QoS Routing in Networks with Inaccurate Information: Theory and Algorithms

Roch A. Guérin and *Ariel Orda*

Abstract— **This paper investigates the problem of routing flows with Quality-of-Service (QoS) requirements through one or more networks, when the information available for making such routing decisions is inaccurate. Inaccuracy in the information used in computing QoS routes, e.g., network state such as link and node metrics, arises naturally in a number of different environments that are reviewed in the paper. Our goal is to determine the impact of such inaccuracy on the ability of the path selection process to successfully identify paths with adequate available resources. In particular, we focus on devising algorithms capable of selecting path(s) that are most likely to successfully accommodate the desired QoS, in the presence of uncertain network state information. For the purpose of our analysis, we assume that this uncertainty is expressed through probabilistic models, and we briefly discuss sample cases that can give rise to such models. We establish that the impact of uncertainty is minimal for flows with only bandwidth requirements, but that it makes path selection intractable when end-to-end delay requirements are considered. For this latter case, we provide efficient solutions for special cases of interest and develop useful heuristics.**

Keywords—**Routing, Networks, QoS, Bandwidth, Delay, Inaccuracy.**

I. Introduction

A. Motivations and Results Overview

QoS routing is one of the tools available to network operators to improve their ability to accommodate flows which expect certain QoS guarantees from the network. Specifically, QoS routing enables a network operator to identify, for every new flow, a path through the network, that has sufficient resources to meet the flow's requirements (see [1], [2] for examples). These requirements are typically in the form of bandwidth or end-to-end delay guarantees, so that identifying a path capable of meeting them implies some knowledge of the availability of resources throughout the network.

For purposes of clarity, we assume in the paper an environment where a source node is presented with a request to establish a new flow with specific QoS requirements, e.g., bandwidth or end-to-end delay, and is responsible for finding a suitable path to the destination. In other words, we consider a (loose) source routing model[1] as in [1], [4]. In addition, we follow the link state model of [1], [2], [4], where a network topology database is available that keeps state information about nodes and links in the network. This information is then used by the path selection process, to identify paths with sufficient resources to accommodate the requirements of new flows.

It should be noted that in addition to just finding a path with sufficient resources, there are other criteria that a network operator may want to consider. For example, it may want to optimize network utilization, carried load, number of flows successfully routed, etc. Each of these criteria can influence the outcome of the path selection process, e.g., the network may identify a path capable of satisfying the requirement of the flow, but decide not to use it because it is too expensive in terms of the amount of resources being consumed. This paper does not attempt to address all these issues. Instead, we focus on the aspect of *finding* a path capable of satisfying the requirements of a new flow. However, we believe that such criteria can often be either handled after a path has been selected or directly incorporated into the model we develop in the paper for path selection. We briefly touch on this aspect in Section V.

When considering the task of finding a path capable of accommodating a new request, it is important to realize that this ability is very much dependent on the accuracy of the information specifying the availability of network resources (see [5], [6] for examples of this sensitivity). Unfortunately, there are many reasons for why this information may be inaccurate, and we review several of them later in this section. Our main concern is, therefore, with the impact of inaccurate state information in the topology database used by path selection. As is to be expected, this impact depends on the kind of QoS requirements expressed by a new flow. In particular, it seems intuitive that so called *bottleneck* requirements such as bandwidth, i.e., a certain amount of bandwidth is required on each link, would react differently to inaccuracy than *additive* requirements such as end-to-end delay. As a result, we investigate separately the cases of flows with bandwidth requirements and of flows which request a bound on their end-to-end delay.

For both, our focus is on finding "good" paths, i.e., paths with enough resources to accommodate new requests, despite the possibility that the state information in the topology database may not accurately reflect the actual availability of resources. In other words, the criterion we use to assess the goodness of a path is the likelihood it will have sufficient resources to accommodate a new flow. In that context, we deem the best possible path to be the one *most likely* to have the necessary resources. In order to asses such path qualities, a source node needs to associate some stochastic behavior, namely probability distribution functions (p.d.f.'s) to a performance metric associated with the various network components. However, it is important to understand that this does not imply that p.d.f.'s are necessarily *advertised* by the network nodes, as part of the link-state protocol; rather, a source node may *construct* the p.d.f.'s, based on the (standard) advertised parameters and some knowledge of the characteristics and possible range of their inaccuracy.

For flows with bandwidth requirements, we show that under certain assumptions on how inaccuracy in network state information is represented, this best possible path can be computed using relatively standard algorithms. In the case of flows which request end-to-end delay bounds, the situation is unfortunately

A short version of this paper was presented at INFOCOM'97.

R. Guérin (guerin@ee.upenn.edu) is with the Dept. Elec. Eng., U. Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Part of this work was done while at the IBM T.J. Watson Research Center.

A. Orda (ariel@ee.technion.ac.il) is with the Dept. Elec. Eng., Technion - I.I.T., Haifa, 32000 - Israel. Part of this work was done while visiting the IBM T.J. Watson Research Center.

[1] As discussed in [2], a hop-by-hop model can also be considered, but requires additional care to avoid loops (see [3] for a discussion of this issue).

not as favorable, and we establish a number of intractability results when it comes to computing the path most likely to meet a given end-to-end delay bound. As a result, we then investigate several approximations, special cases of interest, and heuristics. We do so in the context of two different models for the provision of end-to-end delay bounds, and develop separate solutions for each.

First, we focus on a model that assumes the "rate-based" approach of [7] for providing end-to-end delay bounds to flows. This model essentially amounts to coupling delay and bandwidth guarantees, i.e., providing a bound on the queuing delay experienced by the packets of a flow is accomplished by ensuring a minimum service rate to the flow. Such a model imposes additional constraints on network nodes as it mandates the use of schedulers capable of enforcing relatively strict rate guarantees. However, it also means that rate (bandwidth) is the main network resource whose state needs to be considered when computing a path. As we shall see, this simplifies, to some extent, the path computation process, so that tractable solutions can be provided in a number of interesting cases.

Next, we extend our investigation to a second model, a "delay-based" model, where end-to-end delay bounds are now guaranteed by concatenating local delay guarantees provided at each node/link on the path of a flow. This model is consistent with the service and signaling specifications of [1], [8], [9]. The reliance on local delay guarantees for providing end-to-end delay bounds affects the nature of the network state information that needs to be provided, as it must now include the local delay guarantees that each node/link can provide. This in turn affects the path computation process, and as we will again see, makes it typically more difficult to find the best feasible path than with the rate-based model.

B. Sources of Inaccuracy in Network State Information

As mentioned earlier, QoS routing relies on state information specifying resource availability at network nodes and links, and uses it to find paths with enough free resources to accommodate new flows. In turn, the successful routing of new flows together with the termination of existing ones, induce constant changes in the amount of resources available. These must then be communicated back to QoS routing to ensure it makes its decision based on correct information. Unfortunately, communicating such changes in a timely fashion is expensive and, at times, not even feasible [6], [10]. As a result, changes in resources availability are usually communicated either infrequently, e.g., only when they are big enough, or imprecisely, e.g., after aggregating network states.

Such limitations can introduce substantial inaccuracy in the information used by path selection to identify good paths through the network, and, as stated before, this is the problem we want to address in the paper. However, before we proceed with our investigation, we briefly review some of the underlying parameters that determine the extent of inaccuracy we can expect in network state information. There are two main components to the cost of timely distribution of changes in network state: the number of entities generating such updates, and the frequency at which each entity generates updates.

Limiting the number of entities (nodes and links) generating updates on their state, is a generic scalability issue that is not specific to QoS routing. Indeed, as network sizes grow, scalability quickly becomes a generic concern that has been the source of the many hierarchical schemes in use by network protocols. Two good examples are provided by the OSPF [11] and PNNI [1] protocols. OSPF supports a two-level hierarchy, with different granularity in terms of topology and routing information for each level. In particular, detailed information about topology and routes is available *within* an area, while only much coarser information is available about the "backbone" and remote areas. The PNNI protocol generalizes this concept to an arbitrary number of levels, and defines a progressive aggregation procedure that combines multiple networks into single "nodes" as they get more remote.

These various aggregation steps abstract multiple physical nodes and links into a much smaller number of logical entities. As a result, information about the state of individual nodes and links is often lost. For example, the PNNI standard outlines several possible approaches for aggregating an entire network into a single node, where the metrics, e.g., available bandwidth, associated with the aggregate node are typically obtained by "averaging" corresponding individual metrics. This loss of accuracy in state information can have a substantial impact on the path selection process. For example, the knowledge that a remote network, or set of networks, has an amount B of available bandwidth, must be interpreted only as an indication that if a flow requests that amount, it is *likely* to be accepted. This is because the quantity B is now only a summary of the amount of bandwidth actually available on the many different paths across this remote network.

For path selection, the main consequence of this loss of accuracy in network state information, is that it now needs to consider not only the amount of resources that are available, but also the level of certainty with which these resources are indeed available. For example, a link which *guarantees* 10 Mbps of available bandwidth may be more desirable than one which *advertises* 20 Mbps based on some average computed over multiple different paths. Again, our goal is to identify the path most likely to accommodate the requirements expressed by a new flow.

The second contributor to the cost of maintaining accurate state information is the frequency of state changes, and therefore updates. Specifically, each advertisement of a state change consumes both network bandwidth on all the links over which it is sent, and processing cycles at all nodes where it is received. Keeping this overhead to a minimum is, therefore, desirable, if not mandatory. There are many different methods that can be used to achieve such a goal (see [5], [6] for in-depth investigations of this issue and its impact on QoS routing), but they typically involve waiting either for a large enough change or until a minimum amount of time has passed.

As a result, the actual state of a remote node or link can drift away from the value known to other nodes, without them being aware of it. The size of the gap between the actual state and its last advertised value clearly depends on the specifics of the mechanisms used to control distribution of state information.

For example, a mechanism that relies on thresholds, i.e., an update is triggered if the difference between the current value and the last advertised one is greater than, say 50%, implies some maximum value for this gap. This information can then be used during the path selection phase to assess the likelihood that a given amount of bandwidth is available. For example, [12] and [13] illustrate how this can be used together with the path selection method described in Section II of this paper. However, there are other cases where estimating the value of this gap is more difficult. In particular, as illustrated in [6], timer-based mechanisms can induce more arbitrary inaccuracies, which will require different approaches for selecting good paths.

C. Scope of the Paper and Relation to Previous Works

As was outlined above, there are many possible sources of inaccuracy in network state information, which all translate into different models and levels of inaccuracy. In addition, parameters such as network topology, traffic load, etc., all contribute to defining a broad problem space and, therefore, solution space. As a result, providing a comprehensive set of solutions for computing good paths in the presence of inaccuracy is a daunting task that goes well beyond the scope of a single paper. Devising solutions that explicitly address specific cases is certainly of importance, and some initial results have been reported in [5], [6], [12] for the case of the OSPF protocol. However, this is not the focus of this paper.

Instead, the goal of this paper is to explore fundamental aspects associated with the impact of information inaccuracy in the context of QoS routing. In particular, we concentrate on the formal specification of the problem, the derivation of basic complexity results, and the investigation and validation (or invalidation) of a representative sample of algorithmic approaches. Clearly, in the context of such an open-ended investigation, the focus is less on providing specific justifications for the assumptions underlying each scenario, and more on acquiring some basic understanding of the issues at stake. We feel that the latter is an important first step in exploring this complex problem. One that can then help better assess possible trade-offs between performance and complexity, and provide a useful guide for future investigations, both at the conceptual (e.g., [14], [15]) as well as the applied level (e.g., [5], [6], [12]). This is even more true given the lack of previous work on the specific topic of this paper.

The most relevant body of work to our problem of QoS routing in the presence of inaccurate information, is a set of papers aimed at exploring state aggregation issues and their impact on routing performance in large networks. [16] provided an initial investigation of the problem of hierarchical routing in large networks, and focused on the problem of cluster design to ensure maximum reduction of routing table sizes without substantial increases in path length (hop count). [17] focused on the problem of state aggregation in the context of link state routing protocols whose goal was to compute shortest paths. The problem of state aggregation in the context of QoS routing was described in [18], with specific aggregation methods being described and evaluated in [19], [20]. Finally, [21]–[25] developed a number of fundamental results for "good" aggregation techniques that minimize inaccuracy in network state information, while allowing substantial reductions in the amount of state data.

However, despite their relevance to our problem, none of these prior works truly addressed the problem of evaluating the fundamental impact of inaccuracy in state information, on the performance of QoS routing. This is the focus of this paper, and the rest of our investigation is structured as follows. In Section II, we investigate the problem of computing good paths for flows with bandwidth requirements. We show how a simple adaptation of a standard shortest path algorithm can be used to compute "optimal" (most likely to be feasible) paths. Sections III and IV address the case of flows that require end-to-end delay bounds. Section III deals with the previously mentioned rate-based model, while Section IV considers the case of the delay-based model. Because of the greater complexity of the delay-based model, we investigate a number of heuristics. In particular, we develop an approximation specifically targeted to the case of hierarchical network models, one which exploits the specific characteristics of the type of inaccuracies they introduce. The findings of the paper are briefly summarized in Section V, which also points to possible extensions. A key algorithmic technique used in the paper is formulated and validated in an Appendix. Several technical proofs and details are omitted and can be found in a technical report [26].

II. Flows with Bandwidth Requirements

In this section, we address the problem of computing paths for flows which have specific bandwidth requirements, when the information available at the source node of such a flow only consists of an inaccurate estimate of the actual amount of bandwidth available at other nodes/networks, represented by a graph $G(V, E)$. Specifically, the information known at the source node regarding the available bandwidth on each link $l \in E$, is in the form of quantities $p_l(x)$, where $p_l(x)$ is the probability that link l can accommodate a flow which requires x units of bandwidth.

In the case of a link state protocol, this distribution could be derived using the last received value for the advertised available bandwidth on the link, together with some information on the possible excursions around this value based on the triggering mechanism in use, e.g., as outlined in [12] for threshold based triggers. The basic premise is that the link state information cannot be taken at face value, and a probabilistic model is used to capture the fact that a range of values is instead possible. As mentioned earlier, our goal in this paper is not so much to assume and justify a specific distribution $p_l(x)$ and investigate its impact on the selection of paths for flows with bandwidth requirements. Instead, we wish to gain some general understanding of the added complexity imposed on the path selection by the need to deal with imprecise link bandwidth information. This understanding can then drive the exploration of specific cases, and the construction of distributions which are not only realistic models to represent state inaccuracy in real networks, but also enable tractable solutions to the QoS routing problem (see [13] for a detailed experimental investigation of this issue, and its use in the context of the methods proposed in this paper).

In this context, for a new flow with bandwidth requirement w, we wish to find the path that is most likely to be able to accom-

modate this new request. For that purpose, we let $p_l = p_l(w)$ be the probability of "success" on link l, i.e., the probability that w units of bandwidth are indeed available on link l. For a path $\mathbf{p}$, $\Pi_{l\in\mathbf{p}} p_l$ corresponds then to the probability of success of the path, i.e., the probability that at least the amount of requested bandwidth is available on *all* the links of the path.[2] The problem we are trying to solve can then be expressed as follows.

Problem B: For a given bandwidth requirement w, find a path $\mathbf{p}^*$ such that, for any path $\mathbf{p}$:

$$\Pi_{l\in\mathbf{p}^*} p_l(w) \geq \Pi_{l\in\mathbf{p}} p_l(w).$$

Next, we show how this problem has a rather straightforward solution, using a standard *Most Reliable Path (MRP)* algorithm. Such an algorithm consists of computing shortest paths for properly selected link weights, i.e., weights which are the negative logarithm of the original link weights. This way, the multiplication of problem **B** is transformed into the usual additive path length computation. In other words, we have:

Algorithm (MRP):

1. Let $w_l = -\log p_l$, for all $l \in E$.
2. Find the shortest path according to the metric $\{w_l\}$.

In [26] we formally establish that the *MRP* algorithm solves problem **B**, i.e., it yields a path which has the maximum probability of satisfying the requested bandwidth w. This essentially says that inaccuracies in the actual bandwidth available on links/nodes can be dealt with relatively simply, using a shortest path algorithm. As we shall see in the next sections, this property is not shared when dealing with end-to-end delay guarantees.

III. Flows with End-to-End Delay Requirements: Advertising of Rate Guarantees

In this section and in the next one, we consider the case of flows which require an end-to-end delay bound for their packets. The goal of path selection is then to identify a path that can both accommodate the traffic generated by the source, and guarantee that the end-to-end delays experienced by packets from the flow remain below a given value. This "guarantee" can be strict (hard delay bound) or loose (bound on the average delay). Such differences clearly affect resource allocation and scheduling support, but do not have a significant impact on the path selection problem that we consider as they typically translate into constraints of a similar form. As a result and for purposes of simplicity, we limit our attention to the case where delay guarantees are in terms of a hard upper bound.

This section considers the *rate-based* service model of [7] for providing end-to-end delay bounds. As mentioned earlier, this model requires the use of specific schedulers at all network nodes, e.g., a Weighted Fair Queuing scheduler [27] or a Rate Controlled Earliest Deadline First scheduler [28]. Under the assumption that such schedulers are in use, the end-to-end delay bound $d(\mathbf{p})$ that the network can guarantee on an n-hop path $\mathbf{p}$ is of the following form:

$$d(\mathbf{p}) = \frac{\sigma}{r} + \frac{\sum_{l\in\mathbf{p}} c_l}{r} + \sum_{l\in\mathbf{p}} d_l, \qquad (1)$$

[2]Note that we make here the assumption that the corresponding random variables on different links are independent of each other; such an assumption is made in all the models considered in this paper.

where σ is the size of the flow's burst, c_l is a fixed quantity at link l, typically the maximum packet length for the flow, d_l is a static delay value, typically the link propagation delay, and r is the minimal rate that can be guaranteed to the flow at each link along the path. In the rest of the paper, we assume that $c_l \equiv c$, e.g., it is the flow's maximum packet size, which is true for many scheduling policies (e.g., [27], [28]). Thus, for an n-hop path $\mathbf{p}$, we have

$$d(\mathbf{p}) = \frac{\alpha_n}{r} + \sum_{l\in\mathbf{p}} d_l$$

where $\alpha_n = \sigma + cn$.

As can be seen from the above expression, a major benefit of the rate-based model is that the dependency of the end-to-end delay bound on network resources is only in terms of available bandwidth on each link, i.e., the rate r. As a result, the link metric of most significance in this setting is as before the available bandwidth. In that context, we assume again a network represented by a graph $G(V, E)$, and denote $N = |V|, M = |E|$. The metrics (state) associated with each link l consist of a fixed propagation delay d_l, and a residual rate available to new flows. As propagation delays are not subject to (significant) fluctuations, the residual rate is the only quantity that is deemed variable and, therefore, subject to some inaccuracy.

As in the case of flows with bandwidth requirements, we capture this inaccuracy by assuming that the residual rate is a random variable with p.d.f. $p_l(r)$ corresponding to the probability of being able to allocate rate r on link l. As before, this distribution would likely be based on the last advertised value and some estimate of the possible variations around it. Again, our purpose is not to study or justify the use of a specific distribution, but to gain some understanding of how uncertainty in the amount of available bandwidth affects our ability to compute paths which satisfy end-to-end delay bounds.

Note that in cases where the source of uncertainty in state information is state aggregation, propagation delays are aggregated as well, and hence also carry some level of uncertainty. However, it is expected that the range of variations for these quantities will be much less than that of advertised residual rates. As a result, assuming that only rates are not known precisely remains a reasonable model even in that setting. Moreover, as can be seen from equation (1), the overall effect of imprecision in propagation delays can be expected to be smaller than for rates, since errors on different links can compensate for each other.

Before we return to assessing the impact of rate inaccuracy on our ability to select a good path capable of meeting a given end-to-end delay bound, we note that the relative weight of the two delay terms in equation (1) can greatly vary based on path length, traffic characteristics, and rate allocation. But even in the case of long haul connections, e.g., cross country, the impact of the rate-dependent term need not be negligible. For example, a video connection with a burst size of 32kbytes, a packet size of 1.5kbytes, an allocated rate of 2Mbits/sec, and taking 10 hops to cross the continental US, corresponds to a worst case queuing delay of 188msec. This is to be compared to a coast-to-coast propagation delay of about 20msec.

A. Intractability

Given the source and destination nodes of a new flow, a maximum delay requirement D for the new flow, and a path $\mathbf{p}$, we define $\pi_D(\mathbf{p})$ as the probability that $d(\mathbf{p}) \leq D$. We denote the problem of finding a path that maximizes the probability of satisfying the end-to-end delay requirement D of a flow in this setting as Problem **R-D**. Referring to equation (1), we note that, despite the fact that as with flows with bandwidth requirements the rate r is the only random variable, substantial differences exist between the two problems. Not only does the rate r appear in a denominator position, but the additive nature of the propagation delay term also affects the nature of the problem. As we shall see next, these differences drastically affect the impact of inaccuracy when it comes to computing paths capable of ensuring delay guarantees. Specifically, we establish that the presence of inaccuracies in residual rates, as we have just defined them, makes the path selection problem intractable. This is stated in the next proposition.

Proposition III.1: Problem **R-D** is NP-complete.

Proof: Through a reduction to a shortest weight-constrained path problem, which is known to be NP-complete [29]. Given a graph with two positive values, a_l and b_l, associated with each link l, and a positive bound B, the shortest weight-constrained path problem is to find a path that minimizes the sum of the a_l's, with the constraint that the sum of the b_l's does not exceed B. The transformation into problem **R-D** is done as follows. Set $d_l \leftarrow b_l$, and $\alpha_n \leftarrow \sigma + cn \geq 1$ for all n (the requirement that $\alpha_n \geq 1$ for all n is easily enforced by properly selecting σ and c). The rate at each link can be either infinite or $\frac{1}{B+1}$, with the following probabilities:

$$Prob\{r_l = \infty\} = e^{-a_l}$$
$$Prob\{r_l = \frac{1}{B+1}\} = 1 - e^{-a_l}.$$

The delay constraint is chosen as $D = B$. It is easy to verify that a path $\mathbf{p}$ that satisfies the constraint D must have $r = \infty$ and $\sum_{l \in \mathbf{p}} d_l \leq B$. Thus, a path that is a solution for problem **R-D**, i.e., that maximizes the probability of not exceeding the delay constraint, is also a path that minimizes $\sum_{l \in \mathbf{p}} a_l$ while obeying the bound $\sum_{l \in \mathbf{p}} b_l \leq B$, i.e., that solves the shortest weight-constrained path problem. Therefore, problem **R-D** is NP-hard. Since it can be transformed (polynomially) into a decision problem that is in NP, problem **R-D** is NP-complete. □

In the next section, we show that while the original problem is intractable, a pseudo-polynomial solution can be constructed.

B. Pseudo-Polynomial Solution

Assume that d_l takes integer values. A pseudo-polynomial solution for problem **R-D** can then be constructed as follows.

1. For each $1 \leq d \leq D$, for each $1 \leq n < N$:

(a) $r \leftarrow \frac{\alpha_n}{D-d}$.

(b) Among paths $\mathbf{p}$ with at most n hops and for which $\sum_{l \in \mathbf{p}} d_l = d$, find one that maximizes $\Pi_{l \in \mathbf{p}} p_l(r)$.

2. Among the $O(ND)$ [3] chosen paths, choose the one with maximal probability of success.

Using a standard dynamic programming approach, step (1.(b)) of the above algorithm can be performed in $O(DNM)$ steps. Thus, the overall complexity of the above algorithm is $O(D^2N^2M)$.

While the availability of a pseudo-polynomial solution is certainly a desirable feature, it is often not adequate in practice due to the typically large value of the term D. As a result, it is natural to ask if narrowing the problem by making additional assumptions on the distribution of the residual rate, might yield tractable (polynomial) solutions. As mentioned earlier, this is useful not only because certain distributions may be good candidates for capturing the impact of different sources of inaccuracy, but also because this sampling of the problem space can provide a better understanding of the source of the additional complexity introduced by inaccurate state information.

C. Special Cases

C.1 Deterministic Case

This first case is little more than a "sanity check" to verify that the absence of inaccuracy indeed yields a tractable solution. For that purpose, we assume that each link has a deterministic rate (r_l) associated with it. In that case, problem **R-D** is equivalent to the Quickest Path problem [30], for which a simple solution of $O(K(N \log N + M))$ is known, where K is the number of different values for r_l (thus $K \leq M$). The solution amounts to running a shortest path algorithm for each possible value of r. In [31] it is shown how complexity can be substantially reduced by resorting to ϵ-accurate solutions, and how the path selection can be refined in order to incorporate network optimization criteria.

C.2 Identical d_l's

This next case attempts to simplify the path selection problem by making the additive component of the end-to-end delay, i.e., the propagation delay, be essentially a function of the hop count. For that purpose, we assume that all propagation delays are identical, i.e., $d_l \equiv d$. From a practical point of view, this may be an appropriate assumption in a local environment, where propagation will be nearly insignificant so that a single value could then be used. In this case, the following algorithm provides a solution.

1. For each $1 \leq n \leq N$:

- Find a path of at most n hops that maximizes $p_l(r)$, where $r = \frac{\alpha_n}{D-nd}$. (This step is carried through an *MRP* algorithm, as described in Section II, which computes shortest paths through a *Bellman-Ford* shortest path scheme [32].)

2. Among the $O(N)$ selected paths, choose the one with maximal probability.

The complexity of this algorithm is simply $O(N^2M)$.

C.3 Identical P.d.f.'s

In this next case, we now explore the effect of a more restrictive model for the rate component of the end-to-end delay. Specifically, we assume that the same distribution function characterizes the residual rate on all links, i.e., $p_l(r) \equiv p(r)$. This

[3] Following standard terminology, we say that a function $f(n)$ is $O(g(n))$ (respectively, $\Omega(g(n))$) whenever there exists a constant c such that $|f(n)| \leq c \cdot |g(n)|$ (respectively, $|f(n)| \geq c \cdot |g(n)|$) for all values of $n \geq 0$. If $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$, we say that $f(n)$ is $\Theta(g(n))$.

may be an appropriate assumption for an homogeneous environment, or one for which little is known regarding the actual state of any link so that a common distribution is one possible representation of their state. It can then be seen that among paths of n-hops, an optimal one maximizes $p\left(\frac{\alpha_n}{D-\sum d_l}\right)$, which amounts to minimizing $\frac{\alpha_n}{D-\sum d_l}$, i.e., minimizing $\sum d_l$. Thus, a solution can be found through a single execution of a *Bellman-Ford* shortest path algorithm.

Remark: A tractable solution exists in this case even when $c_l \not\equiv c$. We then need to minimize $\frac{\sigma+\sum_{l\in\mathbf{p}} c_l}{D-\sum_{l\in\mathbf{p}} d_l}$, which can be done through algorithm *min-PR* (see the Appendix).

C.4 Exponential Distributions

This last special case is one which may be of more practical interest. It yields a tractable exact solution, while maintaining the main characteristics of the original problem, i.e., it preserves most of the generality of equation (1) and its choice for a distribution of available link bandwidth is one that corresponds to an intuitively reasonable model. Specifically, we consider the case where an exponential distribution is used to model the distribution of residual rate on all links, i.e., $p_l(r) = e^{-\mu_l r}$.

Such a model may be appropriate in the context of link state protocols that advertise a single value to characterize the available bandwidth on each link. This value could be chosen to represent the actual available bandwidth at a given instant in time, which would then fluctuate as new flows get established and older ones are released. Those fluctuations may not map exactly onto an exponential distribution, but it nevertheless represents a reasonable choice which captures the expected behavior of small bandwidth requests being likely to be accepted and larger ones becoming increasingly unlikely. Furthermore, nodes could also use this exponential model as a trigger for advertising updated values, when it is deemed that the previous exponential distribution is not adequate anymore.

Under the assumption of an exponential distribution, the probability of success over an n-hop path $\mathbf{p}$, is found to be simply given by:

$$\pi_D(\mathbf{p}) = \Pi_{l\in\mathbf{p}} e^{\frac{-\mu_l \alpha_n}{D-\sum_{j\in\mathbf{p}} d_j}} = e^{\frac{-\alpha_n \sum_{l\in\mathbf{p}} \mu_l}{D-\sum_{l\in\mathbf{p}} d_l}}$$

Thus, an n-hop path that maximizes $\pi_D(\mathbf{p})$ is one that minimizes $\frac{\sum_{l\in\mathbf{p}} \mu_l}{D-\sum_{l\in\mathbf{p}} d_l}$. Hence a solution to problem **R-D** can be obtained through algorithm *min-PR* (see the Appendix). This result is summarized in the following statement.[4]

Proposition III.2: When residual rates are exponentially distributed, an optimal solution to problem **R-D** can be found through algorithm *min-PR*, resulting in $O(N^2 M(\log(N \cdot D \cdot \frac{\max_{l\in E}\mu_l}{\min_{l\in E}\mu_l})))$ steps. □

Extensions of the above result are possible. For example, in [26] we show that Proposition III.2 holds also in a more general case of a shifted exponential distribution, i.e., each link can guarantee some minimal, non-zero value of residual rate.

[4] For simplicity of exposition, the result is stated under the assumption that d_l and D take integer values.

The above result not withstanding, it may, however, still be desirable to determine if tractable and near-optimal solutions can be constructed, that do not require specific assumptions on the distribution of the available rate on each link. Investigating this aspect is the topic of the next section.

D. An ϵ-Optimal Solution

In the previous section, we have identified a number of special cases for which simple solutions exist, and which have helped us gain some understanding into the impact of state inaccuracy on the complexity of path selection. In this section, we show that simple solutions can also be found, if one is willing to abandon the requirement of strict optimality. Specifically, we describe an approach that yields paths that are arbitrary close to the optimal, i.e., ϵ-optimal. This approach is based on quantizing the metrics used in selecting a path.

We consider a value $p_{min} > 0$, which is the minimal allowed success probability on any link (i.e., we make the assumption that a probability of less than, say, $p_{min} = 0.1$, results in a prohibitively high risk of failure for the resulting path). In other words, we assume that $p_l(r)$ is either higher than p_{min} or else equal to 0; without loss of generality, we shall further assume that $p_l(r)$ is always greater than p_{min}. Furthermore, we assume that the rates r_l on link l can only take K_l different values, where K_l is a fixed (and reasonably small) number and $K = \sum_{l\in E} K_l$. For example, this would be the case when using one of the class-based or quantized triggering policies of [6] to determine when to advertise new rate values.

An ϵ-optimal solution can then be constructed based on the following quantization of the residual rate p.d.f.'s on each link. Let $w_l(r) = -\log p_l(r)$. Each $w_l(r)$ is rounded up based on a quantization step η, yielding a new value denoted as $\tilde{w}_l(r)$. Letting $I = \lceil \frac{-\log p_{min}}{\eta} \rceil$, we note that $\tilde{w}_l(r) \in \{0, \eta, 2\eta, \ldots, I\eta\}$. The quantization factor η is chosen as $\eta = \frac{\log \frac{1}{1-\epsilon}}{N}$. Note that η is a function of the level of inaccuracy one is willing to tolerate. We point out that we describe here a general procedure that works irrespective of the number K of different values; further optimization of this procedure is possible by accounting for the impact of K.

The algorithm for selecting a path based on these quantized probabilities is then as follows.

Algorithm QP:

1. For all K possible values of r:
For all $1 \le n \le N$:
For all $0 \le m \le n$:
For all $0 \le i \le I$:
 (a) Find a path $\mathbf{p}(n, m, i)$ of at most n hops and for which $\sum_{l\in\mathbf{p}(n,m,i)} \tilde{w}_l(r) \le m \cdot i \cdot \eta$, that is shortest w.r.t. the metric $\{d_l\}$.
 (b) If $\frac{\alpha_n}{r} + \sum_{l\in\mathbf{p}(n,m,i)} d_l < D$ then compute $\Pi_{l\in\mathbf{p}(n,m,i)} p_l(r)$, the (real) probability of success of $\mathbf{p}(n, m, i)$ for the considered value of r; otherwise, the probability of success of $\mathbf{p}(n, m, i)$ for the considered value of r is 0.
2. Among the paths identified in the previous step, choose the one with the highest probability of success.

Theorem III.1: With the assumptions specified in this section, algorithm *QP* is an ϵ-optimal solution to problem **R-D**,

namely: the ratio between the probability of success of a path chosen by algorithm *QP* to that of an optimal path is bounded (from below) by $1 - O(\epsilon)$. Its complexity is $O(N^3 M \frac{1}{\epsilon})$, i.e., polynomial in the input size N, M, and the level of inaccuracy $\frac{1}{\epsilon}$. □

The proof can be found in [26].

IV. Flows with End-to-End Delay Requirements: Advertising of Delay Guarantees

This section deals with the alternate *delay-based* model outlined in Section I for providing end-to-end delay guarantees. In this model, end-to-end delay bounds are provided through the concatenation of local delay bounds at each node. As a result, the state information that a node now needs to advertise is not as in the previous section in the form of a residual rate or bandwidth, but instead in terms of its ability to provide specific delay guarantees. It is then this local delay information, which we will consider subject to some inaccuracy, and our problem can be stated as follows.

Given: A maximum delay requirement D for a new flow between given source and destination nodes, a network represented by a graph $G(V, E)$, $N = |V|$, $M = |E|$, p.d.f.'s $p_l(d)$ for all $l \in E$, such that $p_l(d)$ is the probability that (for the new flow) link l will introduce a delay of at most d units, i.e., that $d_l \leq d$. As with the rate model, an important issue is again the identification of appropriate distributions for the local link delays. However, as before, our focus is not so much on justifying a particular distribution and deriving a specific solution for it. In that respect, much of the discussions of Section III apply here as well, and we proceed along essentially the same lines in our investigation of the delay-based model.

Specifically, for a path $\mathbf{p}$, let $\pi_D(\mathbf{p})$ be the probability that $\sum_{l \in \mathbf{p}} d_l \leq D$, and define the path selection problem as:

Problem D: Find a path $\mathbf{p}^*$ such that, for any path $\mathbf{p}$: $\pi_D(\mathbf{p}^*) \geq \pi_D(\mathbf{p})$.

In other words, find the path $\mathbf{p}^*$ that is most likely to accommodate the new flow and provide it with an end-to-end delay guarantee less than or equal to D. Next, we establish that, as with the rate-based model, this problem is intractable.

A. Intractability of Problem D

Consider first the (simpler) problem of determining whether, for a given path $\mathbf{p}$ and value π, $\pi_D(\mathbf{p}) \geq \pi$. Call it Problem **P($\pi$)**.

Lemma IV.1: Problem **P(π)** is NP-hard.

Proof: Consider an instance of the Kth largest subset problem (see [29]). Given a finite set A, a size $s(a) \in Z^+$ for each $a \in A$, positive integers K and B, the problem is to determine whether there are K or more distinct subsets $A' \subseteq A$ for which the sum of the sizes of the elements in A' does not exceed B. The Kth largest subset problem is known to be NP-hard, and we will transform it into an instance of problem **P(π)** in the following manner. The elements of A are ordered arbitrarily to constitute a path $\mathbf{p}$, in which each $a \in A$ uniquely corresponds to a link in $\mathbf{p}$. The delay d_a of a link $a \in \mathbf{p}$ takes the values 0 and $s(a)$, each with probability of 0.5. The delay requirement for path $\mathbf{p}$ is chosen as $D = B$, and the probability bound is chosen as $\pi = \frac{K}{2^{|A|}}$. Then, it is easy to see that $\pi_D(\mathbf{p}) \geq \pi$ *iff* the answer to the Kth largest subset problem is affirmative. We thus conclude that problem **P(π)** is NP-hard. □

Corollary IV.1: Problem **D** is NP-hard.

Proof: Consider an instance of problem **P(π)**. Construct a network G, composed of only two parallel paths: the first is path $\mathbf{p}$ (of problem **P(π)**), the second is a single link, which takes 0 delay with probability $\pi - \epsilon$ (where π is the input probability of problem **P(π)** and ϵ is a sufficiently small positive number), and delay $D + 1$ with probability $1 - \pi + \epsilon$. The answer to problem **P(π)** is affirmative *iff* path $\mathbf{p}$ is chosen when solving problem **D**. □

Suppose, that in an attempt to override the intractability obstacle, we would be satisfied with a path that is "reasonably good", albeit not necessarily optimal, e.g., a path $\mathbf{p}$ for which $\pi_D(\mathbf{p}) \geq \pi_{min}$, where π_{min} is some (large enough) positive number. Unfortunately, even this relaxed version is intractable. Indeed, a solution for the relaxed version could be transformed into a solution for problem **D**, through a binary search on the values of π_{min} between 0 and 1, whose complexity is polynomial in the problem size (the details can be found in [26]).

Given the intractability of (even very simple instances of) problem **D**, we will resort to approximations and heuristics. However, we first turn our attention to some special cases to see if some simplifications are possible when considering a less general problem. The investigation of such special cases often provides insight into what might be good heuristics for the general problem.

B. Special Cases

B.1 Identical P.d.f.'s

The first special case we consider is that of identical p.d.f.'s, i.e., we assume that $p_l(d) \equiv p(d)$ for all $l \in E$. In this case, it is easy to see that any minimal hop path is an optimal solution to problem **D**.

Remark: Unfortunately, the above does not hold even when we "mildly" break symmetry. For example, in [26] we show that when link delays are uniformly distributed on $(\mu_l - \delta_l, \mu_l + \delta_l)$ for link $l \in E$, where $\delta_l \equiv \delta$, but μ_l may take more than one value, a minimal hop path may not be optimal.

B.2 Tight Constraints

In this special case, we attempt to simplify the computation of $\pi_D(\mathbf{p})$ by forcing *each and every* link of a path to provide a low delay, in order for the path $\mathbf{p}$ to have a non-zero probability of satisfying the delay constraint. For example, this could be representative of an environment where optimization of path selection is triggered only when resources become scarce, and accommodating new requests requires such tight optimization. Specifically, we assume here that link delays are uniformly distributed, and we will consider two cases. In the first, "proportional window" case, link delays are uniformly distributed on $(\mu_l \cdot (1 - \delta/2), \mu_l \cdot (1 + \delta/2))$, whereas in the second, "constant window" case, link delays are uniformly distributed on $(\mu_l - \frac{\delta}{2}, \mu_l + \frac{\delta}{2})$. In both cases we assume that the end-to-end delay bound is tight, so that, on any path, the delay on any link cannot be equal to its upper bound.

Consider first the case of proportional windows. The assumption of tight constraints transforms here into the following: for each path $\mathbf{p}$, there is some d, $0 < d < \delta \cdot \min_{l\in\mathbf{p}} \mu_l$, such that $D - \sum_{l\in\mathbf{p}} \mu_l \cdot (1 - \delta/2) = d$. In other words, the total delay margin d on path $\mathbf{p}$ is such that no link can afford to contribute its worst case delay. Under such a constraint, it is possible to obtain a simple explicit expression for the probability $\pi_D(\mathbf{p})$.

Lemma IV.2: In the case of tight constraints and proportional windows, the probability that the delay of an n-hop path $\mathbf{p}$ does not exceed D is given by

$$\pi_D(\mathbf{p}) = \frac{1}{n!}\left(\frac{d}{\delta}\right)^n \cdot \frac{1}{\Pi_{l\in\mathbf{p}}\mu_l}.$$

□

The above lemma allows us to then state the following corollary.

Corollary IV.2: In the case of tight constraints and proportional windows, among n-hop paths, the optimal one maximizes

$$\left(\frac{d}{\delta}\right)^n \cdot \frac{1}{\Pi_{l\in\mathbf{p}}\mu_l} = \frac{(D - \sum_{l\in\mathbf{p}} \mu_l(1 - \frac{\delta}{2}))^n}{\Pi_{l\in\mathbf{p}}\mu_l} \cdot \frac{1}{\delta^n} \quad (2)$$

□

Unfortunately, while the additional assumption of tight constraints helped simplify the computation of the probability that a given path meets the end-to-end delay requirement, solving the path optimization (implied by (2) for proportional windows) seems to be still intractable. However, if $\delta \cdot \min_{l\in E} \mu_l$ is a reasonably small number, a pseudo-polynomial algorithm of acceptable complexity can be formulated, as described in [26].

Consider now the case of constant windows, i.e., the delays d_l are uniformly distributed over $(\mu_l - \frac{\delta}{2}, \mu_l + \frac{\delta}{2})$. Following the same lines as above, it can be shown that the success probability for an n-hop path $\mathbf{p}$ is given by

$$\frac{1}{n!}\left(\frac{d}{\delta}\right)^n. \quad (3)$$

Therefore, the optimal path among those having at most n hops maximizes d, i.e., minimizes $\sum_{l\in\mathbf{p}} \mu_l$, and hence it is a shortest path w.r.t. the mean values μ_l. The global solution can then be identified by comparing the probabilities of the n-hop-constrained shortest paths for all n, $1 \le n \le N$. We summarize the above discussion in the following statement.

Proposition IV.1: In the case of tight constraints and constant windows, an optimal path can be found by identifying, for each value of n in the range $1 \ldots N$, an n-hop path that is shortest w.r.t. the mean values μ_l, and choosing among these N paths the one that maximizes (3). The complexity of the solution is $O(NM)$. □

While we have been able to develop tractable solutions for some cases of tight constraints, it should be noted that these cases are relatively limited, i.e., resources are scarce on *all* links. Therefore, even if the optimization of path selection is clearly beneficial in such cases, there are many other scenarios for which it is also important to be able to identify good paths. As a result, it remains desirable to provide a general and tractable solution capable of handling a wide variety of load conditions. This we do in the next section, where we attempt to develop some heuristics to tackle the general problem **D**. We focus on heuristics for which a tractable and efficient solution can eventually be constructed.

C. Split-Constraints Heuristics

In this section, we investigate a simplification of the general problem, which is based on the idea of splitting *a priori* the end-to-end constraint D among the links, i.e., transform a global constraint into local constraints. In the next few sections, we review several such heuristics and determine if and when they yield practical solutions. The first heuristics serve as building blocks for the ultimate one, which, as shall be shown, suits well the case where state inaccuracy is primarily caused by state aggregation.

C.1 Split-Constraints Heuristic - Version 1 (S1)

In the first version S1 of the split-constraint heuristic, the delay split is performed so that the probability of meeting the (local) constraint is equal along all links in the path. The path selection problem is then to find the local constraint value that yields the best path. In other words, given a path $\mathbf{p}$ and some value $0 < p \le 1$, we split D into D_l's, $l \in \mathbf{p}$, such that, for each link $l \in \mathbf{p}$:

$$p_l(D_l) = p, \text{ or } p_l(D_l) = 1 \quad (4)$$

(the second possibility is introduced in order to deal with links for which only a deterministic value is available), and[5]:

$$\sum_{l\in\mathbf{p}} D_l = D. \quad (5)$$

We further assume that the delays d_l on link l are uniformly distributed on $(\gamma_l, \gamma_l + \delta_l)$, and that, for simplicity, D, γ_l, and δ_l take integer values.

It is then easy to see that:
1. There is a solution *iff* $\sum_{l\in\mathbf{p}} \gamma_l < D$ (thus: $D - \sum_{l\in\mathbf{p}} \gamma_l \ge 1$).
2. If $D > \sum_{l\in\mathbf{p}}(\gamma_l + \delta_l)$, then the probability of success in $\mathbf{p}$ is equal to 1.

Thus, we first check if the shortest distance in the network G, w.r.t. $\{\gamma_l\}$, is no more than D (otherwise, there is no solution), and if the shortest distance w.r.t. $\{\gamma_l + \delta_l\}$ is more than D (otherwise, a shortest path $\mathbf{p}$ w.r.t. $\{\gamma_l + \delta_l\}$ is optimal, with $\pi_D(\mathbf{p}) = 1$). Accordingly, we assume from now on that there is a path $\mathbf{p}'$, such that:

$$\sum_{l\in\mathbf{p}'} \gamma_l < D \quad (6)$$

and that, for all paths $\mathbf{p}$:

$$\sum_{l\in\mathbf{p}} (\gamma_l + \delta_l) > D. \quad (7)$$

It can then be verified that expressions (4)–(7) have the following unique solution:

$$D_l = (D - \sum_{j\in\mathbf{p}} \gamma_j)\frac{\delta_l}{\sum_{j\in\mathbf{p}} \delta_j} + \gamma_l, \quad (8)$$

for which:

$$p_l(D_l) = \frac{D - \sum_{j\in\mathbf{p}} \gamma_j}{\sum_{j\in\mathbf{p}} \delta_j}. \quad (9)$$

Thus, for a path $\mathbf{p}$ with n hops:

[5] In equation (5) we ignore the possibility that $\sum_{l\in\mathbf{p}} D_l < D$ with probability 1; as shall become clear subsequently, this is done without loss of generality.

$$\pi_D(\mathbf{p}) \geq \left(\frac{D - \sum_{j\in\mathbf{p}} \gamma_j}{\sum_{j\in\mathbf{p}} \delta_j} \right)^n. \quad (10)$$

Therefore, heuristic S1 selects the path in the following way.

Heuristic S1:

1. If the shortest distance in the network G, w.r.t. $\{\gamma_l\}$, is more than D, then stop (there is no solution).
2. If the shortest distance, w.r.t. $\{\gamma_l + \delta_l\}$, is less than D, then stop (a shortest path $\mathbf{p}$ w.r.t. $\{\gamma_l + \delta_l\}$ is optimal, with $\pi_D(\mathbf{p}) = 1$).
3. For $1 \leq n \leq N$, run algorithm *min-CTW(n)* (see the Appendix), in order to find an n-hop walk $\mathbf{p}(n)$[6] that minimizes the following expression:

$$\frac{\sum_{j\in\mathbf{p}} \delta_j}{D - \sum_{j\in\mathbf{p}} \gamma_j}. \quad (11)$$

4. Choose the path $\tilde{\mathbf{p}}$ that maximizes (10) among all $\mathbf{p}(n)$'s.[7]

In view of the complexity discussion presented in the Appendix, the complexity of heuristic S1 is:

$$O(N^2 M \cdot (\log(N \cdot D \cdot \max_{l\in E} \delta_l))).$$

The solution $\tilde{\mathbf{p}}$ generated by heuristics S1 has a number of "nice" properties which we briefly review.

1. It is feasible, i.e., for all $l \in \tilde{\mathbf{p}}$, $\gamma_l \leq D_l \leq \gamma_l + \delta_l$.
2. For $\delta_l = 0$, we have $D_l = \gamma_l = d_l$, i.e., a deterministic link is assigned its deterministic delay.
3. In a network with identical p.d.f.'s we have, for an n-hop path: $D_l(n) = \frac{D}{n}$, i.e., a symmetric assignment. Therefore, $\tilde{\mathbf{p}}$ is a minimum hop path, i.e., the (real) optimal path.
4. More generally: expression (10) grows with D and decreases with $\sum \gamma_l$, $\sum \delta_l$ and n. This means that we prefer paths with fewer hops (n), higher slack ($D - \sum \gamma_l$), and smaller uncertainty ($\sum \delta_l$), which seems like an intuitive thing to do.

Despite the above benefits, heuristic S1 also has shortcomings, in particular when links are heterogeneous. Indeed, in that case, it may be a bad idea to impose the same success probability on all links. For example, consider a path composed of two links, with delays uniformly distributed on $(0.5, 1.5)$ and $(50, 150)$, respectively, and assume a delay constraint of $D = 51$. The probability of success on the second link can only be small, and heuristic S1 will thus also force a small probability of success on the first link. This is inefficient as there are other partitions of D into D_1 and D_2 that result in much higher success probability for the path. However, there is a special case for which this drawback is not present, and we discuss it next.

Assume that $\delta_i \equiv \delta$. Then, the probability of success over an n hop path $\mathbf{p}$ is

$$\left(\frac{D - \sum_{j\in\mathbf{p}} \gamma_j}{n\delta} \right)^n, \quad (12)$$

and the partition is

$$D_l = \frac{D - \sum_{j\in\mathbf{p}} \gamma_j}{n} + \gamma_l \quad (13)$$

which can be shown to also be the best possible *a priori* partition, that maximizes the success probability of path $\mathbf{p}$ (this assertion is formally established in [14], under a more general setting). In that case, the optimal path can be found simply by running N shortest-path algorithms. Thus, for the case of identical δ_l's, we derive the following heuristic, denoted as SI for Split constraint heuristics with Identical δ_l's.

Heuristic SI:

1. If the shortest distance in the network G, w.r.t. $\{\gamma_l\}$, is more than D, then stop (there is no solution).
2. If the shortest distance, w.r.t. $\{\gamma_l + \delta\}$, is less than D, then stop (a shortest path $\mathbf{p}$ w.r.t. $\{\gamma_l + \delta\}$ is optimal, with $\pi_D(\mathbf{p}) = 1$).
3. For all n, $1 \leq n < N$, find an n-hop path that is shortest w.r.t. $\{\gamma_j\}$.
4. Choose the best, among the $O(N)$ selected paths, either according to expression (12), or by exact computation of the path probability of success (if such a computation, involving an N-stage convolution, is reasonably tractable).

Heuristic SI has the following desirable properties.

1. The desirable properties of S1.
2. Optimality of the split of D into D_l's, along any chosen path.
3. In the case of tight constraints, and when the choice among the $O(N)$ selected paths is done according to an exact computation, the chosen path is optimal.
4. Noting that step (3) can be done through a *Bellman-Ford* algorithm, we conclude that the complexity is simply $O(N \cdot M)$.

In [26] we present a heuristic that attempts to generalize the above result, i.e., obtain the above optimality property (2) also when the δ_l's are not identical. Unfortunately, while we are able to develop a simple procedure to pick the best delay split for any given path, this does not extend to a simple (tractable) path selection procedure.[8]

In the next section, we design one last split-constraints heuristic that not only leverages the above findings, but also incorporates information on the overall structure of the network. Specifically, we assume hierarchical network structure as in the PNNI protocol [1], where each layer in the hierarchy represents an additional level of aggregation of network/link state information.

C.2 Applying SI in a Hierarchical network model (SIH)

As mentioned earlier, it is often the case that networks are organized according to some hierarchy, which determines how network state information is to be aggregated. Knowledge of this underlying hierarchy can provide useful guidelines when devising heuristics to accommodate the loss of information that aggregation entails. However, before investigating such a possibility, we proceed to first describe more precisely the characteristics of a hierarchical network model.

As before, the network or set of networks across which flows need to be routed is represented by a graph $G(V, E)$, where V is now the set of *layer 1* nodes, and E is the set of physical links interconnecting them. $G(V, E)$ will be referred as *the actual network*. Layer 1 nodes are clustered to form layer 2 nodes, which are in turn clustered into layer 3 nodes etc., up to the last, say L-th, layer. Nodes of the same layer i, which are clustered

[6] As explained in the Appendix, $\mathbf{p}(n)$ may contain loops, thus it is a "walk" and not necessarily a (simple, i.e. loopless) "path".

[7] As explained in the Appendix, although $\mathbf{p}(n)$ may contain loops for some values of n, $\tilde{\mathbf{p}}$ is a (simple) path.

[8] However, in [14] it is shown that, by imposing a convexity assumption related to the p.d.f.'s, an ϵ-optimal path selection procedure of polynomial complexity can be constructed.

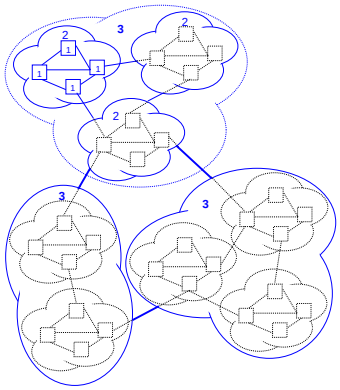


Fig. 1. Example of Hierarchical Network

into the same $(i+1)$-layer node, are said to belong to the same *peer group* [1]. Layer-i nodes that form a layer-$(i+1)$ node are called its *children*, and the upper layer node is called their *parent*. An *ancestor* of a node is either its parent or an ancestor's parent. We assume that each peer group is composed of at most m nodes of the lower layer, where m is a ***fixed*** (reasonably small) number. Since $|V| = \Theta(m^L)$, we conclude that the number of layers L is $O(\log|V|)$. For each layer 1 node i, the above hierarchical procedure derives a corresponding aggregated representation of the network, which will be referred as *the image* (of the network $G(V,E)$) *at node* i.

The above terminology is illustrated in Figure 1. The "squares" correspond to layer 1 nodes, whereas the "clouds" represent layer 2 and layer 3 nodes. The nodes and links drawn with full lines correspond to the network image at any of the layer 1 nodes located at the upper-left part of the figure. Accordingly, the dashed nodes and links correspond to the components that are invisible at those nodes. In this example we have $m = 4$, $L = 3$ and $|V| = 32$.

Associated with each layer-i node, say $v(i)$, are a set of numbers, that aggregate the state and metrics information of the layer-1 actual network components of which $v(i)$ is the corresponding ancestor. The way by which this metric aggregation is performed is out of the scope of this paper (see, e.g., [18] and references therein for typical examples). Nevertheless, the following general observations can be made, independently of the precise method of metric aggregation:

H1 The amount of aggregated information, and thus its imprecision, grows with the layer index. Usually, at each layer the aggregation process encapsulates $\Theta(m^2)$ information into an $O(m)$ (or even an $O(1)$) database.

H2 Some aggregated values grow with the layer index. For example, the delay associated with a layer i node aggregates the delay over entire paths at the lower, $i-1$'st, layer.

H3 A link interconnecting two nodes of a layer i, say $u(i)$ and $v(i)$, represents some set S of actual links that interconnect the corresponding actual nodes, aggregated by $u(i)$ and $v(i)$. The metric associated with link $(u(i),v(i))$ combines the metric associated with each of the two end-nodes, together with an aggregation of the metric of the links in the set S. For example, the delay associated with $(u(i),v(i))$ combines the aggregated delay associated with $u(i)$, together with the aggregated delay of the links in S. It should be noted that, in general, aggregation is a complex process, that introduces dependencies between link metrics not only across layers but also within the same (higher) layer. However, accounting for such dependencies during the path selection process is a difficult, if not impossible task. Furthermore, assuming that aggregation is successful, as it should, at properly ordering link metrics in each level, its impact on the hierarchical routing heuristic we describe next should be relatively small.

Requests are placed to path selection processes at source (layer 1) nodes. The information available to a path selection process at a node i corresponds to the image of the network at that node. Considering a specific request at some node i, we denote by M the number of links in the image of the network at i; similarly, we denote by N the maximal number of nodes on a (simple, loopless) path between source and destination on the image of the network at that node. Since a path for a flow is composed of up to m nodes at each of the L layers, we conclude that $N = O(m \cdot \log|V|) = O(\log|V|)$, i.e., on the network's image the maximal size of a path grows logarithmically with the size of the actual network. Similarly, $M = O(m^2 \cdot L) = O(\log|V|)$.

As an example, consider again Figure 1. Suppose that a request is placed at one of the (layer 1) nodes drawn with full lines at the upper left part of the figure. It is easy to verify that any path to the destination, on the image of the network at the source node, is composed of at most four nodes of layer 1, two nodes of layer 2 and two nodes of layer 3, i.e., $N = 8$. The network image consists of five links between layer 1 nodes, three links between layer 2 nodes and three links between layer 3 nodes, i.e., $M = 11$.

The aggregation process induced by such a hierarchical model can be used to justify a number of assumptions regarding the advertised network state, which can help formulate heuristics to deal with our path selection problem. Specifically, we can assume that the parameters that characterize network "links" at each layer have the following properties, where for ease of exposition, we assume that link delays d_l are uniformly distributed in$(\gamma_l, \gamma_l + \delta_l)$:

- At each layer i, all δ_l's are identical, and denoted by $\delta(i)$.

Justification: the actual value is likely to be heavily based on the level of aggregation implied by l, i.e., on l's layer.

- For a link l in layer i and for a path $\mathbf{p}$ wholly in layer $i-1$, $\gamma_l = \Theta(\sum_{j\in\mathbf{p}} \gamma_j)$.

Justification: a single link at layer i amounts to crossing a whole peer group of layer $i-1$. (See observation H2 above.)

- The δ_l of layer i is $\Omega(m)$ larger than that of layer $i-1$.

Justification: the aggregation process encapsulates $\Theta(m^2)$ information into an $O(m)$ database. (See observation H1 above.)

In view of the above observations, we formulate heuristic SIH on the basis that links in a higher layer have greater weight than those of lower layers. Hence, it is preferable to start with them when attempting to select good paths. This heuristic is embodied in the following algorithm which, due to space constraints, is only described informally here (the reader is referred to [26] for a formal description).

Since a given layer only plays a minor role in the performance of the global path as compared to the next higher layer, the path will be constructed top-down, i.e., we first choose the path suffix

which corresponds to the last layer, and then proceed sequentially to the lower layers. However, we do let lower layers affect to some extent the choice of higher layer paths. In order to allow such feedback, at each layer i we do not limit ourselves to a single path, but rather choose several, say K, layer-i paths. Then, when moving to the $i-1$-st layer, we will consider each of the K layer-i paths and identify a corresponding layer-$(i-1)$-path. Then, by comparing the K possibilities, obtained by concatenating each layer-i path with its corresponding layer-$(i-1)$ path, we identify the best solution for the i-th layer, among the K layer-i paths. We then continue the recursion with layer $i-1$.

It remains to determine the rule according to which paths are evaluated and chosen, which we do next. At each layer i we consider a delay bound $D(i)$, which corresponds to what is "left" from the original value D after having consumed part of it at the upper layers (the precise computation of $D(i)$ appears in [26]). Thus, at layer i, paths will be selected so as to maximize the probability of success assuming an optimal partition of $D(i)$ into link constraints D_l along the path. Suppose first that $K=1$, i.e., at each layer we only identify the "best" path. Since at each layer all δ_l's are equal, it follows from the discussion on heuristic SI (Section IV-C.1) that the path selection should be done in the following way. We identify the shortest n hop path with respect to $\{\gamma_l\}$, for each possible number of hops n; the best path is chosen among these shortest paths so as to maximize the approximated probability of success given by (12). Accordingly, for a value $K>1$, we find the K shortest paths for each hop number n, and then choose the best K (according to (12)) among these paths.

The full details can be found in [26]. We mention here that, for $K=1$, the complexity of heuristic SIH is $O(m\cdot M)$, which compares well with $O(N\log N+M)$, i.e., the complexity of a regular shortest-path algorithm run on the topology visible to the source; for $K>1$, the complexity is $O(K\cdot m^3\cdot M)$, which is still reasonable, as the hierarchical representation should be designed with a small value of m.

In [26] the potential efficiency of the heuristic is discussed. Since the discussion is based on the detailed specification of the algorithm, we will confine ourselves here to a few important observations. For $K=1$, SIH has all the desirable properties of SI. Higher values of K can be expected to improve the quality of the solution, at the expense of increasing the complexity. However, in [26] it is demonstrated that, typically, the lower layers indeed have only a minor effect on the path, thus a small value of K, even $K=1$, should usually suffice. In [26] we also discuss other, more complex, variations of the algorithm.

V. Conclusion

In this paper, we have investigated the impact on the path selection process for flows which require QoS guarantees, of inaccuracies in the available network state and metrics information. These inaccuracies can have many sources such as the need to limit the rate at which state updates are distributed, the aggregation of state information to ensure scalability, or in general the use of generic state representations that do not account for the specific details of a node/network internal structure. Our focus was on gaining some basic understanding into the impact of such inaccuracy on our ability to select good paths, i.e., paths likely to have sufficient resources, for flows with bandwidth requirements and end-to-end delay guarantees.

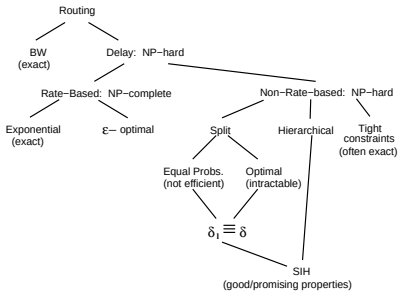


Fig. 2. Summary of Results

We showed that in the case of flows with only bandwidth requirements, the impact of inaccuracies is relatively minimal, in that a good path could be identified based on an algorithm which was essentially a shortest path algorithm. Unfortunately, the same could not be said for flows with end-to-end delay guarantees, for which we found that inaccuracies had a major impact on the complexity of the path selection process, that typically became intractable.

We then proceeded to study this latter case further for two different models, namely a rate-based model and a delay-based model. We showed that for the case of a rate-based model, the problem is intractable. However, we identified a number of special cases of practical interest, for which tractable solutions exist. We note that the derivation of some of these solutions (presented in the Appendix) is in itself a contribution of this paper. In addition, we showed that by introducing metrics quantization, near-optimal solutions can be constructed at a reasonable cost in terms of complexity. For the delay-based model, we also established intractability and then investigated a number of heuristics that involved splitting the end-to-end constraints into local constraints. Using these heuristics, we developed an approach that provides a solution of acceptable complexity for the case where the source of inaccuracies is the aggregation process that occurs in hierarchically interconnected networks.

A summary of the paper's results is shown in Figure 2, and we hope that they will foster additional works in what we feel is an important area. Some follow-on studies are reported in [5], [6], [12], [13], [14], [15].

Several aspects clearly require further investigation, and we proceed to name a few major ones. One aspect is identifying probability distributions that can realistically capture the inaccuracy in state information. The exponential distribution is an attractive candidate because it is an intuitively reasonable model, and one which also yields practical solutions. However, as illustrated in [6], state inaccuracies can be of many different forms based, for example, on the trigger mechanisms used for state updates (see [13] for an experimental investigation of possible distributions in the case of connections with bandwidth requirements). As a result, we do not expect that a single model will be adequate for all environments. This wide range of possible scenarios was indeed one of the primary motivations for seeking some basic understanding of the problem as carried out in this

paper. Another aspect is the incorporation of network optimization criteria within the path selection process, e.g., network utilization, carried load, number of flows successfully routed, etc. Here, we note that one possible approach for incorporating such criteria into our model, is to appropriately "distort" the p.d.f.'s used to account for state inaccuracy. For example, if network optimization requires to discourage the use of a link, its related p.d.f. can be (artificially) changed so as to manifest lower success probabilities. While this seems a plausible and promising approach, understanding the appropriate level of distortion required to comply with a given criteria, is clearly an issue that calls for further investigation. Finally, another important aspect is an extension to multicast routing, and some initial, related results can be found in [15].

REFERENCES

[1] R. Cherukuri, D. Dykeman (eds.), and M. Goguen (chair), "PNNI Draft Specification," ATM Forum 94-0471, November 1995.

[2] G. Apostolopoulos, R. Guérin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, "QoS routing mechanisms and OSPF extensions," Internet Draft, `draft-guerin-qos-routing-ospf-05.txt`, December 1998, (Work in Progress).

[3] R. Guérin, S. Kamat, and S. Herzog, "QoS path management with RSVP," in *Proceedings of GLOBECOM*, Phoenix, AZ, November 1997.

[4] T. E. Tedijanto, R. O. Onvural, D. C. Verma, L. Gün, and R. A. Guérin, "NBBS Path Selection Framework," *IBM Systems Journal: Special Issue on* Networking BroadBand Services, vol. 34, no. 4, pp. 629–639, November 1995.

[5] A. Shaikh, J. Rexford, and K. Shin, "Dynamics of quality-of-service routing with inaccurate link-state information," Technical Report CSE-TR-350-97, University of Michigan, November 1997.

[6] G. Apostolopoulos, R. Guérin, S. Kamat, and S. Tripathi, "Quality of service based routing: A performance perspective," in *Proceedings of SIGCOMM*, Vancouver, Ontario, CANADA, September 1998, pp. 17–28.

[7] S. Shenker, C. Partridge, and R. Guérin, "Specification of guaranteed quality of service," Request For Comments (Proposed Standard) RFC 2212, Internet Engineering Task Force, September 1997.

[8] S. Sathaye, "ATM Forum Traffic Management Specification Version 4.0," ATM Forum 95-0013, December 1995.

[9] P. Samudra, "ATM User-Network Interface (UNI) Signalling Specification Version 4.0," ATM Forum 95-1434, December 1995.

[10] G. Apostolopoulos, R. Guérin, and S. Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in *Proceedings of INFOCOM'99*, New York, NY, March 1999, (To appear).

[11] J. Moy, "OSPF Version 2," Request For Comments (Standard) RFC 2178, Internet Engineering Task Force, July 1997.

[12] R. Guérin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proceedings of GLOBECOM*, Phoenix, AZ, November 1997.

[13] G. Apostolopoulos, R. Guérin, S. Kamat, and S. Tripathi, "Improving QoS routing performance under inaccurate link state information," in *Proceedings of ITC'16*, June 1999, (To appear).

[14] D. H. Lorenz and A. Orda, "QoS Routing in Networks with Uncertain Parameters," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 768–778, December 1998.

[15] D. H. Lorenz and A. Orda, "Optimal Partition of QoS Requirements on Unicast Paths and Multicast Trees," in *Proceedings of INFOCOM*, New York, NY, March 1999.

[16] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks - performance evaluation and optimization," *Computer Networks*, vol. 1, pp. 82–92, 1977.

[17] A. Bar-Noy and P. M. Gopal, "Topology distribution cost vs. efficient routing in large networks," in *Proceedings of SIGCOMM*, Philadelphia, PA, September 1990, pp. 242–252.

[18] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," in *Proceedings of SIGCOMM*, April 1995, pp. 82–92.

[19] W. C. Lee, "Spanning tree methods for link state aggregation in large communication networks," in *Proceedings of INFOCOM*, Boston, MA, April 1995, pp. 297–302.

[20] U. Gremmelmaier, J. Puschner, M. Winter, and P. Jocher, "Performance evaluation of the PNNI routing protocol using and emulation tool," in *Proceedings of ISS*, Toronto, Ontario, CANADA, September 1997, vol. 1, pp. 401–408.

[21] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, "Fast network decomposition," in *Proceedings of 11th Annual Symp. Principles of Distributed Computing*, Vancouver, BC, CANADA, August 1992, pp. 169–177.

[22] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, "Near linear cost sequential and distributed constructions of sparse neighborhood covers," in *Proceedings of 34th Annual Symp. Foundations of Computer Science*, Palo Alto, CA, November 1993, pp. 638–647.

[23] N. Linial and M. Saks, "Low diameter graph decompositions," *Combinatorica*, vol. 13, no. 4, pp. 441–454, 1993.

[24] Y. Bartal, "Probabilistic approximationof metric spaces and its algorithmic applications," in *Proceedings of 37th Annual Symp. Foundations of Computer Science*, Burlington, VT, October 1996, pp. 184–193.

[25] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, "Fast distributed network decomposition and covers," *J. Parallel Distrib. Comput.*, vol. 39, no. 2, pp. 105–114, December 1996.

[26] R. Guérin and A. Orda, "QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms," Research Report RC 20515, IBM, T. J. Watson Research Center, July 1996.

[27] A. K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Multiple Node Case," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 137–150, 1994.

[28] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan, "Efficient Network QoS Provisioning Based on per Node Traffic Shaping," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 482–501, August 1996.

[29] M.R. Garey and D.S. Johnson, *Computers and Intractability*, Freeman, San Francisco, 1979.

[30] J. B. Rosen, S. Z. Sun, and G. L. Xue, "Algorithms for the Quickest Path Problem," *Computers and Operations Research*, vol. 18, pp. 579–584, 1991.

[31] A. Orda, "Routing with End to End QoS Guarantees in Broadband Networks," *IEEE/ACM Transactions on Networking*, 1999, To appear.

[32] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[33] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Networks Flows*, Prentice-Hall, New Jersey, 1993.

[34] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[35] N. Megiddo, "Combinatorial Optimization with Rational Objective Functions," *Mathematics of Operations Research*, vol. 4, pp. 414–424, 1979.

[36] R. K. Ahuja, J. L. Batra, and S. K. Gupta, "Combinatorial Optimization with Rational Objective Functions: a Communication," *Mathematics of Operations Research*, vol. 4, pp. 314, 1983.

APPENDIX

PATH OPTIMIZATION WITH RATIONAL OBJECTIVE FUNCTIONS

Several times in this paper, we face the need to solve a path optimization problem with a rational objective function of the following form.

Problem min-PR: Given a graph $G(V, E)$ with a distinguished pair of source and destination nodes, an integer B, and, for all $l \in E$, a number $a_l > 0$ and an integer $b_l > 0$.[9] Denote by $\mathcal{P}^+$ the set of (simple, loopless) paths between the source and the destination, for which $\sum b_l < B$, i.e., $\mathcal{P}^+ = \{\mathbf{p} | \sum_{l \in \mathbf{p}} b_l < B\}$. Find a path $\mathbf{p}^* \in \mathcal{P}^+$, that minimizes $\frac{\sum_{l \in \mathbf{p}} a_l}{B - \sum_{l \in \mathbf{p}} b_l}$.

Problem **min-PR** resembles a class of combinatorial optimization problems, broadly known as "Minimum Cost-to-Time ratio problems (**min-CT**)" (see, e.g., [33] and the references therein), which can be generally stated as follows.

Problem min-CT: Given a graph $G(V, E)$, for all $l \in E$, two number x_l and y_l, such that $x_l > 0$ on all l and, on any path (alternatively, cycle) $\mathbf{p}$, $\sum_{l \in \mathbf{p}} y_l > 0$ holds. Among all (simple) paths $\mathbf{p}$ between a distinguished pair of source and destination

[9] We note that the results that follow hold also when B and b_l are non-integers, with a slight modification of the time complexity.

nodes (alternatively, all cycles $\mathbf{p}$), find a path (alternatively, cycle) $\mathbf{p}^*$ that minimizes $\frac{\sum_{l\in\mathbf{p}} x_l}{\sum_{l\in\mathbf{p}} y_l}$.

In [34], a polynomial algorithm was presented for the corresponding **min-CT** *cycle* optimization problem. The algorithm depends on both the number of inputs (i.e., on the size of the graph) and on a logarithmic term of the maximum size of an input value. Later, in [35] a general approach was described for solving minimum ratio problems, which, as a special case, yields a strongly polynomial algorithm (i.e., that depends on only the number of inputs but not on their size) for the **min-CT** *cycle* optimization problem. In [36] it was shown that, in general, the corresponding **min-CT** *path* optimization problem is NP-complete.

The above intractability result might indicate that our problem **min-PR** is intractable since we also attempt to optimize paths. However, the following discussion will show the opposite. We begin by noting that the intractability of (the path version of) problem **min-CT** stems from the requirement to limit the search on simple paths. If the solution is allowed to contain loops, then, as shall be demonstrated, the problem becomes tractable. Following common terminology, we shall call a path that may contain loops *a walk*. As a first step, then, we replace problem **min-CT** with a corresponding walk-optimization problem **min-CTW**. The key point in our approach is to note that the solution of problem **min-PR** is necessarily a (loopless) path, even when the search is conducted on the broader domain of walks. This fact will be formally established subsequently.

A second problem that we encounter when trying to solve problem **min-PR** through a solution to problem **min-CTW**, is that the latter assumes that the denominator of the objective function is positive on all paths/walks. Indeed, the problem does not make sense if the denominator takes a negative value; in terms of the various problems analyzed in the paper (i.e., those that boil down to a **min-PR**-type problem), a negative denominator corresponds to a path on which the end-to-end delay bound is violated with probability 1. Thus, one might suspect that problem **min-PR** would not be solved through a solution of problem **min-CTW**, unless the search is somehow limited to only paths/walks with positive denominators, which seems to be an intractable task. However, as we show below, it turns out that the basic solution to problem **min-CTW** operates correctly, even if its search includes forbidden paths/walks with negative denominators.

We now turn to a formal description of the problem, its solution and the corresponding correctness proof.

Given a graph $G(V,E)$, with a distinguished pair of source and destination nodes, and, for all $l \in E$, two numbers $x_l > 0$ and y_l (note that y_l may be negative). Denote by $\mathcal{W}^+$ the set of walks $\mathbf{w}$ between the source and the destination, for which $\sum y_l \geq 0$, i.e., $\mathcal{W}^+ = \{\mathbf{w} | \sum_{l\in\mathbf{w}} y_l > 0\}$. For a walk $\mathbf{w} \in \mathcal{W}^+$, denote $F(\mathbf{w}) = \frac{\sum_{l\in\mathbf{w}} x_l}{\sum_{l\in\mathbf{w}} y_l}$.

Problem min-CTW: Find a walk $\mathbf{w}^* \in \mathcal{W}^+$, between the source and the destination, of at most $N-1$ hops, such that, for all $\mathbf{w} \in \mathcal{W}^+$, $F(\mathbf{w}^*) \leq F(\mathbf{w})$.

For ease of presentation, we consider also problem **min-CTW(n)**, which is a restriction of problem **min-CTW** on walks of at most n hops. We proceed with a solution to this last problem.

Algorithm min-CTW(n):

1. If the shortest distance of n-hop walks between the source and the destination according to the metric $\{-y_l\}$ is nonnegative (i.e., no n-hop walk in $\mathcal{W}^+$) then $\lambda(n) \leftarrow \infty$ and Stop.
2. $L \leftarrow \frac{\min_{l\in E} x_l}{\max_{l\in E} n\cdot|y_l|}$, $H \leftarrow \frac{\max_{l\in E} n\cdot x_l}{\min_{l\in E} |y_l|}$.
3. $\lambda \leftarrow \frac{L+H}{2}$.
4. For all $l \in E$: $\bar{x}_l \leftarrow x_l - \lambda y_l$.
5. Among walks of n hops between the source and the destination, find a shortest walk (possibly containing loops) $\mathbf{w}$ according to the metric $\bar{x}_l$, and denote its length by Λ.[10]
6. If $\Lambda < 0$ then $H \leftarrow \lambda$.
7. If $\Lambda > 0$ then $L \leftarrow \lambda$.
8. If $\Lambda \neq 0$ then go to step (3), else ($\Lambda = 0$):
   (a) $\mathbf{w}(n) \leftarrow \mathbf{w}$.
   (b) $\lambda(n) \leftarrow \lambda$.
   (c) Stop.

Note that the returned values are $\lambda(n)$ and, whenever $\lambda(n) < \infty$, also $\mathbf{w}(n)$.

Lemma A.1: Algorithm *min-CTW(n)* solves problem **min-CTW(n)**. Its complexity is

$$O(n \cdot M \cdot \log(n \cdot \frac{\max_{l\in E} x_l}{\min_{l\in E} x_l} \cdot \frac{\max_{l\in E} y_l}{\min_{l\in E} y_l})).$$

Proof: First, note that for a walk $\mathbf{w} \notin \{\mathcal{W}^+\}$, we have, for any $\lambda > 0$,

$$\sum_{l\in\mathbf{w}} \bar{x}_l = \sum_{l\in\mathbf{w}} x_l - \lambda \sum_{l\in\mathbf{w}} y_l > 0;$$

also, it is clear from the algorithm that $\lambda > 0$ always holds.

Consider some hop-count n. Accordingly, all the following references to walks are to n-hop walks between the source and the destination. Step (1) identifies the case in which there is no n-hop walk in $\mathcal{W}^+$. Therefore, assume that $\mathcal{W}^+$ is nonempty, and let $\mathbf{w}^*(n)$ be an optimal n-hop walk, with a corresponding optimal value $\lambda^*(n) = F(\mathbf{w}^*(n))$.

Consider an arbitrary iteration of the algorithm. Suppose that the current λ is such that $\lambda = \lambda^*(n)$. This means that the length of $\mathbf{w}^*(n)$ according to the metric $\bar{x}_l$ is 0. Any other walk in $\mathcal{W}^+$ cannot have a smaller length since this would contradict the optimality of $\mathbf{w}^*(n)$. As shown above, walks not in $\mathcal{W}^+$ have a length larger than 0. We conclude that, for $\lambda = \lambda^*(n)$, the algorithm ends with the correct value.

Suppose now that the current λ is such that $\lambda > \lambda^*(n)$. Applying an argument similar to the above, it can be verified that the shortest walk (according to the metric $\bar{x}_l$) has length less than 0. Thus, the algorithm proceeds by decreasing the value of λ, i.e., it gets closer to $\lambda^*(n)$.

Finally, suppose that the current λ is such that $\lambda < \lambda^*(n)$. Any $\mathbf{w} \in \mathcal{W}^+$ must have $\sum_{l\in\mathbf{w}} x_l > \lambda \sum_{l\in\mathbf{w}} y_l$. This, together with the observation made earlier for walks not in $\mathcal{W}^+$, implies that, for any walk $\mathbf{w}$, $\sum_{l\in\mathbf{w}} \bar{x}_l > 0$. Thus, the algorithm proceeds by increasing the value of λ, i.e., it gets closer to $\lambda^*(n)$.

We conclude that the binary search advances in the right direction. The rest of the proof is similar to that of the corresponding algorithm for the **min-CT** problem (see [34], [33]). □

[10] This step can be carried out through a *Bellman-Ford* shortest path algorithm.

We are now ready to present a solution to the original problem, **min-PR**.

Algorithm min-PR:

1. $n \leftarrow 1$.
2. set $x_l \leftarrow a_l$, $y_l \leftarrow \frac{B}{n} - b_l$, for all $l \in E$.
3. Call algorithm *min-CTW(n)*, which returns $\lambda(n)$ and $\mathbf{w}(n)$.
4. $n \leftarrow n + 1$.
5. If $n < N$ then go to step (2).
6. $\mathbf{p}^*$ is a path $\mathbf{w}(m)$ for which $\lambda(m) \leq \lambda(n)$ for all $1 \leq n < N$.

The following lemma establishes the correctness of algorithm *min-PR*. In particular, it shows that $\mathbf{p}^*$ is, necessarily, a (simple) path.

Lemma A.2: The walk $\mathbf{p}^*$, identified by algorithm *min-PR*, is a (simple) path that solves problem **min-PR**.

Proof: It is straightforward that $\mathbf{p}^*$ is a walk that solves the corresponding problem **min-CTW**. Therefore, and since $\mathcal{P}^+ \subseteq \mathcal{W}^+$, it suffices to show that $\mathbf{p}^*$ does not contain loops. Suppose that $\mathbf{p}^*$ does contain a loop, denoted by $\mathbf{c}$. Denote $\mathbf{w} = \mathbf{p}^* \setminus \mathbf{c}$, and note that $\mathbf{w}$ is also a walk between the same source-destination pair. Let n and m be the number of hops on $\mathbf{c}$ and $\mathbf{w}$, respectively (therefore, the number of hops on $\mathbf{p}^*$ is $n+m$).

First, we establish that $\mathbf{w} \in \mathcal{W}^+$. Since $\mathbf{p}^*$ solves problem **min-CTW**, we have $\mathbf{p}^* \in \mathcal{W}^+$, therefore:

$$\begin{aligned} 0 &< \sum_{l\in\mathbf{p}^*} y_l = \sum_{l\in\mathbf{p}^*} (\frac{B}{n+m} - b_l) = B - \sum_{l\in\mathbf{p}^*} b_l \\ &= B - \sum_{l\in\mathbf{w}} b_l - \sum_{l\in\mathbf{c}} b_l < B - \sum_{l\in\mathbf{w}} b_l = \\ &= \sum_{l\in\mathbf{w}} (\frac{B}{m} - b_l) = \sum_{l\in\mathbf{w}} y_l, \end{aligned} \tag{14}$$

therefore $\mathbf{w} \in \mathcal{W}^+$.

We now show that the loop $\mathbf{c}$ contradicts the established optimality of $\mathbf{p}^*$ w.r.t. problem **min-CTW**.

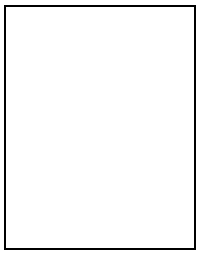
$$\begin{aligned} \frac{\sum_{l\in\mathbf{p}^*} x_l}{\sum_{l\in\mathbf{p}^*} y_l} &= \frac{\sum_{l\in\mathbf{p}^*} a_l}{\sum_{l\in\mathbf{p}^*} (\frac{B}{n+m} - b_l)} = \frac{\sum_{l\in\mathbf{p}^*} a_l}{B - \sum_{l\in\mathbf{p}^*} b_l} \\ &= \frac{\sum_{l\in\mathbf{w}} a_l + \sum_{l\in\mathbf{c}} a_l}{B - \sum_{l\in\mathbf{w}} b_l - \sum_{l\in\mathbf{c}} b_l} \\ &= \frac{\sum_{l\in\mathbf{w}} a_l + \sum_{l\in\mathbf{c}} a_l}{\sum_{l\in\mathbf{w}} (\frac{B}{m} - b_l) - \sum_{l\in\mathbf{c}} b_l} > \frac{\sum_{l\in\mathbf{w}} a_l}{\sum_{l\in\mathbf{w}} (\frac{B}{m} - b_l)} \\ &= \frac{\sum_{l\in\mathbf{w}} x_l}{\sum_{l\in\mathbf{w}} y_l}. \end{aligned} \tag{15}$$

Since $\mathbf{w} \in \mathcal{W}^+$, (15) implies that $\mathbf{w}$ is a better solution for problem **min-CTW** than $\mathbf{p}^*$, which is a contradiction. □

We thus obtain the following result:

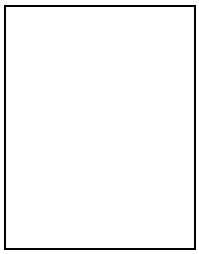
Theorem A.1: Algorithm *min-PR* solves problem **min-PR**. Its complexity is $O(N^2 M \cdot \log(N \cdot \frac{\max_{l\in E} a_l}{\min_{l\in E} a_l} \cdot B))$.

Proof: Follows from Lemmas A.1 and A.2. □



Roch Guérin (S'84, M'86, SM'91) received the "Diplôme d'Ingénieur" from the École Nationale Supérieure des Télécommunications, Paris, France, in 1983, and his M.S. and Ph.D. from the California Institute of Technology, both in Electrical Engineering, in 1984 and 1986 respectively. He recently joined the department of Electrical Engineering of the University of Pennsylvania, where he is the Alfred Fitler Moore Professor of Telecommunications Networks. Before joining the University of Pennsylvania, he had been with IBM at the Thomas J. Watson Research Center, Yorktown Heights, New York, since 1986, where he was the Manager of the Network Control and Services department prior to his departure. His current research interests are in the general area of Quality-of-Service support in high-speed networks, and in particular QoS routing and scheduling and buffer management mechanisms. He is also interested in aggregation techniques, in terms of both protocols and service models, for scalable service deployment.

Dr. Guérin is a member of Sigma Xi and a Senior member of the IEEE Communications Society. He is an editor for the IEEE/ACM Transactions on Networking, an area editor for the IEEE Communications Surveys, and the editor of the ACM SIGCOMM Computer Communication Review. He is the current chair for the IEEE Technical Committee on Computer Communications, served at the General Chair for the IEEE INFOCOM'98 conference, and has been active in the organization of many conferences and workshops such as the ACM SIGCOMM conference, the IEEE ATM Workshop, etc. He was an editor for the IEEE Transactions on Communications and the IEEE Communications Magazine. In 1994 he received an IBM Outstanding Innovation Award for his work on traffic management in the BroadBand Services Network Architecture. His email address is: guerin@ee.upenn.edu.



Ariel Orda (S'94-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 a Senior Lecturer and the Academic Head of the Computer Networking Laboratory. In 1993–1994, he was with the Center for Telecommunication Research, Columbia University, New York, NY, as a Visiting Scientist, and with AT&T Bell Laboratories, Murray Hill, NJ, as a Scientific Consultant. During the summers of 1992 and 1995–1998, he held visiting positions at AT&T Bell Laboratories, Murray Hill, NJ, IBM Watson, Hawthorne, NY, and Bell Laboratories - Lucent Technologies, Holmdel, NJ. Since 1991 he has held several consulting positions with the Israeli industry.

His current research interests include QoS routing, the application of game theory to computer networking, network pricing, and distributed network algorithms.

Dr. Orda received the Award of the Chief Scientist in the Ministry of Communication in Israel, a Gutwirth Award for outstanding distinction, and the Research Award of the Association of Computer and Electronic Industries in Israel. His email address is: ariel@ee.technion.ac.il.