

Dynamic Routing of Restorable QoS Connections in MPLS Networks

Gustav Rosenbaum, Chun Tung Chou and Sanjay Jha
School of Computer Science and Engineering
University of New South Wales, Australia
Email: {filipr, ctchou, sjha}@cse.unsw.edu.au

Deep Medhi
School of Computing and Engineering
University of Missouri-Kansas City
Email: dmedhi@umkc.edu

Abstract—In this paper we consider dynamic routing of restorable Quality of Service (QoS) connections in multi-protocol label switched (MPLS) networks under a single link failure model. To route a restorable QoS connection, two link disjoint label switched paths from the ingress to the egress node need to be computed such that both paths comply with the QoS constraints. When looking at QoS constraints like bandwidth guarantee and end-to-end delay bound, the current approach described in the literature converts the end-to-end delay bound into an effective bandwidth in a pre-processing step before computing two link disjoint bandwidth guaranteed paths. We argue that the current approach results in poor network performance and suggest a new approach that exploits the dependency between the end-to-end delay, chosen path and provisioned bandwidth. Thus, the two link disjoint paths are computed dynamically with their respective bandwidths so that they meet both the bandwidth guarantee and the end-to-end delay bound. In this paper, we present a new generic algorithm and two new linear programming formulations that implement the new approach. The two linear programming formulations are not intended to be used in a production environment due to their extensive running times, up to 2.5 minutes on average per request, but rather to benchmark approximation algorithms that in turn can be used in production. Furthermore our experiments show that the network performance improvement when exploiting the dependency between end-to-end delay, chosen path and provisioned bandwidth is substantial.

Index Terms—MPLS, Network resilience, QoS routing, Restoration routing

I. INTRODUCTION

Dynamic routing of restorable quality of service (QoS) connections in multi-protocol label switched (MPLS) networks is motivated by network providers' needs to satisfy customers who want both QoS and resilience guarantees simultaneously for their aggregated flows. A restorable QoS connection is a logical connection between two network nodes, an ingress and an egress node, that is resilient to network failures and meet the specified QoS constraints.

In this paper we consider a common restoration framework in which a restorable QoS connection is protected from end to end in a single link failure model [1], [2], [3]. Furthermore, we assume that the network provider uses MPLS and that requests for restorable QoS connections are routed dynamically one by one as they arrive without any knowledge of future requests. In the given framework, each restorable QoS connection is realized by two link-disjoint LSPs from the ingress to the

egress node. Thus when a request for a restorable QoS connection arrives, the network provider needs to first compute the two LSPs and then set them up in the network. The LSP set up can be handled using RSVP-TE [4]. The focus of this paper is on the path computation.

A typical scenario where a customer needs both QoS guarantees and resilience for its aggregated flows is when the customer uses real time applications like e-commerce and VoIP in a provider based virtual private network (VPN). Here, the aggregate flows are forwarded along a restorable QoS connection between two customer sites attached to an ingress and egress node in the provider network. When a customer wants to connect two sites, the customer requests a new restorable QoS connection between them from its network provider. The problem that a network provider faces when such a request is received, is how to compute two link disjoint LSPs from the ingress to egress node such that both paths meet the QoS constraints.

Motivated by the network providers' needs to provide bandwidth guaranteed services that are both delay-sensitive and resilient we focus this paper on the routing problem of restorable QoS connection subject to bandwidth guarantee and end-to-end delay constraints. The current approach described in the literature that deals with this problem is to convert the end-to-end delay bound into an effective bandwidth in a pre-processing step before computing the two LSPs [2], [3]. The pre-processing step effectively transforms the problem into computing two link disjoint paths using only a static bandwidth as QoS constraint.

Observing that the end-to-end delay depends on both the chosen path and the provisioned bandwidth, a new approach to the described routing problem can be taken. In the new approach, this dependency is exploited so that the two link disjoint paths are computed dynamically with their respective bandwidths. Thus, a solution to the routing problem of restorable QoS connections will not only contain two LSPs but their respective bandwidths as well.

There are basically two ways to approach the routing problem of restorable QoS connections with dynamic bandwidth allocation. One way is to decompose it into problems where existing solutions already exists. For instance, one can decompose the problem into QoS routing of non-restorable connections where known solutions like the ones presented

in [5], [6], [7], [8] can be applied. Another example is to decompose it into a problem of routing restorable bandwidth guaranteed connections [1], [2], [3] and end-to-end delay verification [9]. Decomposing the problem inevitably trades accuracy for simplicity, hence a given decomposition might not find a solution even though one exists. It is therefore important to look at the combined way to approach the problem whereby the two LSPs and their respective bandwidths are computed simultaneously.

The decomposed approach has not yet been fully understood in the literature and the combined approach is to our knowledge not addressed at all. In this paper we discuss the two approaches and propose three new algorithms, one generic algorithm that implements a decomposed approach and two new linear programming (LP) formulations that implement the combined approach. The presented LP formulations are not suitable for a production environment due to their inhabited complexity. However, they are suitable to benchmark approximation algorithms that in turn can be used in production.

The rest of this paper is organized as follows. Problem formulation is given in section II followed by a number of decomposed computation approaches in section III and combined computation approaches in section IV. Section V presents our simulations, and discusses the results. Conclusion is given in section VI.

II. PROBLEM FORMULATION

In this paper we are concerned with the problem of routing connections that carry an aggregate of flows with both QoS and resilience requirements in a label switched network. In particular, we consider an end-to-end restoration scenario under a single link failure model where two link disjoint LSPs have to be set up from the ingress to the egress node for each admitted connection request. We will call the two LSPs service path P_s and backup path P_b . In our framework, connection requests are routed one by one as they arrive with no knowledge of future requests. When a connection request arrives it is only admitted into the network if the provider can find a service path and a backup path that both meet the QoS requirements, otherwise is the request rejected.

Motivated by the service provider's needs to provide bandwidth guaranteed connections that are both delay-sensitive and resilient, we will use bandwidth guarantee and bounded end-to-end delay as our QoS constraints. We will refer to such connections as restorable delay bounded connections.

Formally, we can specify a restorable delay bounded connection request as a 4-tuple $Req = (s, d, TSpec, RSpec)$, where the first field s is the ingress node, d is the egress node, $TSpec$ is the aggregate traffic specification defined by a vector (M, r, t, b) where M is the maximum packet size, r is the sustainable rate, t is the peak rate and b is the burst tolerance. $RSpec$ specifies the upper end-to-end delay bound d_m . Thus given a restorable delay bounded connection request and a network modeled by G , the problem is to find a service

path and a backup path from s to d such that both the TSpec and RSpec is satisfied on both paths.

To approach this problem we need a method that describes how the end-to-end delay relates to a given path and provisioned bandwidth. One such method is deterministic network calculus [10], which provides a deterministic worst case end-to-end delay guarantee. One drawback of the deterministic approach is that it provides an over-estimate of the bandwidth requirement because it does not take statistical multiplexing into account. There are some recent efforts in creating a stochastic network calculus counterpart [11], [12] but the results are still under development. We have therefore chosen to use the deterministic network calculus where mature understanding of the theories exist. Note that, whether the deterministic or stochastic framework is used, it does not affect the arguments in Section III. Also the optimization formulations in Section IV can be readily adapted if we are to use the stochastic framework though the formulations may not remain linear.

In the deterministic network calculus framework it has been proved that the general form of QoS routing of non-restorable connections is NP-complete [13]. Stiliadis et. al. showed that when the QoS constraints are bandwidth guarantee and end-to-end delay, the problem can be reduced to polynomial time through introducing latency rate (LR) servers [9]. A LR server is a network node that runs a packet scheduler like Weighted Fair Queuing, Virtual Clock or Frame-based Fair Queuing. They showed that, in a network consisting of LR servers, a tight upper bound of the end-to-end delay D_m on a path P from s to d is:

$$D_m = \frac{(t - R)}{(t - r)} \cdot \frac{b}{R} + \sum_{(i,j) \in P} \left(\frac{M}{R} + \frac{M_{ij}^m}{C_{ij}} + prop_{ij} \right) \quad (1)$$

Where C_{ij} and $prop_{ij}$ are respectively the capacity and the propagation delay of link (i, j) . M_{ij}^m is the maximum packet size of all LSPs that use link (i, j) . R is the minimum of the allocated bandwidths associated with the LSP in the traversed nodes. In other words $R = \min(R_1, R_2, \dots, R_{|P|})$ where $r \leq R_i \leq t$ is the allocated bandwidth for the LSP in node i and $|P|$ is the length of P . However, in this paper we are considering LSPs where all nodes reserve the same amount of bandwidth and hence $R_j = R_i \forall i, j \in P$.

In Eq. (1) the first term represents the shaper delay at the ingress node. The second term, the sum, corresponds to the queuing related delay induced by transmission buffers and propagation delays. No transmission buffers are considered in d since that is the egress node.

As evident in Eq. (1), the end-to-end delay bound depends not only on the chosen path but also on the provisioned bandwidth R . Evidently, increasing the provisioned bandwidth R results in shorter end-to-end delay and vice versa. In this paper we seek to exploit the dependency between the end-to-end delay, chosen path and bandwidth when satisfying a restorable delay bounded connection request.

Now, the problem we consider in this paper can be formulated as follows. Given a network graph $G = (V, E)$, where V

is a set of LR servers and E is a set of edges connecting the servers, and a request $Req = (s, d, TSpec, RSpec)$. Find two link disjoint LSPs P_s and P_b , from s to d , and their respective bandwidths R_s and R_b such that both $TSpec$ and $RSpec$ are satisfied.

When the restoration latency bound is very tight, a restoration routing scheme called 1+1 should be used. In this scheme, the traffic on a connection is forwarded along both the service and backup path simultaneously and the egress node then listens on the path with the strongest signal. This scheme does have the advantage of achieving very fast recovery from link failures but to the cost of doubling the traffic volume.

If the restoration latency bound is relaxed, it is possible to use a more bandwidth efficient restoration routing scheme called 1:1. In the 1:1 scheme traffic is switched to the backup path only after a failure has been detected on the service path. Since the backup path is idle as long as the service path is in operation the backup path can be shared between different service paths as long as the service paths are link disjoint.

We will address the problem of routing restorable delay bounded connections for both 1+1 and 1:1 restoration schemes, that is without and with backup path resource sharing. Independently of routing scheme, an algorithm that solves this problem need not only compute two LSPs but also their respective bandwidths. The problem can be approached in two ways. One way is to decompose the problem into sub-problems to which known solutions already exist. The other way is to search for the two LSPs and their respective bandwidths simultaneously.

III. DECOMPOSED COMPUTATION

In this section we approach the problem of routing restorable delay bounded connections through decomposition. The problem can be decomposed in a number of different ways.

One way to decompose the problem is to first convert the end-to-end delay bound into an effective bandwidth and then search for two disjoint bandwidth guaranteed LSPs simultaneously using a known algorithm such as the ones presented in [2] and [1]. This approach is suggested by Kodialam et. al. [2]. However, when converting the end-to-end delay bound into an effective bandwidth, we need to know the path that will be used. Since we do not have the path at hand we need to estimate or guess the path. If we want to be sure that the computed path will not breach the end-to-end delay bound we have to make a pessimistic estimate and use the longest (in end-to-end delay) possible loop free path from s to d . Hence, the resulting effective bandwidth will be higher than required in most cases since the computed paths are likely to be shorter than the longest path. This approach can be refined in a number of different ways nevertheless it will always be subject to an estimated path. Clearly, this decomposition does not exploit the end-to-end delay dependency on the chosen path and provisioned bandwidth.

Another way is to decompose the problem into a sub-problem of finding a single QoS constrained LSP P_s and its

associated bandwidth R_s , from s to d in $G = (V, E)$ using a known algorithm such as the one presented by Steenkiste et. al. [6] or by Pornavalai et. al. [7]. Once P_s is computed, the second LSP P_b and its associated bandwidth R_b , is computed using the same algorithm but on network graph $G' = (V, E \setminus E(P_s))$, where $E(P_s)$ is the set of edges on P_s . Though this approach is simple, it has been shown that joint computation of two bandwidth guaranteed LSPs leads to higher network performance than computing them in sequence [1]. Here, the end-to-end delay dependency on the chosen path and provisioned bandwidth is exploited when using the algorithm provided by Steenkiste et. al. but not when using Pornavalai's algorithm.

Yet another way to decompose the routing problem of restorable delay bounded connections is to first search for two disjoint paths again using a known algorithm [2], [1], with a bandwidth at least as high as the requested sustainable rate r . If two paths were found, then if the end-to-end delay bound can be verified on both paths the request is admitted into the network, otherwise it is rejected. This decomposition does not exploit the end-to-end delay dependency on chosen path and provisioned bandwidth either.

One way to possibly improve the performance of this decomposition, is to exploit the dependency between the end-to-end delay and allocated bandwidth. The baseline is that increasing the bandwidth allocation shortens the end-to-end delay. Thus, iterating over a path finding algorithm using an increasing amount of bandwidth can improve the chances to find two paths that meet the end-to-end delay. This approach is expressed in Algorithm (1). The algorithm takes a network

```

Input:  $G, Req$ 
Output:  $P_s, R_s, P_b, R_b$ 

 $bw \leftarrow r$ ;
while  $bw \leq t$  do
   $P_s, P_b \leftarrow \text{FindPaths}( G, s, d, bw )$ ;
  if paths not found then
    return false;
  end
  if  $\text{VerifyDelay}( G, P_s, P_b, bw )$  then
     $R_s, R_b \leftarrow \text{Trim}( G, P_s, P_b, Req )$ ;
    return true;
  end
   $bw \leftarrow bw + \frac{t-r}{q}$ ;
end
return false

```

Algorithm 1:

graph $G = (V, E)$ and a request $Req = (s, d, TSpec, RSpec)$ for a delay bounded restorable connection as input and returns a service path P_s and backup path P_b , with their respective bandwidths R_s and R_b . The algorithm first initiates the bandwidth bw to the sustainable bandwidth r as specified in the request's $TSpec$. Then while bw is less or equal to the peak

rate t it first searches for two disjoint paths through a call to *FindPaths*. *FindPaths* is typically implemented using existing algorithms. If it is run in a 1+1 protection routing context an algorithm like the one proposed by Suurballe can be used [1], which finds two bandwidth guaranteed link disjoint paths from s to d that combined have the lowest cost. In case of a 1:1 protection routing context, backup bandwidth sharing algorithms like the one proposed by Kodialam et. al. [2] is preferably used.

If *FindPaths* failed to find two paths then the algorithm returns false since if *FindPaths* failed to find two paths using bw it is certain to fail using some bandwidth greater than bw .

Given that *FindPaths* found two paths, the end-to-end delay bound is checked in *VerifyDelay*. *VerifyDelay* uses Eq. (1) to verify the delay and returns true if and only if $D_m \leq d_m$ on both P_s and P_b when $R = bw$. Now, if both paths passed the end-to-end delay bound test, their respective bandwidths R_s and R_b are calculated using *Trim*. Reason for trimming the bandwidths is that the bandwidth bw used to calculate the paths is an upper bound on the required bandwidth. *Trim* basically calculates a more precise bandwidth using Eq. (1) with $D_m = d_m$ and $P = P_s$ and $P = P_b$ respectively. After the bandwidths have been trimmed, the algorithm returns true indicating that the request can be admitted into the network.

If either of the two paths did not meet the end-to-end delay bound, then the bandwidth bw is adjusted to a higher value. The granularity of the bandwidth adjustment is controlled through the constant q . The higher value of q , the smaller adjustments to bw and consequently the more iterations. Thus q can be used as a trade-off mechanism between solution precision and running time.

If no paths have been found that satisfies the end-to-end delay bound even when the peak rate is used, the algorithm returns false indicating that the request is to be rejected.

This solution approach cannot guarantee that it will find a solution if it exists. The reason is that it tries to find two paths using the same bandwidth. Assume that there are exactly two paths P_1 and P_2 in $G = (V, E)$ from s to d and that $bw = k$ in the current iteration of Algorithm (1). Assume furthermore that the two paths' respective residual capacities are $H_{P_1} \leq k$ and $H_{P_2} > k + \Delta$ for some $\Delta > 0$, also let their respective delay bounds be $D_{P_1}^k \leq d_m$ and $D_{P_2}^k > d_m$. Now, in the algorithm, *FindPaths* will return P_1 and P_2 because both of them have enough residual capacity to host the request but *VerifyDelay* will return false since P_2 does not comply with the end-to-end delay bound. So the algorithm adjusts the bandwidth to $bw = k + \epsilon < t$ for some $\epsilon \in (0, \Delta]$ and calls *FindPaths* again. This time *FindPaths* will return false since P_1 does not have enough residual capacity and the request is subsequently rejected even if $D_{P_2}^{k+\epsilon} \leq d_m$ is true and a solution exists, namely $P_s = P_1$ and $P_b = P_2$ with their respective bandwidths $R_s = k$ and $R_b = k + \epsilon$.

In this section we approach the online routing problem of restorable delay bounded connections through combined computation of paths and bandwidths. Hence, when a request arrives, the algorithm searches for a service path P_s and a backup path P_b along with their respective bandwidths R_s and R_b simultaneously. We base these algorithms on linear programming and provide two LP formulations, one for the 1+1 and one for the 1:1 restoration routing scheme. The difference being that in the 1:1 restoration routing scheme backup path resources can be shared between different service paths as long as the service paths are link disjoint.

In the LP formulations presented below we allow the provisioned bandwidth to adjust up to the requested peak rate t , since we base our end-to-end delay calculations on Eq. (1), which has t as a theoretical upper bound. However, the requested peak rate t can be much greater than the requested sustainable rate r . Therefore a tighter upper rate bound can be chosen to possibly improve the overall network performance. To get a fair comparison when using the presented LP-formulations as a benchmark for approximation algorithms, the same upper rate bound should be chosen for both the approximation algorithm and for the LP formulation.

A. 1+1 Protection Routing

When using 1+1 protection routing, the bandwidth reserved on the backup path cannot be shared amongst different service paths so the problem here is: Given a network $G = (V, E)$ and a request $(s, d, TSpec, RSpec)$, find a service path P_s and an exclusive link disjoint backup path P_b from s to d and their respective bandwidths R_s and R_b such that both the TSpec and RSpec are met on both P_s and P_b . Furthermore, we want the total cost of the two paths to be minimized.

Let the cost of using link (i, j) on the service path be defined as $R_s F_{ij}^R$, where F_{ij}^R is a constant used to balance the network load. F_{ij}^R is chosen to reflect the amount of residual capacity on link (i, j) , typically the more residual capacity the lower the value of F_{ij}^R . Similarly, let the cost of using link (i, j) on the backup path P_b be defined as $R_b F_{ij}^R$.

Let x_{ij} be a decision variable that indicates if link (i, j) is used on P_s or not. Define x_{ij} to be 1 if $x_{ij} \in P_s$ and 0 otherwise. Likewise, we define y_{ij} to indicate if link (i, j) is used on P_b or not, with a value of 1 if $y_{ij} \in P_b$ and 0 otherwise. Given these decision variables, the cost of using link (i, j) on the service path can be formulated as $z_{ij} = R_s F_{ij}^R x_{ij}$ and on the backup path as $w_{ij} = R_b F_{ij}^R y_{ij}$. Note that z_{ij} is a non-linear function of decision variables R_s and x_{ij} and likewise w_{ij} is a non-linear function of R_b and y_{ij} . We will show later how z_{ij} and w_{ij} can be reformulated as linear expressions.

To constrain P_s and P_b and their respective bandwidths R_s and R_b so that the given delay bound d_m is met we use Eq. (1). The baseline is that $d_m \geq D_m$ and since we are considering linear programming we need to transform Eq. (1) into a linear expression. First we multiply both sides with R which gives

us:

$$R d_m \geq b \cdot \frac{(t-R)}{(t-r)} + \sum_{(i,j) \in P} \left(M + R \left(\frac{M_{ij}^m}{C_{ij}} + prop_{ij} \right) \right) \quad (2)$$

Furthermore, we need to convert the expression into a more general form where the sum is over all links in the network, not only the links on the path. Using the service path P_s as an example we get:

$$R_s d_m \geq b \cdot \frac{(t-R_s)}{(t-r)} + \sum_{(i,j) \in E} \left(M x_{ij} + \left(\frac{M_{ij}^m}{C_{ij}} + prop_{ij} \right) R_s x_{ij} \right) \quad (3)$$

We can transform the non-linear expression $R_s x_{ij}$ in Eq. (3) into a linear expression through the observation that $R_s x_{ij} = z_{ij} \frac{1}{F_{ij}^R}$. Furthermore, substituting the constant $\frac{1}{F_{ij}^R} \left(\frac{M_{ij}^m}{C_{ij}} + prop_{ij} \right)$ with K_{ij} gives us the following linear expression for the end-to-end delay bound on the service path:

$$R_s d_m \geq b \cdot \frac{(t-R_s)}{(t-r)} + \sum_{(i,j) \in E} (M x_{ij} + K_{ij} z_{ij}) \quad (4)$$

where z_{ij} , x_{ij} and R_s are decision variables. Eq. (4) can be used to compute the end-to-end delay bound on the backup path as well after substituting z_{ij} , x_{ij} and R_s with w_{ij} , y_{ij} and R_b respectively.

Now we are ready to present the LP formulation. The objective is to minimize the combined cost of the service path and backup path as expressed in Eq. (5).

$$\min \left(\sum_{(i,j) \in E} z_{ij} + \sum_{(i,j) \in E} w_{ij} \right) \quad (5)$$

under the following constraints:

$$z_{ij} = R_s F_{ij}^R x_{ij} = \begin{cases} R_s F_{ij}^R, & \text{if } x_{ij} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$$w_{ij} = R_b F_{ij}^R y_{ij} = \begin{cases} R_b F_{ij}^R, & \text{if } y_{ij} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$r \leq R_s \leq t \quad (8)$$

$$r \leq R_b \leq t \quad (9)$$

$$R_s \leq H_{ij}, \quad \text{if } x_{ij} \geq 1, \quad \forall (i,j) \in E \quad (10)$$

$$R_b \leq H_{ij}, \quad \text{if } y_{ij} \geq 1, \quad \forall (i,j) \in E \quad (11)$$

$$R_s d_m \geq b \frac{(t-R_s)}{(t-r)} + \sum_{(i,j) \in E} (M x_{ij} + K_{ij} z_{ij}) \quad (12)$$

$$R_b d_m \geq b \frac{(t-R_b)}{(t-r)} + \sum_{(i,j) \in E} (M y_{ij} + K_{ij} w_{ij}) \quad (13)$$

$$\sum_j x_{ij} - \sum_j x_{ji} = 0, \quad \forall i \neq s, d \quad (14)$$

$$\sum_j x_{sj} - \sum_j x_{js} = 1 \quad (15)$$

$$\sum_j x_{dj} - \sum_j x_{jd} = -1 \quad (16)$$

$$\sum_j y_{ij} - \sum_j y_{ji} = 0, \quad \forall i \neq s, d \quad (17)$$

$$\sum_j y_{sj} - \sum_j y_{js} = 1 \quad (18)$$

$$\sum_j y_{dj} - \sum_j y_{jd} = -1 \quad (19)$$

$$x_{ij}, y_{ij} \in \{0, 1\}, \quad \forall (i,j) \in E \quad (20)$$

$$x_{ij} + y_{ij} \leq 1, \quad \forall (i,j) \in E \quad (21)$$

Eq. (6) and (7) provide the cost for using link (i,j) on the service and backup path respectively. For readability we formulate them as logical expressions, it is straightforward to convert them into linear expressions, see Appendix A-1. In general we provide the logical formulation in the remainder of this paper for *if-then* and *if-then-else* expressions to improve readability.

Eq. (8) and (9) provide the limits for the provisioned service and backup bandwidths. Followed by the link capacity constraints given in Eq. (10) and (11), which say that the provisioned bandwidth on link (i,j) has to be smaller than the link's residual capacity, H_{ij} . These constraints cannot be formulated as $R_s x_{ij} \leq H_{ij}$ since that is a non-linear expression. The end-to-end delay constraints for the service and backup path are given in Eq. (12) and (13), which correspond to Eq. (4).

Eq. (14) - (19) are the flow preservation constraints for the service path and backup path. Eq. (20) assures that x and y only take on value 0 and 1. The last constraint, Eq. (21) states that the service and backup paths have to be disjoint.

B. 1:1 Protection Routing

When using 1:1 protection routing, the resources reserved on the backup path can be shared amongst different service paths as long as the service paths are link disjoint. Thus the problem in this scenario is: Given a network $G = (V, E)$ and a request $(s, d, TSpec, RSpec)$, find a service path P_s and a shareable link disjoint backup path P_b from s to d and their respective bandwidths R_s and R_b such that both the TSpec and RSpec are met. Furthermore, we want the total cost of the two paths to be minimized.

The cost z_{ij} of using link (i,j) on the service path is defined in the same way as in the LP formulation for the 1+1 protection routing scheme. Before we can define the cost of using link (i,j) on the backup path we need to look at the implications of backup path bandwidth sharing.

We are concerned with resilience under single link failures, hence the baseline for sharing is, that as long as two or more service paths do not have any links in common they can share the same backup path resources.

Example 1: Suppose that we have two link disjoint service paths P_{s1} and P_{s2} ($P_{s1} \cap P_{s2} = \emptyset$), that use R_{b1} and R_{b2} units of bandwidth respectively on their backup paths. Now if both P_{s1} and P_{s2} use link (u,v) on their backup paths, then it is sufficient to allocate $\max(R_{b1}, R_{b2})$ units of backup bandwidth on (u,v) since not both P_{s1} and P_{s2} will fail at the same time.

In the 1+1 protection scenario it was sufficient to keep track of the capacity on the links. However, when utilizing backup bandwidth sharing that is not sufficient. Here, we need to keep track of all currently admitted requests in the network as well, including their associated service and backup paths and respective provisioned bandwidths. Let A_{ij} represent all admitted requests that use link (i, j) on their service path and let B_{uv} represent all admitted requests that use link (u, v) on their backup path.

Now we define ϕ_{ij}^{uv} as the set of requests that use link (i, j) on their service path and link (u, v) on their backup path, as expressed in Eq. (22).

$$\phi_{ij}^{uv} = A_{ij} \cap B_{uv} \quad (22)$$

Furthermore, define δ_{ij}^{uv} as the amount of backup bandwidth on link (u, v) that depends on requests that use link (i, j) on their service path. δ_{ij}^{uv} is given by the sum of the backup bandwidths used by requests in ϕ_{ij}^{uv} .

$$\delta_{ij}^{uv} = \sum_{k \in \phi_{ij}^{uv}} R_b^k \quad (23)$$

where R_b^k is the provisioned backup bandwidth by request k . Note that δ_{ij}^{uv} does not express how much backup bandwidth that is reserved on link (u, v) , it only tells how much backup bandwidth on link (u, v) that depends on requests whose service paths use link (i, j) . However, the reserved backup bandwidth N_{uv} on link (u, v) is always the highest value of δ_{ij}^{uv} , that is $N_{uv} = \max_{(i,j) \in E} \delta_{ij}^{uv}$.

Example 2: Looking back on Example 1 we see that δ_{ij}^{uv} for link (u, v) is either 0, R_{b1} , or R_{b2} , assuming that no other admitted requests use (u, v) on their backup path. So N_{uv} is simply the largest of these three values. What if the two service paths in the example are only partly link disjoint? Say that $P_{s1} \cap P_{s2} = \{(e, f)\}$. Then for any link (u, v) that is shared by their backup paths, we have that $\delta_{ef}^{uv} = R_{b1} + R_{b2}$ so $N_{uv} = \max(0, R_{b1}, R_{b2}, R_{b1} + R_{b2}) = R_{b1} + R_{b2}$. Hence, no sharing will take place.

In general, the amount of bandwidth to reserve on backup link (u, v) when admitting a request using R_b units of bandwidth on the backup path depends on how much bandwidth that can be shared on link (u, v) . Shareable bandwidth is the difference between the already allocated backup bandwidth N_{uv} and the current maximum dependency of this bandwidth to links used by the current request's service path P_s . The shareable bandwidth can thus be expressed as $N_{uv} - \max_{(ij) \in P_s} \delta_{ij}^{uv}$. More specific, the shareable bandwidth on link (u, v) when used to backup link (i, j) is $N_{uv} - \delta_{ij}^{uv}$. If R_b exceeds the shareable bandwidth, we refer to the difference $R_b - (N_{uv} - \delta_{ij}^{uv})$ as the excess bandwidth.

Now we define Θ_{ij}^{uv} as the cost of using link (u, v) to backup link (i, j) as:

$$\Theta_{ij}^{uv} = \begin{cases} R_b F_{ij}^F, & \text{if } R_b \leq N_{uv} - \delta_{ij}^{uv} \\ (N_{uv} - \delta_{ij}^{uv}) F_{ij}^F + \\ (R_b - (N_{uv} - \delta_{ij}^{uv})) F_{ij}^R, & \text{otherwise} \end{cases} \quad (24)$$

Eq. (24) says that the cost to use link (u, v) to backup link (i, j) , when there is no excess bandwidth, is $R_b F_{ij}^F$. Here F_{ij}^F

is a constant used to load balance the use of *free* bandwidth in a similar way as F_{ij}^R is used to load balance explicit bandwidth reservations. Even though a link potentially can be used on a backup path without explicitly reserving more bandwidth, it does increase the bandwidth dependencies. Thus we treat it as a network resource that should be balanced. If there is excess bandwidth, the cost Θ_{ij}^{uv} is the sum of the cost for the *free* portion of R_b and the cost for the excess portion. Since the excess results in explicit bandwidth reservation we use F_{ij}^R .

Now we can calculate the cost q_{uv} of using link (u, v) on the backup path when the service path is P_s as $q_{uv} = \max_{(ij) \in P_s} \Theta_{ij}^{uv} = \max_{(ij) \in E} \sigma_{ij}^{uv}$, where σ_{ij}^{uv} equals Θ_{ij}^{uv} if and only if link (i, j) is used on the service path and link (u, v) is used on the backup path, otherwise it equals 0.

Definitions of x_{ij} and y_{ij} in the following LP formulation are the same as in the 1+1 protection routing scenario. Now the LP formulation can be stated as follows. The objective is to minimize the combined cost of the service path and backup path as expressed in Eq (25):

$$\min \left(\sum_{(i,j) \in E} z_{ij} + \sum_{(u,v) \in E} q_{uv} \right) \quad (25)$$

subject to the following constraints

$$z_{ij} = R_s F_{ij}^R x_{ij} = \begin{cases} R_s F_{ij}^R, & \text{if } x_{ij} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (26)$$

$$r \leq R_s \leq t \quad (27)$$

$$R_s \leq H_{ij}, \quad \text{if } x_{ij} \geq 1, \quad \forall (i, j) \in E \quad (28)$$

$$R_s d_m \geq b \frac{(t - R_s)}{(t - r)} + \sum_{(i,j) \in E} (M x_{ij} + K_{ij} z_{ij}) \quad (29)$$

$$q_{uv} \geq \sigma_{ij}^{uv}, \quad \forall (i, j) \in E \quad (30)$$

$$\sigma_{ij}^{uv} = \begin{cases} \Theta_{ij}^{uv}, & \text{if } x_{ij} + y_{uv} \geq 2 \\ 0, & \text{otherwise} \end{cases} \quad (31)$$

$$\Theta_{ij}^{uv} = \begin{cases} R_b F_{ij}^F, & \text{if } R_b \leq N_{uv} - \delta_{ij}^{uv} \\ (N_{uv} - \delta_{ij}^{uv}) F_{ij}^F + \\ (R_b - (N_{uv} - \delta_{ij}^{uv})) F_{ij}^R, & \text{otherwise} \end{cases} \quad (32)$$

$$r \leq R_b \leq t \quad (33)$$

$$R_b \leq H_{uv} + N_{uv} - \delta_{ij}^{uv}, \quad \text{if } x_{ij} + y_{uv} \geq 2 \quad (34)$$

$$R_b d_m \geq b \frac{(t - R_b)}{(t - r)} + \sum_{(i,j) \in E} (M y_{ij} + K_{ij} w_{ij}) \quad (35)$$

$$w_{ij} = R_b F_{ij}^R y_{ij} = \begin{cases} R_b F_{ij}^R, & \text{if } y_{ij} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (36)$$

$$\sum_j x_{ij} - \sum_j x_{ji} = 0, \quad \forall i \neq s, d \quad (37)$$

$$\sum_j x_{sj} - \sum_j x_{js} = 1 \quad (38)$$

$$\sum_j x_{dj} - \sum_j x_{jd} = -1 \quad (39)$$

$$\sum_j y_{ij} - \sum_j y_{ji} = 0, \quad \forall i \neq s, d \quad (40)$$

$$\sum_j y_{sj} - \sum_j y_{js} = 1 \quad (41)$$

$$\sum_j y_{dj} - \sum_j y_{jd} = -1 \quad (42)$$

$$\sum_j y_{ij} \leq 1, \quad \forall i \neq d \text{ and } \forall j \neq s \quad (43)$$

$$\sum_j y_{is} = 0 \quad (44)$$

$$\sum_j y_{dj} = 0 \quad (45)$$

$$x_{ij}, y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in E \quad (46)$$

$$x_{ij} + y_{ij} \leq 1, \quad \forall (i, j) \in E \quad (47)$$

The service path constraints are given in Eq. (26) - (29), which are identical to the ones in the 1+1 protection routing scenario. However, the backup path constraints are different since we consider backup path resource sharing in this scenario.

Eq. (30) and (31) together express the cost of using link (u, v) on the backup path. While Eq. (32) specifies the cost of using link (u, v) to backup link (i, j) as defined in Eq. (24).

The backup bandwidth bounds are given in Eq. (33) where the upper bound t is determined by the the delay calculation formula provided in Eq. (1).

The capacity constraints are given in Eq. (34), which basically says that the provisioned bandwidth on the backup path has to be smaller or equal to the sum of the residual capacity and the shareable bandwidth.

Eq. (35) and (36) bound the provisioned bandwidth on the backup path from below so that the delay constraint is met. These constraints are identical to the constraints in Eq. (13) and (7) for the 1+1 protection scenario.

The flow preservation constraints, Eq. (37) - (46), are the same as in the 1+1 protection scenario as well. If F_{ij}^F is 0 then the possibility to use a link for “free” on the backup path potentially creates loops. To prevent loops from forming in this case we need a few additional constraints. Eq. (43) ensures that at most one incoming link per node on the backup path is used, Eq. (44) prevents use of incoming links to the source node and Eq. (45) prevents use of outgoing links from the destination node on the backup path. Finally, Eq. (47) provides the constraints for disjointness, which again are the same as in the 1+1 protection scenario.

V. SIMULATION RESULTS

In this section we present simulations that aim to outline the performance difference between the dynamic and static bandwidth allocation approach to the routing problem of restorable delay bounded connections. Furthermore, the simulations aim to reveal how extensive the computation times are for the proposed LP formulations.

The two approaches are compared in both the 1+1 and 1:1 protection routing contexts. In the dynamic bandwidth allocation approach, we use the presented LP formulations

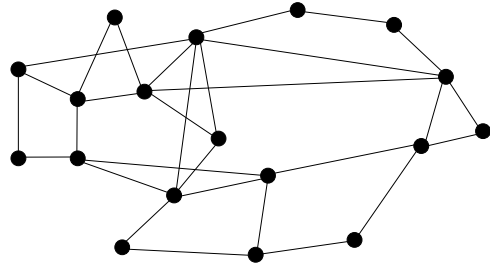


Fig. 1. Network representing a typical ISP network

in their respective routing context. In the static bandwidth allocation approach, we use Algorithm (1) in both routing contexts. To make it behave as a static bandwidth allocation algorithm, we prevent it from iterating by choosing q less than one. Furthermore, in the 1+1 protection routing context, we implement *FindPaths* using an algorithm proposed by Suurballe [1], which computes two link disjoint paths that combined have the least cost. In the 1:1 protection routing context we implement *FindPaths* using a LP formulation called Complete Information Scenario (CIS) proposed by Kodialam et. al. [2]. CIS uses backup bandwidth sharing and, like Suurballe’s algorithm, it computes two link disjoint paths that combined have the lowest cost. Neither of these two algorithms provide a load balancing mechanism, so for the sake of fairness, we “turn off” the load balancing in our proposed LP formulations through selecting $F_{ij}^R = 1$ and $F_{ij}^F = 0$. All code is written in C++ and CPLEX, an optimization package from ILOG.

The network we ran the simulations in is shown in Fig. 1. It represents a typical ISP network with 18 nodes and 60 unidirectional links. This network has been used for comparable simulations in [14], [15]. The capacity is set to 4 Mbps and the propagation delay to 1 millisecond on all links. Similar results as presented in this paper were obtained running the simulation on 2 other networks. In the simulation, restorable delay bounded requests arrive one by one according to a Poisson distribution with mean arrival rate λ . Admitted requests remain in the system for a time span drawn from an exponential distribution with mean μ . We chose to keep μ constant at one time unit and λ to 55 requests per time unit in the 1+1 protection routing context and to 85 requests per time unit in the 1:1 routing context, which gives a rejection ratio of about 5% when the delay bound is relaxed.

The simulation emulates restorable delay bounded connections used for voice applications. Assuming that the voice applications use a G.726 codec with a bit rate of 40 kbps, we get a maximum packet size M of 84 bytes. The other parameters in the TSpec, r , t and b are chosen to be respectively 400 kbps, 1000 kbps and $18M = 1512$ bytes, which gives a maximum buffer delay at the ingress node of about 30 milliseconds. The end-to-end delay bound d_m , as specified in the RSpec, is used as a variable ranging from 50 – 200 milliseconds.

In the static bandwidth allocation approach, it is not necessary to use the sustainable rate r as the bandwidth. In our

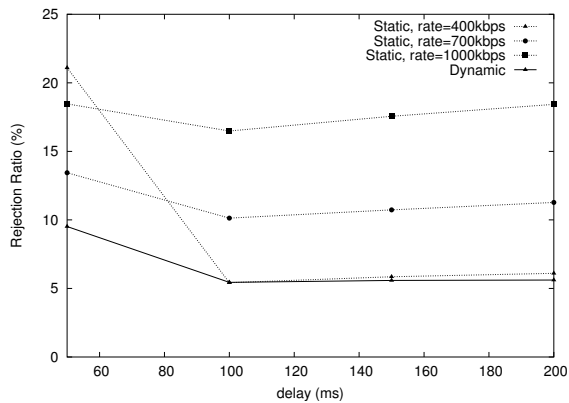


Fig. 2. Dynamic vs. static bandwidth allocation in a 1+1 protection routing context $\lambda = 55$

simulations we want to see how the network performance is affected by choosing different values of the provisioned bandwidth. We run the simulations thrice, choosing the provisioned bandwidth to be either 400, 700 or 1000 kbps through initializing bw to the chosen bandwidth in Algorithm (1).

Once the network has reached a steady-state, the measurements commence and run for 15 time units. We measure the rejection ratio, which is plotted as a function of the end-to-end delay bound d_m . A solid line represents the dynamic bandwidth allocation approach and a dotted line represents the static bandwidth allocation approach.

The average computation time per accepted request in the 1:1 protection scenario is up to 146 seconds and in the 1+1 scenario up to 0.5 seconds on a 3.2 GHz Intel CPU with 2 GB RAM memory. Due to the inhabited complexity of the restorable delay bounded routing problem, the computation time does in some cases exceed 40 hours for one request in the combined computation approach. Therefore, we limit the allowed computation time to 30 minutes per request. If this time limit is reached before a solution is found, the request is rejected. The only rejections due to the maximum time limit in our simulations occurred at 50 milliseconds in the 1:1 protection routing context for the dynamic bandwidth allocation approach, where 3 requests were rejected.

Looking at the results from the simulation in the 1+1 routing context shown in Fig. 2, we see that as expected the dynamic bandwidth allocation performs better than the static scheme. This is especially true when the end-to-end delay bound is tight, less than 100 milliseconds.

Note that when a static rate of 700 and 1000 kbps is used a better network performance is achieved than when using the sustainable rate 400 kbps for delay bounds tighter than 80 and 60 milliseconds respectively. However, when the delay bound is relaxed, static bandwidth allocation with 400 kbps outperforms the other two rates with a rejection ratio of about 6% versus 11% and 17%. The explanation is that, when using for instance a rate of 700 kbps, any link with less residual capacity than 700 kbps will not even be considered during the path computation no matter how relaxed the delay bound

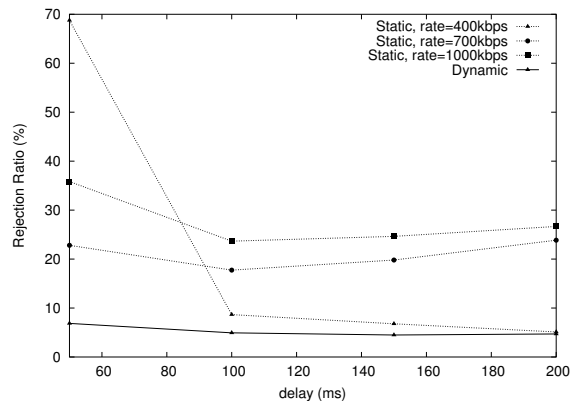


Fig. 3. Dynamic vs. static bandwidth allocation in a 1:1 protection routing context $\lambda = 85$

is. Furthermore, when the delay bound is tighter than 80 milliseconds, it apparently pays off to look for paths that can use up to 700 kbps since the higher rate makes it more likely that the paths comply with the delay bound. Remember that the bandwidth is trimmed, so that the bandwidth actually reserved in the network is in most cases less than 700 kbps per admitted request. Clearly there is a tradeoff between the chosen static bandwidth and the network performance in presence of an end-to-end delay bound.

Given a static rate of 400 kbps we see that the impact that the end-to-end delay bound has on the network performance of the two approaches is rapidly decreasing until around 100 milliseconds after which they perform similarly. Hence, using dynamic bandwidth allocation only makes sense when the delay bound has an impact on the path computation. What happens is that the problem is effectively transformed into a pure bandwidth guaranteed problem of restorable connections when the delay bound is relaxed. In other words, if the delay bound is relaxed enough, it will not be breached no matter how long the chosen path is.

Another interesting point in the graph is that for the static bandwidth allocation approach, the rejection ratio slightly increases from about 100 milliseconds onwards. One explanation for this is that around 100 milliseconds, requests that require long paths to be setup are rejected due to the delay bound and thus leaving more resources available for “shorter” requests. This effect subsides when the delay bound is relaxed resulting in the slight increase of the rejection ratio.

We see similar patterns when turning to the results in the 1:1 protection routing context shown in Fig. 3. It is worth to point out that even though the arrival rate is about 50% higher than in the 1+1 routing context the dynamic bandwidth allocation algorithm performs better than the dynamic bandwidth allocation algorithm in the 1+1 context as an effect of backup resource sharing.

One noticeable difference between the two routing contexts is that the improvement of using dynamic bandwidth allocation is higher in the 1:1 routing context. Reason being

that the backup path resource sharing encourage “long” backup paths since its resources can potentially be used for free or at a subsidized cost. The longer path, the more unlikely to comply with the delay bound and hence the more likely to get rejected. The same reason is behind the high rejection ratio for the static bandwidth allocation with rate 400 kbps when the delay bound is tight.

The response times of the LP formulations are long, up to 3 minutes in the 1+1 routing context and more than 40 hours in the 1:1 context. The average is up to 0.5 seconds in the 1+1 and up to 2.5 minutes in the 1:1 routing context. The poor response times make the LP formulations unsuitable for a production environment that typically requires a guaranteed response time in fractions of a second. However, they do result in very high network performance and thus they are suitable as a benchmark for approximation algorithms.

VI. CONCLUSION

In this paper we addressed the problem of routing restorable delay bounded connections. We proposed a new approach to solve this problem in which the two paths and their respective bandwidths are computed dynamically. The dynamic bandwidth allocation approach exploits the dependency between the end-to-end delay, chosen path and bandwidth allocation. To realize our approach, we presented three algorithms, one generic algorithm that decomposes the problem into problems where known solutions already exists. The other two algorithms are LP formulations that implement a combined approach where the paths and their respective bandwidths are computed simultaneously.

Our initial results show that the network performance improves significantly when using a dynamic bandwidth allocation approach as opposed to the current static bandwidth allocation approach. The tighter delay bound the better improvement. The improvement is more significant in the 1:1 protection routing context than in the 1+1 context. The presented LP formulations are not suitable for a production environment since they have too long response times, up to 2.5 minutes on average. They are however suitable for benchmarking approximation algorithms that in turn can be used in production.

Future work aim to deepen the understanding of the relation between network performance and the dynamic bandwidth allocation approach. We are also developing new approximation algorithms that address the problem of routing restorable delay bounded connections with dynamic bandwidth allocation. These approximation algorithms will use the presented LP formulations as a benchmark.

ACKNOWLEDGMENT

The support of the Co-operative Research Centre for Smart Internet Technology (<http://www.smartinternet.com.au>) for this work is hereby acknowledged.

REFERENCES

- [1] J. Suurballe and R. Tarjan, “A quick method for finding shortest pairs of disjoint paths,” *Networks*, vol. 14, pp. 325–336, 1984.
- [2] M. Kodialam and T. V. Lakshman, “Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information,” *Transaction on networking*, vol. 11, no. 3, pp. 399–410, June 2003.
- [3] S. Norden, M. M. Buddhikot, M. Waldvogel, and S. Suri, “Routing bandwidth guaranteed paths with restoration in label switched networks,” in *9th International Conference on Network Protocols (ICNP 2001)*, Nov. 2001, pp. 71–79.
- [4] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, “RSVP-TE: Extensions to RSVP for LSP tunnels,” December 2001, RFC 3209.
- [5] Q. Ma and P. Steenkiste, “Quality of service routing for traffic with performance guarantees,” in *Proceedings of IFIP International Workshop on Quality of Service*, May 1997.
- [6] Q. Ma and P. Steenkiste, “Routing traffic with quality-of-service guarantees in integrated services networks,” in *Proceedings of NOSSDAV*, July 1998.
- [7] C. Pornavalai, G. Chakraborty, and N. Shiratori, “Qos based routing algorithm in integrated services packet networks,” *Journal of High Speed Networks*, vol. 7, no. 2, pp. 99–112, 1998.
- [8] T. Korkmanz and M. Krunz, “Bandwidth-delay constrained path selection under inaccurate state information,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 384–398, 2003.
- [9] D. Stiliadis and A. Varma, “Latency-rate servers: A general model for analysis of traffic scheduling algorithms,” *Transactions on networking*, vol. 6, no. 5, pp. 611–624, 1998.
- [10] J. Y. Le Boudec, “Applications of network calculus to guaranteed service networks,” *IEEE/ACM Transactions on Information Theory*, vol. 44, no. 3, pp. 1087–1097, 1998.
- [11] C. Li, A. Burchard, and J. Liebeherr, *A network calculus with effective bandwidth*. Technical Report CS-2003-20, University of Virginia, Computer Science Department, November 2003.
- [12] F. Ciucu, A. Burchard, and J. Liebeherr, “A network service curve approach for the stochastic analysis of networks,” in *Proceedings of ACM Sigmetrics*, June 2005.
- [13] J. M. Jaffe, “Algorithms for finding paths with multiple constraints,” *Networks*, vol. 14, pp. 95–116, 1984.
- [14] K. Kar, M. Kodialam, and T. Lakshman, “Routing restorable bandwidth guaranteed connections using maximum 2-route flows,” in *Proceedings of INFOCOM*, June 2002.
- [15] G. Apostopoulos, R. Guerin, S. Kamat, A. Orda, and S. K. Tripathi, “Quality of service based routing: A performance perspective,” in *Proceedings of ACM Sigcomm*, September 1998, pp. 17–28.

APPENDIX

A-1. IF-THEN CONVERSION

A logical expression like $if\ f(x_1, x_2, \dots, x_n) > 0\ then\ g(y_1, y_2, \dots, y_m) \geq 0$ can be transformed into a linear expression through introducing a variable $t \in \{0, 1\}$ and a constant M chosen so that $M > f$ for all possible values of f and $M > g$ for all possible values of g . Now the logical $if - then$ expression can be transformed into:

$$-g \leq M t \quad (A-I)$$

$$f \leq M(1 - t) \quad (A-II)$$

Furthermore, an expression $if - then - else$ can be divided into two $if - then$ expressions. Consider the following, $if\ f > 0\ then\ g \geq 0\ else\ h \geq 0$ can be divided into $if\ f > 0\ then\ g \geq 0$ and $if\ f \leq 0\ then\ h \geq 0$. Now if f is an integer function then $f \leq 0$ can be written as $1 - f > 0$ after which the described transformation can be directly applied.