

Algorithms for Computing QoS Paths with Restoration

Yigal Bejerano, Yuri Breitbart, *Member, IEEE*, Ariel Orda, *Senior Member, IEEE*, Rajeev Rastogi, and Alexander Sprintson, *Member, IEEE*

Abstract— There is a growing interest among service providers to offer new services with *Quality of Service* (QoS) guarantees that are also resilient to failures. Supporting QoS connections requires the existence of a routing mechanism, that computes the *QoS paths*, i.e., paths that satisfy QoS constraints (e.g., delay or bandwidth). Resilience to failures, on the other hand, is achieved by providing, for each primary QoS path, a set of alternative QoS paths used upon a failure of either a link or a node. The above objectives, coupled with the need to minimize the global use of network resources, imply that the cost of both the primary path and the restoration topology should be a major consideration of the routing process.

We undertake a comprehensive study of problems related to finding suitable restoration topologies for QoS paths. We consider both bottleneck QoS constraints, such as bandwidth, and additive QoS constraints, such as delay and jitter. This is the first study to provide a rigorous solution, with proven guarantees, to the combined problem of computing *QoS paths with restoration*. It turns out that the widely used approach of disjoint primary and restoration paths is not an optimal strategy. Hence, the proposed algorithms construct a *restoration topology*, i.e., a set of *bridges*, each bridge protecting a portion of the primary QoS path. This approach guarantees to find a restoration topology with low cost when one exists.

Index Terms— QoS routing, Restoration, Restricted Shortest Path, Approximation Algorithms.

I. INTRODUCTION

There is a growing interest among service providers to offer their customers new revenue-generating services with *Quality of Service* (QoS) guarantees. This is facilitated by current efforts to provide resource reservations and explicit path routing, e.g., *MultiProtocol Label Switching* (MPLS). On the other hand, physical network infrastructures may be prone to failures; for example, in optical networks, a single link failure is frequent enough in order to warrant consideration [8]. Therefore, a key requirement for such services is that they also be resilient to failures. This goal, namely, providing QoS paths with failure resilience, can be achieved by provisioning *primary* and *restoration* paths that satisfy the QoS constraints. The primary QoS path is used during normal network operation; upon failure of a network element (node or link) in the primary path, the traffic is immediately switched to a restoration path. To facilitate this seamless recovery to

a restoration path in the event of a failure, it is necessary to reserve network resources (e.g., bandwidth) on both the primary and restoration QoS paths. Such resources should be consumed in a networkwide efficient manner. A common way for modeling the impact of such resource consumption on each link is by associating “costs” with the links. Accordingly, a major problem is to find primary and restoration paths that satisfy end-to-end QoS constraints at minimum cost. This problem is the subject of this study.

QoS constraints occur naturally in a number of practical settings involving bandwidth and delay sensitive applications such as voice over IP, audio and video conferencing, multimedia streaming, etc. QoS constraints can be divided into *bottleneck* constraints, such as bandwidth and *additive* constraints, such as delay or jitter.

QoS routing has been the subject of several recent studies and proposals (see, e.g., [4], [16], [18], [19] and references therein). However, none of the prior studies on QoS routing consider the problem of provisioning QoS paths with restoration. Similarly, path restoration and routing over alternate paths has also attracted a large body of research (see, e.g., [9]–[11], [13], [14]). Most of the proposed solutions, however, consider only bottleneck QoS constraints. The few studies that do consider additive constraints, focus on heuristic approaches and do not provide proven performance guarantees.

Bottleneck QoS constraints can be efficiently handled by pruning infeasible links. However, additive QoS constraints are more difficult to handle. Indeed, the basic problem of finding an optimal path that satisfies an additive QoS constraint is \mathcal{NP} -hard [6]. Moreover, it turns out that, in the presence of additive QoS constraints, the widely used approach of disjoint primary and restoration paths is not an optimal strategy. A better solution is to provide a *restoration topology*, i.e., a set of *bridges*, with each bridge protecting a portion of the primary path. The advantage of the disjoint paths strategy is its ability to switch promptly from the primary path to the backup path in the event of a failure. While a restoration topology requires more sophisticated switching with a proper signaling mechanism, it has several advantages over the disjoint paths strategy. First, it provides a cheaper solution in terms of resource consumption. Second, it may find a solution when one does not exist for the disjoint paths strategy [13]. Third, a restoration topology strategy uses fewer backup links upon a failure, which facilitates more efficient sharing of backup bandwidth [11]. Finally, the restoration topology strategy enables the network to recover from a failure by simply activating a local bridge, rather than switching to a completely new path.

Accordingly, this study investigates the problem of provisioning primary and restoration paths that satisfy QoS constraints. Since this problem is \mathcal{NP} -hard, we present solutions

Yigal Bejerano and Rajeev Rastogi are with Bell Labs, Lucent Technologies, Murray Hill, New Jersey, USA 07974 (email {bej,rastogi}@research.bell-labs.com).

Yuri Breitbart is with the Department of Computer Science, Kent State University, Kent, Ohio, USA 44240 (email yuri@cs.kent.edu).

Ariel Orda is with the Department of Electrical Engineering, Technion — Israel Institute of Technology, Haifa 32000, Israel (email: ariel@ee.technion.ac.il).

Alexander Sprintson (corresponding author) is with the Department of Engineering and Applied Science, California Institute of Technology, Pasadena, California, USA 91125 (email: spalex@caltech.edu).

that are guaranteed to be within a certain factor of the optimum. Focusing on the fundamental problem of resilience to a single failure, the paper makes two major contributions. First, we address the issue of link sharing by different bridges, which complicates the process of finding the set of optimal bridges, and we prove that there is an optimal restoration topology in which each link is shared by at most two bridges. This enables us to identify restoration topologies whose cost is at most two times more than the optimum. The second contribution is the novel concept of *adjusted delay*, which allows us to represent the set of bridges that compose a restoration topology as a single *walk*¹ between the source and the destination nodes. This concept makes it possible to adapt standard schemes such as Bellman-Ford's Shortest Path algorithm [3] for identification of restoration topologies.

To further illustrate the effectiveness of our proposed solutions, we conduct some simulation experiments. The simulation results demonstrate that our proposed algorithms perform well in practice, and, in a number of cases, return a restoration topology even when prior approaches fail to do so.

The remainder of the paper is organized as follows. In Section II, we present the network model and formulate the problems considered in this paper. In Section III, we discuss a fundamental property of optimal restoration topologies, namely, the sharing of links by several bridges. In Section IV, we introduce the basic concepts of adjusted delay and feasible walk. In Section V, we provide approximation algorithm for provisioning of the restoration topologies. In Section VI, we extend our results for directed networks. Simulation results are presented and discussed in Section VII. Finally, conclusions appear in Section VIII. Due to space limits, some proofs are omitted and can be found in [2].

II. MODEL AND PROBLEM FORMULATION

In this section, we describe the network model and the two problems addressed in this paper. For simplicity of exposition, we use the terms *bandwidth* and *delay* requirements in order to generically refer to *bottleneck QoS constraints* and *additive QoS constraints*, respectively. Table I summarizes the notation used throughout the paper.

A. The Network Model

We represent the *network* by an undirected graph $G(V, E)$, where V is the set of nodes and E is the set of links. We assume that the network does not contain parallel links (i.e., two or more links that connect the same nodes). We denote by N and M the number of network nodes and links, respectively, i.e., $N = |V|$ and $M = |E|$. An (s, t) -*walk* is a finite sequence of nodes $\mathcal{W} = (s = v_0, v_1, \dots, t = v_n)$, such that, for $0 \leq i \leq n - 1$, $(v_i, v_{i+1}) \in E$. Here, $n = |\mathcal{W}|$ is the *hop count* of \mathcal{W} . Note that nodes and links may appear in a walk several times. An (s, t) -*path* \mathcal{P} is an (s, t) -walk whose nodes are distinct. The subwalk (subpath) of \mathcal{W} (\mathcal{P}) that extends from v_i to v_j is denoted by $\mathcal{W}_{(v_i, v_j)}$ ($\mathcal{P}_{(v_i, v_j)}$). Let \mathcal{W}_1 be a (v, u) -walk and \mathcal{W}_2 be a (u, w) -walk; then, $\mathcal{W}_1 \circ \mathcal{W}_2$ denotes the (v, w) -walk formed by the concatenation of \mathcal{W}_1 and \mathcal{W}_2 .

Each link $l \in E$ offers a bandwidth guarantee (which is typically the available bandwidth on l), and a delay guarantee

$G(V, E)$	A graph that represents a network
V	The set of nodes of G
E	The set of links of G
$N = V $	The number of nodes of G
$M = E $	The number of links of G
\mathcal{P}	A path
\mathcal{W}	A walk
$\mathcal{P}_{(v_i, v_j)}$	The subpath of \mathcal{P} that extends from v_i to v_j
$\mathcal{W}_{(v_i, v_j)}$	The subwalk of \mathcal{W} that extends from v_i to v_j
$\mathcal{W}_1 \circ \mathcal{W}_2$	The walk formed by concatenating \mathcal{W}_1 and \mathcal{W}_2
s	The source node
t	The destination node
$\hat{\mathcal{P}}$	The primary path
d_l	The delay of link $l \in E$
c_l	The cost of link $l \in E$
$C(\mathcal{P})$ ($C(\mathcal{W})$)	The cost of path \mathcal{P} (walk \mathcal{W})
$D(\mathcal{P})$ ($D(\mathcal{W})$)	The delay of path \mathcal{P} (walk \mathcal{W})
\hat{d}	The delay constraint
$\hat{D}(\mathcal{W})$	The adjusted delay of the walk \mathcal{W}
\mathcal{B}	A bridge
Δ	The difference between the delay constraint and the delay of the primary path, i.e., $\Delta = \hat{d} - D(\hat{\mathcal{P}})$
\mathcal{R}	A restoration topology
OPT	The minimum cost of a restoration topology
ε	A constant that captures the trade-off between the quality of an approximation and its computational complexity

TABLE I
NOTATION USED IN THE PAPER.

d_l . The bandwidth of a walk is identical to the bandwidth of its worst link. The delay $D(\mathcal{W})$ of a walk \mathcal{W} is the sum of the QoS requirements of its links, i.e., $D(\mathcal{W}) = \sum_{l \in \mathcal{W}} d_l$.

In order to satisfy QoS constraints, certain resources such as bandwidth and buffer space must be reserved along QoS paths. In order to optimize the global resource utilization, we need to identify QoS paths that consume as few network resources as possible. Accordingly, we associate with each link l a nonnegative cost c_l , which estimates the quality of the link in terms of resource utilization. The link cost may depend on various factors, e.g., the link's available bandwidth and its location. The cost $C(\mathcal{W})$ of a walk \mathcal{W} is defined to be the sum of the costs of its links, i.e., $C(\mathcal{W}) = \sum_{l \in \mathcal{W}} c_l$.

Network links are prone to failures. Following [10] and [12], we assume that only a single failure can occur at a time. Indeed, protection from multiple failures would have incurred excessively high cost in terms of network utilization, which, typically, is not justified by the rare occurrence of simultaneous failures.

In order to model networks with nodes connected by asymmetric or unidirectional links, we also consider directed graphs (Section VI). For instance, in such networks, for a pair of connected nodes (v_i, v_j) , the bandwidth allocated on the link in the direction from v_i to v_j may be much larger than the allocated bandwidth in the opposite direction. In addition, the delay and cost characteristics of a link (v_i, v_j) may be very different from those of the reverse link (v_j, v_i) .

B. QoS paths

A fundamental problem in QoS routing is to identify a minimum cost path between a source s and a destination t that satisfies some delay and bandwidth constraints. Bandwidth requirements can be efficiently handled by simply pruning

¹In contrast to a path, a walk may include loops.

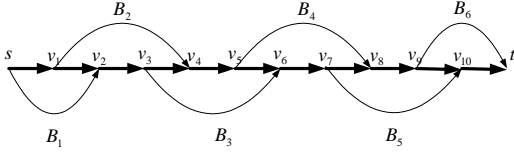


Fig. 1. An example of a network. The delay of all links is 10, except for bold links whose delay is 1.

infeasible links from the graph, i.e., links l whose bandwidth is lower than the constraint. Thus, in the rest of the paper, we only consider delay requirements. Accordingly, the fundamental problem is to find a minimum cost path that satisfies a given delay constraint. This can be formulated as the *Restricted Shortest Path* problem.

Problem RSP (Restricted Shortest Path): Given a source node s , a destination node t and a delay constraint \hat{d} , find an (s, t) -path $\hat{\mathcal{P}}$ such that

- 1) $D(\hat{\mathcal{P}}) \leq \hat{d}$, and
- 2) $C(\hat{\mathcal{P}}) \leq C(\mathcal{P})$ for every other (s, t) -path \mathcal{P} that satisfies $D(\mathcal{P}) \leq \hat{d}$.

In general, Problem RSP is intractable, i.e., \mathcal{NP} -hard [6]. However, there exist pseudo-polynomial solutions, which give rise to fully polynomial approximation schemes² (FPAS), whose computational complexity is reasonable (see [5], [7] and [15]). The most efficient algorithm, presented in [15], has a computational complexity of $\mathcal{O}(MN(\frac{1}{\varepsilon} + \log \log N))$, and computes a path with delay at most \hat{d} and cost at most $(1 + \varepsilon)$ times the optimum. We refer to this algorithm as Algorithm RSP.

C. Bridges and a Restoration Topology

As mentioned earlier, our study focuses on provisioning QoS paths with restoration. The QoS path that is used during normal network operation is referred to as the *primary* path. Upon failure of a network element (node or link) in the primary path, the traffic is immediately switched to a *restoration* path. Thus, we require that in addition to the primary path, the restoration paths also satisfy the delay constraint \hat{d} . In this paper, we primarily focus on link failures, but our results can be easily extended to deal with node failures by using standard node splitting technique (see, e.g., [20]).

A common approach for path restoration is to provision two disjoint paths that satisfy the delay constraint. However, as we illustrate below (see also [13]), in some cases, such disjoint paths do not exist, although it is possible to provision a primary path with a set of restoration paths. Consider the network depicted in Fig. 1. Here, the delay of all links is 10, except for the links marked by bold lines, whose delay is 1. The only two disjoint paths between the source node s and the destination node t are $\mathcal{P}_1 = \{s, v_1, v_4, v_5, v_8, v_9, t\}$ and $\mathcal{P}_2 = \{s, v_2, v_3, v_6, v_7, v_{10}, t\}$. For a delay constraint $\hat{d} = 20$, \mathcal{P}_1 and \mathcal{P}_2 cannot be used as primary and restoration paths, because $D(\mathcal{P}_1) = D(\mathcal{P}_2) = 33 > 20$. However, it is possible to provision a primary path $\hat{\mathcal{P}}$ and a set of restoration paths

that satisfy the delay constraint $\hat{d} = 20$. Specifically, we use the primary path $\hat{\mathcal{P}} = \{s, v_1, v_2, \dots, v_{10}, t\}$ and restoration paths $\{\hat{\mathcal{P}}_1, \dots, \hat{\mathcal{P}}_6\}$ defined as follows. Upon failure of links (s, v_1) or (v_1, v_2) we use restoration path $\hat{\mathcal{P}}_1 = \mathcal{B}_1 \circ \hat{\mathcal{P}}_{(v_2, t)}$ with $D(\hat{\mathcal{P}}_1) = 19$, while upon failure of link (v_2, v_3) path $\hat{\mathcal{P}}_2 = \hat{\mathcal{P}}_{(s, v_1)} \circ \mathcal{B}_2 \circ \hat{\mathcal{P}}_{(v_4, t)}$ with $D(\hat{\mathcal{P}}_2) = 18$ is used. Similarly, we construct restoration paths $\hat{\mathcal{P}}_3, \dots, \hat{\mathcal{P}}_6$. As demonstrated in this example, a restoration path comprises portions of the primary path and a *bridge*, which serves as a backup for the failed segment of the primary path. For example, in Fig. 1, the restoration paths $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$ include the bridges $\mathcal{B}_1 = \{s, v_2\}$ and $\mathcal{B}_2 = \{v_1, v_4\}$, respectively.

Definition 1 (Bridge for a link failure): Let $\hat{\mathcal{P}} = \{s = v_0, \dots, t = v_n\}$ be a QoS path and $\hat{\mathcal{P}}_{(v_i, v_j)}$ be a subpath of $\hat{\mathcal{P}}$. A path \mathcal{B} between $v_i \in \hat{\mathcal{P}}$ and $v_j \in \hat{\mathcal{P}}$ that has no common links with $\hat{\mathcal{P}}$ is referred to as a *bridge*. We say that bridge $\mathcal{B} = \{v_i, \dots, v_j\}$ *protects* the subpath $\hat{\mathcal{P}}_{(v_i, v_j)}$ of $\hat{\mathcal{P}}$.

Recall that each restoration path must satisfy the delay constraint \hat{d} . This implies that the delay $D(\mathcal{B})$ of a bridge \mathcal{B} must also be constrained. Specifically, the delay of a bridge $\mathcal{B}_i = \{s_i, \dots, t_i\}$ must be at most $D(\hat{\mathcal{P}}_{(s_i, t_i)}) + \hat{d} - D(\hat{\mathcal{P}})$, where $\hat{\mathcal{P}}_{(s_i, t_i)}$ is the subpath of $\hat{\mathcal{P}}$ protected by \mathcal{B}_i . We denote the quantity $\hat{d} - D(\hat{\mathcal{P}})$ by Δ . Clearly, for larger values of Δ , it is possible to find cheaper bridges $\mathcal{B}_i = \{s_i, \dots, t_i\}$ that satisfy $D(\mathcal{B}_i) \leq D(\hat{\mathcal{P}}_{(s_i, t_i)}) + \Delta$.

A set of bridges that provides a restoration path for the failure of any link $l \in \hat{\mathcal{P}}$ is referred to as a *restoration topology*.

Definition 2 (Restoration topology for link failures): Let \hat{d} be a QoS constraint and $\hat{\mathcal{P}} = \{s = v_0, \dots, v_n = t\}$ be a QoS path that satisfies \hat{d} (i.e., $D(\hat{\mathcal{P}}) \leq \hat{d}$). Then, a *restoration topology* \mathcal{R} for $(\hat{\mathcal{P}}, \hat{d})$ is a set of bridges $\{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_h\}$ such that:

- 1) for each bridge $\mathcal{B}_i = \{s_i, \dots, t_i\}$, it holds that $D(\mathcal{B}_i) \leq D(\hat{\mathcal{P}}_{(s_i, t_i)}) + \hat{d} - D(\hat{\mathcal{P}})$, and
- 2) for each link $l \in \hat{\mathcal{P}}$, there exists a bridge $\mathcal{B}_i = \{s_i, \dots, t_i\}$ that “protects” l , i.e., l is included in the subpath $\hat{\mathcal{P}}_{(s_i, t_i)}$ of $\hat{\mathcal{P}}$ protected by \mathcal{B}_i .

We refer to \mathcal{R} as a *feasible restoration topology*, in order to emphasize that each restoration path satisfies the delay constraint \hat{d} .

Let $E(\mathcal{R})$ be the set of links that belong to bridges of \mathcal{R} , i.e., $E(\mathcal{R}) = \{l \mid l \in \mathcal{B}_i, \mathcal{B}_i \in \mathcal{R}\}$. The cost of a restoration topology is defined as the total cost of links in $E(\mathcal{R})$, i.e., $C(\mathcal{R}) = \sum_{l \in E(\mathcal{R})} c_l$. Note that the cost of each link is counted only once, even if it belongs to several bridges. We denote by $|E(\mathcal{R})|$ the number of links in the restoration topology.

We seek restoration topologies that minimize the usage of network resources. Since the cost of a link is a measure of its desirability for routing (with lower cost links being more desirable), our goal is to find a (feasible) restoration topology \mathcal{R} with minimum cost $C(\mathcal{R})$. Note that, depending on how costs are assigned to links, our approach enables a wide range of restoration topologies to be selected. For example, associating unit costs with all links would translate into computing restoration topologies with a minimum number of links. Also observe that in an optimal restoration topology, the subpaths of $\hat{\mathcal{P}}$ protected by two bridges \mathcal{B}_i and \mathcal{B}_j are not nested, one within the other. Thus, for any two bridges

²A *Fully Polynomial Approximation Scheme* (FPAS) provides a solution whose cost is at most $(1 + \varepsilon)$ times more than the optimum with a time complexity that is polynomial in the size of the input and $1/\varepsilon$.

Problem	Undirected	Directed
RT	$(2 \cdot (1 + \varepsilon), 1)$	$(2 \cdot (1 + \varepsilon), 2)$
P+RT	$(3 \cdot (1 + \varepsilon), 2)$	$(3 \cdot (1 + \varepsilon), 3)$

TABLE II
APPROXIMATION RATIOS OBTAINED BY OUR ALGORITHMS

$\mathcal{B}_i = \{s_i, \dots, t_i\}$ and $\mathcal{B}_j = \{s_j, \dots, t_j\}$ in \mathcal{R} , if s_i precedes s_j in $\hat{\mathcal{P}}$ then t_i also precedes t_j in $\hat{\mathcal{P}}$, and vice versa. For clarity of presentation, we assume that the bridges in \mathcal{R} are enumerated such that the source s_i of bridge \mathcal{B}_i is either identical to or a predecessor of the destination t_{i-1} of bridge \mathcal{B}_{i-1} in $\hat{\mathcal{P}}$.

D. Problem Statement

We are now ready to formulate the two problems that we consider in this study. The first problem seeks to compute a suitable restoration topology for a given QoS path.

Problem RT (Restoration topology for a QoS Path): Given an (s, t) -path $\hat{\mathcal{P}}$ and a QoS constraint \hat{d} , such that $D(\hat{\mathcal{P}}) \leq \hat{d}$, find a minimum cost restoration topology \mathcal{R} for $(\hat{\mathcal{P}}, \hat{d})$.

We denote by *OPT* the minimum cost of a restoration topology for $(\hat{\mathcal{P}}, \hat{d})$. Next, we consider the problem of provisioning a QoS path with a restoration topology.

Problem P+RT (QoS Path and Restoration topology):

Given a source s , a destination t and a QoS constraint \hat{d} , find an (s, t) -path $\hat{\mathcal{P}}$ that satisfies $D(\hat{\mathcal{P}}) \leq \hat{d}$ and a restoration topology \mathcal{R} for $(\hat{\mathcal{P}}, \hat{d})$ such that their total cost $C(\hat{\mathcal{P}}) + C(\mathcal{R})$ is minimum.

Each of the above problems, namely RT and P+RT, includes Problem RSP as a special case; hence, they are both \mathcal{NP} -hard. Furthermore, as discussed below, in most cases we cannot provide an efficient solution without violating the delay constraint in the restoration paths. Accordingly, we introduce the following definition of (α, β) -approximations.

Definition 3 ((α, β)-approximation): For constants α and β , an (α, β) -approximate solution to, either Problem RT or Problem P+RT is a solution, for which:

- 1) the cost is at most α times more than the optimum;
- 2) the primary path satisfies the delay constraint;
- 3) each restoration path violates the delay constraint by a factor of at most β .

E. Our results

In table II, we summarize the approximation ratios that we obtain for the above two problems. In this table, ε is a parameter that captures the trade-off between the quality of the approximations and the running time of the algorithms. Specifically, the time complexity of the algorithms is proportional to $1/\varepsilon$; thus smaller values of ε yield better approximate solutions at the expense of higher running times.

The solution of Problem P+RT is the main contribution of this paper. We emphasize that our solutions may violate the delay constraint only for restoration paths, while *the primary paths always satisfy the QoS constraints*. Therefore, such delay violations have no effect during normal network operation. Moreover, many time-sensitive applications can tolerate short-term delay violations (until the failed link is repaired), e.g., by way of buffering.

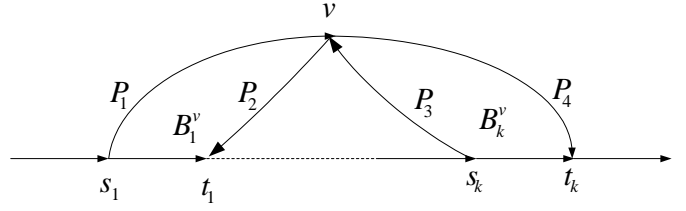


Fig. 2. Node v is shared by bridges $\mathcal{B}_1^v = \mathcal{P}_1 \circ \mathcal{P}_2$ and $\mathcal{B}_k^v = \mathcal{P}_3 \circ \mathcal{P}_4$.

III. PROPERTIES OF RESTORATION TOPOLOGIES

Finding an optimal restoration topology is a complicated problem due to the fact that bridges may share links. However, in this section we show that this obstacle can be overcome with a minimum penalty in terms of the cost of the solution obtained. We begin by establishing the existence of an optimal restoration topology in which the number of appearances of a node or a link is bounded by 2.

Lemma 1: Given an undirected graph G , a delay constraint \hat{d} , an (s, t) -path $\hat{\mathcal{P}}$, $D(\hat{\mathcal{P}}) \leq \hat{d}$ and a restoration topology \mathcal{R} for $(\hat{\mathcal{P}}, \hat{d})$, there exists a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, \hat{d})$ such that $C(\hat{\mathcal{R}}) \leq C(\mathcal{R})$, and each node $v \in \hat{\mathcal{R}}$ or link $l \in \hat{\mathcal{R}}$ is included in at most two bridges.

Proof: Let $\hat{\mathcal{R}}$ be a restoration topology for $(\hat{\mathcal{P}}, \hat{d})$ such that $E(\hat{\mathcal{R}}) \subseteq E(\mathcal{R})$ and $|E(\hat{\mathcal{R}})|$ is minimum. We prove that each node of $\hat{\mathcal{R}}$ is included in at most two bridges.

By way of contradiction, let $\{\mathcal{B}_1^v, \mathcal{B}_2^v, \dots, \mathcal{B}_k^v\}$, $k \geq 3$ be the set of bridges of $\hat{\mathcal{R}}$ that contain v , sorted according to their indexes. Since $|E(\hat{\mathcal{R}})|$ is minimum, it follows that the subpaths $\hat{\mathcal{P}}_{(s_1, t_1)}$ and $\hat{\mathcal{P}}_{(s_k, t_k)}$ of $\hat{\mathcal{P}}$ protected by bridges \mathcal{B}_1^v and \mathcal{B}_k^v , respectively, are disjoint (i.e., t_1 is a predecessor of s_k in $\hat{\mathcal{P}}$), otherwise we can omit from $\hat{\mathcal{R}}$ the bridges $\mathcal{B}_2^v, \dots, \mathcal{B}_{k-1}^v$. We denote the subpaths $\{s_1, \dots, v\}$ and $\{v, \dots, t_1\}$ of \mathcal{B}_1^v by \mathcal{P}_1 and \mathcal{P}_2 , respectively (see Fig. 2). The subpaths $\{s_k, \dots, v\}$ and $\{v, \dots, t_k\}$ of \mathcal{B}_k^v are denoted by \mathcal{P}_3 and \mathcal{P}_4 , respectively.

The delay of each bridge \mathcal{B} exceeds the delay of the subpath of $\hat{\mathcal{P}}$ protected by \mathcal{B} by at most Δ , i.e.:

$$\begin{aligned} D(\mathcal{P}_1) + D(\mathcal{P}_2) &\leq D(\hat{\mathcal{P}}_{(s_1, t_1)}) + \Delta \text{ and} \\ D(\mathcal{P}_3) + D(\mathcal{P}_4) &\leq D(\hat{\mathcal{P}}_{(s_k, t_k)}) + \Delta. \end{aligned}$$

It is easy to verify that one of the following two conditions must hold:

$$\begin{aligned} D(\mathcal{P}_1) + D(\mathcal{P}_3) &\leq D(\hat{\mathcal{P}}_{(s_1, t_1)}) + \Delta \text{ or} \\ D(\mathcal{P}_2) + D(\mathcal{P}_4) &\leq D(\hat{\mathcal{P}}_{(s_k, t_k)}) + \Delta. \end{aligned}$$

If the first condition holds, then we can substitute bridge \mathcal{B}_1^v in $\hat{\mathcal{R}}$ by a new bridge $\hat{\mathcal{B}}_1^v$ formed by concatenating \mathcal{P}_1 and the reverse path of \mathcal{P}_3 . This substitution yields a valid restoration topology with fewer links than in $\hat{\mathcal{R}}$ which contradicts our assumption that $|E(\hat{\mathcal{R}})|$ is minimum. Note that this substitution yields a restoration topology with fewer links even if \mathcal{P}_2 has common links with \mathcal{P}_3 or \mathcal{P}_4 . Similarly, we can show a contradiction if the second condition holds.

We thus conclude that each node $v \in \hat{\mathcal{R}}$ is shared by at most two bridges. It follows that each link $l \in \hat{\mathcal{R}}$ is also shared by at most two bridges. ■

Clearly, the fact that a link can be shared by several bridges reduces the cost of the restoration topology, since the cost

of the shared link is counted only once. However, finding restoration topologies with shared links is a difficult task that incurs a high computational complexity. An alternative approach is to ignore such link sharing, i.e., count the cost of a shared link as many times as it appears in the bridges.

Corollary 1: Let \mathcal{R} be a restoration topology for $(\hat{\mathcal{P}}, \hat{d})$, where $\hat{\mathcal{P}}$ is a primary path and \hat{d} is a delay constraint. We denote by $\hat{C}(\mathcal{R}) = \sum_{\mathcal{B} \in \mathcal{R}} \sum_{l \in \mathcal{B}} c_l$ the cost of \mathcal{R} when sharing is not considered, i.e., the cost of each shared link is counted several times. Then, there exists a feasible restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, \hat{d})$ such that $C(\hat{\mathcal{R}}) = OPT$ and $\hat{C}(\hat{\mathcal{R}}) \leq 2 \cdot OPT$, where OPT is minimum cost of a feasible restoration topology for $(\hat{\mathcal{P}}, \hat{d})$.

Proof: Immediate from Lemma 1. \blacksquare

Corollary 1 implies that, by ignoring link sharing (i.e., finding a minimum cost topology $\hat{\mathcal{R}}$ with respect to cost $\hat{C}(\mathcal{R})$), we can identify a restoration topology whose cost is at most twice more than that of an optimal solution (with sharing). We adopted this approach in order to construct efficient approximation algorithms.

In Section VI, we derive a similar bound of 2 on the degree of sharing for each link in a directed network graph. But first, in the following two sections, we focus on developing approximation algorithms for Problems RT and P+RT for undirected graphs.

IV. ADJUSTED DELAY AND FEASIBLE WALK CONCEPTS

In this section, we introduce the basic concepts of *adjusted delay* and *feasible walk*, which lay the foundations for our efficient approximation algorithms for Problems RT and P+RT, presented in Section V.

A. A Simple Algorithm

In order to set the stage for the concept of adjusted delay, we first present a simple algorithm for Problem RT. The algorithm, at a high level, consists of the following steps. First, we compute for each node pair (v_i, v_j) in $\hat{\mathcal{P}}$, the cheapest bridge $\mathcal{B}_{(i,j)} = \{v_i, \dots, v_j\}$ whose delay is at most $D(\hat{\mathcal{P}}_{(v_i, v_j)}) + \Delta$. To that end, we delete all the links of the path $\hat{\mathcal{P}}$ from G and apply Algorithm RSP [15] to the resulting graph. Next, we construct a restoration topology by selecting a subset of bridges $\{\mathcal{B}_{(i,j)}\}$ such that each link of $\hat{\mathcal{P}}$ is protected and the total cost is minimum. To achieve this, we construct an auxiliary directed graph \tilde{G} whose nodes are essentially the nodes of $\hat{\mathcal{P}}$. Further, for each link $(v_i, v_{i+1}) \in \hat{\mathcal{P}}$ we add to \tilde{G} a link (v_{i+1}, v_i) and assign it a zero cost. Also, for each pair (v_i, v_j) of nodes of $\hat{\mathcal{P}}$, such that $j > i$, we add to \tilde{G} a link $l = (v_i, v_j)$ whose cost is identical to the cost of the bridge $\mathcal{B}_{(i,j)}$. We now show that each (s, t) -path in \tilde{G} corresponds to a feasible restoration topology. Consider an (s, t) -path \mathcal{P} in \tilde{G} , and let \mathcal{S} be the set of bridges that correspond to links in \mathcal{P} . Consider two successive bridges $\mathcal{B}_i = \{s_i, t_i\}$ and $\mathcal{B}_{i+1} = \{s_{i+1}, t_{i+1}\}$ in \mathcal{S} . Note that s_{i+1} either precedes t_i or else coincides with t_i , while t_{i+1} succeeds t_i in the path $\hat{\mathcal{P}}$. This ensures that every link in $\hat{\mathcal{P}}$ is protected by a bridge, and thus \mathcal{S} corresponds to a feasible restoration topology. Hence, a near-optimal restoration topology can be determined by finding the shortest (s, t) -path in \tilde{G} .

Specifically, Lemma 1 implies that there exists an (s, t) -path in \tilde{G} whose cost is at most $2(1 + \varepsilon)OPT$. Indeed, let \mathcal{R} be an optimal restoration topology and let $\hat{\mathcal{R}}$ be a restoration topology whose existence is guaranteed by Lemma 1. Then, we can construct a path $\hat{\mathcal{P}} \in \hat{\mathcal{R}}$ such that each bridge in $\mathcal{B} \in \hat{\mathcal{R}}$ corresponds to a link $l \in \hat{\mathcal{P}}$, and $c_l \leq (1 + \varepsilon)C(\mathcal{B})$. The last inequality follows from the fact that, for each pair of nodes (i, j) , Algorithm RSP returns a path whose cost is at most $(1 + \varepsilon)$ times more than the optimum.

The above algorithm, while conceptually simple, is computationally expensive, because it applies Algorithm RSP [15] for each pair of nodes in $\hat{\mathcal{P}}$. Since the time complexity of the RSP algorithm is $\mathcal{O}(MN(1/\varepsilon + \log \log N))$, it requires a total of $\mathcal{O}(MN^3(1/\varepsilon + \log \log N))$ time. In the following sections, we describe Algorithm RT, which employs similar ideas, but whose computational complexity is comparable to that of Algorithm RSP.

B. The Adjusted Delay Concept

The algorithm presented in the previous section exploited the relationship between the shortest path in an auxiliary graph and the restoration topology. In this section, we use this idea again, but for devising a more efficient algorithm. We construct a directed auxiliary graph G' from G by reversing each link $l \in \hat{\mathcal{P}}$ and assigning it a zero cost. In addition, we also substitute each link $l = (u, v) \in G, l \notin \hat{\mathcal{P}}$ by two directed links $l_1 = (u, v)$ and $l_2 = (v, u)$ such that $c_{l_1} = c_{l_2} = c_l$ and $d_{l_1} = d_{l_2} = d_l$. Clearly, each (s, t) -walk in the auxiliary graph G' corresponds to a set of bridges that protects each link $l \in \hat{\mathcal{P}}$. For example, Fig. 3 depicts the auxiliary graph for the network depicted in Fig. 1 and the primary path $\hat{\mathcal{P}} = \{s, v_1, v_2, \dots, t\}$. The walk $\mathcal{W} = \{s, v_2, v_1, v_4, v_3, \dots, t\}$ in auxiliary graph corresponds to a set of bridges $\{\mathcal{B}_1, \dots, \mathcal{B}_6\}$. In general, however, as explained below, not every (s, t) -walk in the auxiliary graph corresponds to a *feasible* restoration topology, i.e., one that satisfies the delay constraint.

One of the key contributions of this study is an efficient method for verifying, during its construction, whether a given walk represents a feasible restoration topology. This method is used as a basic building block in our algorithm, and enables us to find a low-cost feasible restoration topology. In order to identify a walk in G' that corresponds to a feasible restoration topology, we introduce the notion of *adjusted delay* for a walk \mathcal{W} in G' .

Consider a walk $\mathcal{W} = \{s = u_0, \dots, u_{k-1}, u_k, \dots, t\}$ in G' that defines a set $\mathcal{S} = \{\mathcal{B}_i = \{s_i, \dots, t_i\}\}$ of bridges, and let \mathcal{P}_i be the restoration path obtained by activating the bridge \mathcal{B}_i ; thus, $\mathcal{P}_i = \hat{\mathcal{P}}_{(s, s_i)} \circ \mathcal{B}_i \circ \hat{\mathcal{P}}_{(t_i, t)}$. We refer to nodes s_i and t_i as the start-point and termination-point of bridge \mathcal{B}_i , respectively. Recall that a bridge \mathcal{B}_i satisfies the delay constraint only if $D(\mathcal{B}_i) \leq D(\hat{\mathcal{P}}_{(s_i, t_i)}) + \Delta$, or, equivalently,

$$D(\mathcal{P}_{i(s, t_i)}) \leq D(\hat{\mathcal{P}}_{(s, t_i)}) + \Delta, \quad (1)$$

where $\Delta = \hat{d} - D(\mathcal{P})$.

Furthermore, every subwalk $\mathcal{W}_{(s, u)}$ of \mathcal{W} corresponds to a subset \mathcal{S}' of complete bridges in \mathcal{S} and, possibly, a subpath of an additional bridge $\mathcal{B}_j \in \mathcal{S}$. The adjusted delay $\tilde{D}(\mathcal{W}_{(s, u)})$ of the walk $\mathcal{W}_{(s, u)}$ maintains the following invariant: if all the bridges in \mathcal{S}' satisfy the delay constraint, then $\tilde{D}(\mathcal{W}_{(s, u)})$

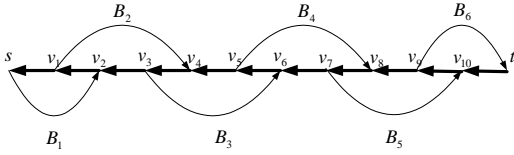


Fig. 3. An auxiliary graph for the network depicted on Fig. 1.

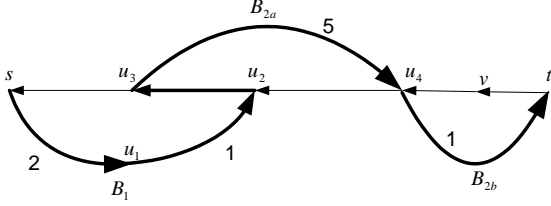


Fig. 4. An example of a walk in the auxiliary graph G' .

represents the delay between the source node s and the node $u \in \mathcal{B}_j$ along \mathcal{P}_j , i.e., $\tilde{D}(\mathcal{W}_{(s,u)}) = D(\mathcal{P}_j(s,u))$. Otherwise, if there is a bridge in \mathcal{S}' that does not satisfy the delay constraint, $\tilde{D}(\mathcal{W}_{(s,u)})$ is set to infinity, thus indicating that the restoration topology formed by the walk $\mathcal{W}_{(s,u)}$ is infeasible. Thus, by applying Condition (1), the adjusted delay enables us to check easily whether bridge \mathcal{B}_j satisfies the delay constraint, when its termination-point t_j is reached.

The adjusted delay $\tilde{D}(\mathcal{W})$ of a walk $\mathcal{W} = \{s = u_0, \dots, u_{k-1}, u_k, \dots, t\}$ is calculated in a recursive manner. The adjusted delay of an empty walk is zero, that is $\tilde{D}(\{s\}) = 0$. Now, let us turn to compute the adjusted delay of $\mathcal{W}_{(s,u_k)}$ and suppose that we have already calculated the adjusted delay of the sub-walk $\mathcal{W}_{(s,u_{k-1})}$. Let $D = \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) + d_{(u_{k-1},u_k)}$, where $d_{(u_{k-1},u_k)}$ is the delay of the link (u_{k-1}, u_k) . Generally speaking, the adjusted delay $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D$, except for the case when $u_k \in \hat{\mathcal{P}}$. In this case, a special procedure is required for verifying if node u_k is the termination-point of a bridge and whether the newly formed bridge satisfies the delay constraint. Node $u_k \in \hat{\mathcal{P}}$ is not necessarily a termination-point of a bridge, since a bridge may have several common nodes with the primary path. For instance, in Fig. 4, bridge \mathcal{B}_2 comprises of the two segments \mathcal{B}_{2a} and \mathcal{B}_{2b} , and node $u_4 \in \hat{\mathcal{P}}$ is not a termination-point of a bridge. However, if for link (u_k, u_{k+1}) of \mathcal{W} , it holds that $(u_{k+1}, u_k) \in \hat{\mathcal{P}}$, then u_k must be the termination-point of a bridge since a bridge cannot share links with the primary path. As illustrated in Fig. 4, node u_2 must be the termination-point of the bridge \mathcal{B}_1 , since its successor node in the walk, node u_3 , is also included in $\hat{\mathcal{P}}$.

Based on the above observations, the adjusted delay of a walk ending at node $u_k \in \hat{\mathcal{P}}$ is defined as follows. (Below, $D = \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) + d_{(u_{k-1},u_k)}$).

- **Case 1:** If $(u_k, u_{k-1}) \notin \hat{\mathcal{P}}$ and $D > D(\hat{\mathcal{P}}_{(s,u_k)}) + \Delta$, then node u_k cannot be the termination-point of a valid bridge and it may only be an internal node of a bridge. Thus, $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D$.
- **Case 2:** If $(u_k, u_{k-1}) \notin \hat{\mathcal{P}}$ and $D \leq D(\hat{\mathcal{P}}_{(s,u_k)}) + \Delta$ then node u_k may be either the termination-point or an internal node of a bridge. As a result, since u_k may be

the starting-point of a new bridge, we set $\tilde{D}(\mathcal{W}_{(s,u_k)}) = \min\{D(\hat{\mathcal{P}}_{(s,u_k)}), D\}$.

- **Case 3:** If $(u_k, u_{k-1}) \in \hat{\mathcal{P}}$ and $\tilde{D}(\mathcal{W}_{(s,u_{k-1})}) \leq D(\hat{\mathcal{P}}_{(s,u_{k-1})})$, then it follows that node u_k is not a termination-point of a bridge and all the bridges included by the walk satisfy the delay constraint. Since u_k may be the starting-point of a new bridge, its adjusted delay is $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D(\hat{\mathcal{P}}_{(s,u_k)})$.
- **Case 4:** If $(u_k, u_{k-1}) \in \hat{\mathcal{P}}$ and $\tilde{D}(\mathcal{W}_{(s,u_{k-1})}) > D(\hat{\mathcal{P}}_{(s,u_{k-1})}) + \Delta$, then it follows that, if node u_{k-1} is the termination-point of a bridge (that is, the predecessor of u_{k-1} in walk \mathcal{W} does not belong to $\hat{\mathcal{P}}$), then the bridge ending at u_{k-1} does not satisfy the delay constraint (due to Condition (1)). On the other hand, if u_{k-1} is not the termination-point of a bridge, then an induction argument can be used to show that some bridge preceding u_{k-1} in walk \mathcal{W} does not satisfy the delay constraint. Hence, $\tilde{D}(\mathcal{W}_{(s,u_k)})$ is set to ∞ .

Note 1: Cases 1-4 above consider all possibilities except the case that $(u_k, u_{k-1}) \in \hat{\mathcal{P}}$ and $D(\hat{\mathcal{P}}_{(s,u_{k-1})}) < \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) \leq D(\hat{\mathcal{P}}_{(s,u_{k-1})}) + \Delta$. However, it can be shown by an inductive argument that this case is impossible.

We proceed to present a formal definition of adjusted delay that considers the four cases mentioned above.

Definition 4 (Adjusted Delay): Let $\hat{\mathcal{P}}$ be a primary path and let G' be the auxiliary graph formed from G by reversing each link $l \in \hat{\mathcal{P}}$. Then, the *adjusted delay* $\tilde{D}(\mathcal{W})$ of a walk $\mathcal{W} = \{s = u_0, \dots, u_{k-1}, u_k\}$ in G' is defined recursively as follows:

- 1) The adjusted delay of an empty walk (i.e., $k = 0$) is 0: $\tilde{D}(\{s\}) = 0$;
- 2) Otherwise, the adjusted delay $\tilde{D}(\mathcal{W})$ of a walk $\mathcal{W} = \{u_0, \dots, u_{k-1}, u_k\}$ is
 - $\min\{D(\hat{\mathcal{P}}_{(s,u_k)}), D\}$ if $(u_k, u_{k-1}) \notin \hat{\mathcal{P}}$, and $D \leq D(\hat{\mathcal{P}}_{(s,u_k)}) + \Delta$,
 - $D(\hat{\mathcal{P}}_{(s,u_k)})$ if $(u_k, u_{k-1}) \in \hat{\mathcal{P}}$ and $\tilde{D}(\mathcal{W}_{(s,u_{k-1})}) \leq D(\hat{\mathcal{P}}_{(s,u_{k-1})})$,
 - ∞ if $(u_k, u_{k-1}) \in \hat{\mathcal{P}}$ and $\tilde{D}(\mathcal{W}_{(s,u_{k-1})}) > D(\hat{\mathcal{P}}_{(s,u_{k-1})}) + \Delta$,
 - D otherwise,

where $D = \tilde{D}(\mathcal{W}_{(s,u_{k-1})}) + d_{(u_{k-1},u_k)}$.

We illustrate the calculation of the adjusted delay using the walk shown in bold in Fig. 4. Here, the primary path is $\hat{\mathcal{P}} = \{s, u_3, u_2, u_4, v, t\}$. The delay of every link $l \in \hat{\mathcal{P}}$ is $d_l = 1$ and the delay constraint $\hat{d} = 7$. Thus, $D(\hat{\mathcal{P}}) = 5$ and $\Delta = 2$. The delay of every other link $l \notin \hat{\mathcal{P}}$ is depicted in the figure. Let us calculate the adjusted delay of the walk $\mathcal{W}_{(s,t)}$ and its various prefixes. At the base of the recursion, $\tilde{D}(\{s\}) = 0$. Since node u_1 does not belong to $\hat{\mathcal{P}}$, we set $\tilde{D}(\mathcal{W}_{(s,u_1)}) = 2$. For computing the adjusted delay of node u_2 we calculate the value $D = \tilde{D}(\mathcal{W}_{(s,u_1)}) + d_{(u_1,u_2)} = 3$. Since $D \leq D(\hat{\mathcal{P}}_{(s,u_2)}) + \Delta = 4$, node u_2 may be either the termination-point of a bridge that satisfies the delay constraint or an internal node of a bridge. To allow these two possibilities, we set $\tilde{D}(\mathcal{W}_{(s,u_2)}) = \min\{D(\hat{\mathcal{P}}_{(s,u_2)}), D\} = \min\{2, 3\} = 2$. Since $(u_3, u_2) \in \hat{\mathcal{P}}$, node u_2 must be the termination-point of the bridge \mathcal{B}_1 . Since the bridge \mathcal{B}_1 satisfies the delay

constraint, we have $\tilde{D}(\mathcal{W}_{(s,u_3)}) = D(\hat{\mathcal{P}}_{(s,u_3)}) = 1$. Now, to compute $\tilde{D}(\mathcal{W}_{(s,u_4)})$, we calculate the corresponding value $D = \tilde{D}(\mathcal{W}_{(s,u_3)}) + d_{(u_3,u_4)} = 6$. Because $D > D(\hat{\mathcal{P}}_{(s,u_4)}) + \Delta = 3 + 2 = 5$, node u_4 can only be an internal node inside a bridge and its adjusted delay is $\tilde{D}(\mathcal{W}_{(s,u_4)}) = 6$. Finally, when computing the adjusted delay of $\mathcal{W}_{(s,t)}$, $D = \tilde{D}(\mathcal{W}_{(s,u_4)}) + d_{(u_4,t)} = 7$. Node t is the termination-point of a valid bridge \mathcal{B}_2 and $\tilde{D}(\mathcal{W}_{(s,t)}) = \min\{D(\hat{\mathcal{P}}), D\} = 5$. We conclude that the given walk represents a feasible restoration topology.

C. Feasible Walk Concept

An (s, t) -walk whose adjusted delay does not exceed the delay $D(\hat{\mathcal{P}})$ of the primary path $\hat{\mathcal{P}}$ is referred to as a *feasible (s, t) -walk*. From Lemma 2 below, it follows that there is a one-to-one correspondence between feasible walks and feasible restoration topologies.

Lemma 2:

- 1) Let $\mathcal{R} = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_h\}$ be a feasible restoration topology and \mathcal{W} be the corresponding (s, t) -walk in G' . Then, $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$.
- 2) Let $\hat{\mathcal{P}}$ be a primary path and \mathcal{W} be an (s, t) -walk in the auxiliary graph G' such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$. Then, there exists a feasible restoration topology \mathcal{R} that corresponds to \mathcal{W} .

Proof: See [2]. ■

In general, there may be more than one way to decompose a walk into a set of bridges, i.e., there are several sets of bridges that can be constructed out of a single walk. For example, in Fig. 4 there are two sets of bridges S_1 and S_2 that correspond to walk \mathcal{W} : S_1 is formed by bridges $\{\mathcal{B}_1, \mathcal{B}_{2a}, \mathcal{B}_{2b}\}$, while in S_2 , the bridges \mathcal{B}_{2a} and \mathcal{B}_{2b} are combined into a single bridge \mathcal{B}_2 . Note that only some of these sets constitute feasible restoration topologies. One can construct a feasible restoration topology from a feasible walk by simply choosing as the termination-point for a bridge, the first node $u_k \in \hat{\mathcal{P}}$ in the bridge for whom $\tilde{D}(\mathcal{W}_{(s,u_k)}) = D(\hat{\mathcal{P}}_{(s,u_k)})$.

We denote by \mathcal{W}^{opt} the minimum cost feasible (s, t) -walk in the auxiliary graph G' and by c^{opt} the cost of \mathcal{W}^{opt} . We show a relationship between \mathcal{W}^{opt} and the optimum restoration topology (whose cost is denoted by OPT).

Suppose that we assign a cost of 0 to each link in the auxiliary graph G' that originated from a link $l \in \hat{\mathcal{P}}$. Clearly, due to Lemmas 1 and 2, it follows that there exists a feasible walk \mathcal{W} such that $C(\mathcal{W}) \leq 2 \cdot OPT$. Thus, $c^{opt} \leq 2 \cdot OPT$. Further, note that the cost of a restoration topology constructed from a walk never exceeds the cost of the walk itself. Thus, if we could compute the optimal feasible walk, then we could compute a (2,1)-approximation to the optimal restoration topology.

V. APPROXIMATION ALGORITHMS FOR PROVISIONING OF RESTORATION TOPOLOGIES

We are now in a position to present efficient approximation algorithms for Problems RT and P+RT. We begin with a pseudo-polynomial algorithm for Problem RT, which serves as the basic building block for the approximation algorithm for Problem RT presented in Section V-B. Finally, in Section V-C we present the approximation algorithm for Problem P+RT.

A. Pseudo-polynomial Solution for Problem RT

The fundamental concepts of adjusted delay and feasible (s, t) -walk give rise to a pseudo-polynomial algorithm for Problem RT, i.e., an algorithm whose running time is proportional to the cost of the optimal solution. The algorithm, referred to as Algorithm PP is presented in this section. Because of its simplicity, the algorithm can be easily implemented in practice. However, in the worst case, its running time can be very high.

Algorithm PP computes a minimum-cost feasible (s, t) -walk \mathcal{W} and the corresponding restoration topology. The algorithm is a natural extension of the well-known Bellman-Ford algorithm and uses the dynamic-programming technique of *relaxation* [3]. The algorithm assumes that the costs of links are integer values greater than 0, and an upper bound U on the cost of the solution is given.

We first describe the *relaxation* technique used by Algorithm PP. For each node $v_i \in V$, we maintain an array $D_{v_i}[c]$ of *minimum delay estimates*. The array $D_{v_i}[c]$ stores, for each cost c , the minimum adjusted delay of an (s, v_i) -walk, whose cost is at most c . Initially, $D_{v_i}[c] = \infty$ for every $v_i \in S \setminus s$ and $c \geq 0$. We only relax links whose cost does not exceed the current budget restriction c . The process of relaxing a link (v_i, v_j) consists of testing whether we can improve the best (s, v_j) -walk (i.e., the walk whose adjusted delay is minimum) found so far to v_j by going through v_i without exceeding the current budget restriction c and if so, updating $D_{v_j}[c]$. The relaxation technique is implemented by Procedure RELAX (see Fig. 5).

Next, we proceed to describe Algorithm PP, whose pseudo-code appears in Fig. 5. The purpose of Algorithm PP is to check, for a given value of upper bound U , whether there exists an (s, t) -walk \mathcal{W} in G' , such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$ and $C(\mathcal{W}) \leq U$, and if so, to find a minimum cost (s, t) -walk \mathcal{W} such that $\tilde{D}(\mathcal{W}) \leq D(\hat{\mathcal{P}})$. We start with a zero budget for c and increment it by a value of 1 in each iteration until either $D_t[c] \leq D(\hat{\mathcal{P}})$, i.e., there exists a walk between s and t whose adjusted delay is at most $D(\hat{\mathcal{P}})$, or else $c = U$. In each iteration, we process each node $v_j \in G'$ by relaxing all links entering v_j .

As discussed below, in our approximation algorithm, Algorithm PP is applied to graphs in which $c_l \geq 1$ for each link l . Thus, the cost of each link $l \in G'$ is at least 1, except for links in G' that originate from the primary path $\hat{\mathcal{P}}$. Note that for each link (v_i, v_j) with zero cost, node v_i must be processed before v_j . Accordingly, the nodes $v_j \in G'$ are processed in an order such that v_j is before $v_{j'}$ if v_j is a successor of $v_{j'}$ in $\hat{\mathcal{P}}$.

Also, in Step 15, the algorithm identifies a walk $\mathcal{W} = \{s = u_0, \dots, u_k = t\}$ whose adjusted delay is at most $D(\hat{\mathcal{P}})$ using backtracking. Suppose that c is the value at which Algorithm PP breaks out of the for loop (in Step 16), that is, $D_t[c] \leq D(\hat{\mathcal{P}})$. Then, beginning with node t , for each node u_i , the backtracking procedure adds to the returned walk, the link (u_{i-1}, u_i) that resulted in the value $D_{u_i}[c_i]$ until node s is reached, where the c_i values for u_i are computed as follows: for the initial node $u_k = t$, $c_k = c$ and for every subsequent node u_{i-1} , $c_{i-1} = c_i - c_{(u_{i-1}, u_i)}$. Thus, the cost of the walk \mathcal{W} can be shown to be at most c . The computational complexity of Algorithm PP is $\mathcal{O}(M \cdot U)$.

Algorithm PP ($G(V, E), \hat{\mathcal{P}}, \hat{d}, U$):

parameters:
 $G(V, E)$ - network,
 $\{d_l, c_l\}_{l \in E}$ - delays and costs of the network links,
 $\hat{\mathcal{P}} = \{s = v_0, v_1, \dots, t = v_n\}$ - QoS path,
 \hat{d} - delay constraint,
 U - the upper bound on the cost of \mathcal{R} .

```

1  $\Delta \leftarrow \hat{d} - D(\hat{\mathcal{P}})$ 
2  $E' \leftarrow E$ 
3 for each link  $l = (v_i, v_{i+1}) \in \hat{\mathcal{P}}$  do
4    $E' \leftarrow E' \setminus \{(v_i, v_{i+1}) \in \hat{\mathcal{P}}\}$ 
5    $E' \leftarrow E' \cup \{(v_{i+1}, v_i) \in \hat{\mathcal{P}}\}, c_{(v_{i+1}, v_i)} \leftarrow 0$ 
6 for all  $v_i \in V$  do
7    $D_{v_i}[0] \leftarrow \infty$ 
8    $D_s[0] \leftarrow 0$ 
9   for  $c = 1, 2, \dots, U$  do
10    for each  $v_j \in V$  in order such that  $v_j$  is before  $v_{j'}$  if  $v_j$ 
        is a successor of  $v_{j'}$  in  $\hat{\mathcal{P}}$  do
11       $D_{v_j}[c] \leftarrow D_{v_j}[c - 1]$ 
12      for each link  $l = (v_i, v_j) \in E'$  do
13        RELAX( $l(v_i, v_j), c, \Delta$ )
14      if  $D_t[c] \leq D(\hat{\mathcal{P}})$  then
15        determine walk  $\mathcal{W}$  by backtracking
16      return  $\mathcal{W}$ .
17 return FAIL

Procedure RELAX ( $l = (v_i, v_j), c, \Delta$ )
1 if  $(v_j, v_i) \in \hat{\mathcal{P}}$  then
2   if  $D_{v_i}[c] \leq D(\hat{\mathcal{P}}_{(s, v_i)})$  then
3      $D_{v_j}[c] \leftarrow \min\{D_{v_j}[c], D(\hat{\mathcal{P}}_{(s, v_j)})\}$ 
4 else
5   if  $c_l \leq c$  then
6      $D_{v_j}[c] \leftarrow \min\{D_{v_j}[c], D_{v_i}[c - c_l] + d_l\}$ 
7   if  $v_j \in \hat{\mathcal{P}}$  and  $D_{v_j}[c] \leq D(\hat{\mathcal{P}}_{(s, v_j)}) + \Delta$  then
8      $D_{v_j}[c] \leftarrow \min\{D_{v_j}[c], D(\hat{\mathcal{P}}_{(s, v_j)})\}$ 

```

Fig. 5. Algorithm PP.

Theorem 1: Let c^{opt} denote the minimum cost of a feasible (s, t) -walk in G' . Then, the following two conditions hold:

- 1) The walk \mathcal{W} returned by Algorithm PP is feasible.
- 2) If $c^{opt} \leq U$ then Algorithm PP returns a minimum cost feasible walk \mathcal{W} .

Proof: See [2]. ■

B. Approximation Algorithm for Problem RT

In this section, we develop an approximation algorithm for computing the minimum cost feasible (s, t) -walk, and use this near-optimal walk to construct a near-optimal restoration topology. The technique we use is similar to the one presented in [15].

We begin with a high-level overview of the approximation algorithm. A critical building block of the algorithm is Procedure SCALE (Fig. 6), which uses *scaling* and *rounding* in order to efficiently find an approximate solution. The efficiency of Procedure SCALE depends on the tightness of the lower and upper bounds, L, U , on the cost of the optimal solution. Thus, to compute sufficiently tight lower and upper bounds, we rely on two procedures, namely Procedure BOUND and Procedure TEST. The former is used for obtaining initial values

of L, U such that $U/L < 2 \cdot N$, while the latter performs iterations to tighten the bounds further. Finally, we combine all the ideas in Procedure RT.

B.1. Scaling and Rounding

Recall that the computational complexity of Algorithm PP is $\mathcal{O}(M \cdot U)$, where U is an upper bound on the minimum cost c^{opt} of a feasible (s, t) -walk in G' . Thus, the computational complexity of Algorithm PP effectively depends on c^{opt} , which in turn depends on the values of the link costs. This implies that the computational complexity is high for large values of link costs. The idea of the scaling method is to reduce link costs, and, in turn, the computational complexity of the algorithm. Scaling introduces a certain penalty in terms of the quality (i.e., cost) of the obtained solution and a keypoint is to perform it in a way that keeps this penalty low. Specifically, we substitute the cost c_l of each link $l \in E$ by a new value c'_l , as follows:

$$c'_l = \left\lfloor \frac{c_l}{S} \right\rfloor + 1,$$

where $S = \frac{L\varepsilon}{2N}$. Clearly, with the new costs c'_l , there must exist a feasible walk with cost at most $\left\lfloor \frac{c^{opt}}{S} \right\rfloor + 2N$ and no more than $2N$ links (due to Lemma 1). Thus, the actual cost of the path returned by Algorithm PP (in the final step) is no more than $c^{opt} + 2NS \leq (1 + \varepsilon) \cdot c^{opt}$. It follows that if Procedure SCALE is invoked with valid lower and upper bounds, i.e., $L \leq c^{opt} \leq U$, then it returns a walk whose cost is at most $(1 + \varepsilon)$ times more than the optimum. The formal description of Procedure SCALE appears in Fig. 6.

We summarize this discussion in the following lemma:

Lemma 3: If Procedure SCALE is invoked with valid upper bound, i.e., $c^{opt} \leq U$, then it returns a walk whose cost is at most $(1 + \varepsilon)$ times more than the optimum.

We note that Procedure SCALE might return a feasible walk even if it is invoked with non valid upper bound, i.e., $c^{opt} > U$. However, we show that the cost of a walk \mathcal{W} returned by the algorithm always satisfies $C(\mathcal{W}) \leq L\varepsilon + U$. We use this property later to compute tight lower and upper bounds on c^{opt} .

Lemma 4: Any walk \mathcal{W} returned by Procedure SCALE is feasible and satisfies $C(\mathcal{W}) \leq L\varepsilon + U$.

Proof: Let $\hat{\mathcal{W}}$ be a walk returned by Procedure SCALE. Note that $\hat{\mathcal{W}}$ was returned by Algorithm PP. By Theorem 1 (Part 1), $\hat{\mathcal{W}}$ is a feasible walk. Let $C'(\hat{\mathcal{W}})$ be the cost of $\hat{\mathcal{W}}$ with respect to the scaled link costs c'_l . Since $C'(\hat{\mathcal{W}}) \leq \tilde{U} = \left\lfloor \frac{U}{S} \right\rfloor + 2N$, and since $c_l \leq c'_l \cdot S$ for each $l \in E$, we have $C(\hat{\mathcal{W}}) \leq U + 2NS = L \cdot \varepsilon + U$. ■

The running time of Procedure SCALE is $\mathcal{O}\left(\frac{MNU}{\varepsilon L}\right)$. Thus, if we can compute tight lower and upper bounds on c^{opt} such that the ratio U/L is a constant, then we can reduce the computation time of Procedure SCALE to $\mathcal{O}\left(\frac{MN}{\varepsilon}\right)$. We next show how to compute these tight bounds.

B.2. Lower and upper bounds

In this subsection, we present Procedure BOUND (see Fig. 6), which identifies lower and upper bounds L, U on the minimum cost c^{opt} of a feasible walk c^{opt} such that $U/L \leq 2N$.

We begin by constructing a set $\{c_l \mid l \in G\}$ that contains the values of the link costs. Then, we sort this set in order to obtain the distinct cost values $c^1 < c^2 < \dots < c^r$. Note that this operation requires $\mathcal{O}(M \log N)$ time. Our goal is to find the maximum cost value $c^* \in \{c^i\}$ such that the graph G'' derived from G' by omitting all links whose cost is greater than c^* , does not contain a feasible (s, t) -walk. Clearly, a feasible (s, t) -walk contains at least one link whose cost is c^* or more, hence c^* is a lower bound on c^{opt} . In addition, there exists a feasible (s, t) -walk that comprises of links whose cost is c^* or less, and whose hop count is, by Lemma 1, at most $2N$. We conclude that $2N \cdot c^*$ is an upper bound on c^{opt} .

Procedure BOUND performs a binary search on the values c^1, c^2, \dots, c^r . At each iteration, we need to check whether $c \leq c^*$, where c is the current estimate of c^* . For this purpose, we remove from G all links whose cost is more than c , and assign the unit cost to the remaining links. Then, we apply Algorithm PP on the resulting graph, with the parameter $U = 2N$. If Algorithm PP returns a feasible walk, then $c \geq c^*$; otherwise, $c < c^*$. The computational complexity of Procedure BOUND is $\mathcal{O}(MN \log N)$.

B.3. A testing procedure

In order to tighten the bounds further, we make use of Procedure TEST (shown in Fig. 6). Procedure TEST performs the following 2-approximation test: if the procedure returns a positive answer, then definitely $c^{opt} < 2B$; otherwise, it is the case that $c^{opt} \geq B$.

Procedure TEST is implemented by invoking Procedure SCALE with $U = L = B$ and $\varepsilon = 1$.

Lemma 5: If Procedure TEST returns a positive answer, then $c^{opt} \leq 2B$. Otherwise, $c^{opt} > B$.

Proof: If Procedure TEST returns a positive answer, then Procedure SCALE does not fail. Thus, by Lemma 4, Procedure SCALE returns a feasible walk \mathcal{W} , for which $C(\mathcal{W}) \leq 2B$, hence $c^{opt} \leq C(\mathcal{W}) \leq 2B$.

We proceed to prove the second part of the lemma. By way of contradiction, assume that $c^{opt} \leq B$. By Lemma 3, Procedure SCALE does not return FAIL, hence Procedure TEST returns a positive answer, which contradicts the condition of the lemma. ■

B.4. Putting it all together

We are now in a position to combine the results of the previous subsections in order to present our final approximation algorithm, referred to as Algorithm RT (see Fig. 6).

The algorithm begins by applying Procedure BOUND, which provides the lower and upper bounds L and U on c^{opt} such that $U/L \leq 2N$. Then, we iteratively apply Procedure TEST to improve these bounds until the ratio U/L falls below 8. As we show below (proof of Lemma 6), this requires only a small number (at most $\log \log(2N)$) of iterations.

In each iteration, we invoke Procedure TEST with $B = \sqrt{L \cdot U}$. If Procedure TEST returns a positive answer, then, $c^{opt} < 2B$, hence U is set to $2B$. Otherwise, it is the case that $c^{opt} > B$, hence L is set to B . Note that, if the ratio U/L is equal to x at the beginning of an iteration, then at the end of the iteration we have $(U/L) \leq 2\sqrt{x}$. Thus, since the above process terminates once $U/L \leq 8$, the number of iterations performed can be shown to be $\mathcal{O}(\log \log N)$.

Algorithm RT ($G(V, E), \hat{\mathcal{P}}, \hat{d}, \varepsilon$):

parameters:

$G(V, E)$ - network

$\hat{\mathcal{P}} = \{s = v_1, v_2, \dots, t = v_n\}$ - QoS path,

\hat{d} - delay constraint

ε - approximation ratio

```

1  $L, U \leftarrow \text{BOUND}(G(V, E), \hat{\mathcal{P}}, \hat{d})$ 
2 do
3    $B \leftarrow \sqrt{L \cdot U}$ 
4   if  $\text{TEST}(G(V, E), \hat{\mathcal{P}}, \hat{d}, B)$  returns YES then
5      $L \leftarrow B$ 
6   else
7      $U \leftarrow 2 \cdot B$ 
8   until  $U/L \leq 8$ .
9  $\mathcal{W} \leftarrow \text{SCALE}(G(V, E), \hat{\mathcal{P}}, \hat{d}, L, U, \varepsilon)$ 
10 return the restoration topology that corresponds to  $\mathcal{W}$ .
```

Procedure $\text{SCALE}(G(V, E), \hat{\mathcal{P}}, \hat{d}, L, U, \varepsilon)$

```

1  $S \leftarrow \frac{L\varepsilon}{2N}$ 
2 for each link  $l \in E$  do
3    $c'_l \leftarrow \lfloor \frac{c_l}{S} \rfloor + 1$ 
4    $\tilde{U} \leftarrow \lfloor \frac{U}{S} \rfloor + 2N$ 
5 return  $\text{PP}(G(V, E), \{d_l, c'_l\}_{l \in E}, \hat{\mathcal{P}}, \hat{d}, \tilde{U})$ 
```

Procedure $\text{TEST}(G(V, E), \hat{\mathcal{P}}, \hat{d}, B)$

```

1 Apply Procedure SCALE for  $(G(V, E), \hat{\mathcal{P}}, \hat{d}, B, B, 1)$ 
2 if Algorithm SCALE returned FAIL then
3   return NO
4 else
5   return YES
```

Procedure $\text{BOUND}(G(V, E), \hat{\mathcal{P}}, \hat{d})$

```

1 let  $c^1 < c^2 < \dots < c^r$  the distinct costs values of the links.
2  $low \leftarrow 1$ ;  $high \leftarrow r$ 
3 while  $low < high - 1$ 
4    $j \leftarrow \lfloor (high + low)/2 \rfloor$ 
5    $E'' \leftarrow \{l \mid c_l \leq c^j\}$ 
6   set  $c_l \leftarrow 1$  for each  $l \in E''$ 
7   apply Algorithm PP on  $(G''(V, E''), \hat{\mathcal{P}}, \hat{d}, 2N)$ 
8   if Algorithm PP returned FAIL then
9      $high \leftarrow j$ 
10  else
11     $low \leftarrow j$ 
12   $U \leftarrow 2N \cdot c^{high}$ ;  $L \leftarrow c^{high}$ ;
13 return  $L, U$ ;
```

Fig. 6. Algorithm RT.

Having obtained lower and upper bounds L, U such that $U/L \leq 8$, we use Procedure SCALE to find a feasible walk \mathcal{W} , whose cost is at most $(1 + \varepsilon) \cdot c^{opt}$. Finally, we return the restoration topology corresponding to \mathcal{W} .

Lemma 6: The computational complexity of Algorithm RT is $\mathcal{O}(MN(1/\varepsilon + \log N))$.

Proof: Procedure BOUND requires $\mathcal{O}(MN \log N)$ time; the execution of Procedure SCALE in line 9 requires $\mathcal{O}(MN/\varepsilon)$ time.

We proceed to analyze the computational complexity of the loop of lines 2-8. We denote by x_i the ratio U/L at the beginning of iteration i . Initially, we have $x_1 \leq 2N$. As discussed above at iteration i , $x_i \leq 2\sqrt{x_{i-1}}$. It follows that

$$x_i \leq 2 \cdot 2^{1/2} \cdot 2^{1/4} \dots \cdot 2^{1/2^i} \cdot (2N)^{1/2^i} \leq 4 \cdot (2N)^{1/2^i}.$$

At iteration $k = \log \log(2N)$ we have $x_k \leq 8$. We conclude that the loop performs $\mathcal{O}(\log \log N)$ iterations. At each iteration we execute Procedure TEST, which requires $U = \mathcal{O}(MN)$ time. We conclude that the total running time of the loop is $\mathcal{O}(MN \log \log N)$.

We conclude that the total running time of Algorithm RT is $\mathcal{O}(MN(1/\varepsilon + \log N))$. ■

We summarize our results in the following theorem.

Theorem 2: Given an undirected graph G , a primary QoS path $\hat{\mathcal{P}} \in G$, a delay constraint \hat{d} and an approximation ratio ε , Algorithm RT identifies, in $\mathcal{O}(MN(1/\varepsilon + \log N))$ time, a feasible restoration topology \mathcal{R} for $(\hat{\mathcal{P}}, \hat{d})$, whose cost is at most $2 \cdot (1 + \varepsilon)$ times more than the optimum.

Proof: By Lemma 5, lines 4-7 ensure that at each iteration L and U are valid bounds, i.e., $L \leq c^{opt} \leq U$. Thus, Procedure SCALE is called at line 9 with valid bounds, hence by Lemma 3 it finds a feasible walk \mathcal{W} whose cost $C(\mathcal{W}) \leq (1 + \varepsilon) \cdot c^{opt}$. Further, from Lemmas 1 and 2, it follows that $c^{opt} \leq 2 \cdot OPT$. Thus, $C(\mathcal{W}) \leq 2(1 + \varepsilon) \cdot OPT$ and the cost of the restoration topology \mathcal{R} corresponding to \mathcal{W} satisfies $C(\mathcal{R}) \leq 2(1 + \varepsilon) \cdot OPT$. ■

C. Approximation Algorithm for Problem P+RT

The approximation algorithm for simultaneous provisioning of a primary QoS path and the restoration topology is implemented as follows. First, using Algorithm RSP, we identify a \hat{d} -delay constrained (s, t) -path $\hat{\mathcal{P}}$ in G whose cost is at most $(1 + \varepsilon)$ times the optimum. Then we apply Algorithm RT with parameters G , $\hat{\mathcal{P}}$, $(\hat{d} + D(\hat{\mathcal{P}}))$ and ε . The resulting algorithm is referred to as Algorithm P+RT.

Theorem 3: Algorithm P+RT identifies, in $\mathcal{O}(MN(1/\varepsilon + \log N))$ time, a $(3 \cdot (1 + \varepsilon), 2)$ -approximate solution for Problem P+RT.

Proof: The computational complexity of Algorithm P+RT is identical to that of Algorithm RT.

For a path \mathcal{P} we denote by $E(\mathcal{P})$ the set of links in \mathcal{P} . Similarly, $E(\mathcal{B})$ denotes the set of links in bridge \mathcal{B} .

Let $(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ be the output of Algorithm P+RT and let $(\hat{\mathcal{P}}^{opt}, \hat{\mathcal{R}}^{opt})$ be the optimal solution to Problem P+RT. We prove that there exists a restoration topology \mathcal{R}' for $(\hat{\mathcal{P}}, \hat{d} + D(\hat{\mathcal{P}}))$ such that $E(\mathcal{R}') \subseteq E(\hat{\mathcal{P}}^{opt}) \cup E(\hat{\mathcal{R}}^{opt})$.

For each link $l \in \hat{\mathcal{P}}$ we identify a bridge $\mathcal{B}_l = \{s_l, \dots, t_l\}$ such that \mathcal{B}_l protects l and $E(\mathcal{B}_l) \subseteq E(\hat{\mathcal{P}}^{opt}) \cup E(\hat{\mathcal{R}}^{opt})$. We consider the following two cases.

- **Case 1.** If $l = (v_i, v_{i+1}) \in \hat{\mathcal{P}}$ and $l \notin \hat{\mathcal{P}}^{opt}$ then we choose \mathcal{B}_l to be the subpath $\hat{\mathcal{P}}_{(s_l, t_l)}^{opt}$ of $\hat{\mathcal{P}}^{opt}$, where s_l is the first predecessor of v_i in $\hat{\mathcal{P}}$ that belongs to $\hat{\mathcal{P}}^{opt}$ and t_l is a first successor of v_{i+1} in $\hat{\mathcal{P}}$ that belongs to $\hat{\mathcal{P}}^{opt}$ (see Fig. 7 (a)).
- **Case 2.** If $l = (v_i, v_{i+1}) \in \hat{\mathcal{P}}$ and $l \in \hat{\mathcal{P}}^{opt}$ then let $\mathcal{B}' = \{s', \dots, t'\}$ be a bridge of $\hat{\mathcal{R}}^{opt}$ that protects l . We denote by \mathcal{P}' the path $\hat{\mathcal{P}}_{s, s'}^{opt} \circ \mathcal{B}' \circ \hat{\mathcal{P}}_{t', t}^{opt}$, i.e., the restoration path for link l in the optimal solution. Then, we choose \mathcal{B}_l to be the subpath $\mathcal{P}'_{(s_l, t_l)}$ of \mathcal{P}' , where s_l is the first predecessor of v_i in $\hat{\mathcal{P}}$ that belongs to \mathcal{P}' and t_l is a first successor of v_{i+1} in $\hat{\mathcal{P}}$ that belongs to \mathcal{P}' (see Fig. 7 (b)).

We note that, in both cases, $D(\mathcal{B}_l) \leq \hat{d}$. Indeed, in the first case, \mathcal{B}_l is a subpath of the optimal path $\hat{\mathcal{P}}^{opt}$, hence

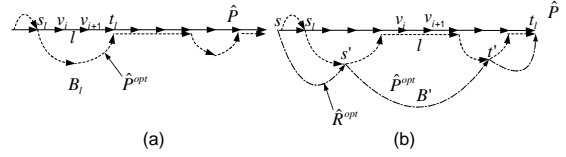


Fig. 7. (a) The optimal primary path $\hat{\mathcal{R}}^{opt}$ and an (s, t) -path $\hat{\mathcal{P}}$. (b) The optimal solution $(\hat{\mathcal{P}}^{opt}, \hat{\mathcal{R}}^{opt})$ to Problem P+RT and an (s, t) -path $\hat{\mathcal{P}}$.

$D(\mathcal{B}_l) \leq D(\hat{\mathcal{P}}^{opt}) \leq \hat{d}$; and in the second case, \mathcal{B}_l is a subpath of the restoration path \mathcal{P}' used by the optimal solution, whose delay is at most \hat{d} .

Let \mathcal{R}' be a restoration topology formed by bridges $\{\mathcal{B}_l \mid l \in \hat{\mathcal{P}}\}$. We observe that each link $l \in \mathcal{P}$ is protected by a bridge in \mathcal{R}' . Thus, as for each bridge $\mathcal{B}_l \in \mathcal{R}'$ it holds that $D(\mathcal{B}_l) \leq \hat{d}$, \mathcal{R}' is a feasible restoration topology for $(\hat{\mathcal{P}}, \hat{d} + D(\hat{\mathcal{P}}))$.

We also note that $C(\mathcal{R}') \leq C(\hat{\mathcal{P}}^{opt}) + C(\hat{\mathcal{R}}^{opt})$, since, \mathcal{R}' contains only links from $\hat{\mathcal{P}}^{opt}$ and $\hat{\mathcal{R}}^{opt}$. By Theorem 2, the cost $C(\mathcal{R})$ of \mathcal{R} is at most $2 \cdot (1 + \varepsilon)$ times more than the cost of the optimal restoration topology for $(\hat{\mathcal{P}}, \hat{d} + D(\hat{\mathcal{P}}))$, thus $C(\mathcal{R}) \leq 2(1 + \varepsilon) \cdot C(\mathcal{R}')$. As a result, $C(\mathcal{R}) \leq 2(1 + \varepsilon)(C(\hat{\mathcal{P}}^{opt}) + C(\hat{\mathcal{R}}^{opt}))$. Since $C(\hat{\mathcal{P}}) \leq (1 + \varepsilon)C(\hat{\mathcal{P}}^{opt})$, we have $C(\hat{\mathcal{P}}) + C(\mathcal{R}) \leq 3(1 + \varepsilon)(C(\hat{\mathcal{P}}^{opt}) + C(\hat{\mathcal{R}}^{opt}))$. Since $\hat{d} + D(\hat{\mathcal{P}}) \leq 2 \cdot \hat{d}$, it follows that \mathcal{R} is a $(3 \cdot (1 + \varepsilon), 2)$ -approximate solution for Problem P+RT. ■

VI. DIRECTED NETWORKS

In this section, we extend the previous results and apply them for directed networks, modeled as directed graphs. In such networks, for a pair of connected nodes (v_i, v_j) , the bandwidth provisioned on the link in the direction from v_i to v_j may be much larger than the allocated bandwidth in the opposite direction. In addition, the delay and cost characteristics of a link (v_i, v_j) may be very different from those of the reverse link (v_j, v_i) .

We observe that, with such asymmetric links, it is possible that a node or a link is shared by several bridges, which constitutes a major obstacle for identifying efficient solutions. For example, consider the network depicted on Fig. 8(a). The numbers show the link delays. For a delay constraint $\hat{d} = 10$, there exists only one feasible restoration topology $\mathcal{R} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4\}$, $\mathcal{B}_1 = \{v_0, v, u, v_4\}$, $\mathcal{B}_2 = \{v_1, v, u, v_5\}$, $\mathcal{B}_3 = \{v_2, v, u, v_6\}$ and $\mathcal{B}_4 = \{v_3, v, u, v_7\}$. Note that the nodes v, u and the link (v, u) are shared by all the bridges of \mathcal{R} .

We overcome this obstacle by combining bridges. Specifically, suppose that \mathcal{R} is an optimal restoration topology for $(\hat{\mathcal{P}}, \hat{d})$. We combine the bridges \mathcal{B}_1 and \mathcal{B}_4 into a single bridge \mathcal{B} , such that $\mathcal{B} = \mathcal{B}_1(v_0, v) \circ \mathcal{B}_4(v, v_7)$, as depicted in Fig. 8(b). We note that for each bridge $\mathcal{B}_i \in \mathcal{R}$ it holds that the delay of \mathcal{B}_i exceeds the delay of the subpath of $\hat{\mathcal{P}}$ protected by \mathcal{B}_i by at most $\Delta = \hat{d} - D(\hat{\mathcal{P}})$. Thus, we have $D(\mathcal{B}_1) \leq D(\hat{\mathcal{P}}_{(v_0, v_4)}) + \Delta$ and $D(\mathcal{B}_4) \leq D(\hat{\mathcal{P}}_{(v_3, v_7)}) + \Delta$. This implies that

$$D(\mathcal{B}) \leq D(\mathcal{B}_1) + D(\mathcal{B}_4) \leq D(\hat{\mathcal{P}}_{(v_0, v_4)}) + D(\hat{\mathcal{P}}_{(v_3, v_7)}) + 2\Delta \leq D(\hat{\mathcal{P}}_{(v_0, v_7)}) + 2\hat{d} - D(\hat{\mathcal{P}}),$$

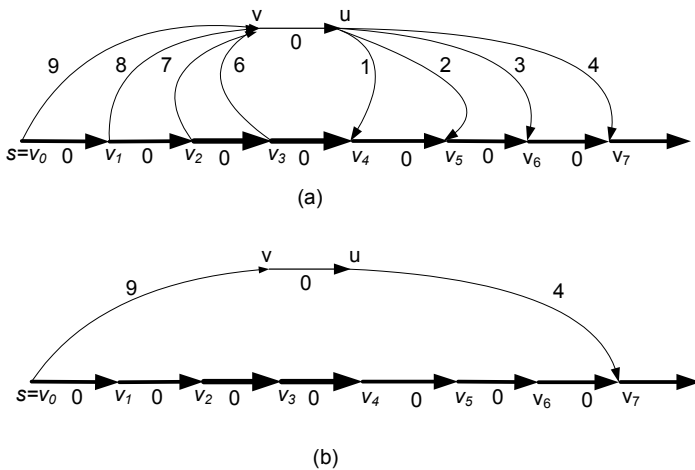


Fig. 8. (a) An example of a restoration topology \mathcal{R} formed by bridges $\mathcal{B}_1 = \{v_0, v, u, v_4\}$, $\mathcal{B}_2 = \{v_1, v, u, v_5\}$, $\mathcal{B}_3 = \{v_2, v, u, v_6\}$ and $\mathcal{B}_4 = \{v_3, v, u, v_7\}$. Link $l(v, u)$ is shared by the bridges $\mathcal{B}_1, \dots, \mathcal{B}_4$. (b) The bridges \mathcal{B}_1 and \mathcal{B}_4 are replaced by a single bridge.

where the last inequality follows from the fact that $D(\hat{\mathcal{P}}_{(v_0, v_4)}) + D(\hat{\mathcal{P}}_{(v_3, v_7)}) = D(\hat{\mathcal{P}}_{(v_0, v_7)}) + D(\hat{\mathcal{P}}_{(v_3, v_4)}) \leq D(\hat{\mathcal{P}}_{(v_0, v_7)}) + D(\hat{\mathcal{P}})$. We conclude that the delay of the restoration path that includes the bridge \mathcal{B} is at most $D(\mathcal{B}) + D(\hat{\mathcal{P}}) - D(\hat{\mathcal{P}}_{v_0, v_7}) \leq 2\hat{d}$. We employ this observation in order to prove the following lemma, which is the counterpart of Lemma 1 for undirected networks.

Lemma 7: Given a directed graph G , a delay constraint \hat{d} and an (s, t) -path, $D(\hat{\mathcal{P}}) \leq \hat{d}$, there exists a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, 2\hat{d})$ such that $C(\hat{\mathcal{R}}) \leq OPT$, and each node $v \in \hat{\mathcal{R}}$ or link $l \in \hat{\mathcal{R}}$ is shared by at most two bridges.

Proof: See [2]. \blacksquare

Lemma 7 implies that $\hat{\mathcal{R}}$ is a feasible restoration topology with respect to the delay constraint $\hat{d}' = 2\hat{d}$. Thus, in order to achieve an efficient solution, we relax the delay constraint by using \hat{d}' instead of \hat{d} . For example, by invoking the (simple) algorithm presented in Section IV-A with the delay constraint \hat{d}' , we obtain a $(2 \cdot (1 + \varepsilon), 2)$ -approximate solution for Problem RT.

In this section we denote by \mathcal{W}^{opt} the optimal walk with respect to \hat{d}' . The cost of \mathcal{W}^{opt} is denoted by c^{opt} .

Generally, the approximation algorithm for directed networks is similar to that for the undirected case, except for the following:

- 1) the adjusted delay is defined with respect to the delay constraint $\hat{d}' = 2\hat{d}$;
- 2) we use $\Delta' = 2\hat{d} - D(\hat{\mathcal{P}})$ instead of $\Delta = \hat{d} - D(\hat{\mathcal{P}})$;
- 3) Algorithm PP is applied with the delay constraint \hat{d}' (instead of \hat{d});
- 4) a more elaborate procedure is required for finding the lower and upper bounds L, U on c^{opt} .

A more detailed discussion follows.

A. Approximation Algorithm for Problem RT

Recall that our approach for the undirected case was to first identify lower and upper bounds, L and U , on the cost of an optimum walk c^{opt} , such that $U/L \leq 2N$, and then iteratively improve these bounds by employing scaling on the

link costs. For directed networks, however, computing lower bound L on c^{opt} incurs high complexity. This is because any optimum walk might have a large hop count (recall that, in the undirected case, by Lemma 1, there exists an optimum walk whose hop count is at most $2N$). Our approach is to consider, for the purpose of computing the lower bound, only walks whose hop counts do not exceed $2N$. More specifically, we denote by \mathcal{W}_{2N}^{opt} the feasible walk of minimum cost whose hop count is at most $2N$, and by c_{2N}^{opt} the cost of \mathcal{W}_{2N}^{opt} . We then use the lower bound L on c_{2N}^{opt} instead of c^{opt} .

We identify initial bounds L and U by invoking Procedure BOUND (See Section V-B) for $(G(V, E), \hat{\mathcal{P}}, \hat{d}')$. More specifically, for a given value c , we construct an auxiliary graph G'' out of G by omitting all links whose cost is greater than c and assigning the unit cost to the remaining links. Then, we find the maximum cost value c^* such that Algorithm PP, applied to $(G'', \hat{\mathcal{P}}, 2\hat{d}, 2N)$, returns FAIL for any value $c \geq c^*$. Clearly, any feasible walk whose length is at most $2N$ contains at least one link whose cost is c^* , hence we set $L = c^*$. In addition, Algorithm PP identifies a feasible walk that includes at most $2N$ links, such that the cost of each link is at most c^* , hence we set $U = 2Nc^{opt}$. Then, the bounds L and U are iteratively improved until either a suitable walk is found or $U/L \leq 8$. In the latter case, we apply Procedure SCALE with parameters L and U in order to find a suitable walk.

Our algorithm is based on the following two lemmas.

Lemma 8: $c_{2N}^{opt} \leq 2 \cdot OPT$.

Proof: By Lemmas 2 and 7, there exists a feasible walk $\hat{\mathcal{W}}$ such that $|\hat{\mathcal{W}}| \leq 2N$ and $C(\hat{\mathcal{W}}) \leq 2 \cdot OPT$. Hence, $c_{2N}^{opt} \leq C(\hat{\mathcal{W}}) \leq 2 \cdot OPT$. \blacksquare

Lemma 9: If $c_{2N}^{opt} \leq U$, then Procedure SCALE returns a feasible walk \mathcal{W} whose cost is at most $(1 + \varepsilon)$ times more than c_{2N}^{opt} , i.e., $C(\mathcal{W}) \leq (1 + \varepsilon) \cdot c_{2N}^{opt}$.

Proof: Similar to the proof of Lemma 3, but using Lemma 8 instead of Lemma 1. \blacksquare

We proceed to describe the approximation algorithm in more detail. First, we invoke Procedure BOUND with parameters $G(V, E), \hat{\mathcal{P}}, \hat{d}'$. It is easy to verify that the procedure returns a lower bound L on c_{2N}^{opt} and an upper bound U on c^{opt} such that $U/L \leq 2N$. Next, we use the following iterative process in order to improve the bounds U and L . At each iteration, we compute $B = \sqrt{L \cdot U}$ and apply Procedure SCALE for $L = U = B$. There are several possible cases.

- **Case 1:** Procedure SCALE returns FAIL. Then, due to Lemma 9, $c_{2N}^{opt} > U$, i.e., there exists no feasible walk \mathcal{W} such that $C(\mathcal{W}) \leq B$ and $|\mathcal{W}| \leq 2N$. Accordingly, we set $L \leftarrow B$;
- **Case 2:** Procedure SCALE returns a feasible walk \mathcal{W} such that $C(\mathcal{W}) \leq L$. Then, the algorithm halts and returns the walk \mathcal{W} . Note that $C(\mathcal{W}) \leq c_{2N}^{opt} \leq 2 \cdot OPT$.
- **Case 3:** Procedure SCALE returns a feasible walk \mathcal{W} such that $L < C(\mathcal{W}) \leq 2B$. In this case we set $U \leftarrow 2B$.

Note that, by Lemma 4, if Procedure SCALE does not fail, it returns a feasible walk \mathcal{W} , whose cost is at most $2B$, hence all possible cases are covered. The process stops when a suitable walk is found or $U/L \leq 8$.

Having obtained a lower bound L on c_{2N}^{opt} and an upper bound U on c^{opt} such that $U/L \leq 8$, we apply Procedure SCALE for L and U . If the algorithm fails, then $L > U$, hence we return the walk \mathcal{W} due to which the upper

bound U was assigned its current value. Note that, due to Lemma 8, $C(\mathcal{W}) \leq L \leq c_{2N}^{opt} \leq 2 \cdot OPT$. Otherwise, due to Lemma 9, Procedure SCALE returns a walk \mathcal{W} such $C(\mathcal{W}) \leq (1 + \varepsilon)c_{2N}^{opt}$. In both cases, we identify a feasible restoration topology $\hat{\mathcal{R}}$ that corresponds to \mathcal{W} . It follows that $C(\hat{\mathcal{R}}) \leq 2(1 + \varepsilon)OPT$. The formal description of the algorithm, referred to as Algorithm DRT, appears in Fig. 9.

```

Algorithm DRT (  $G(V, E), \hat{\mathcal{P}}, \hat{d}, \varepsilon$ ):
  parameters:
     $G(V, E)$  - network
     $\hat{\mathcal{P}} = \{s = v_1, v_2, \dots, t = v_n\}$  - QoS path,
     $\hat{d}$  - delay constraint
     $\varepsilon$  - approximation ratio

  1  $\hat{d}' \leftarrow 2\hat{d}$ 
  2  $L, U \leftarrow \text{BOUND}(G(V, E), \hat{\mathcal{P}}, \hat{d}')$ 
  3 do
  4    $B \leftarrow \sqrt{L \cdot U}$ 
  5   Apply Procedure SCALE for  $G(V, E), \hat{\mathcal{P}}, \hat{d}', B, B, 1$ )
  6   if Procedure SCALE return FAIL then
  7      $L \leftarrow B$ 
  8   else
  9     Set  $\mathcal{W}$  be the walk returned by Procedure SCALE
 10     if  $C(\mathcal{W}) \leq L$  then
 11       return the restoration topology  $\mathcal{R}$  that corresponds to
            $\mathcal{W}$ .
 12   else
 13      $U \leftarrow 2 \cdot B$ ,
 14   until  $U/L \leq 8$ .
 15 Apply Procedure SCALE for  $(G(V, E), \hat{\mathcal{P}}, \hat{d}', L, U, \varepsilon)$ 
 16 if Procedure SCALE does not fail then
 17   Set  $\mathcal{W}$  be the walk returned by Procedure SCALE
 18 return the restoration topology  $\mathcal{R}$  that corresponds to  $\mathcal{W}$ .

```

Fig. 9. Algorithm DRT.

We summarize our results in the following theorem.

Theorem 4: Given are a directed graph G , a primary QoS path $\hat{\mathcal{P}} \in G$, a delay constraint \hat{d} and an approximation ratio ε . Then, Algorithm DRT identifies, in $\mathcal{O}(MN(1/\varepsilon + \log N))$ time, a feasible restoration topology \mathcal{R} for $(\hat{\mathcal{P}}, 2\hat{d})$, whose cost is at most $2(1 + \varepsilon)$ times more than OPT, i.e., a $(2(1 + \varepsilon), 2)$ -approximate solution to Problem RT.

B. Approximation Algorithm for Problem P+RT

The approximation algorithm for identifying the primary QoS path and restoration topology is similar to the undirected case. Namely, we first identify a \hat{d} -delay constrained (s, t) -path $\hat{\mathcal{P}}$ in G , whose cost is at most $(1 + \varepsilon)$ times more than the optimum. Then, we apply Algorithm DRT with parameters $G, \hat{\mathcal{P}}, (\hat{d} + D(\hat{\mathcal{P}}))$ and ε . The resulting algorithm is referred to as Algorithm DP+RT.

Theorem 5: Algorithm DP+RT identifies, in $\mathcal{O}(MN(1/\varepsilon + \log N))$ time, a $(3(1 + \varepsilon), 3)$ -approximate solution for Problem P+RT in directed graphs.

Proof: The proof follows similar lines as in Theorem 3. Let $(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ be the output of Algorithm P+RT and let $(\hat{\mathcal{P}}^{opt}, \hat{\mathcal{R}}^{opt})$ be the optimal solution to Problem P+RT. By Theorem 4, Algorithm DRT returns a restoration topology $\hat{\mathcal{R}}$ for $(\hat{\mathcal{P}}, 2\hat{d} + D(\hat{\mathcal{P}}))$ such that $C(\hat{\mathcal{R}}) \leq 2(1 + \varepsilon) \cdot C(\hat{\mathcal{R}}^{opt})$. Since $2\hat{d} + D(\hat{\mathcal{P}}) \leq 3 \cdot \hat{d}$, $(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ is a $(3(1 + \varepsilon), 3)$ -approximate solution for Problem P+RT.

The running time of the algorithm is dominated by Algorithm DRT, hence it is $\mathcal{O}(MN(1/\varepsilon + \log N))$. ■

VII. SIMULATION EXAMPLES

In order to further illustrate the efficiency of the proposed solutions, we conducted some simulation experiments. Our experiments included the following steps. First, we generated a network topology G and chose source and destination nodes s, t and a delay constraint \hat{d} . Then, we computed a primary path \mathcal{P}_1 between s and t that satisfies the delay constraint \hat{d} at minimum cost (by using Algorithm RSP). Finally, we compared the following two algorithms for computing the restoration topology:

- **Algorithm DP** - Two Disjoint Paths. Find a minimum cost path \mathcal{P}_2 that does not have common links with path \mathcal{P}_1 and satisfies the delay constraint \hat{d} . The path is computed by using Algorithm RSP.
- **Algorithm RT** - Restoration Topology. We use Algorithm RT (described in Section V-B) to provision the restoration topology \mathcal{R} for (\mathcal{P}_1, \hat{d}) .

Recall that Algorithms RSP and RT use an approximation parameter ε . In our experiments we chose ε to be a fairly small constant.

A. Network Generation Models

We used two different methods for generating the undirected network topologies, using the BRITE topology generation tool [17]. The first is Waxman's method [21] and the second is Barabasi and Albert's [1], described below. Both network generators assign delays to links based on the distance between the link's endpoints. Further, we assigned costs to links uniformly and randomly out of a fixed interval.

- **Waxman model** [21]. In this model, nodes are placed on a plane; the probability of interconnecting two nodes decreases exponentially with the Euclidean distance between them. We set the value for parameters α and β to 0.15 and 0.2, respectively.
- **Barabasi-Albert model** [1]. In this model, the node connectivity follows a power-law rule: very few nodes have high connectivity, and the number of nodes with lower connectivity increases exponentially as the connectivity decreases.

B. Experimental Results

In our experiments, we compared the costs of the restoration restoration topology computed by Algorithm RT with the cost of the restoration path computed by Algorithm DP for different values of the delay constraint \hat{d} . Fig. 10 depicts the experimental results for (a) Waxman and (b) Barabasi-Albert models in a network of 7000 nodes. In this figure, the x -axis depicts the delay ratio \hat{d}/d_{sp} , where d_{sp} is the minimum delay of an (s, t) -path; while the y -axis depicts the relative improvement $(c_{dp} - c_{rt})/c_{dp}$ achieved by Algorithm RT over Algorithm DP. Here, c_{rt} and c_{dp} are the provisioning costs of the solutions computed by Algorithms RT and DP, respectively. In addition, we present the values of the delay constraint for which the algorithms fail, i.e., cannot identify a restoration path or topology that satisfies the constraint.

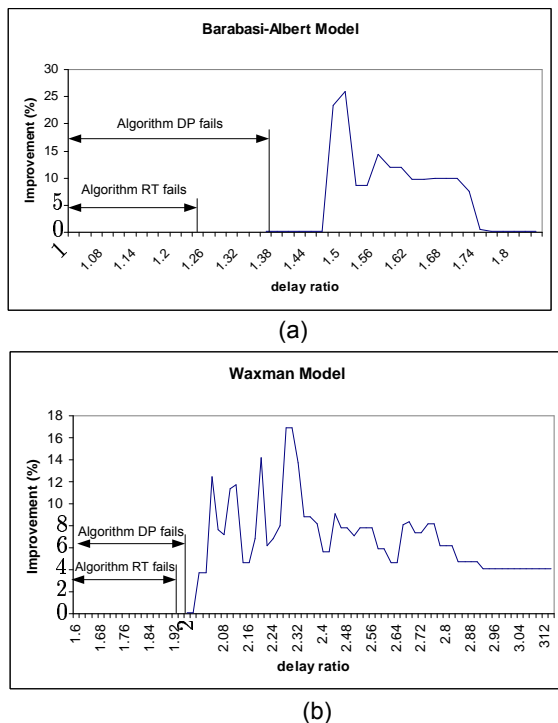


Fig. 10. Effect of delay ratio on the algorithm performance

The following observations can be made from the simulation results.

- In many of the cases in which Algorithm DP fails to find a pair of disjoint paths, Algorithm RT still computes a (feasible) primary path and restoration topology solution with a low cost. For example, for the Barabasi-Albert model, for values \hat{d}/d_{sp} between 1.24 and 1.36, Algorithm DP fails while Algorithm RT still provides a feasible solution of low cost. The same occurs for $\hat{d}/d_{sp} \leq 1.96$ in the Waxman model. This clearly demonstrates the advantage of the restoration topology strategy over the traditional disjoint-paths approach.
- Algorithm RT always exhibits superior performance (i.e., finds paths of lower cost) over Algorithm DP.
- The cost benefits due to Algorithm RT are particularly significant (around 15%) when the delay constraint is tight, i.e., closer to the minimum delay of an (s, t) -path.

VIII. CONCLUSION

In this paper, we investigated the problem of provisioning QoS paths with restoration. Specifically, we developed algorithms that compute a *primary QoS path* and a *restoration topology* comprising of a set of *bridges*, each of which protects a different part of the primary QoS path.

A major contribution of this paper is the concept of *adjusted delays*, which allows existing path algorithms (e.g., Bellman-Ford [3], Hassin's [7]) to be adapted in order to identify suitable restoration topologies. This enabled us to devise efficient approximation algorithms with proven performance guarantees. Specifically, we presented an $\mathcal{O}(MN(1/\epsilon + \log N))$ approximation algorithm (Algorithm P+RT) that provides $(3 \cdot (1 + \epsilon), 2)$ -approximate solutions for link failures. We

extended the algorithm for directed networks and achieved a $(3 \cdot (1 + \epsilon), 3)$ -approximate solution. We emphasize that, in our algorithms, the delay violation may occur only in the restoration paths, while *the primary path always satisfies the QoS constraint*.

There are several interesting topics for future research, which we proceed to describe. A challenging direction is to devise approximation algorithms for all problems considered in this study that yield better approximation ratios (i.e., lower delay violation and lower cost). In addition, the case of multiple link failures needs to be addressed. Such multiple failures could be handled by protecting each link by multiple bridges, which are mutually link-disjoint. This problem is very difficult and poses major challenges even if the number of simultaneous link failures is limited to just 2. Yet another important research topic is to devise efficient algorithms for concurrently computing primary paths and the corresponding restoration topologies for multiple connections, such that each connection has a different source-destination pair and delay requirement. Clearly, an efficient solution for this problem must use the network resources (such as link bandwidth) in a network-wide efficient manner. We believe that the concepts and techniques presented in this paper would be useful in the investigation of these problems.

REFERENCES

- [1] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- [2] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, and A. Sprintson. Algorithms for Computing QoS Paths with Restoration. CCIT Pub. No. 395, Department of Electrical Engineering, Technion, Haifa, Israel, August 2002. Available from: <http://ftp.technion.ac.il/pub/supported/ee/Network/os02.pdf>.
- [3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
- [4] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet – RFC No. 2386. Internet RFC, August 1998.
- [5] F. Ergun, R. Sinha, and L. Zhang. An Improved FPTAS for Restricted Shortest Path. *Information Processing Letters*, 83(5):237–293, September 2002.
- [6] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.
- [7] R. Hassin. Approximation Schemes for the Restricted Shortest Path Problem. *Mathematics of Operations Research*, 17(1):36–42, February 1992.
- [8] R. R. Iraschko and W. D. Grover. A Highly Efficient Path-Restoration Protocol for Management of Optical Network Transport Integrity. *IEEE Journal on Selected Areas in Communications*, 18(5):779–793, May 2000.
- [9] G. Italiano, R. Rastogi, and B. Yener. Restoration Algorithms for Virtual Private Networks in the Hose Model. In *Proceedings of IEEE INFOCOM'2002*, New York, NY, June 2002.
- [10] K. Kar, M. Kodialam, and T. V. Lakshman. Routing Restorable Bandwidth Guaranteed Connections using Maximum 2-Route Flows. *IEEE/ACM Transactions on Networking*, 11(5):772–781, Oct 2003.
- [11] M. S. Kodialam and T. V. Lakshman. Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration. In *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000.
- [12] R.R. Kodialam and T.V. Lakshman. Restorable Dynamic QoS Routing. *IEEE Communications Magazine*, 40(6):72–81, June 2002.
- [13] G. Krishna, M. Pradeep, and C. R. Murthy. A Segmented Backup Scheme for Dependable Real Time Communication in Multihop Networks. In *Proceedings of Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2000)*, Cancun, Mexico, May 2000.
- [14] G. Li, D. Wang, C. Kalmanek, and R. Doverspike. Efficient Distributed Path Selection for Shared Restoration Connections. In *Proceedings of IEEE INFOCOM'2002*, New York, NY, June 2002.
- [15] D.H. Lorenz and D. Raz. A Simple Efficient Approximation Scheme for the Restricted Shortest Path Problem. *Operations Research Letters*, 28(5):213–219, June 2001.

- [16] Q. Ma and P. Steenkiste. Quality of Service Routing for Traffic with Performance Guarantees. In *Proceedings of International Workshop on Quality of Service (IWQoS'97)*, Columbia University, New York, NY, May 1997.
- [17] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Universal topology generation from a user's perspective. In *Proceedings of Workshop the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, Cincinnati, OH, August 2001.
- [18] A. Orda. Routing With End to End QoS Guarantees in Broadband Networks. *IEEE/ACM Transactions on Networking*, 7(3):365–374, June 1999.
- [19] J.L. Sobrinho. Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet. *IEEE/ACM Transactions on Networking*, 10(4):541–550, Aug 2002.
- [20] J. Suurballe. Disjoint Paths in a Network. *Networks*, 4:125–145, 1974.
- [21] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1671–1622, 1988.

PLACE
PHOTO
HERE

Rajeev Rastogi is the Director of the Internet Management Research Department at Bell Laboratories, Lucent Technologies. He received the B. Tech degree in Computer Science from the Indian Institute of Technology, Bombay in 1988, and the masters and Ph.D. degrees in Computer Science from the University of Texas, Austin, in 1990 and 1993, respectively. He joined Bell Laboratories in Murray Hill, New Jersey, in 1993 and became a Bell Labs Fellow in 2003.

Rajeev Rastogi is active in the field of databases and has served as a program committee member for several conferences in the area. He currently serves on the editorial board of IEEE Transactions on Knowledge and Data Engineering. His writings have appeared in a number of ACM and IEEE publications, and other professional conferences and journals. His research interests include database systems, network management, and knowledge discovery. His most recent research has focused on the areas of network topology discovery, monitoring, configuration and provisioning, XML publishing, approximate query answering and data stream analysis.

PLACE
PHOTO
HERE

Yigal Bejerano received his B.Sc. in Computer Engineering in 1991 (summa cum laude), his M.Sc. in Computer Science in 1995, and his Ph.D. in Electrical Engineering in 2000, from the Technion - Israel Institute of Technology, Haifa, Israel. He is currently a member of the technical staff (MTS) at Bell Laboratories, Lucent Technologies. His research interests are mainly management aspects of high-speed and wireless networks, including the areas of mobility management, network monitoring, topology discovery and QoS routing. Dr. Bejerano is on the

technical program committee (TPC) of the INFOCOM -2002, 2003, 2004 and 2005 conferences.

PLACE
PHOTO
HERE

Yuri Breitbart (ACM F '00). Dr. Breitbart is Ohio Board of Regents Distinguished Professor of Computer Science in the Department of Computer Science at Kent State University. He has joined the Department in January 2002 after almost 6 years as a Member of Technical Staff and Distinguished Member of Technical Staff at the Bell Laboratories. Prior to that he was Professor and Chair in the Department of Computer Science at University of Kentucky. Dr. Breitbart has held visiting positions at the Swiss Technological Institute (ETH) in Zurich

and at HP Research Center in Palo Alto. Dr Breitbart's research is in the area of distributed information system, network management systems and bio-informatics. The focus of his research effort is on dealing with algorithmic and system issues of highly distributed autonomous and collaborative systems. Dr. Breitbart has received DSc degree in Computer Science from the Department of Computer Science at Israel Institute of Technology (Technion). He is a Fellow of the ACM and member of IEEE Computer Society and SIGMOD.

PLACE
PHOTO
HERE

Ariel Orda (S'84-M'92-SM'97) received the B.Sc. (summa cum laude), M.Sc., and D.Sc. degrees in Electrical Engineering from the Technion – Israel Institute of Technology, Haifa, Israel, in 1983, 1985, and 1991, respectively. Since 1994, he has been with the Department of Electrical Engineering at the Technion, where he is currently an Associate Professor and the Academic Head of the Computer Networking Laboratory. He has held visiting and research positions at the Center for Telecommunication Research, Columbia University, New York, NY,

Bell Laboratories, NJ, and IBM Watson Research Center, NY. In addition, he has held several consulting positions with Israeli industry. His current research interests include network routing, QoS provisioning, wireless networks, the application of game theory to computer networking and network pricing. He received the Award of the Chief Scientist in the Ministry of Communication in Israel, a Gutwirth Award for Outstanding Distinction, the Research Award of the Association of Computer and Electronic Industries in Israel, and the Jacknow Award for Excellence in Teaching. He served as Technical Program co-chair of IEEE Infocom 2002. He is an Editor of the IEEE/ACM Transactions on Networking and of the Journal of Computer Networks.

PLACE
PHOTO
HERE

Alexander Sprintson (S'00-M'03) received the B.Sc. degree (summa cum laude), M.Sc. and Ph.D. degrees in Electrical Engineering from the Technion – Israel Institute of Technology, Haifa, Israel, in 1995, 2001, and 2003, respectively. Since 2003, he has been with the Department of Engineering and Applied Science at the California Institute of Technology, Pasadena, California, where he is currently a postdoctoral research fellow. During the summers of 2002 and 2003 he was with the Internet Management Research Department at Bell Laboratories, Murray Hill, NJ. His research interests lie in the areas of QoS routing, resource allocation and scheduling in communication networks.

Dr. Sprintson received the Wolf Award for Distinguished Ph.D. Students, the Gutwirth Award for Outstanding Distinction and the Knesset (Israeli Parliament) Award for Distinguished Students.