G.

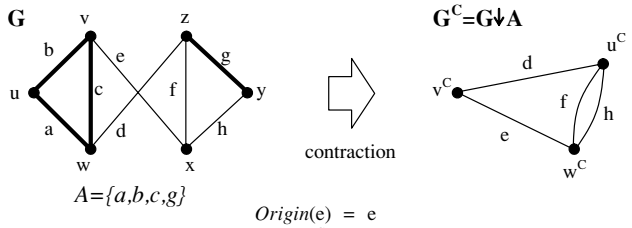


Fig. 2. Contraction of a graph G on a set of edges $A = \{a, b, c, g\}$. The origins of some elements of $G^C = G \downarrow A$ are also shown (bottom).

According to this definition, the result of the $Origin(\cdot)$ function is the *maximal* subgraph transformed in its argument. For example, we could say that in Fig. 2, the vertex $z \in G$ was transformed into the vertex $u^C \in G^C$, however $z \neq Origin(u^C)$ because it is not the only element that was transformed into u^C by contraction. The maximal subgraph in this case is $(\{y, z\}, g) = Origin(u^C)$.

D. Survivability and Piecewise Survivability

Let M_{E^L} be a mapping of the logical topology G^L on the physical topology G^ϕ. Assume that a physical link e^ϕ fails. Each logical link in G^L using e^ϕ in its mapping (lightpath) will then be cut. This may cause a disconnection of G^L. If, after any single physical link failure, the graph G^L remains connected, then the pair $\left[G^L, M_{E^L}\right]$ is declared *survivable*. We extend this property to a family of graphs constructed from the logical topology in the following definition:

Definition 5 (Survivability): Let $G^L = (V, E^L)$, $A \subset E^L$ and $G^C = (V^C, E^C) = G^L \downarrow A$. Take any connected subgraph $G^C_{sub} = (V^C_{sub}, B)$ of the contracted topology G^C, and let M_B be a mapping of the set B of logical links. The pair $\left[G^C_{sub}, M_B\right]$ is *survivable* if the failure of any single physical link e^ϕ does not disconnect the graph G^C_{sub}.

A direct consequence of Definition 5 is that if $\left[G^C_{sub}, M_B\right]$ is survivable, then $\left[G^C_{sub}, M_{B'}\right]$ is also survivable, for any $B \subset B' \subseteq E^L$.

In Definition 5, G^C_{sub} represents a large family of graphs obtained from the logical topology. If $A = \emptyset$, then $G^C = G^L$ and G^C_{sub} is any connected subgraph of G^L (including G^L itself). If $A \neq \emptyset$, then G^C_{sub} is any connected subgraph of $G^L \downarrow A$. The different instances of G^C_{sub} and survivable pairs are given in Fig. 1 and described in the following three examples:

Example 2: It is easy to check that in Fig. 1c the pair $\left[G^L, M_{E^L}\right]$ is survivable.

Example 3: In Fig. 1a, let $G^L_{\{a^L,b^L,c^L\}}$ be the subgraph of G^L defined by the edges a^L, b^L, c^L and their end-vertices. The pair $\left[G^L_{\{a^L,b^L,c^L\}}, M_A\right]$ is survivable, because the failure of any physical link does not disconnect $G^L_{\{a^L,b^L,c^L\}}$. Similarly, the pair $\left[G^L_{\{f^L,g^L,h^L\}}, M_A\right]$ is also survivable.

Example 4: In Fig. 1a, the contracted topology G^C is the result of the contraction of the logical topology on the set A, i.e., $G^C = G^L \downarrow A$. Take $G^C_{sub} = G^C$. It consists of two logical links, d^L and e^L. A possible mapping of the set $B = \{d^L, e^L\}$ is the mapping M_B shown in Fig 1b. Consider the pair $\left[G^C, M_B\right]$; it is survivable, because a single physical link failure cannot bring down both d^L and e^L at the same time, hence G^C remains connected.

Definition 6 (Piecewise Survivability): Let M_A be a mapping of a set $A \subset E^L$ on the physical topology. The pair $\left[G^L, M_A\right]$ is *piecewise survivable* if, for every vertex v^C of the contracted logical topology $G^L \downarrow A$, the pair $\left[Origin(v^C), M_A\right]$ is survivable.

In contrast to survivability, piecewise survivability is defined only for the entire logical topology G^L. We will say that a mapping M_A is (piecewise) survivable, if the pair $\left[G^L, M_A\right]$ is (piecewise) survivable (i.e., we take G^L as the default topology).

Example 5: In Fig. 1a, the pair $\left[G^L, M_A\right]$ is piecewise survivable. To prove it, we have to show that for vertices u^C and v^C of $G^L \downarrow A$, the pairs $\left[Origin(u^C), M_A\right]$ and $\left[Origin(v^C), M_A\right]$ are survivable. Here we have $Origin(u^C) = G^L_{\{a^L,b^L,c^L\}}$ and $Origin(v^C) = G^L_{\{f^L,g^L,h^L\}}$. We have shown in Example 3 that each of these two graphs forms a survivable pair with M_A.

III. Fundamental properties of survivable and piecewise survivable mappings

In this section we prove three useful properties of survivable and piecewise survivable mappings. We will often use them in the following sections.

A. The Expansion of Survivability

Given a piecewise survivable mapping, the logical topology can be viewed as a set of survivable 'pieces'. This is a general property of a piecewise survivable mapping. (For instance in Example 5, given the piecewise survivable mapping M_A, there are two survivable 'pieces' of G^L: $G^L_{\{a^L,b^L,c^L\}} \subset G^L$ and $G^L_{\{f^L,g^L,h^L\}} \subset G^L$.) The following theorem enables us to merge some of these pieces, resulting in a single large survivable piece.

Theorem 1 (Expansion of Survivability): Let M_A be a mapping of a set of logical edges $A \subset E^L$ on the physical topology G^ϕ, such that the pair $\left[G^L, M_A\right]$ is piecewise survivable. Let $G^C = G^L \downarrow A$. Take any subgraph of G^C, call it $G^C_{sub} = (V^C_{sub}, B)$. Let M_B be a mapping of the set B of edges of G^C_{sub} on G^ϕ. If the pair $\left[G^C_{sub}, M_B\right]$ is survivable then the pair $\left[Origin(G^C_{sub}), M_A \cup M_B\right]$ is also survivable.

Proof: See Appendix. ■

The following example illustrates this theorem.

Example 6: In Example 5 we have shown that in Fig. 1a, the pair $\left[G^L, M_A\right]$ is piecewise survivable. Take $G^C_{sub} = G^C = G^L \downarrow A$ and take M_B as in Fig. 1b. From Example 4, we know that the pair $\left[G^C, M_B\right]$ is survivable. Now, by Theorem 1, the pair $\left[Origin(G^C), M_A \cup M_B\right] = \left[G^L, M_A \cup M_B\right]$ is survivable. So beginning from the piecewise survivable mapping M_A and adding the mapping M_B, we merged the two survivable pieces $G^L_{\{a^L,b^L,c^L\}}$ and $G^L_{\{f^L,g^L,h^L\}}$ into a single, large, survivable piece. In this example the resulting survivable piece is the entire logical topology G^L. The full mapping $M_A \cup M_B = M_{E^L}$ is shown in Fig. 1c.

B. Invariance of Survivability Under Contraction

Theorem 2 (Invariance of Survivability Under Contraction): Let $G^C_{sub} = (V^C_{sub}, B)$ be a subgraph of some contracted topology G^C. If M_B is a mapping such that the pair $\left[G^C_{sub}, M_B\right]$ is survivable, then for any set $A \subset B$ of logical links the pair $\left[G^C_{sub} \downarrow A, M_B\right]$ is also survivable.

Proof: See Appendix. ■

In other words, Theorem 2 says that if we can map in a survivable way some subgraph G^C_{sub} of the logical or contracted logical topology, then the subgraph obtained by contracting some additional set A of edges can always be mapped in a survivable way, whatever the choice of A.

Example 7: Take $G^C_{sub} = G^L$ and $M_B = M_{E^L}$ as in Fig. 1c. We know that the pair $\left[G^L, M_{E^L}\right]$ is survivable. Theorem 2 implies that for any set of logical edges $A \subset E^L$ the pair $\left[G^L \downarrow A, M_{E^L}\right]$ is also survivable. In particular, for the set A as defined in Fig. 1a, $\left[G^L \downarrow A, M_{E^L}\right]$ is survivable, which was shown in Example 4 ($M_B \subset M_{E^L}$).

Note that we do not impose any requirements (such as e.g., preserving piecewise survivability) on the contracted edges A. Moreover, we do not have any restrictions on what happens with the rest of the contracted topology, i.e., in $G^C \setminus G^C_{sub}$.

C. The Existence of a Survivable Mapping

In general, for a given pair of physical and logical topologies, it is very difficult to verify the existence of a survivable mapping. A heuristic approach, if it fails, does not give any answer. The ILP approach or an exhaustive search could provide us with the answer, but due to their high computational complexity their application is limited to the topologies of several nodes. The following theorem shows how this verification problem can be substantially reduced:

Theorem 3 (Existence of a survivable mapping): Let M_A be a mapping of a set of logical edges $A \subset E^L$, such that the pair $\left[G^L, M_A\right]$ is piecewise survivable. A survivable mapping $M^{surv}_{E^L}$ of G^L on G^ϕ exists if and only if there exists a mapping $M^{surv}_{E^L \backslash A}$ of the set of logical links $E^L \backslash A$ on G^ϕ, such that the pair $\left[G^L \downarrow A, M^{surv}_{E^L \backslash A}\right]$ is survivable.

Proof: See Appendix. The proof uses Theorems 1 and 2. ■

The following example illustrates this theorem.

Example 8: In Fig. 1 delete edge b^ϕ from the physical topology G^ϕ. Now, for the logical topology G^L and the physical topology $G^\phi \setminus \{b^\phi\}$, a survivable mapping does not exist. To prove it, note that we can still find a mapping M_A of G^L on $G^\phi \setminus \{b^\phi\}$ that is piecewise survivable. For instance, we can take M_A defined as follows: $a^L \mapsto (c^\phi)$, $b^L \mapsto (a^\phi)$, $c^L \mapsto (d^\phi, e^\phi, f^\phi, g^\phi)$, $f^L \mapsto (d^\phi, c^\phi, a^\phi, g^\phi)$, $g^L \mapsto (f^\phi)$ and $h^L \mapsto (e^\phi)$. However, the remaining two logical links d^L and e^L cannot be mapped edge-disjointly on $G^\phi \setminus \{b^\phi\}$. Therefore no survivable mapping $M_{\{d^L, e^L\}}$ of the contracted logical topology $G^L \downarrow A$ on $G^\phi \setminus \{b^\phi\}$ exists. Consequently, by Theorem 3 we know that no survivable mapping of G^L on $G^\phi \setminus \{b^\phi\}$ exists, which was to be proved. Note that to prove it, we only considered the two-edge topology $G^L \downarrow A$ instead of the entire G^L, which greatly simplified the problem. Clearly, the larger the set A, the more we benefit from Theorem 3.

IV. The SMART Algorithm

In this section we present an algorithm that searches for a survivable mapping, which we call SMART. It maps the topology part by part, gradually converging to a final solution. By formal graph theoretic analysis, we prove that if SMART converges completely, a *survivable* mapping is found. Otherwise, when the algorithm terminates before its complete convergence, the returned mapping is *piecewise survivable* and no survivable solution exists.

A. The pseudo-code of SMART

Step 1 Begin from the full logical topology $G^C = G^L$, and an empty mapping $M_A = \emptyset, A = \emptyset$;
Step 2 Take some subgraph $G^C_{sub} = (V^C_{sub}, B)$ of G^C and find a mapping M_B, such that the pair $\left[G^C_{sub}, M_B\right]$ is survivable. IF no such pair exists, THEN RETURN M_A AND $G^C = G^L \downarrow A$, END.
Step 3 Update the mapping by merging M_A and M_B, i.e., $M_A := M_A \cup M_B$;
Step 4 Contract G^C on B, i.e., $G^C := G^C \downarrow B$;
Step 5 IF G^C is a single node, THEN RETURN M_A, END.
Step 6 GOTO Step 2

The SMART algorithm starts from an empty mapping $M_A = \emptyset$. At each iteration it maps some set B of logical links (Step 2), and extends the mapping M_A by M_B (Step 3). Meanwhile, the contracted topology G^C gradually shrinks (Step 4).

B. The Correctness of the SMART Algorithm

We declare that:

- *SMART converges* if the contracted topology G^C converges to a *single* node. We prove later in Corollary 1, that the mapping M_A returned in step 5 is then a survivable solution;
- *SMART does not converge* if SMART terminates before G^C converges to a single node, i.e., in Step 2. We prove below in Theorem 4 that the mapping M_A returned in Step 2 piecewise survivable. Moreover, we show in Corollary 1 that in this case a survivable solution does not exist. The graph $G^C = G^L \downarrow A$ (also returned in Step 2) is called the *remaining contracted logical topology* as it consists of unmapped logical links $E^L \backslash A$.

Theorem 4 (SMART's piecewise survivability): After each iteration of the SMART algorithm, the pair $\left[G^L, M_A\right]$ is piecewise survivable.

Proof: See Appendix. ■

Theorem 4 leads us to the following important property of the SMART procedure:

Corollary 1 (SMART's Convergence): The SMART algorithm returns a single node contracted topology G^C, if and only if there exists a survivable mapping of the logical graph G^L on the physical graph G^ϕ. In this case the returned mapping M_A is survivable.

Proof: See Appendix. ■

G^C may converge to a single node topology with *self-loops*; they form a set of remaining unmapped logical links $E^L \backslash A$. However, this does not affect the result, because the links of $E^L \backslash A$ may be mapped in any way (e.g. shortest path) to obtain a full survivable mapping M_{E^L}.

C. The Order of a Sequence of Subgraphs

Recall that in Step 2 of the SMART algorithm we take some subgraph $G^C_{sub} = (V^C_{sub}, B)$ of the contracted topology G^C. We do not specify which subgraph to take; if there are more candidates G^C_{sub} that meet the condition given in Step 2 (which is usually the case), we are free to choose any of them. This raises a natural question: How does the choice of G^C_{sub} affect

the convergence of the SMART algorithm? In the following theorem we show that this choice does *not* affect the outcome of the SMART algorithm.

Theorem 5 (SMART's Uniqueness):
The contracted topology G^C_{min} (excluding self-loops) returned by SMART is unique.

Proof: See Appendix. ■

V. Implementation - SMART-H

In the previous section we have proposed a general procedure, called SMART, for which we have proved a number of important properties. In practice, however, an exact implementation of SMART (as described by the pseudo-code in Section IV-A) is not feasible, because Step 2 is in general an NP-complete problem. For this reason, for large topologies we use a heuristic to solve Step 2. We will refer to the SMART procedure that takes use of any heuristic in Step 2 as "SMART-H." In this section we first discuss which theoretical results carry over from SMART to SMART-H. Next, we give a simple and effective example of a heuristic for solving Step 2.

A. Which theoretical results hold for SMART-H?

With a heuristic used to solve Step 2, SMART-H can terminate in Step 2 not only if a pair $\left[G^C_{sub}, M_B\right]$ does not exist, but also if our heuristic was not able to find one. However, most of the theoretical results obtained for SMART in Section IV carry over to SMART-H and can be successfully applied as we show in Sections VI and VII. In particular:

- Theorem 4 holds.
- Corollary 1 holds only in one direction: If SMART-H converges, then the returned mapping is survivable (see Proof of Corollary 1). Otherwise, the returned mapping is piecewise survivable (by Theorem 4), but a survivable solution might still exist. However, the piecewise survivability of the returned mapping can be effectively exploited to verify the existence of a survivable solution, as we demonstrate in Section VI-A.
- Theorem 5 does no longer hold. In practice, however, it is highly probable that if our heuristic for Step 2 can find a survivable mapping of some G^C_{sub} then it will also find a survivable mapping of its contracted version $G^C_{sub} \downarrow A$. (For instance, it is harder to map disjointly all edges of a cycle, than only a subset of them.) With this property, distinct runs of SMART-H converge to the same contracted topology. We have found an excellent confirmation of this claim in simulations. We have tested many pairs of physical and logical topologies for which a survivable mapping does not exist (so G^C_{min} is always larger than a one-node topology). To ensure a variety of sequences of cycles considered in Step 2 of SMART-H, we randomly permute every time the list of candidates. Even with this approach, for a given pair of topologies, more than 99% of distinct runs of SMART-H resulted in the same contracted topology.

B. A heuristic for Step 2 of SMART-H

In our implementation of Step 2 of SMART-H we take the graph G^C_{sub} in the form of a *cycle*. Thus we will systematically contract cycles (or 'rings') found in the contracted logical topology, which explains the name of the algorithm ("Survivable Mapping Algorithm by Ring Trimming"). G^C_{sub} in the form of a cycle requires the mapping M_B (Step 2) to be *edge-disjoint*. (Otherwise, if the same physical link e^ϕ is used by two or more logical links in G^C_{sub}, a failure of e^ϕ will bring these links down, disconnecting the cycle G^C_{sub}.) Since finding it is equivalent to the NP-complete edge-disjoint paths problem [17], we applied a simple heuristic, as follows. Let each physical edge have a weight (these weights will be used only by this heuristic) that is initially set to 1. At each iteration, map the logical links from G^C_{sub} with the shortest path. If no physical link is used more than once, the disjoint solution is found. Otherwise, the weight of each physical link used more than once is increased, and a new iteration starts. After several unsuccessful iterations the heuristic fails.

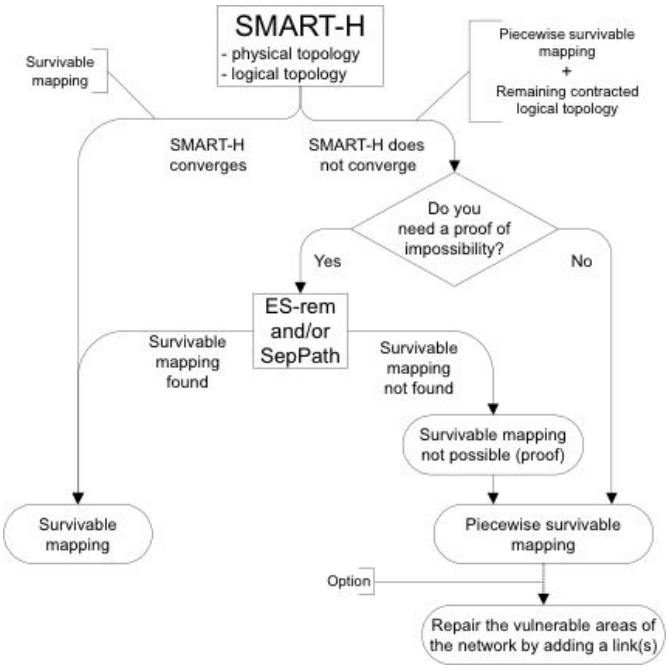


Fig. 3. Applications of the SMART-H algorithm.

VI. SMART-H Applications

We can apply the SMART-H algorithm in a number of ways. The general scheme can be found in Fig. 3. The option we choose depends on the nature of the results we want to obtain. Specifically we can distinguish:

- the formal verification of the existence of a survivable mapping,
- a tool tracing and repairing the vulnerable areas of the network,
- a fast heuristic.

Note that the first two applications are specific to SMART-H; no other heuristic proposed to date has this property. We discuss each of the three applications separately, in the following sections.

A. Formal Verification of the Existence of a Survivable Mapping (ExSearch and SepPath)

Run SMART-H to map a logical topology G^L on the physical topology G^ϕ. If SMART-H converges, a survivable map-

ping exists and is returned (see the comment on Corollary 1 in V-A). If SMART-H does not converge, it returns a mapping M_A and a remaining contracted logical topology $G^L \downarrow A$. What makes SMART-H very different from other heuristics is that, by Theorem 4, the returned mapping M_A is piecewise survivable. This allows us to apply Theorem 3, which reduces the task of verifying the existence of a survivable mapping for the entire G^L, to the same verification for $G^L \downarrow A$. This property is a key feature of SMART-H: if there is a survivable mapping of G^L on G^ϕ, then SMART-H *will never miss it*, because the set of the remaining logical links $E^L \backslash A$ can be still mapped in a way that preserves the survivability of $G^L \downarrow A$ (and hence of G^L). In other words, *no backtracking is needed* in order to assess if there exists a survivable mapping for G^L; we obtain exactly the same answer by studying $G^L \downarrow A$. Although this verification is NP-complete for both G^L and $G^L \downarrow A$, in practice the size of $G^L \downarrow A$ is often very small, which makes the verification feasible. We use two methods to verify the existence of a survivable mapping for $G^L \downarrow A$:

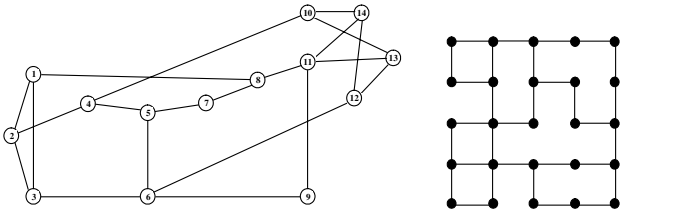
1) Exhaustive Search (*ExSearch*) uses exhaustive search to find a survivable mapping of the contracted logical topology $G^L \downarrow A$.
2) Separated Path check (*SepPath*) is defined as follows. If the contracted logical topology $G^L \downarrow A$ contains a path p^C such that all nodes on p^C but the first and the last ones, are of degree two, then clearly all the logical links in p^C must be mapped edge-disjointly to enable survivability. Therefore the failure of an exhaustive search for an edge-disjoint mapping of p^C will prove impossibility. Otherwise, nothing can be concluded.

Compared to an unrestricted exhaustive search, the exhaustive search respecting the edge-disjointness constraint is relatively easy (although still NP-complete). For this reason SepPath is better suited to larger topologies than ExSearch.

The implementations that first use SMART-H, and then apply ExSearch or SepPath to the returned contracted topology $G^L \downarrow A$, will be called SMART-H + ExSearch and SMART-H + SepPath, respectively. We test the efficiency of these approaches in Section VII-B; we apply them to verify the existence of a survivable mapping for various topologies in Section VII-C.

B. A Tool Tracing and Repairing the Vulnerable Areas of the Network

We have developed two methods to verify the existence of a survivable mapping: ExSearch and SepPath. Once we know that a particular pair of physical and logical topologies cannot be mapped in a survivable way, a natural question is to modify the topologies to enable such a mapping. Where should a new link be added? The SMART-H algorithm helps us in answering this question. Run SMART-H and wait until it terminates. The remaining contracted logical topology $G^L \downarrow A$ and the piecewise-survivable mapping M_A are returned. Choose at random two nodes u^C, v^C in $G^L \downarrow A$ and pick any two nodes u, v in G^L, such that $u \in Origin(u^C)$ and $v \in Origin(v^C)$. Now connect u and v with an additional logical/physical link (remember that we assume identical vertices at both layers). If this link already exists, repeat the procedure.



a) NSFNET b) f-lattice (2-node-connected)

Fig. 4. Physical topologies used in simulations. (a) NSFNET; (b) f–lattice constructed from full square lattice by deleting fraction f of links, while preserving 2–node–connectivity (here $f \simeq 0.25$).

The simulation results in Section VII-D discuss the efficiency of this approach.

C. A Fast Heuristic

With SMART-H, a survivable mapping is found orders of magnitude more rapidly and usually more often than with other heuristics proposed in the literature to date (see [15]).

VII. Simulation Results

In this section we test in simulations the novel applications of SMART-H, i.e., these described in Section VI-A and VI-B. The performance of SMART-H as a fast heuristic (see VI-C) has already been evaluated in [15]; we do not repeat it here due to space constraints.

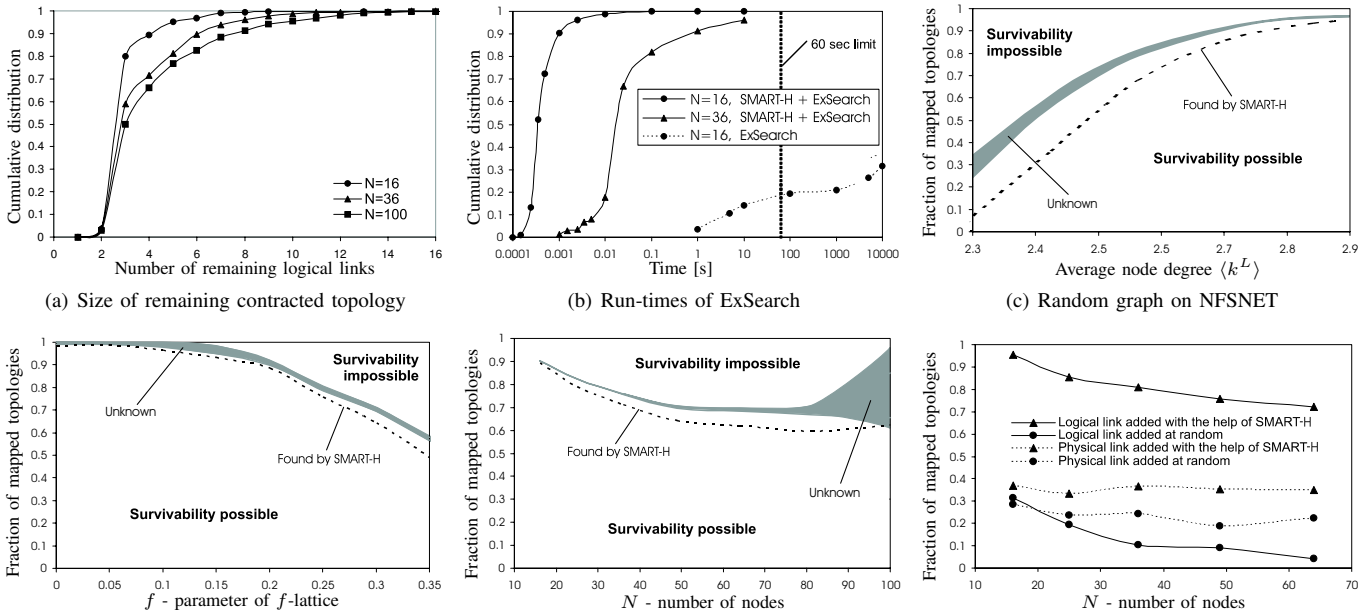
A. Physical and Logical Topologies

In the simulations we use various topologies. A relatively small physical topology is NSFNET (14 vertices, 21 edges) presented in Fig. 4a. To imitate larger real-life physical topologies, we also generate square lattices in which a fraction f of edges is deleted, as shown in Fig. 4b; we call them f-lattices. The parameter f is often fixed to $f = 0.3$, which resulted in an f-lattice with an average vertex degree slightly smaller than that of NSFNET. As the IP graph is less regular (for instance, there is no reason why it should be planar), the logical topologies are 2-edge-connected random graphs of various average vertex degree. (Clearly, 2-edge-connectivity of both physical and logical topologies is a necessary condition for the existence of a survivable mapping.)

B. ExSearch and SepPath Efficiency, and 'Unknown Area'

In Section VI-A we defined two methods of verification of the existence of a survivable mapping, ExSearch and SepPath. In this section we examine the benefits of these approaches.

The physical topology is an f-lattice with the parameter $f = 0.3$. The logical topology is a random graph with average vertex degree $\langle k^L \rangle = 4$. For each number of nodes N, we generate a number of physical/logical topology pairs, and keep the first 1000 for which SMART-H *does not converge*. In Fig. 5a, we present the cumulative distribution function (CDF) of the number of logical links in the remaining contracted logical topology $G^L \downarrow A$ returned by the algorithm. We can see that, if SMART-H does not converge, the size of $G^L \downarrow A$ is usually relatively small. For instance, for $N = 36$, SMART-H



(a) Size of remaining contracted topology (b) Run-times of ExSearch (c) Random graph on NFSNET

(d) Random graph on f-lattice, $N = 49$ (e) Random graph on f-lattice, $f = 0.3$ (f) Introduction of a new link

Fig. 5. Survivability under various scenarios. Parameters: N–number of nodes; f–parameter of the f-lattice physical topology; $\langle k^L \rangle$–average node degree of the logical topology. (a) CDF of the number of logical links in the remaining contracted logical topology. $f = 0.3$, $N = 16 \ldots 100$; (b) CDF of ExSearch times with and without prior contraction by SMART-H. $f = 0.3$, $N = 16 \ldots 100$. (c) Random graph logical topologies mapped on NSFNET. $N = 14$, $\langle k^L \rangle = 2.3 \ldots 2.9$; (d) Random graph logical topologies mapped on f-lattices. $N = 49$, $f = 0 \ldots 0.35$, $\langle k^L \rangle = 4$; (e) Random graph logical topologies mapped on f-lattices. $N = 16 \ldots 100$, $f = 0.3$, $\langle k^L \rangle = 4$; (f) Enabling survivability by introducing an additional link. Random graph logical topologies mapped on f-lattices. $N = 16 \ldots 100$, $f = 0.3$, $\langle k^L \rangle = 4$.

leaves six or fewer logical links out of the total number of 72, in about 80% of cases. Moreover, this property seems to depend only slightly on the topology size.

Now we apply EsSearch and SepPath to the contracted $G^L \downarrow A$ returned by SMART-H. The distribution of run-times of SMART-H + ExSearch is plotted in Fig. 5d. For $N = 16$, about 90% of topologies need less than 0.001 sec to run SMART-H + ExSearch.[3] Only very few need more than 0.1 sec. For comparison purposes, we also ran a complete exhaustive search ExSearch without prior contraction by SMART-H for $N = 16$. We observe the difference in run-times of at least 7 orders of magnitude. Most of the runs of the fully exhaustive search last more than 10000 seconds ($\sim$ 3 hours), the maximal time allowed in the simulations. This limits the application of the full exhaustive search to the topologies of a very small size.

Fig. 5d also exemplifies the tradeoff we faced in simulations. On one hand, the SMART-H + ExSearch runs quickly for the majority of the topologies, but on the other hand, the remaining few topologies take orders of magnitude more time. We observed the same phenomenon when applying the SMART-H + SepPath verification method. Therefore we have decided to use a strict, *one minute stopping time*. If neither SMART-H + ExSearch nor SMART-H + SepPath finishes the computation within 60 seconds, the question of the existence of a survivable mapping is left unanswered. As the result, the figures in the following sections display two curves: the lower one is the percentage of survivable mappings found within 1 minute, the upper one is the percentage of logical topologies proved to be unmappable in a survivable way within 1 minute. The curves are separated by an 'unknown area' set in gray.

[3]We implemented the all algorithms in C++ and ran it on a Pentium 4 machine.

C. Survivability of Random Graphs on Various Physical Topologies

It is interesting to see what fraction of randomly chosen topologies can/cannot be mapped in a survivable way. To the best of our knowledge, this is the first time these results can be obtained in a reasonable time for moderate and large topologies.

For a particular pair of physical and logical topologies, we first apply the SMART-H algorithm. If SMART-H does not converge, we try ExSearch and SepPath to verify the existence of a survivable solution. Their run-times are restricted to the 'one minute bound', as explained in Section VII-B.

In Fig. 5c we present the results of the mapping of random graph logical topologies on NSFNET. We vary the average vertex degree $\langle k^L \rangle$ of the logical graph; for each value of $\langle k^L \rangle$ we generate 1000 topologies. Observe that the results strongly depend on $\langle k^L \rangle$.

In order to examine a larger spectrum of physical topologies and topology sizes, in Figs. 5de we map a random graph logical topology on the f-lattice physical topology. This time we fix the average vertex degree of the logical topology $\langle k^L \rangle = 4$ and we vary the parameter f of the physical topology (Fig. 5d) or the number of nodes N (Fig. 5e). We generated 1000 topologies for each parameter. Fig. 5d shows that the fraction of topologies mappable in a survivable way decreases with growing f. This was expected, because it is more difficult to map the logical topology on a sparser physical graph. In

Fig. 5e, the 'unknown area' quickly widens for $N > 80$ because of the 'one minute bound'.

The dashed curves in Figs. 5cde show the fraction of topologies mapped in a survivable way by SMART-H alone, without being followed by ExSearch or SepPath. The distances between these curves and the mapping-impossible areas are relatively small, which confirms the high efficiency of SMART-H as a heuristic.

D. Introduction of an Additional Link

Another property of the SMART-H algorithm is the ability to trace and repair the vulnerable areas of the network. In particular, in Section VI-B we described a way to introduce an additional logical or physical link to enable a survivable mapping. In this section we verify the efficiency of that approach.

We map random graph logical topologies on f-lattices and vary N. For each N, we generate 1000 pairs of physical and logical topologies, such that for each pair separately, a survivable mapping does not exist. For each topology pair, we add one logical or physical link with the help of SMART-H, as described in Section VI-B. Next, the existence of a survivable mapping is verified again, for this extended pair of topologies. For comparison purposes we also simulate a completely random placement of an additional link.

The results are shown in Fig. 5f. For better readability, we do not include the 'unknown area', which lies above each curve. The application of SMART-H enables a very efficient placement of an additional *logical* link, which helps in 70% to 95% of cases (depending on N). In contrast, the completely random placement helps far less, and only for small topologies - for larger N its efficiency becomes insignificant. This is because only new logical links connecting *different* nodes in $G^L \downarrow A$ (i.e., different survivable pieces in G^L) may help; the larger the topology, the lower the probability of achieving it with a completely random placement.

The efficiency of the placement of a new *physical* link has a more random nature. Again, the SMART-H approach helps, but its effect is not as significant nor dependent on N, as in the case of logical links. This is because the introduction of a new physical link *within the same* survivable piece may also help. For instance, in Example 8, for the logical topology G^L and the physical topology $G^\phi \backslash \{b^\phi\}$, the piecewise survivable mapping M_A consists of two survivable pieces $\{a^L, b^L, c^L\}$ and $\{f^L, g^L, h^L\}$. We have shown that for this pair of topologies a survivable mapping does not exist. But it is enough to add the physical link b^ϕ to make a survivable mapping possible, as shown in Fig. 1c. Note that the link b^ϕ lies within the survivable piece $\{a^L, b^L, c^L\}$.

VIII. Capacity Constraints

The SMART approach described in this paper is a powerful tool for studying the survivability of two-layer systems (such as IP-over-WDM) from a *topological* perspective. Therefore, as in the approaches in [9], [12], we have assumed infinite capacities (number of wavelengths) on each physical fiber. This has pros and cons. On one hand, this makes 'negative results' more general: if we prove that a survivable mapping does not exist for a particular pair of physical and logical topologies with infinite physical capacities, then this proof holds for any combination of finite capacities. On the other hand, a 'positive result' (i.e., a survivable mapping) found for infinite capacities is not necessarily applicable to a scenario with given capacity constraints.

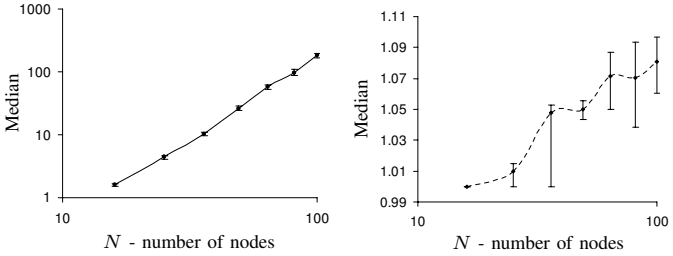
In this section we show that SMART, although primarily devised to tackle purely topological aspects of the survivability problem, can also incorporate some additional real-life constraints, which makes our approach an interesting alternative for the heuristics available in the literature.

We first review the theorems given in this paper. Assume that the capacity constraints are respected at every moment. This means, for example, that in Step 2 of the SMART algorithm the pair $\left[G^C_{sub}, M_B\right]$ must be not only survivable, but that the resulting mapping $M_A \cup M_B$ (Step 3) should not exceed the capacity of any physical link. With this assumption it is easy to see that Theorems 1 and 4 hold. Corollary 1 holds only in one direction, i.e., if the contracted topology G^C converges to a single node then the underlying mapping is survivable and capacity constraints are not violated. Unfortunately, Theorems 2, 3, and 5 do not hold, because their proofs implicitly require unlimited capacities of physical links. Therefore the SMART algorithm cannot be used to verify the existence of a survivable mapping respecting capacity constraints; its functionality is reduced to that of a pure heuristic. In the reminder of this section we evaluate the effectiveness and speed of this heuristic.

In the implementation of the SMART algorithm with the capacity constraints (called "SMART-C") we have exploited the following idea. First, we construct a "light" survivable mapping that uses relatively few logical links. This is achieved by accepting in Step 2 only the mappings that are not significantly longer than their shortest path counterpart. Next, we map the remaining logical links trying to satisfy the capacity constraints, instead of applying the shortest path mapping suggested in Section IV.

As a benchmark, we take one of the most efficient and widely used heuristic to solve a survivable mapping problem, Tabu Search [12], [18], [13], [11]. We have followed the implementation given in [18]; it is a version of [12] that takes the capacity constraints into account. We refer to this algorithm as Tabu98.

We compare SMART-C with Tabu98 in Fig. 6. On one hand, Tabu98 always finds slightly better solutions than SMART-C - on average Tabu98 needs several percent smaller physical link capacities to succeed (see Fig. 6b). On the other hand, the comparison of the run–times of the examined algorithms (see Fig. 6a) reveals a big difference that grows with the network size to several orders of magnitude. These results are not surprising. At every iteration, Tabu98 examines the mapping of the entire logical topology, whereas SMART-C processes only a small portion of it. This gives Tabu98 a global view and results in a slightly better routing of lightpaths, but at the very significant cost of speed and scalability. Therefore, with finite link capacities, the two approaches are complementary.



(a) Time ratio ($\frac{T_{tabu}}{T_{smart}}$) (b) Capacity ratio ($\frac{C_{smart}}{C_{tabu}}$)

Fig. 6. Survivability with capacity constraints: SMART-C vs. Tabu98. Two–edge–connected random logical topologies of average node degree $\langle k^L \rangle = 4$ are mapped on the f-lattice physical topology, with $f = 0.3$. The size of topologies ranges from $N = 16$ to 100. For each value of N we generate 500 topology pairs and run SMART-C and Tabu98 until they succeed (i.e., find a survivable mapping that meets the capacity constraints) or the maximum number of iterations is reached. For the plots we took only the cases where both SMART-C and Tabu98 succeeded (Tabu98 succeeded less often then SMART-C). All fibers in physical topology have the same capacity, chosen separately for every pair of topologies. For every pair of topologies we compare SMART-C and Tabu98 with respect to two metrics: (a) (log-log) Comparison of run-times T_{smart} of SMART-C with run-times T_{tabu} of Tabu98. We fix the capacities of the physical links to the lowest value C_{smart} for which SMART-C succeeds. Tabu98 is run using the same capacity C_{smart}. (b) (log-lin) Comparison of minimal capacities. We compare the value C_{smart} with the minimal value C_{tabu} of capacities for which Tabu98 succeeds. In (b) the run-times are ignored. In both figures, all points are the medians over all topology pairs taken into account; the confidence interval for medians are computed at 0.95 confidence level.

IX. Conclusion

In this paper we defined a *piecewise survivable mapping* which preserves the survivability of some subgraphs of the logical topology. The formal analysis of the piecewise survivable mapping enabled us to specify the necessary and sufficient conditions for the existence of a survivable mapping. This has led us to the SMART algorithm that is guaranteed to converge to a survivable mapping if and only if it exists. As one iteration of SMART is an NP-complete problem, we adapted it by using a heuristic, which results in the SMART-H algorithm. Most of the theoretical results obtained for SMART carry over to SMART-H, which makes the latter a practical tool that substantially simplifies the verification of the existence of a survivable mapping. A second application of SMART-H is tracing vulnerable areas in the network and pointing where new link(s) should be added to enable a survivable mapping. Finally, SMART-H can serve as an efficient and scalable heuristic that searches for a survivable mapping. To conclude, the formal analysis of the piecewise survivable mapping gives us a powerful tool to designing, diagnosing and upgrading the topologies in IP over WDM networks.

X. Appendix

In this section we prove all Theorems introduced before. For this purpose we use the following definition of survivability, equivalent to Definition 5.

Definition 7 (Survivability): Let $G^L = (V, E^L)$, $A \subset E^L$ and $G^C = (V^C, E^C) = G^L \downarrow A$. Take any connected subgraph $G^C_{sub} = (V^C_{sub}, B)$, $B \subseteq E^C$, of the contracted topology G^C, and let M_B be a mapping of the set B of logical links. The pair $\left[G^C_{sub}, M_B\right]$ is *survivable* if for any physical link e^ϕ and for any two vertices $u, v \in V^C_{sub}$, there exists a path $p^C_{u,v}$ in G^C_{sub} between vertices u and v, such that $e^\phi \notin M_B(p^C_{u,v})$.
(Note that every path in the contracted topology, e.g., $p^C_{u,v}$, actually consists of logical links.)

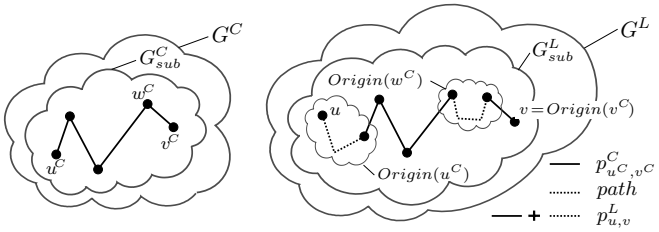


Fig. 7. Illustration of proof of Theorem 1. A first portion of the path $p^L_{u,v}$ is the path $p^C_{u^C,v^C}$ found in G^C_{sub}. Next it is completed, where necessary, with the patches found in origins of the nodes of $p^C_{u^C,v^C}$.

Proof of Theorem 1: (Please refer to Fig. 7.)
First note that since $G^C = G^L \downarrow A$, no logical edge from the set A can be found in G^C, which implies that $A \cap B = \emptyset$. Therefore the operation $M_A \cup M_B$ is always well defined, as in (2) and (3).
Let $M_{A\cup B} = M_A \cup M_B$ and $G^L_{sub} = Origin(G^C_{sub})$. We have to prove that the pair $\left[G^L_{sub}, M_{A\cup B}\right]$ is survivable. Take any single physical link e^ϕ and two vertices $u, v \in G^L_{sub}$. According to Definition 7 we have to show that there exists a path $p^L_{u,v}$ in G^L_{sub} such that $e^\phi \notin M_{A\cup B}(p^L_{u,v})$. The path $p^L_{u,v}$ is constructed in two steps, (i) and (ii).
(i) A first portion of $p^L_{u,v}$ is found in the contracted graph G^C (recall that G^C consists of logical edges), as follows. Call $u^C, v^C \in V^C_{sub}$ the vertices in $G^C_{sub} = (V^C_{sub}, B)$ whose origins contain u and v, respectively, i.e., such that $u \in Origin(u^C)$ and $v \in Origin(v^C)$. Find a path $p^C_{u^C,v^C}$ in G^C_{sub}, such that $e^\phi \notin M_B(p^C_{u^C,v^C})$. This is always possible since the pair $\left[G^C_{sub}, M_B\right]$ is survivable. We take $p^C_{u^C,v^C}$ as the first portion of $p^L_{u,v}$.

(ii) We now turn our attention to the origins of vertices in the path $p^C_{u^C,v^C}$. Take any two consecutive edges a^L and b^L of $p^C_{u^C,v^C}$, and let w^C be their common end–node in G^C_{sub}. If $Origin(w^C)$ is not a single node in G^L_{sub}, then a^L and b^L might not have a common end–node in G^L_{sub}. However, by piecewise survivability of $\left[G^L, M_A\right]$, the pair $\left[Origin(w^C), M_A\right]$ is survivable. Therefore, if we denote respectively by $v_a, v_b \in Origin(w^C)$ the end–nodes of a^L and b^L, that belong to $Origin(w^C)$, we can find a logical path $p^L_{v_a,v_b}$ in $Origin(w^C)$ connecting v_a and v_b, such that $e^\phi \notin M_A(p^L_{v_a,v_b})$. We call this path a patch of w^C and denote it by $patch(w^C)$. If for a given w^C, the edges a^L and b^L have a common end–node v^L in G^L_{sub} then $patch(w^C) = v^L$.
For every vertex $w^C \in p^C_{u^C,v^C}$, find $patch(w^C)$. If $w^C = u^C$ then $patch(u^C)$ will connect the logical vertex u with an end–node of the first logical edge in $p^C_{u^C,v^C}$, instead of connecting two end–nodes. The same holds for $w^C = v^C$.
To summarize, in step (i) we have found the path $p^C_{u^C,v^C}$ in the contracted subgraph G^C_{sub}. Next, in step (ii), we have constructed a set of *patches* for each vertex of this path. Now

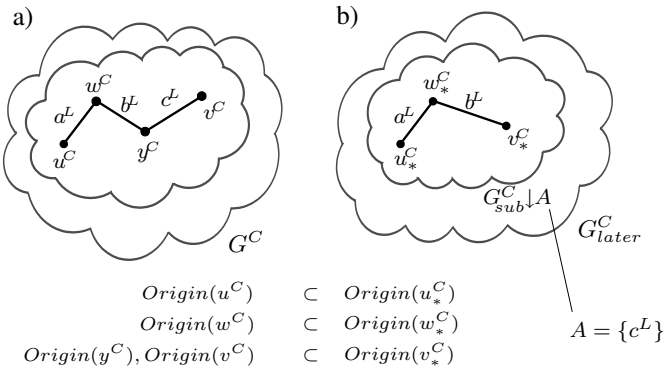


Fig. 8. Illustration of the proof of Theorem 2. (a) The original subgraph G^C_{sub} and a path $p^C_{u^C,v^C}$ that avoids e^ϕ in its mapping. (b) The subgraph G^C_{sub} contracted on the set $A = \{c^L\}$ of logical edges; the resulting subgraph is denoted by $G^C_{sub}\downarrow A$. The path $p^C_{u^C_*,v^C_*}$ originates from $p^C_{u^C,v^C}$, hence it also avoids e^ϕ in its mapping.

we combine steps (i) and (ii) to obtain the full path $p^L_{u,v}$:

$$p^L_{u,v} = p^C_{u^C,v^C} \cup \Big\{ \bigcup_{w^C \in p^C_{u^C,v^C}} patch(w^C) \Big\}. \quad (4)$$

The logical path $p^L_{u,v}$ connects the vertices u and v and has been constructed in such a way, that

$$e^\phi \notin M_B(p^C_{u^C,v^C}) \quad (5)$$
$$e^\phi \notin M_A(patch(w^C)) \quad \text{for every } w^C \in p^C_{u^C,v^C}. \quad (6)$$

Since $M_A \cup M_B = M_{A\cup B}$ and $A \cap B = \emptyset$, we can rewrite (5) and (6) as

$$e^\phi \notin M_{A\cup B}(p^C_{u^C,v^C}) \quad (7)$$
$$e^\phi \notin M_{A\cup B}(patch(w^C)) \quad \text{for every } w^C \in p^C_{u^C,v^C}. \quad (8)$$

Combining (4), (7) and (8) yields finally that $e^\phi \notin M_{A\cup B}(p^C_{u,v})$, which proves the claim. ■

Proof of Theorem 2 [Please refer to Fig. 8]
Take any single physical link $e^\phi \in E^\phi$ and two vertices $u^C_*, v^C_* \in G^C_{sub}\downarrow A$. According to Definition 7 we have to show that there exists a path $p^C_{u^C_*,v^C_*}$ in $G^C_{sub}\downarrow A$ such that $e^\phi \notin M_B(p^C_{u^C_*,v^C_*})$.
First, find in G^C_{sub} two vertices $u^C, v^C \in V^C_{sub}$, such that

$$Origin(u^C) \subseteq Origin(u^C_*), \text{ and} \quad (9)$$
$$Origin(v^C) \subseteq Origin(v^C_*). \quad (10)$$

Note that since $G^C_{sub}\downarrow A$ is created by contracting some edges in G^C_{sub}, vertices u^C and v^C always exist (they are not necessarily unique). Since the pair $\left[G^C_{sub}, M_B\right]$ is survivable, there exists a path $p^C_{u^C,v^C}$ in G^C_{sub} such that $e^\phi \notin M_B(p^C_{u^C,v^C})$. Define a sequence of logical edges p^C_* by contracting in $p^C_{u^C,v^C}$ all edges that exist also in A, i.e.,

$$p^C_* = p^C_{u^C,v^C} \downarrow (A \cap p^C_{u^C,v^C}). \quad (11)$$

Since $p^C_{u^C,v^C}$ is a path in G^C_{sub}, and since the contraction an edge merges its two end-nodes and thus preserves its continuity, p^C_* is a path in $G^C_{sub}\downarrow A$. Moreover, relations (9,10) imply that the path p^C_* connects u^C_* and v^C_* in $G^C_{sub}\downarrow A$. Finally, $e^\phi \notin M_B(p^C_{u^C,v^C})$ and (11) yields that $e^\phi \notin M_B(p^C_*)$. Therefore p^C_* is the path $p^C_{u^C_*,v^C_*}$ that we are searching for. ■

Proof of Theorem 3:
$\Leftarrow$ We know that the pair $\left[G^L, M_A\right]$ is piecewise survivable. Suppose that there exists a mapping $M^{surv}_{E^L\setminus A}$, such that the pair $\left[G^L\downarrow A, M^{surv}_{E^L\setminus A}\right]$ is survivable. Then, by Theorem 1, the pair $\left[Origin(G^L\downarrow A),\ M_A \cup M^{surv}_{E^L\setminus A}\right] = \left[G^L,\ M_A \cup M^{surv}_{E^L\setminus A}\right]$ is also survivable. So the mapping $M^{surv}_{E^L} = M_A \cup M^{surv}_{E^L\setminus A}$ is a survivable mapping of G^L on G^ϕ.
$\Rightarrow$ Assume that a survivable mapping of G^L on G^ϕ exists, call it $M^{surv}_{E^L}$. Now, by taking $G^C_{sub} := G^L$ and $M_B := M^{surv}_{E^L}$, Theorem 2 yields that $\left[G^L\downarrow A, M^{surv}_{E^L}\right]$ is survivable. Consequently, the pair $\left[G^L\downarrow A, M^{surv}_{E^L\setminus A}\right]$ is also survivable. ■

Proof of Theorem 4: [By induction]
INITIALIZATION:
Initially $G^C = G^L$. Therefore the origin of any vertex $v^C \in V^C$ is a single node in G^L, and it cannot be disconnected. Hence for every $v^C \in V^C$, the pair $\left[Origin(v^C), M_A\right]$ is survivable and consequently the pair $\left[G^L, M_A\right]$ is piecewise survivable.
INDUCTION:
Assume that after some iteration the pair $\left[G^L, M_A\right]$ is piecewise survivable. We have to prove that after the next iteration of the algorithm, the updated mapping $\widehat{M}_A$ will still form a piecewise survivable pair $\left[G^L, \widehat{M}_A\right]$.
One iteration of the SMART algorithm consists of Steps 2, 3 and 4, which we recall here:
2. Find $G^C_{sub} = (V^C_{sub}, B)$ and M_B, such that the pair $\left[G^C_{sub}, M_B\right]$ is survivable.
3. $\widehat{M}_A := M_A \cup M_B$
4. $\widehat{G}^C := G^C \downarrow B$
(For clarity we indicated the updated M_A and G^C by a hat: '$\widehat{\ }$')
The updated contracted topology $\widehat{G}^C = (\widehat{V}^C, \widehat{E}^C)$ was created from G^C by replacing $G^C_{sub} = (V^C_{sub}, B)$ by a single node, which we call $\widehat{v}^C_{sub}$; the remaining nodes stayed unchanged. So $\widehat{V}^C = \{\widehat{v}^C_{sub}\} \cup V^C \setminus V^C_{sub}$. Take any $\widehat{v}^C \in \widehat{V}^C$; we have two possibilities:
(i) $\widehat{v}^C = \widehat{v}^C_{sub}$: Since $G^C_{sub} = (V^C_{sub}, B)$ was contracted into $\widehat{v}^C_{sub}$, their origins coincide: $Origin(G^C_{sub}) = Origin(\widehat{v}^C_{sub})$. Since $\widehat{M}_A = M_A \cup M_B$, the pair $\left[Origin(\widehat{v}^C_{sub}), \widehat{M}_A\right] = \left[Origin(G^C_{sub}), M_A \cup M_B\right]$ is survivable by Theorem 1.
(ii) $\widehat{v}^C \neq \widehat{v}^C_{sub}$: In this case $\widehat{v}^C \in V^C \setminus V^C_{sub}$, so $\widehat{v}^C = v^C$. By piecewise survivability of the pair $\left[G^L, M_A\right]$, the pair $\left[Origin(v^C = \widehat{v}^C), M_A\right]$ is survivable. Since $\widehat{M}_A = M_A \cup M_B$, the pair $\left[Origin(\widehat{v}^C), \widehat{M}_A\right]$ is survivable as well.
Combining (i) and (ii), we have proven that for every $\widehat{v}^C \in \widehat{V}^C$, the pair $\left[Origin(\widehat{v}^C), \widehat{M}_A\right]$ is survivable. So, by Definition 6, the pair $\left[G^L, \widehat{M}_A\right]$ is piecewise survivable. ■

Proof of Corollary 1:
$\Rightarrow$ We have to show that if there is only one vertex in G^C then $\left[G^L, M_A\right]$ is survivable.
We have two observations: (i) By Theorem 4, the pair $\left[G^L, M_A\right]$ is piecewise survivable. This means that for every vertex $v^C \in G^C$ the pair $\left[Origin(v^C), M_A\right]$ is survivable. (ii) There is only one vertex in G^C (i.e., $G^C = \{v^C\}$), and

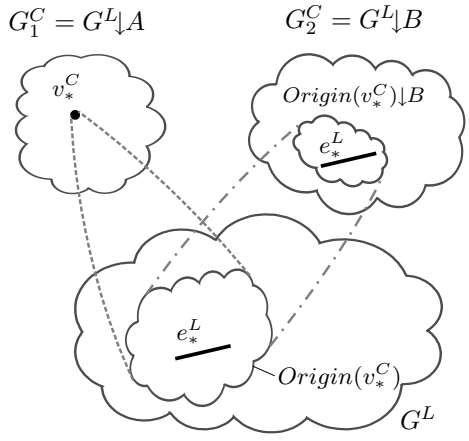


Fig. 9. Illustration of proof of Theorem 5. We start with an edge e_*^L that is in G_2^C, but not in G_1^C. Next, we choose a vertex $v_*^C \in G_1^C$ such that $e_*^L \in Origin(v_*^C)$. In the topology G_2^C, $Origin(v_*^C)$ is contracted to $Origin(v_*^C) \downarrow B$ that contains at least e_*^L. This nonempty subgraph $Origin(v_*^C) \downarrow B$ can be mapped in a survivable way using the mapping M_A, which leads to contradiction.

therefore $Origin(v^C) = G^L$. Combining (i) and (ii), we have that $\left[G^L, M_A\right]$ is survivable.

$\Leftarrow$ We have to show that if the contracted topology G^C has more than one node then a survivable mapping of G^L on G^ϕ does not exist.

By Theorem 4, the pair $\left[G^L, M_A\right]$ is piecewise survivable. Since the algorithm has terminated before converging to a single node (i.e., in Step 2), there exists no pair $\left[G_{sub}^C, M_B\right]$ that is survivable. In particular, if we take $G_{sub}^C = G^C = G^L{\downarrow}A$, there exists no pair $\left[G^L{\downarrow}A, M_*\right]$ that is survivable. Now, by Theorem 3 there exists no survivable mapping of G^L on G^ϕ. ■

Proof of Theorem 5 [By contradiction, Please refer to Fig. 9] Let us assume that two different runs of SMART converge to two different contracted topologies $G_1^C = G^L{\downarrow}A$ and $G_2^C = G^L{\downarrow}B$, and the mappings M_A and M_B, respectively. The SMART algorithm terminated in Step 2, which implies that no subgraph G_{sub1}^C of G_1^C can be mapped in a survivable way; similarly, no subgraph G_{sub2}^C of G_2^C can be mapped in a survivable way. Assume, without loss of generality, that there exists an edge e_*^L such that $e_*^L \in G_2^C$ and $e_*^L \notin G_1^C$. (If such an edge does not exist, an edge satisfying a converse condition must exist, because $G_1^C \neq G_2^C$.) Since $e_*^L \notin G_1^C$, there exists $v_*^C \in G_1^C$ such that $e_*^L \in Origin(v_*^C)$. By Theorem 4, the pair $\left[G_1^C, M_A\right]$ is piecewise survivable, which implies that $\left[Origin(v_*^C), M_A\right]$ is survivable. Now, by Theorem 2, the pair $\left[Origin(v_*^C){\downarrow}B, M_A\right]$ is also survivable. By construction the subgraph $Origin(v_*^C) \downarrow B$ contains at least the edge e_*^L. Therefore, there exists a non-empty subgraph $G_{sub}^C = Origin(v_*^C){\downarrow}B$ of G_2^C that can be mapped in a survivable way (using the mapping M_A), which is impossible because no subgraph G_{sub2}^C of G_2^C can be mapped in a survivable way.

■

REFERENCES

[1] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee, "Fault management in IP-Over-WDM Networks: WDM Protection vs. IP Restoration," *IEEE J. Select. Areas Commun.*, vol. 20, no. 1, January 2002.

[2] A. Fumagalli and L. Valcarenghi, "IP Restoration vs. WDM Protection: Is There an Optimal Choice?" *IEEE Network*, Nov/Dec 2000.

[3] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier-1 backbone," *Sprint ATL Research Report Nr. RR03-ATL-030666*, 2003.

[4] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of Failures in an IP Backbone," *Proc. IEEE INFOCOM'04*, 2004.

[5] M. Kurant and P. Thiran, "On Survivable Routing of Mesh Topologies in IP-over-WDM Networks," *Proc. Infocom*, 2005.

[6] ——, "Survivable Routing in IP-over-WDM Networks in the Presence of Multiple Failures," *http://arxiv.org/abs/cs.NI/0604053*, 2006.

[7] L.-W. Chen and E. Modiano, "Efficient Routing and Wavelength Assignment for Reconfigurable WDM Networks with Wavelength Converters," *Proc. IEEE INFOCOM 2003*, 2003.

[8] H. Lee, H. Choi, S. Subramaniam, and H.-A. Choi, "Survival Embedding of Logical Topology in WDM Ring Networks," *Information Sciences : An International Journal, Special Issue on Photonics, Networking and Computing*, 2002.

[9] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: a new approach to the design of WDM-based networks," *IEEE J. Select. Areas Commun.*, vol. 20, no. 4, pp. 800–809, May 2002.

[10] A. Sen, B. Hao, B. Shen, and G. Lin, "Survivable routing in WDM networks logical ring in arbitrary physical topology," *Proc. IEEE International Communication Conference ICC02*, 2002.

[11] F. Giroire, A. Nucci, N. Taft, and C. Diot, "Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection," *Proc. IEEE INFOCOM 2003*, 2003.

[12] J. Armitage, O. Crochat, and J. Y. L. Boudec, "Design of a Survivable WDM Photonic Network," *Proc. IEEE INFOCOM 97*, April 1997.

[13] A. Nucci, B. Sansò, T. Crainic, E. Leonardi, and M. A. Marsan, "Design of Fault-Tolerant Logical Topologies in Wavelength-Routed Optical IP Networks," *Proc. IEEE Globecom 2001*, 2001.

[14] F. Ducatelle and L. Gambardella, "Survivable routing in IP-over-WDM networks: An efficient and scalable local search algorithm," *Optical Switching and Networking*, vol. 2, pp. 86–99, September 2005.

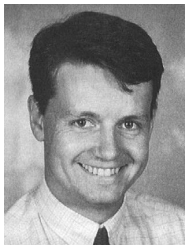
[15] M. Kurant and P. Thiran, "Survivable Mapping Algorithm by Ring Trimming (SMART) for large IP-over-WDM networks," *Proc. of BroadNets*, 2004.

[16] J. Gross and J. Yellen, *Graph Theory and its Applications*. CRC Press, 1999.

[17] A. Frank, *Packing paths, circuits and cuts - a survey (in Paths, Flows and VLSI-Layout)*. Springer, Berlin, 1990.

[18] O. Crochat and J. Y. L. Boudec, "Design Protection for WDM Optical Networks," *IEEE J. Select. Areas Commun.*, vol. 16, no. 7, pp. 1158–1165, September 1998.

Maciej Kurant Maciej Kurant received his M.S. degree from Gdansk University of Technology, Poland, in 2002. He is currently a Ph.D. student at EPFL, Switzerland. His main research interests are in various aspects of multilayer systems in the context of optical and overlay networks, and more generally, of complex networks.



Patrick Thiran Patrick Thiran (S89M97) is an associate professor at EPFL. He received the electrical engineering degree from the Université Catholique de Louvain, Louvain-la-Neuve, Belgium, in 1989, the M.S. degree in electrical engineering from the University of California at Berkeley, USA, in 1990, and the PhD degree from EPFL, in 1996. He became an adjunct professor in 1998, an assistant professor in 2002 and an associate professor in 2006. From 2000 to 2001, he was with Sprint Advanced Technology Labs, Burlingame, CA. His research interests include communication networks, performance analysis, dynamical systems and stochastic models. He is currently active in the analysis and design of wireless multi-hop networks and in network monitoring. He served as an associate editor for the IEEE Transactions on Circuits and Systems (Part II) in 1997-99, and he is currently an associate editor for the IEEE/ACM Transactions on Networking. He was the recipient of the 1996 EPFL Ph.D. award.