

QoS Routing in Networks with Uncertain Parameters

Dean H. Lorenz and Ariel Orda

Department of Electrical Engineering
Technion—Israel Institute of Technology
Haifa 32000, Israel

{deanh@tx, ariel@ee}.technion.ac.il

Abstract— This work considers the problem of routing connections with QoS requirements across networks, when the information available for making routing decisions is inaccurate. This uncertainty about the actual state of a network component arises naturally in a number of different environments, which are reviewed in the paper. The goal of the route selection process is then to identify a path that is most likely to satisfy the QoS requirements. For end-to-end delay guarantees, this problem is intractable. However, we show that by decomposing the end-to-end constraint into local delay constraints, efficient and tractable solutions can be established.

We first consider the simpler problem of decomposing the end-to-end constraint into local constraints, for a given path. We show that, for general distributions, this problem is also intractable. Nonetheless, by defining a certain class of probability distributions, which possess a certain convexity property, and restricting ourselves to that class, we are able to establish efficient and exact solutions. Moreover, we show that typical distributions would belong to that class.

We then consider the general problem, of combined path optimization and delay decomposition. We present an efficient solution scheme for the above class of probability distributions. Our solution is similar to that of the restricted shortest-path problem, which renders itself to near-optimal approximations of polynomial complexity. We also show that yet simpler solutions exist in the special case of uniform distributions.

Keywords—Routing, Networks, QoS, Delay, Metric Inaccuracy, Topology Aggregation.

I. INTRODUCTION

Broadband integrated services networks are expected to support multiple and diverse applications, with various quality of service (QoS) requirements. Accordingly, a key issue in the design of broadband architectures is how to provide the resources in order to meet the requirements of each connection. The establishment of efficient QoS routing schemes is, undoubtedly, one of the major building blocks in such architectures. Indeed, QoS routing has been the subject of several studies and proposals (see, e.g., [2], [4], [5], [9], [12] and references therein). It has been recognized that the establishment of an efficient QoS routing scheme poses several complex challenges. One of the major challenges results from the inherent uncertainty of the information available to the QoS routing process.

As networks grow in size and complexity, full knowledge on network parameters is typically unavailable. Indeed, each single entity in the network cannot be expected to have detailed and instantaneous access to all nodes and links. Routing must therefore rely on partial or approximate information, and still meet the QoS demands. Among the various QoS parameters, the two major ones are bandwidth and end-to-end delay. In the presence of inaccuracies, the former was shown to be polynomially solvable, while the latter poses major obstacles, such as computational intractability [5].

In this work we focus on end-to-end delay guarantees. We explore the impact of inaccurate network information on the QoS routing process, identify useful and problematic properties in this process and present efficient solutions to the various related problems.

We proceed to discuss the possible origins of uncertainty in network parameter.

A. Origins of Uncertain Parameters

Network Dynamics Many parameters associated with delay requirements are affected by temporal conditions, such as congestion. Parameters advertised by a link might be based, for example, on average behavior or on worst case behavior. In either case the advertised parameters are not accurate. This inaccuracy can be eliminated by rapidly advertising the current, updated, accurate conditions. Unfortunately, this is impractical when the network is highly dynamic and changes are frequent. Thus, advertised values should be considered as uncertain. The precise probability distributions associated with each value depends on *a priori* knowledge on the frequency of updates, and the dynamics of the network.

Aggregation in Large Networks In interconnected networks, the growth in information makes it practically impossible to maintain accurate knowledge about all nodes and links. Accordingly, proposals have been made on how to provide the needed information in a scalable form. For example, the ATM Forum PNNI standard [9] introduces a hierarchical process that aggregates information as the network gets more and more remote. However, the aggregation process inherently decreases the accuracy of the information and introduces uncertainty. The semantics of the available parameters depend on the aggregation method used. For instance, we could consider the parameters as averages, or as best or worst cases. Some aggregation schemes may advertise a possible range for each parameter, which may be considered as uniformly distributed within this range. Other schemes may imply different probability distributions, and publish specific parameters associated with these distributions, such as mean and variance.

Rate parameters can be retained through aggregation (see for example [7]). This is an indication that “rate-based” models for delay guarantees can have aggregated delay parameters without any uncertainty. However, such models are not really accurate because they implicitly assume knowledge on the topology, e.g. number of hops, for which aggregation introduces uncertainty. Furthermore, the rate information itself might contain inaccura-

cies [5], thus some uncertainty remains.

Hidden Information Interconnected networks may include private networks that hide some or all of their information. One reason for this could be hiding the network's internal, proprietary, mechanisms. Typically, such networks would advertise information that contains inaccuracies or advertise ranges for specific parameters. We can interpret this information as probability distributions, based on parameters supplied by these networks, or by prior experience.

A second possible cause for hidden information in subnetworks is to maintain some degree of freedom in internal routing. For each request, the subnetwork is free to choose any internal route that satisfies the QoS requirements. The network may advertise the likelihood of path availability for each QoS requirement, in which case the QoS parameters should be treated as random variables.

Approximate Calculation Even the "exact" node and link parameters cannot be assumed to be truly accurate. Typically, they are just approximations of the real parameters and values, since they are based on elaborated models that cannot represent the true intricacy of the devices. The calculated parameters are usually upper bounds (as in [10]) or incorporate some inaccurate assumptions. Approximate calculation is hence yet another source of uncertainty in the advertised parameters.

B. Goals of the Paper

Our overall goal is to investigate the impact of uncertain parameters on routing with end-to-end delay guarantees. We assume a framework where the delay guarantees that are advertised by each link are random variables with known distributions. These variables represent the probability, that a link can satisfy a QoS delay requirement. We further assume that this knowledge is available to us, and that the probability distributions on the links are independent. A source node is presented with a request to establish a new connection that meets given end-to-end delay requirements. The source node seeks a path that is most likely to satisfy these requirements.

The basis for this work was laid in [5]. That study presented the framework of a network with uncertain parameters, and solved the QoS routing problem for rate demands and for delay demands under a rate-based model, as in [11]. It also presented heuristic methods for dealing with end-to-end delay requirements in models that are not rate-based, and optimal solutions for specific cases. The present study extends that framework, and achieves optimal solutions for the general case.

The rest of this paper is organized as follows. In Section II we introduce terminology and definitions, and present different variants of the problem. In Section III we consider the simpler problem of decomposing the end-to-end constraint into local constraints, for a given path. We show that, for general distributions, this problem is also intractable. Nonetheless, by defining a class of probability distributions, which possess a certain convexity property, and restricting ourselves to that class, we are able to establish efficient and exact solutions. Moreover, we show that typical distributions would belong to that class. We then proceed to consider the combined problem of path selection and constraint decomposition. As a first step, in Section IV we discuss the *restricted shortest path* problem, which is closely re-

lated to our problem. Then, in Section V, we present a solution to the QoS routing problem, under the assumption that the end-to-end delay is partitioned along the optimal path. In Section VI we present an improved solution for uniform distributions. Conclusions are presented in Section VII. Due to space limits, many of the proofs and technical details are omitted from this version, and can be found in [8].

II. MODEL AND PROBLEMS

This section introduces the notations and definitions that are used throughout the paper. We begin with a formal definition of the various problems discussed in this work.

A network is represented by a graph $G(V, E)$. There is a single source s and a single destination t , and we need to establish a connection with QoS requirements, namely end-to-end delay requirements. We denote by $|\mathbf{p}|$ the number of links in a path \mathbf{p} .

For each link $l \in E$ we are given a function $f_l(d)$, which is the probability that the delay, d_l , on l is not greater than d . We denote by $\pi_D(\mathbf{p})$ the probability that an end-to-end delay bound D can be guaranteed on the path \mathbf{p} . We shall assume that the functions $\{f_l(d)\}_{l \in E}$ are known and that the probabilities are independent. Note that even in a dynamic network, the functions $f_l(d)$ do not change, rather the dynamics are incorporated in the probabilities.

We need to satisfy a given end-to-end delay constraint. Observe that we do not seek a shortest path, but rather a path that is most likely to satisfy our delay constraints. This can be formalized as an optimization problem with an uncontinuous (threshold) restriction.

Problem MP (Most Probable path): Find a path \mathbf{p}^* s.t. for every other path \mathbf{p} , $\pi_D(\mathbf{p}^*) \geq \pi_D(\mathbf{p})$.

It is important to notice that Problem MP is different from finding the path that is most likely to be shortest.¹ Indeed, it is possible that the path \mathbf{p}^* would unlikely be the shortest path despite being the most likely to satisfy our delay constraints. This is illustrated through the following example.

Example 1: Suppose we have two links a, b connecting the source to the destination. Let

$$f_a(d) = \begin{cases} 0 & \text{if } d < 1, \\ 0.9 & \text{if } 1 \leq d < 3, \\ 1 & \text{if } d \geq 3; \end{cases} \quad f_b(d) = \begin{cases} 0 & \text{if } d < 2, \\ 1 & \text{if } d \geq 2. \end{cases}$$

It is obvious that link a is more likely the shortest path, but if our delay requirement is 2, link b will be our choice.

A possible scheme for satisfying end-to-end delay constraints, which is very often the most practical, is to decompose the constraint into local constraints, each imposed on a link along the path. This means that the total delay guarantee, D , should be partitioned into a set of guarantees $S_D(\mathbf{p}) = \{D_l\}_{l \in \mathbf{p}}$.

Definition 1: Given a path \mathbf{p} , and a set of link delay requirements $S_D(\mathbf{p}) = \{D_l\}_{l \in \mathbf{p}}$, define

$$\pi\left(\{D_l\}_{l \in \mathbf{p}}\right) = \Pr\{d_l \leq D_l \quad \forall l \in \mathbf{p}\} = \prod_{l \in \mathbf{p}} f_l(D_l).$$

¹When considering d_l as a random variable with a distribution f_l .

Different partitions may lead to different probabilities of success, and, as we shall see, finding the best partition (for a given path) is a difficult problem.

Problem OP (Optimal Partition): Given a path \mathbf{p} and a delay D , find a partition $S_D^*(\mathbf{p}) = \{D_l^*\}_{l \in \mathbf{p}}$, s.t. $\pi(S_D^*(\mathbf{p})) \geq \pi(S_D(\mathbf{p}))$, for every (other) partition $S_D(\mathbf{p}) = \{D_l\}_{l \in \mathbf{p}}$.

The assumption that the delay is partitioned imposes a new restriction on the solution of Problem MP. It also changes the probability space, since each event is now determined by the solution to Problem OP. This leads to a revised problem as follows.

Problem OP-MP (Optimally Partitioned MP): Given a total delay D , find a path \mathbf{p}^* s.t. for every other path \mathbf{p} , $\pi(S_D^*(\mathbf{p}^*)) \geq \pi(S_D^*(\mathbf{p}))$.

Problem OP-MP is identical to Problem MP except that the delay is partitioned. Partitioning the delay is a requirement that rises from the actual way in which delay guarantees are given. We shall later see that this partitioning also introduces some very useful properties, which can be exploited in finding the optimal path.

A. Practical Considerations

In practice, there are further restrictions to the above problems.

One such restriction is on the minimal probability of success, *i.e.*, $\pi(S_D(\mathbf{p})) > p_{\min}$; in other words, we do not consider paths with a probability of success smaller than p_{\min} . This imposes a restriction on the minimum probability for each link $f_l(d) > p_0$.² Since $f_l(d)$ is monotonic increasing, we get a restriction on the minimal delay t_l allocated to each link. This means that $f_l(t_l) > p_0$ implies $D_l \geq t_l$ for all $l \in \mathbf{p}$. Note that this restriction always holds, as $p_0 > 0$. Accordingly, in the following we shall assume that for every link l , we have a positive minimal delay t_l . Moreover, we shall assume that there is a non zero probability p_0 , s.t. for every link $l \in E$, we have $f_l(t_l) > p_0$, that is, the probability that the link can guarantee a delay of t_l is positive.

A second restriction is regarding the granularity of the delay values. In practice, the delays cannot be broken into arbitrarily small pieces. We denote the smallest possible change in delay by δ , *i.e.*, δ is the resolution we have for the delay. The smaller δ is the more accurate the solution is, but generally at the cost of more resources needed to solve the problem (or even just to represent the solution). We shall assume, without loss of generality, that $\delta = 1$, and that all delays (d_l, t_l, D_l, D) are integers.

B. Relation to Shortest Path Algorithms

The probability of success on a path is a product of the probability of success on each link. This is readily transformed to a usual sum by defining a proper cost function. Given a probability distribution $f_l(d_l)$, we define a corresponding cost function $c_l(d_l) \equiv -\log f_l(d_l)$. The cost associated with each link is positive and decreases as the delay allocated to the link increases.

²Typically, p_0 is greater than p_{\min} , since the probability of a path is determined by all its links. For instance, if k links have p_0 as their probability of success then we must have $p_0 \geq \sqrt[k]{p_{\min}}$.

Most shortest path problems (and consequently their corresponding algorithms) have an *optimal sub-structure* property [1]: a shortest path between two vertices contains within it other shortest paths. This property is a hallmark of the applicability of both *dynamic-programming* and *greedy* methods.

Specifically, the above property means that if $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$ is a shortest path from s to t , then $u \rightsquigarrow v$ is a shortest path from u to v . Thanks to this property one does not have to check all possible paths, and the complexity reduces from exponential growth to polynomial. Unfortunately, this property *does not* hold in our framework, making it impossible to apply standard shortest path algorithms to our problems. This is illustrated through the next example.

Example 2: Consider a network with three vertices s, v, t and three links: two links a, b from s to v , and a single link c from v to t . Suppose

$$\begin{aligned} f_a(d) &= \begin{cases} \frac{1}{4}d & \text{if } d < 4, \\ 1 & \text{if } d \geq 4; \end{cases} \\ f_b(d) &= \begin{cases} 0 & \text{if } d < 2, \\ 1 & \text{if } d \geq 2; \end{cases} \\ f_c(d) &= \begin{cases} 0 & \text{if } d < C, \\ 1 & \text{if } d \geq C; \end{cases} \end{aligned}$$

where C is some threshold, and with a total delay requirement $D = 10$. Suppose we wish to solve Problem MP. We have to choose either link a or link b . Can we solve this through a sub-problem, *i.e.*, the best path from s to v ? The answer is negative, as the solution depends on the value of C : for $C > 8$, link a is better, whereas for $C \leq 8$, link b is better.

Partitioning the delay, however, introduces a similar, useful property to Problem OP-MP, as follows: if $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$ is a solution of Problem OP-MP with total delay D , and it is partitioned into link delay constraints, such that D' is allocated to the sub-path $u \rightsquigarrow v$, then $u \rightsquigarrow v$ is a solution for total delay D' with the same delay partitioning. This enables us in certain circumstances to solve Problem OP-MP by greedy and dynamic programming methods.

The solution of Problem OP may also introduce some optimal sub-structure property. This is especially true if we try to solve Problem OP-MP with a predefined (non-optimal) solution of Problem OP. For instance, partitioning the delay with equal probabilities for all links allows us to use standard shortest path solutions, as shown in [5].

III. SOLUTION TO PROBLEM OP

In this section we explore Problem OP, and establish an efficient solution for a wide class of probability distributions. Due to space limits, in this version we just outline the results, and the reader is referred to [8] for the full details.

Problem OP can be shown to be NP hard, through a reduction to the *0-1 knapsack problem* [1]. However, in the following we shall show that, by restricting ourselves to a certain class of distributions, exact and efficient solutions can be obtained.

First, we define a family of probability distributions for which we can present efficient solutions. This family consists of distributions with a certain convexity property.

Definition 2: For $c_l(d_l) \equiv -\log f_l(d_l)$, let Ψ be the set of

probability functions $f(d)$ s.t. $\Delta c(d) \equiv c(d) - c(d-1)$ is (strictly) monotonic increasing with d .

Later, we shall identify the probability distributions that belong to Ψ .

Next, we show that any optimal partition is an *equilibrium* point, in the sense that moving a single delay unit from one link to another does not improve the overall probability of success. The following lemma states this property, in terms of the “cost” functions $\{c_l\}_{l \in \mathbf{p}}$, as defined in Section II-B, and the partition $\{D_l\}_{l \in \mathbf{p}}$.

Lemma 1: If $\{D_l\}_{l \in \mathbf{p}}$ is an optimal solution to Problem OP then for any $e, l \in \mathbf{p}$ we have $\Delta c_e(D_e) \leq \Delta c_l(D_l + 1)$.

Proof: Otherwise we can reduce the total cost by adding 1 to D_l at the expense of D_e . The cost will reduce because $((c_l(D_l) + c_e(D_e)) - (c_l(D_l + 1) + c_e(D_e - 1))) = \Delta c_e(D_e) - \Delta c_l(D_l + 1)$. ■

Lemma 1 leads to an important corollary.

Corollary 1: If $\{D_l\}_{l \in \mathbf{p}}$ is an optimal solution to Problem OP then there is a threshold α s.t. for all $l \in \mathbf{p}$ we have $\Delta c_l(D_l) \leq \alpha \leq \Delta c_l(D_l + 1)$.

Proof: Set $\alpha = \max_{l \in \mathbf{p}} \Delta c_l(D_l)$, and the result follows. ■

Focusing on cost functions in Ψ , Δc_l is monotonic increasing hence α determines D_l . We denote the delay allocated to the link l for a specific α by $D_l(\alpha)$, and from the definition of α we have $D_l(\alpha) = \sup \{d : \Delta c_l(d) \leq \alpha\}$. The corresponding delay allocated to the whole path is $D_{\mathbf{p}}(\alpha) = \sum_{l \in \mathbf{p}} D_l(\alpha)$. It can be verified that for distributions in Ψ , $S_D(\mathbf{p}) = \{D_l(\alpha)\}$ is a solution to Problem OP for $D = D_{\mathbf{p}}(\alpha)$.

As a consequence of the above result, we can employ a *greedy* scheme. Specifically, we distribute the total delay piece by piece, giving each bit to the link where it most improves the probability of success. Such a strategy involves D iterations, in each of which we select the optimal link and add to its allocated delay. With a straightforward implementation we get a complexity of $O(|\mathbf{p}|D)$.

This result can be further improved by searching for the threshold α , i.e., we find an α for which $D_{\mathbf{p}}(\alpha) = D$. Since the cost function is monotonic decreasing, α is negative. We can assume that the cost functions have bounded variations ($|\Delta c_l| \leq A$), therefore we must have $\alpha \in (-A, 0)$. We may also assume³ that changes in $D_l(\alpha)$ are of the same magnitude as changes in α , therefore a change of 1 in $D_{\mathbf{p}}(\alpha)$ requires a change of $|\mathbf{p}|^{-1}$ in α . This means that the *effective* size of our search space is of order $O(|\mathbf{p}|A)$. $D_{\mathbf{p}}(\alpha)$ is a monotonic function, thus we can use a binary search with $O(\log(|\mathbf{p}|A))$ iterations. Computing $D_{\mathbf{p}}(\alpha)$ requires $O(|\mathbf{p}|)$, hence the search can be implemented in $O(|\mathbf{p}| \log(|\mathbf{p}|A))$.

It is possible to solve Problem OP by extracting α directly from the equation $D = D_{\mathbf{p}}(\alpha)$. This can be done numerically in the general case, and analytically for certain distributions. One such distribution is the uniform, which we proceed to discuss.

A. Uniform Distribution

It can be verified that uniform distributions belong to Ψ . Next, we show that we can get an implicit expression for the optimal

partition $D_{\mathbf{p}}(\alpha)$.

Theorem 1: Let $f_l(d)$ be uniform, $f_l = \mathbf{U}(t_l, t_l + \delta_l)$, i.e.,

$$f_l = \begin{cases} 0 & \text{if } 0 < d < t_l, \\ \frac{1}{\delta_l}(d - t_l) & \text{if } t_l \leq d \leq t_l + \delta_l, \\ 1 & \text{if } d > t_l + \delta_l; \end{cases}$$

for every $l \in \mathbf{p}$. Let τ be a delay s.t. $\sum_{l \in \mathbf{p}} t_l + \sum_{l \in \mathbf{p}} \min(\tau, \delta_l) = D$.

Then, $S_D = \{D_l = t_l + \min(\tau, \delta_l)\}_{l \in \mathbf{p}}$ is a solution to Problem OP.

Proof: The derivative, $c'_l(d)$, is given by

$$c'_l(d) = -\frac{f'_l(d)}{f_l(d)} = \begin{cases} 0 & \text{if } 0 < d < t_l, \\ \frac{-1}{d-t_l} & \text{if } t_l < d < t_l + \delta_l, \\ 0 & \text{if } t_l + \delta_l < d; \end{cases}$$

so for $\alpha = \frac{1}{\tau}$ we get $D_l(\frac{1}{\tau}) = \sup \{d : c'_l(d) \leq \frac{1}{\tau}\} = t_l + \min(\tau, \delta_l)$. By the assumption on τ we have, for $\alpha = \tau^{-1}$,

$$D_{\mathbf{p}}(\alpha) = \sum_{l \in \mathbf{p}} D_l\left(\frac{1}{\tau}\right) = \sum_{l \in \mathbf{p}} t_l + \sum_{l \in \mathbf{p}} \min(\tau, \delta_l) = D$$

Since $f_l(d) \in \Psi$ and $D_{\mathbf{p}}(\alpha) = D$ we have that

$$S_D = \left\{ D_l(\alpha) = D_l\left(\frac{1}{\tau}\right) = t_l + \min(\tau, \delta_l) \right\}_{l \in \mathbf{p}}$$

is a solution to Problem OP. ■

Note that, since $c_l(d)$ is differentiable, we may apply the previous lemmas on $c'_l(d)$ rather than on $\Delta c_l(d)$.

Theorem 1 leads to the following alternative algorithm that solves Problem OP for uniform distributions. The idea is to seek the τ that appears in the theorem. From the proof of Theorem 1 (the calculation of $c'(d)$), we can realize how we should partition the delay: all links should get exactly the same addition to their minimal requirement, i.e., $D_l - t_l$ is a constant. We only have to compensate for the fact that this addition might be greater than δ_l .

Algorithm UNIFORM-OP: ($\mathbf{p}, f_l = \mathbf{U}(t_l, t_l + \delta_l)_{l \in \mathbf{p}}, D$)

- 1 if $\sum_{l \in \mathbf{p}} t_l > D$ then **Fail**
- 2 Estimate τ as $\tau \leftarrow \frac{1}{|\mathbf{p}|} \left(D - \sum_{l \in \mathbf{p}} t_l \right)$
- 3 $D_l \leftarrow t_l + \min(\tau, \delta_l)$
- 4 Repeat until $\sum_{l \in \mathbf{p}} D_l = D$
- 5 $R \leftarrow \{l \in \mathbf{p} : \tau \leq \delta_l\}$
- 6 $\tau \leftarrow \tau + \frac{1}{|R|} \left(D - \sum_{l \in \mathbf{p}} D_l \right)$
- 7 $D_l \leftarrow t_l + \min(\tau, \delta_l)$
- 8 $S_D \leftarrow \{D_l\}_{l \in \mathbf{p}}$

Theorem 2: If $f_l = \mathbf{U}(t_l, \delta_l)$ then Algorithm UNIFORM-OP solves Problem OP with complexity $O(|\mathbf{p}|)$.

B. Who belongs to Ψ ?

Ψ includes all probability distributions with convex cost functions $c(d)$. In [8] we investigate to what extent the assumption $f_l \in \Psi$ limits the choice of f_l . We proceed to outline the main conclusions.

It is important to note that the assumption is needed only in the region (t_l, D) . In other words, we are only interested in the

³See [8] for a more detailed discussion.

region $p_0 \leq f_l$. Typically, the minimal probability of success on each link, p_0 , should be close to 1 to allow a reasonable probability of success for the whole path.

Proposition 1: If a probability distribution $f_l(d)$ is concave, then $f_l \in \Psi$.

A probability distribution is concave if its density function is monotonic decreasing. This is true for many distributions if p_0 is big enough. For instance, the exponential distribution is concave for any choice of p_0 , and the normal distribution is concave for $p_0 \geq 0.5$.

In [8], we show that the uniform distribution also belongs to Ψ , and observe that, typically, the considered distributions can be expected to belong to Ψ , in the relevant region.

IV. RESTRICTED SHORTEST PATH PROBLEM

In this section we review the well known restricted shortest path problem and a (non-standard) variant of its dynamic programming solution. We then discuss the relation between this problem and Problem OP-MP. These observations shall help us solve Problem OP-MP, in the next section.

We begin by formally presenting the problem.

Problem RSP (Restricted Shortest Path): Given a network $G(V, E)$, a delay and a cost for each link $\{d_l, c_l\}_{l \in E}$ and a maximal delay D , find a path \mathbf{p}^* s.t. $\sum_{l \in \mathbf{p}^*} d_l \leq D$ and $\sum_{l \in \mathbf{p}^*} c_l \leq \sum_{l \in \mathbf{p}} c_l$, for any other path \mathbf{p} that satisfies the restriction $\sum_{l \in \mathbf{p}} d_l \leq D$.

Problem RSP is NP-hard [3], and is generally solved using dynamic programming. The version of the algorithm presented here is not standard, but it simplifies our later discussion.

At iteration n , we calculate the optimal path from s to each vertex v , with a delay limit of n , and store its cost $\mathcal{C}(n, v)$. The cost, $\mathcal{C}(n, u, v)$, is calculated similarly to $\mathcal{C}(n, v)$, except that the last link on the optimal path to v is assumed to be (u, v) . At each iteration, after we have $\mathcal{C}(n, u)$ for each vertex u , we find $\mathcal{C}(n + d_{(u,v)}, u, v)$ for all outgoing links from u .

Finding the optimal path involves calculating $\min_u \mathcal{C}(n, u, v)$ and choosing the *parent*, $\theta(n, v)$, of v on the optimal path. We add a loop (v, v) , to each vertex, with a delay of 1 and a zero cost, i.e., $d_{(v,v)} = 1, c_{(v,v)} = 0$. This addition exempts us from handling the special case, where $\mathcal{C}(n, v) = \mathcal{C}(n - 1, v)$.

Algorithm DYNAMIC-RSP: $(G(V, E), \{d_l, c_l\}_{l \in E}, D)$

```

1  $\mathcal{C}(0, v) \leftarrow \infty \quad \forall v \neq s$ 
2  $\mathcal{C}(0, s) \leftarrow 0$ 
3  $\Theta(v) \leftarrow \{u : (u, v) \in E\}, \theta(v, 0) \leftarrow \text{NIL} \quad \forall v \in V$ 
4  $n \leftarrow 0$ 
5 Repeat until  $n \geq D$ 
6   for all  $l = (u, v) \in E$  do
7      $\mathcal{C}(n + d_l, u, v) \leftarrow \mathcal{C}(n, u) + c_l$ 
8      $n \leftarrow n + 1$ 
9   for all  $v \in V$  do
10     $\mathcal{C}(n, v) \leftarrow \min_{\Theta(v)} \mathcal{C}(n, u, v)^4$ 
11     $\theta(n, v) \leftarrow \arg \min_{\Theta(v)} \mathcal{C}(n, u, v)$ 

```

Since $\left| \bigcup_{v \in V} \Theta(v) \right| = |E|$, calculating $\min_{\Theta(v)} \mathcal{C}(n, u, v)$ requires a total of $O(|E|)$ for all vertices. At each iteration we go over all links in $O(|E|)$, and the number of iterations is D . Thus, the complexity is $O(D|E|)$.

⁴For all existing $\mathcal{C}(n, u, v)$.

A. Application to Problem OP-MP

Observe that Problem OP-MP is also NP-hard. This can be proved by a straightforward reduction to Problem RSP. For a formal proof, see [8].

The crucial difference between Problem RSP and Problem OP-MP is that we do not have a single pair (d_l, c_l) for each link $l = (u, v)$. Rather, we have a complete function $f_l(d_l)$. This means that we cannot calculate $\mathcal{C}(n, u, v)$ in $O(1)$. We must find the minimal among all possibilities for d_l and the corresponding $c_l(d_l)$. In the worst case, this would mean that at each iteration $m, 0 \leq m \leq n - 1$, we recalculate $\mathcal{C}(n, u, v)$ according to $\mathcal{C}(m, u) + c_{(u,v)}(n - m)$. This implies a total complexity of $O(D^2|E|)$.

A better bound can be achieved for discrete probability distributions. Assume there is a bound K such that, for each link $l \in E$, there are no more than K possible delays, that is, $f_l(d_l)$ (and $c_l(d_l)$) are discrete functions with no more than K points. Under this assumption, calculating $\mathcal{C}(n, u, v)$ requires finding a minimum among K possibilities and can be done in $O(K)$, leading to a total complexity of $O(DK|E|)$. Alternately, we could replace each link l with a set of K links $\{l_k\}_{1 \leq k \leq K}$, each with a specific delay $d_{l,k}$. We then get a graph \tilde{G} with $|\tilde{E}| = K|E|$, and the related complexity is again $O(D|\tilde{E}|) = O(DK|E|)$.

Yet a more careful analysis of Problem OP-MP enables to considerably reduce the complexity of the solution, for a wide class of (both discrete and continuous) distributions. This is shown in the next section.

V. PROBLEM OP-MP

In this section we solve Problem OP-MP using dynamic programming methods. The solution uses a modification of Algorithm DYNAMIC-RSP.

Suppose we have a data structure that contains $\mathcal{C}(n, u, v)$, for all $0 \leq n \leq D$. Clearly, this structure must be updated D times, once for each calculation of $\mathcal{C}(m, u), 0 \leq m \leq D - 1$. We will denote by $\mathcal{C}^{(m)}(n, u, v)$, the value of $\mathcal{C}(n, u, v)$ after iteration m .

For specific n and m , where $n > m$, $\mathcal{C}(m, u)$ may affect $\mathcal{C}(n, u, v)$, only if we allocate a delay of $(n - m)$ to the link (u, v) . In this case, we define $\mathcal{C}_m(n, u, v)$ to be the calculated $\mathcal{C}(n, u, v)$, that is $\mathcal{C}_m(n, u, v) \equiv \mathcal{C}(m, u) + c_{(u,v)}(n - m)$. We also define $\mathcal{C}_m(n, u, v) \equiv \infty$ for $n \leq m$.

In each iteration we update, for all n , $\mathcal{C}^{(m)}(n, u, v)$ to $\min\{\mathcal{C}^{(m-1)}(n, u, v), \mathcal{C}_m(n, u, v)\}$. Note that $\mathcal{C}(n, u, v)$ cannot change after iteration n , and for each n , there is some $0 \leq m < n$ for which $\mathcal{C}(n, u, v) = \mathcal{C}_m(n, u, v)$. We will denote this m by *base* (n, u, v) .

We are now ready to present the modified algorithm. The algorithm employs a data structure that holds *base* (n, u, v) , and supports three operations: INIT - initialize the structure; UPDATE - update the structure at each iteration; and GET - get the value of *base* (n, u, v) for a specific n . Note that for each vertex, we need to hold not only its parent on the optimal path, but also the delay allocated to the last hop.

Algorithm DYNAMIC-OP-MP: $(G(V, E), \{f_l\}_{l \in E}, D)$

```

1  $C(0, v) \leftarrow \infty \quad \forall v \neq s$ 
2  $C(0, s) \leftarrow 0$ 
3  $\Theta(v) \leftarrow \{u : (u, v) \in E\}, \theta(v, 0) \leftarrow \text{NIL} \quad \forall v \in V$ 
4 for all  $l = (u, v) \in E$  do
5   INIT( $u, v$ )
6  $n \leftarrow 0$ 
7 Repeat until  $n \geq D$ 
8   for all  $l = (u, v) \in E$  do
9     UPDATE( $n, u, v$ )
10   $n \leftarrow n + 1$ 
11  for all  $l = (u, v) \in \Theta(v)$  do
12     $base(n, u, v) \leftarrow \text{GET}(n, u, v)$ 
13     $C(n, u, v) \leftarrow C_{base(n, u, v)}(n, u, v)$ 
14  for all  $v \in V$  do
15     $C(n, v) \leftarrow \min_{\Theta(v)} C(n, u, v)$ 5
16     $u \leftarrow \arg \min_{\Theta(v)} C(n, u, v)$ 
17     $d \leftarrow n - base(n, u, v)$ 
18     $\theta(n, v) \leftarrow (u, d)$ 

```

The algorithm is essentially the same as Algorithm DYNAMIC-RSP. The main difference is in calculating $C(n, u, v)$. After each calculation of $C(n, u)$, we use the UPDATE procedure to update our data structure. At Lines 11-13, we extract the value of $base(n, u, v)$ from our data structure, and calculate $C(n, u, v)$. We use $base(n, u, v)$ at Line 17 to calculate the delay allocated to the last hop and then store this delay with the parent at Line 18.

With an efficient implementation of $base(n, u, v)$, the complexity of this algorithm is $O(|E| D \log D)$.

A. Implementing $base(n, u, v)$

In the general case, maintaining our data structure requires $O(D)$ in each iteration, which results in a total complexity of $O(D^2 |E|)$. However, we will show that, for the wide class of probability distributions that belong to Ψ , this can be done in $O(\log D)$, reducing the total complexity of Algorithm DYNAMIC-OP-MP to $O(|E| D \log D)$.

As described above, at each iteration we perform the update

$$C^{(m)}(n, u, v) = \min \left\{ C^{(m-1)}(n, u, v), C_m(n, u, v) \right\}$$

for all n . Since $0 \leq n \leq D$, we need $O(D)$ for the update, however the next lemma shows that we can improve upon this for distributions in Ψ .

Lemma 2: If $f_{(u,v)}(d) \in \Psi$, and there exists an n_0 , for which $C^{(m)}(n_0, u, v) = C_m(n_0, u, v)$, then $C^{(m)}(n, u, v) = C_m(n, u, v)$ for all $n \geq n_0$.

Intuitively, Lemma 2 means that $C^{(m-1)}(n, u, v)$ and $C_m(n, u, v)$ have at most a single intersection point. At each iteration we only need to search for the next intersection point. This can be done, using a binary search, in $O(\log D)$. In each update we seek the smallest n_0 that satisfies the condition of Lemma 2, and set $base(n, u, v) \leftarrow m$, for all $n \geq n_0$.

We implement $base(n, u, v)$ by a balanced binary sort tree. The tree is kept balanced by maintaining a subset of a full tree, that is, each level splits the interval exactly by half. We also maintain an in-order linked list of all “live” nodes in the tree. During the search, we create all the nodes on the path leading to

n_0 and update the linked list. When n_0 is found we trim the tree by simply connecting node n_0 to node D .

The following procedure performs the initialization:

```

Procedure INIT:  $(u, v)$ 
1  $tree = tree(u, v), last = last(u, v)$ 
2  $last \leftarrow 0$ 
3  $tree[0].base \leftarrow 0$ 
4  $tree[0].next \leftarrow D$ 
5  $tree[D].base \leftarrow 0$ 
6  $tree[D].next \leftarrow D$ 

```

Line 1 just simplifies the notation. Each tree is initialized with two nodes. Node D is needed because each search for n_0 begins by checking if $n_0 < D$. The initial $base$ value assumes the allocation to the link is $D_l = D$. Node 0 is needed for the beginning of the links list, that is, our initial $base$ value for each iteration is 0. The variable $last$ is used by GET and is explained later.

The next procedure is the core of the algorithm.

```

Procedure UPDATE:  $(m, u, v)$ 
1  $tree = tree(u, v)$ 
2 if  $C_m(D, u, v) > C_{tree[D].base}(D, u, v)$  then return6
3  $L \leftarrow 0, H \leftarrow D$ 
4  $n \leftarrow D$ 
5 repeat
6   if  $C_{tree[n].base}(n, u, v) \leq C_m(n, u, v)$  then
7      $L \leftarrow n$ 
8   else
9      $tree[n].base \leftarrow m$ 
10     $tree[n].next \leftarrow H$ 
11     $H \leftarrow n$ 
12    if  $(H - L) \leq 1$  then return
13     $n \leftarrow \lfloor \frac{L+H}{2} \rfloor$ 
14    if  $tree[L].next = H$  then
15       $tree[n].next \leftarrow H$ 
16       $tree[L].next \leftarrow n$ 
17       $tree[n].base \leftarrow tree[L].base$ 

```

The procedure traverses through the search tree, creating new nodes when needed, and trimming the tree when needed. This procedure is called at each iteration, after $C(m, u)$ is calculated, for all outgoing links (u, v) , from vertex u . The search for n_0 is done in the *repeat* loop, where n is the currently visited node and (L, H) is the current interval.

Lemma 3: Procedure UPDATE updates the tree in $O(\log D)$.

Proof: We assume the tree is correct before the UPDATE call. At each iteration of the *repeat* loop (Line 5), we compare what can be achieved from the node’s $base$ (using $C(base, u)$) with what can be achieved from m (using $C(m, u)$). If the current $base$ is better than m , we must have $n_0 > n$, and we should keep searching in the right branch, *i.e.*, in the interval (n, H) . If m is better, we must have $n_0 < n$, and we should keep searching in the left branch, *i.e.*, in the interval (L, n) . In the latter case, we must also update the $base$ to m , and trim the tree, *i.e.*, update $next$ to H . Thus, we have established that we search in the right direction, and update the nodes along the way correctly.

We need to show how we create new nodes. We have an indication that n is a new node if L and H are neighbors in the linked list (Line 14). In this case, we insert the node n in the list, and set the default $base$ to the base of L . Note that the new node will be examined (and updated if needed) on the next *repeat* iteration.

⁵For all existing $C(n, u, v)$.

⁶ $C(m, v)$ cannot improve $C(n, u, v)$ for any n .

Each node is updated (and if needed, created) in $O(1)$ and the number of nodes is bounded by D . Since the tree is balanced, the search is done in $O(\log D)$ iterations. ■

GET should simply return the value of $tree[n].base$, and can be easily implemented by traversing the tree in $O(\log D)$. However, if we save the last returned value between calls we can implement GET in $O(1)$ as follows:

Procedure GET: (n, u, v)
1 $tree = tree(u, v)$, $last = last(u, v)$
2 If $tree[last].next \leq n$ then $last \leftarrow tree[last].next$
3 return $tree[last].base$

The value of $last$ represents the closest intersection point before n , and is always smaller than n . $last$ is initialized to 0 by INIT and is updated at each iteration.

We can now calculate the complexity of DYNAMIC-OP-MP.

Theorem 3: Algorithm DYNAMIC-OP-MP solves Problem OP-MP with complexity $O(|E| D \log D)$.

Proof: By Lemmas 2 and 3, we conclude that we can maintain a data structure for $base(n, u, v)$, that can be updated in $O(\log D)$. Clearly, the INIT procedure can be done in $O(|E|)$ for all links. Each iteration calls UPDATE and GET once for each link in $O(|E|(\log D + 1))$. There are D iterations, and we get a total complexity of $O(|E| D \log D)$, as claimed. ■

VI. UNIFORM DISTRIBUTION

A. Relation to Problem OP

Our solution to Problem OP-MP uses the fact that the delay is partitioned, to establish an optimal substructure property, which allows the use of dynamic programming. However, it does not make use of the partition strategy itself. The dynamic programming algorithm adds at each iteration a single link to an optimally partitioned path. It is obvious that the delay allocated to that link may result in a non-optimal partition of the whole path. This means that only a specific delay may be allocated to the added link. Thus, we can perform an UPDATE at each iteration in $O(1)$.

For uniform distributions, we can efficiently solve Problem OP, and we can use this result in the solution of OP-MP. By Theorem 1, the optimal partition is $S_D(\mathbf{p}) = \{t_l + \min(\tau, \delta_l)\}_{l \in \mathcal{P}}$, where τ is a delay s.t.

$$\sum_{l \in \mathcal{P}} t_l + \sum_{l \in \mathcal{P}} \min(\tau, \delta_l) = D.$$

The delay τ is a common property for all links along the optimal path (with optimal partition). This implies that any optimal sub-path should be continued only with the same τ . The initial value of τ is determined by the delay allocated to the first link of each path, hence we must consider all possible delay allocations for any links outgoing from s .

B. The Algorithm

The dynamic programming algorithm inspects all possible delay allocations to any link outgoing from s , hence all possible values of τ are considered. We might expect a complexity of $O(|E| D)$ for the solution, however this is not the case. The problem arises when τ is greater than δ for a link outgoing from s . In any optimal solution, the maximal delay allocated to a

link is δ , hence an allocation of δ may correspond to different choices of τ . This problem can be circumnavigated if we assume $\tau \leq \delta_l$, for all $l \in E$,⁷ in which case the optimal partition becomes $S_D(\mathbf{p}) = \{t_l + \tau\}_{l \in \mathcal{P}}$.

The above discussion is summarized in the following algorithm, in which we have added the variables $\tau(n, v)$ and $\theta(n, u, v)$, which store τ , for the corresponding optimal (sub-)paths.

Algorithm UNIFORM-OP-MP: $(G, \{U(t_l, t_l + \delta_l)\}_{l \in E}, D)$
1 $C(0, v) \leftarrow \infty \quad \forall v \neq s$
2 $C(0, s) \leftarrow 0$
3 $\Theta(v) \leftarrow \{u : (u, v) \in E\}$, $\theta(v, 0) \leftarrow \text{NIL} \quad \forall v \in V$
4 $n \leftarrow 0$
5 Repeat until $n \geq D$
6 $n \leftarrow n + 1$
7 for all $l = (s, v) \in E$ do
8 $C(n, s, v) \leftarrow c_l(n)$ (could be ∞ , for $n < t_l$)
9 $\tau(n, s, v) \leftarrow n - t_l$ (may be negative)
10 for all $v \in V$ do
11 $C(n, v) \leftarrow \min_{\Theta(v)} C(n, u, v)$ ⁸
12 $\theta(n, v) \leftarrow \arg \min_{\Theta(v)} C(n, u, v)$
13 $\tau(n, v) \leftarrow \tau(n, \theta(n, v), v)$
14 for all $l = (u, v) \in E$ do
15 $d_l \leftarrow t_l + \tau(n, u)$ ⁹
16 $C(n + d_l, u, v) \leftarrow C(n, u) + c_l(d_l)$ ¹⁰
17 $\tau(n + d_l, u, v) \leftarrow \tau(n, u)$

Lines 14-17 are equivalent to the call to UPDATE in Algorithm DYNAMIC-OP-MP. We continue each optimal path from s to u by allocating to the next hop a delay that corresponds to $\tau(n, u)$.

Lines 7-9 determine the initial τ for each path. All possibilities for τ are considered after D iterations. The actual delay allocated to each link can be obtained from $\tau(n, v)$.

Theorem 4: Let $f_l(d)$ be a uniform distribution, $f_l = U(t_l, t_l + \delta_l)$, then Algorithm UNIFORM-OP-MP solves problem OP-MP, with a complexity of $O(|E| D)$.

Proof: By the previous discussion, each optimal sub-path can continue only with the same τ . Furthermore, τ is determined by the first link of each optimal sub-path, and can be any delay $1 \leq \tau \leq D$. This initial τ is considered, at each iteration, for all possible first links, in $O(|E|)$. All iterations consider all D possibilities for τ , for all those links. We get an overall complexity of $O((|E| + |V| + |E|) D) = O(|E| D)$. ■

C. Generalized Algorithm

In the following we relax the (unreasonable) assumption $\tau \leq \delta_l$ for all $l \in E$. If $\tau > \delta_l$ for some link, then the cost on that link is zero ($f_l(t_l + \delta_l) = 1$), however we must allocate delay to the link. When τ is already known, we can simply allocate $t_l + \min(\delta_l, \tau)$ for the next hop, by changing Line 15 of Algorithm UNIFORM-OP-MP. However, when such a link is the first link in the path, we must determine τ . We should consider all possible values for τ , and in the worst case this could change the complexity to $O(|E| D^2)$. To overcome this problem, we must make sure that paths do not start with such links.

⁷We shall relax this assumption later.

⁸If $C(n, u, v)$ exists, otherwise ∞ .

⁹Recall that we assume $\tau = \min(\delta_l, \tau)$.

¹⁰ $c_l(d_l) = \infty$ for $\tau(n, u) < 0$.

If we sort all links by increasing values of δ_l , i.e., $0 \equiv \delta_0 \leq \delta_1 \leq \dots \leq \delta_{|E|}$, then we can solve the problem $|E|$ times, where at iteration i , we assume $\delta_{i-1} \leq \tau < \delta_i$. For all links that have $\delta_l \leq \delta_{i-1}$, we set $d_l = t_l + \delta_l$, with probability 1. We can now delete all those links from the graph and add their delay to the remaining links. For each remaining link (u, v) we add a new link (s, v) , which has the same δ as (u, v) , but with $t_{(s,v)} = t_{(u,v)} + T(u)$, where $T(u)$ is the delay allocated to the links we removed. $T(u)$ is the minimal distance on them from s to u w.r.t. $(t_l + \delta_l)$ and can be computed using a (standard) shortest path algorithm on a graph consisting of the links we removed.

```

Algorithm GEN-U-OP-MP:  $(G, \{f_l = \mathbf{U}(t_l, t_l + \delta_l)\}_{l \in E}, D)$ 
1  sort the links by increasing values of  $\delta_l$ 
2  for  $i = 1$  to  $|E|$  do
3     $\tilde{E} \leftarrow \{l \in E : l \leq i\}$ 
4     $T(V) \leftarrow \text{SHORTEST-PATH}(G(V, \tilde{E}), \{t_l + \delta_l\}_{l \in \tilde{E}})$ 
5     $\bar{E} \leftarrow \{(s, v) : (u, v) \in E \setminus \tilde{E}\}$ 
6     $f_{(s,v)} \leftarrow \mathbf{U}(t_{(u,v)} + T(u), t_{(u,v)} + T(u) + \delta_{(u,v)})$ 11
7     $\mathbf{p}_i \leftarrow \text{UNIFORM-OP-MP}(G(V, E \cup \bar{E}), \{f_l\}_{l \in E \cup \bar{E}}, D)$ 
8  choose the best  $\mathbf{p}_i$ 

```

Theorem 5: Algorithm GEN-U-OP-MP solves Problem OP-MP for uniform distributions, with complexity

$$O(|E|^2(|V| + D)).$$

Proof: Finding $T(V)$ requires $O(|V||E|)$.¹² In the worst case, the number of links is doubled, hence the complexity of running UNIFORM-OP-MP is still $O(|E|D)$. Thus, the complexity of each iteration is $O(|E|(|V| + D))$. We perform $|E|$ such iterations, hence the total complexity is $O(|E|^2(|V| + D))$. ■

In practice, it is useful to modify Algorithm UNIFORM-OP-MP to take advantage of the limits on τ in each iteration. We may also assume that $|V| = O(D)$, hence the complexity of the Algorithm GEN-U-OP-MP is $O(|E|^2 D)$.

VII. CONCLUSIONS

This work investigated the effects of uncertain parameters on QoS routing with end-to-end delay requirements. We have defined the routing problem and an important variant: *optimally partitioned most probable path* (OP-MP). We have discussed these problems in the context of shortest path problems and pointed out the difficulties in adapting standard shortest path algorithms to our case.

We first focused on the *optimal delay partition problem* (OP). Although Problem OP is intractable in the general case, we established an efficient and exact solution for a wide class of probability distributions, including exponential and normal distributions. The complexity of the solution is $O(|\mathbf{p}| \log(|\mathbf{p}|A))$, where $|\mathbf{p}|$ is the number of links in the path and A is bound on the variations. For uniform distributions, we presented an improved algorithm with a complexity of $O(|\mathbf{p}|)$.

Next, we considered Problem OP-MP and the above class of distributions, and established a pseudo-polynomial solution, based on dynamic programming. Our algorithm has a complex-

ity of $O(|E|D \log D)$, where D is the end-to-end delay demand and $|E|$ is the number of links in the network. This result is comparable with the complexity of $O(|E|D)$ associated with dynamic programming algorithms for the *restricted shortest path* problem (RSP). This means that changing the parameters of each link, from a specific delay with a specific probability of success to a complete probability distribution function, only adds a factor of $\log D$ to the complexity of the solution. Furthermore, we established an $O(|E|^2 D)$ solution for the case of uniform distributions.

For Problem RSP, there are fully polynomial approximation schemes (FPAS), e.g., [6], which are derived from the dynamic programming, exact (pseudo-polynomial) solution. Recently, we have been able to construct a similar FPAS scheme, that can be employed, to get fully polynomial and near-optimal solutions for Problem OP-MP. In particular, we were able to establish an $O(|E||V|\epsilon^{-1} \log(|V|\epsilon^{-1}))$ approximate solution with a cost ratio of $1 + \epsilon$ to the optimal solution. We are currently working on applying the methods presented here to other QoS requirements, and establishing improved solutions to specific probability distributions, other than uniform.

While almost any network control function has to handle some degree of uncertainty, traditional approaches, such as dealing with average values, were often enough to circumvent the problem. However, large-scale broadband networks require more sophisticated solutions. Indeed, the size and architecture of such networks increases the degree of uncertainty, while the stricter service demands complicate the impact of imprecisions. While we believe that the present study offers valuable insight towards the construction of a proper analysis and design methodology, it is clear that much is yet to be done and understood.

REFERENCES

- [1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
- [2] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. Internet Draft, November 1997. work in progress.
- [3] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.
- [4] R. Guérin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS routing mechanisms and OSPF extensions. Internet Draft, April 1997. work in progress.
- [5] R. Guérin and A. Orda. QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms. In *IEEE INFOCOM'97*, pages 75–83, Kobe, Japan, April 1997.
- [6] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, February 1992.
- [7] W. C. Lee. Topology aggregation for hierarchical routing in ATM networks. *Comput. Commun. Rev. (U.S.A.)*, 25(2):82–92, April 1995.
- [8] D. H. Lorenz and A. Orda. QoS Routing in Networks with Uncertain Parameters. Research Report EE Pub. No. 1094. Department of Electrical Engineering, Technion, Haifa, ISRAEL, June 1997. May be obtained from "ftp://ftp.technion.ac.il/pub/supported/ee/Network/lor.qosr97.ps".
- [9] Private Network-Network Interface Specification v1.0 (PNNI). ATM Forum Technical Committee, March 1996.
- [10] 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*, 2 137–150, 1994.
- [11] S. Shenker and C. Partridge and R. Guerin. Specification of Guaranteed Quality of Service, (draft-ietf-intserv-guaranteed-svc-07.txt). Internet Draft, February 1997. work in progress.
- [12] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE JSAC*, 14(7):1288–1234, September 1996.

¹¹ There may be parallel links.

¹² Using the Bellman-Ford algorithm.