An overview of constraint-based path selection algorithms for Qos routing

Kuipers, Fernando; Van Mieghem, Piet; Korkmaz, T; Krunz, M

DOI
https://doi.org/10.1109/MCOM.2002.1106159

Publication date
2002

Document Version
Accepted author manuscript

Published in
IEEE Communications Magazine

Overview of Constraint-Based Path Selection Algorithms for QoS Routing

F.A. Kuipers, Delft University of Technology
T. Korkmaz, University of Texas at San Antonio
M. Krunz, University of Arizona
P. Van Mieghem, Delft University of Technology

Abstract

Constraint-based path selection aims at identifying a path that satisfies a set of quality-of-service (QoS) constraints. In general, this problem is known to be NP-complete, leading to the proposal of many heuristic algorithms. In this paper, we provide an overview of these algorithms, focusing on *restricted shortest path* and *multi-constrained path* algorithms.

1 Introduction

In recent years, there has been an increasing demand for Internet-based multimedia applications. In response to this demand, the research community has been extensively investigating several quality-of-service (QoS) based networking frameworks (e.g., IntServ, DiffServ, MPLS). One of the key issues in all of these frameworks is how to identify efficient paths that can satisfy the given QoS constraints, or what is commonly known as the QoS-based routing problem.

In general, routing (be it QoS-based or not) involves two entities: *routing protocols* and *routing algorithms*. Routing protocols capture the network state information (e.g., available resources) and disseminate it throughout the network, while routing algorithms use this information to compute appropriate paths. While the current best-effort routing simply performs these tasks based on a single, relatively static measure, QoS routing takes into account both the applications requirements and the availability of network resources. As a result, QoS routing has to deal with some challenging issues that are not present in the best-effort routing, including scalable dissemination of dynamic (state-dependent) information, state aggregation, computation of constrained paths. In this paper, our goal is to shed some light on the myriad of existing multi-constrained path selection algorithms proposed for QoS-based *unicast* routing. In all of these algorithms, it is assumed that the network-state information is temporarily static and has been disseminated throughout the network using the underlying QoS-based routing protocols (e.g., QoS-enhanced OSPF).

Before formally defining the problem at hand, we first describe the notation used throughout this paper. Let $G(N, E)$ denote a network topology, where N is the set of nodes and E is the set of links. With a slight abuse of notation, we also use N and E to denote the number of nodes and the number of links, respectively. The source and destination nodes are denoted by s and d, respectively. The number of QoS measures (e.g., delay, hopcount) is denoted by m. Each link is characterized by an

m-dimensional link weight vector, consisting of m non-negative QoS weights ($w_i(u,v)$, $i = 1, ..., m$, $(u,v) \in E$) as components. QoS measures can be roughly classified into *additive* (e.g., delay) and *non-additive* (e.g., available bandwidth, policy flags). In case of an additive measure, the QoS value of a path is equal to the sum of the corresponding weights of the links along that path. For a non-additive measure, the QoS value of a path is the minimum (or maximum) link weight along that path. Constraints are denoted by L_i, $i = 1, ..., m$. In general, non-additive measures can be easily dealt with by pruning from the graph all links (and possibly disconnected nodes) that do not satisfy the requested QoS constraint. Additive measures cause more difficulties. Hence, without loss of generality, we only consider additive measures. The basic problem considered in this paper can be stated as follows:

Definition 1 *Multi-Constrained Path (MCP) problem:* Consider a network $G(N, E)$. Each link $(u,v) \in E$ is associated with m additive weights $w_i(u,v) \geq 0$, $i = 1, ..., m$. Given m constraints L_i, $i = 1, ..., m$, the problem is to find a path P from s to d such that: $w_i(P) \stackrel{def}{=} \sum_{(u,v)\in P} w_i(u,v) \leq L_i$ for $i = 1, ..., m$.

A path obeying the above condition is said to be *feasible*. Note that there may be multiple feasible paths between s and d. A modified (and more difficult) version of the MCP problem is to retrieve the shortest "length" path among the set of feasible paths. This problem is known as the *multi-constrained optimal path* (MCOP) problem, and is attained by adding a second condition on the path P in Definition 1: $l(P) \leq l(Q)$ for any *feasible* path Q between s and d, where $l(.)$ is a path length (or cost) function. A solution to the MCOP problem is also a solution to the MCP problem, but not necessarily vice versa. Considerable work in the literature has focused on a special case of the MCOP problem known as the *restricted shortest path* (RSP) problem, where the goal is to find the least-cost path among those that satisfy only one constraint denoted by D, which bounds the permissible delay of a path.

The MCP problem and its variants are known to be NP-complete [3]. Therefore, they are considered to be intractable for large networks. Accordingly, many heuristics have been proposed for these problems. In the rest of this paper, we briefly describe the lion's share of the existing algorithms. To simplify the discussion, we consider them under two categories: RSP algorithms in Section 2, and MCP algorithms in Section 3. Finally, we conclude the paper in Section 4.

2 RSP Algorithms

Before presenting some of the efficient solutions for the RSP problem, we start by discussing its exact (but computationally more strenuous) solutions.

2.1 Exact Algorithms

An exact solution to the RSP problem can be found by systematically examining every path between s and d in a brute-force manner (e.g., using depth-first search with backtracking). However, since the number of paths grows exponentially with the size of the network, this method may not be useful in practice. An alternative exact solution is known as the Constrained Bellman-Ford (CBF) algorithm. The basic idea here is to systematically discover the lowest-cost paths while monotonically increasing

their total delays. CBF maintains a list of paths from s to every other node with increasing cost and decreasing delay. It selects a node whose list contains a path that satisfies D and that has the minimum cost. CBF then explores the neighbors of this node using a breadth-first search, and (if necessary) adds new paths to the list maintained at each neighbor. This process continues as long as D is satisfied and there exists a path to be explored. Although CBF exactly solves the RSP problem, its execution time grows exponentially in the worst case.

The RSP problem can also be solved exactly via pseudo-polynomial-time algorithms. In general, the complexity of pseudo-polynomial-time algorithms depends on the actual values of the input data (e.g., link delays and the given delay constraint) as well as the size of the network [3]. Pseudo-polynomial-time algorithms incur long execution times if the value of the input data is large. This can happen if the granularity of link weights is very small.

2.2 ϵ-Optimal Approximation

One general approach to deal with NP-complete problems is to look for a polynomial-time algorithm that guarantees finding an approximate, quantifiable solution to the optimal one. An algorithm is said to be ϵ-optimal if it returns a path whose cost is at most $(1+\epsilon)$ times the cost of the optimal path, where $\epsilon > 0$. Two examples of ϵ-optimal approximation algorithms for the RSP problem were provided in [5]. The first ϵ-optimal approximation algorithm initially determines an upper bound (UB) and a lower bound (LB) on the optimal cost denoted by OPT. For this, the algorithm initially starts with $LB = 1$ and UB equals to the sum of the $(N-1)$ largest link-costs, and then it systematically adjusts these bounds using a *testing* procedure. After computing LB and UB, the algorithm bounds the cost of every link by rounding and scaling it by ϵLB. It then applies a pseudo-polynomial-time algorithm on these new weights. The second version is basically an extension of the first one, and uses a technique called *interval partitioning*. The performance of approximation algorithms improves with smaller values of ϵ while causing an increased computational complexity.

2.3 Backward-Forward Heuristic

In *backward-forward* algorithms, the graph is explored based on the concatenation of two segments: (1) the so-far explored path from s to an intermediate node u, and (2) the least-delay or the least-cost path from node u to d. These algorithms are implemented in a centralized or distributed manner. In a distributed implementation, probing and backtracking are used, as follows. The algorithm sends a probe packet over the preferred links one at a time. If the receiving node accepts the probe packet, it forwards it to the next node. Otherwise, if the packet is rejected, the algorithm tries the next preferred link. In a centralized implementation, a backward-forward heuristic (BFH) can be implemented as follows: first determine the least-delay path (LDP) and the least-cost path (LCP) from every node u to d. BFH then starts from s and explores the graph as in Dijkstra's algorithm with the following modification in the relaxation procedure: a link (u, v) is relaxed if it reduces the total cost from s to v, while its approximated end-to-end delay obeys the delay constraint. The latter is obtained from the path up to node v and the LDP from v to d. BFH extracts the next node that has the minimum cost from the heap. The computational complexity of the centralized BFH is three times that of Dijkstra's algorithm.

2.4 Lagrangian-Based Linear Composition

In the Lagrangian-based composition algorithms, the graph is searched based on a single weight that is obtained by linearly combining the delay and cost of each link. Two multipliers, for delay and cost, are used to obtain different linear combinations. A key issue here is how to determine appropriate values for the multipliers. This can be done systematically by iteratively finding the shortest path with respect to (w.r.t.) the linear combination and adjusting the multipliers' values in the direction of the optimal solution. This technique is similar to the well-known Lagrangian relaxation technique used in other constrained optimization problems. It can be shown that if the weights of the paths are uniformly distributed in the delay-cost space, then the search terminates after a finite number of iterations of Dijkstra's algorithm. Several refinements have been proposed to the basic Lagrangian-based composition approach. For example, one can use the k-shortest path algorithm[1] to close the gap between the optimal solution and the returned path based on the linear combination.

2.5 Hybrid Algorithms

It is also possible to devise efficient heuristics for the RSP problem using combinations of the aforementioned approaches. One such heuristic has been provided in [4]. According to this heuristic, the cost of the least-delay path is selected as the cost constraint. The problem is then solved by minimizing a nonlinear length function, analogous to the one used in TAMCRA (see Section 3.3), that gives more priority to lower-cost paths. To minimize the nonlinear length function, a k-shortest-path-based algorithm called DCCR is used. The performance of the DCCR algorithm depends on k; if k is large, the algorithm gives good performance at the expense of an increased execution time. In order to improve the performance with smaller values of k, the search space can be reduced by using a Lagrangian-based algorithm before applying DCCR. The complexity of this final hybrid algorithm called SSR+DCCR depends on that of the Lagrangian-based algorithm and the k-shortest path algorithm.

3 MCP Algorithms

In this section, we present a representative sample of the algorithmic solutions for the MCP problem, and in some cases, for the more difficult MCOP problem.

3.1 Jaffe's Approximation

In [6] Jaffe presented two algorithms for the MCP problem under two constraints ($m = 2$). The first is a pseudo-polynomial-time algorithm that has an unattractive worst-case complexity. The second is an approximation algorithm which is simply called Jaffe's algorithm and discussed here in detail. This algorithm first determines two positive multipliers, namely d_1 and d_2. It then uses these multipliers to assign a composite weight $w(u, v)$ to every link $(u, v) \in E$ by linearly combining the original weights. The algorithm then finds the shortest path w.r.t. w. This search process is illustrated pictorially in Figure 1, where all possible paths between the source and destination nodes are indicated by black circles. Equal-length paths w.r.t. w are indicated by a line. The search for the minimum-length

[1] A k-shortest path algorithm does not stop when the destination has been reached for the first time, but continues until it has been reached through k different paths succeeding each other in length.

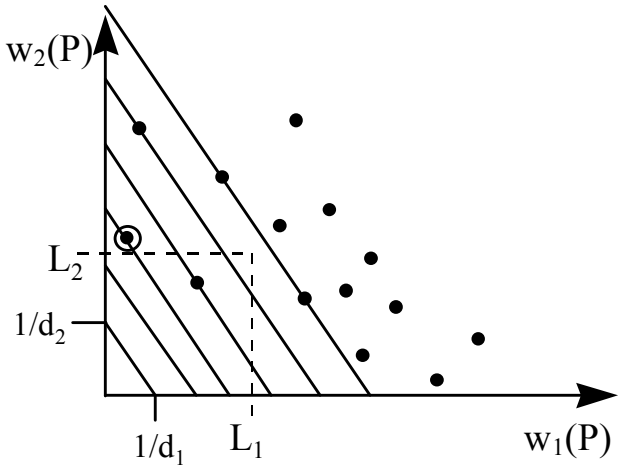

Figure 1: Pictorial representation of the search process in Jaffe's algorithm.

path is equivalent to sliding this line outward from the origin until a path (black circle) is hit. This path is the one returned by the algorithm. Figure 1 also illustrates that the returned path does not necessarily reside within the feasibility region defined by the two constraints. Accordingly, Jaffe suggested using a nonlinear function whose minimization guarantees finding a feasible path, if such a path exists. However, no simple shortest path algorithm is available to minimize such a nonlinear function. Instead, Jaffe provided the above approximation and showed how to determine d_1 and d_2 based on this nonlinear function. Note that Jaffe's algorithm can be extended to an arbitrary number of constraints.

3.2 Fallback Algorithm

In this approach, the algorithm computes the shortest path one at a time w.r.t. individual QoS measures. If the current shortest path w.r.t. a given QoS measure satisfies all the constraints, then the algorithm stops. Otherwise, the search is repeated using another QoS measure until a feasible path is found or until all QoS measures are examined. The worst-case complexity of the algorithm is m times that of Dijkstra. One problem with the fallback approach is that there is no guarantee that optimizing path selection w.r.t. any single measure would lead to a feasible path or even one that is close to being feasible. The fallback approach performs good when the link weights are positively correlated, because then if one weight is small, the other weights are also likely to be relatively small, resulting in a path farthest from the constraints.

3.3 TAMCRA and SAMCRA

TAMCRA [2] and its exact companion SAMCRA [10] are based on three fundamental concepts: (1) a nonlinear measure for the path length, (2) the k-shortest path approach, and (3) the principle of non-dominated paths. Figure 2 explains the first concept pictorially (when $m = 2$). Part (a) depicts the search process using a *linear* composition function, similar to Jaffe's algorithm. If the two path weights are highly correlated, then the linear approach tends to perform well. However, if that is not

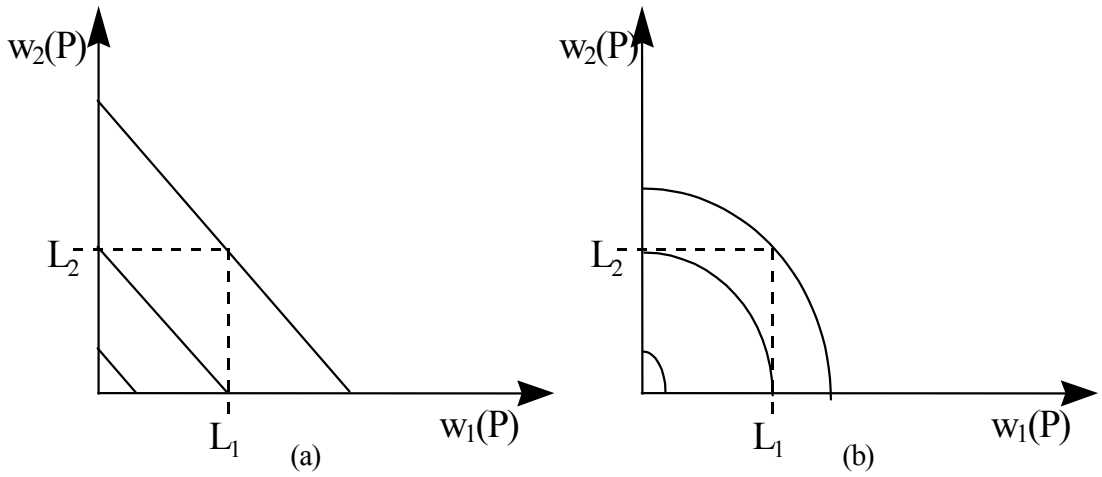

Figure 2: Searching for a feasible path by minimizing: (a) a linear composite function, (b) nonlinear composite function.

the case, then a nonlinear function is more appropriate. Part (b) of the figure depicts the search process using a nonlinear function. Ideally, the equal-length lines should perfectly match the boundaries of the constraints, scanning the constrained area without ever selecting a solution outside it. When all QoS measures are equally treated, this can be achieved by finding a path that minimizes:

$$l(P) = \max_{1 \le i \le m} \left(\frac{w_i(P)}{L_i} \right) \tag{1}$$

An important characteristic of nonlinear path-length functions such as the one in (1) is that *sub-paths of shortest paths are not necessarily shortest paths.* In the path computation, this suggests considering more paths than only the shortest one, leading us to the k-shortest path approach. In TAMCRA the k-shortest path concept is applied to intermediate nodes i on the path from s to d. While traversing the graph, the algorithm keeps track of multiple sub-paths from s to i. Not all sub-paths are stored, but an efficient distinction is made based on the non-dominance of a path. We say that a path Q is dominated by a path P if $w_i(P) \le w_i(Q)$ for all $i = 1, .., m$, with an inequality for at least one i. TAMCRA only considers non-dominated (sub)-paths. This property allows it to efficiently reduce the search space without compromising the solution in finding a feasible paths. Any path P that satisfies $l(P) \le 1$ is a feasible path, and hence is an acceptable solution to the MCP problem. However, this path may not be optimal in terms of its length. In addition to feasibility, SAMCRA address this issue (i.e., provides a solution to the more difficult MCOP problem) by allowing different length functions.

As an exact algorithm, SAMCRA guarantees finding a feasible path, if one exists. Furthermore, it allocates buffer space only when truly needed, and it self-adaptively adjusts the number of stored paths k at each node. Unfortunately, in the worst case, this could lead to an exponentially growing k. In TAMCRA k is predefined and fixed, so its worst-case complexity is polynomial.

3.4 Chen's Approximate Algorithm

Chen and Nahrstedt [1] provided an approximate algorithm for the MCP problem. This algorithm first transforms the MCP problem into a simpler one by scaling down $m-1$ (real-valued) link weights into integer weights, as follows: $w_i^*(u,v) = \left\lceil \frac{w_i(u,v)\cdot x_i}{L_i} \right\rceil$ for $i = 2, 3, \ldots, m$, where the x_i's are predefined positive integers. The simplified problem reduces to finding a path P that minimizes the first (real) weight provided that the other $m-1$ scaled down (integer) weights are within the stricter constraints x_i. To exactly solve this simplified MCP problem, Chen and Nahrstedt proposed two algorithms based on dynamic programming: the Extended Dijkstra's Shortest Path algorithm (EDSP) and the Extended Bellman-Ford algorithm (EBF). When the graph is sparse and the number of nodes is relatively large, EBF is expected to give better performance than EDSP in terms of execution time. However, to achieve good performance, high x_i's are needed, which makes this approach rather computationally intensive for practical purposes.

3.5 Randomized Algorithm

The concept behind randomization is to make random decisions during the execution of an algorithm so that unforeseen traps can potentially be avoided when searching for a feasible path. Using this approach, Korkmaz and Krunz [8] proposed a randomized algorithm for the MCP problem. The algorithm consists of two parts: (a) initialization phase, (b) randomized search. In the initialization phase, the algorithm computes the shortest paths from every node u to d w.r.t. each link weight and w.r.t. a linear combination of all weights. The algorithm then starts from s and explores the graph using a randomized breadth-first search (BFS). In contrast to the conventional BFS, which systematically discovers every node that is reachable from s, the randomized BFS discovers nodes from which there is a good chance to reach d. By using the information obtained in the initialization phase, the randomized BFS can check whether this chance exists before discovering a node. If there is no chance, the algorithm foresees the trap and does not consider such nodes any further. Otherwise, it continues searching by randomly selecting discovered nodes until d is reached. If the first attempt of randomized BFS fails, the search can be repeated again. Because of the nature of randomization, subsequent attempts may succeed in returning a feasible path. The worst-case complexity of the randomized algorithm is $m+1$ times that of Dijkstra.

3.6 H_MCOP

Korkmaz and Krunz [7] also provided a heuristic algorithm called H_MCOP. This heuristic attempts to find a feasible path for any number of constraints while simultaneously minimizing a path length function. The search for a feasible path is done by approximating the nonlinear function (1), which is also used in TAMCRA. To achieve its objectives, H_MCOP executes two modified versions of Dijkstra's algorithm in backward and forward directions. In the backward direction, H_MCOP computes the shortest paths from every node to d w.r.t. a linear combination of all weights. Later on, these (reverse) paths are used to estimate how suitable the remaining sub-paths are. In the forward direction, using Look_Ahead_Dijkstra, H_MCOP starts from s and discovers every node u based on a path P, where P is a heuristically determined complete s-d path that is obtained by concatenating the already traveled sub-path from s to u and the estimated remaining sub-path from u to d. Since the algorithm considers

complete paths before reaching d, it can foresee some feasible paths during the search. If paths seem feasible, then the algorithm can switch to explore these feasible paths for the one with minimum length. If H_MCOP is used for the MCP problem only, then the execution can be stopped once a feasible path is found or foreseen, reducing the execution time.

3.7 Limited Path Heuristic

Yuan [11] presented two heuristics for the MCP problem; the "limited granularity" heuristic and the "limited path" heuristic (LPH). LPH has a very high probability of finding a feasible path, provided that such a path exists. LPH (which is better than the limited granularity, particularly when $m > 3$) is based on the Bellman-Ford algorithm and uses two of the fundamental concepts in TAMCRA, namely non-dominance and storing at most k paths per node. However, while TAMCRA uses a k-shortest path approach, LPH stores the first k (not necessarily the shortest) paths. Moreover, LPH does not check whether a sub-path obeys the constraints; it only does this when d is reached.

3.8 A*Prune

Liu and Ramakrishnan [9] considered the problem of finding not only one but multiple (K) shortest paths that are within the constraints. The linear length function used is the same as that of Jaffe's algorithm. The authors proposed an exact algorithm called A*Prune. If there are no K feasible paths present, the algorithm will only return those that are within the constraints.

For each QoS measure, A*Prune calculates the shortest paths from s to all nodes $i \in N\backslash\{s\}$ and from d to all $i \in N\backslash\{d\}$. The weights of these paths will be used to evaluate whether a certain sub-path can indeed become a feasible path (similar look-ahead features were also used in the H_MCOP algorithm). After this initialization phase, the algorithm proceeds in a Dijkstra-like fashion. The node with the shortest predicted end-to-end length is extracted from a heap and then all of its neighbors are examined. The neighbors that cause a loop or lead to a violation of the constraints are pruned. The A*Prune algorithm continues extracting/pruning nodes until K constrained shortest paths from s to d are found or until the heap is empty. The worst-case complexity of A*Prune grows exponentially with the size of the network. It is possible to implement a Bounded A*Prune algorithm, which runs polynomial in time at the risk of losing exactness.

4 Conclusions

We provided a high-level overview of the main solutions available in the literature for constraint-based path selection. Naturally, these solutions provide different trade-offs between computational complexity and accuracy. An important property of multidimensional routing is that a nonlinear length function is required to obtain exact results. QoS routing algorithms that use a linear definition for the path length will only prove useful when the link weights are positively correlated. In all other cases a nonlinear function is necessary, which significantly complicates the problem, since no simple shortest path algorithm is available to minimize such a nonlinear function. As a consequence, multiple paths must be evaluated, requiring the use of a k-shortest path algorithm. The other important techniques are non-dominance, look-ahead, search-space reducing, rounding and scaling the weights,

and the constraint values themselves. Depending on the availability of resources, these techniques allow for devising efficient tailor-made QoS algorithms.

Acknowledgments

The work of M. Krunz was supported by the National Science Foundation under grants ANI 9733143, CCR 9979310, and ANI 0095626.

References

[1] S. Chen and K. Nahrstedt. On finding multi-constrained paths. In *Proceedings of the ICC '98 Conference*, volume 2, pages 874–879. IEEE, 1998.

[2] H. De Neve and P. Van Mieghem. TAMCRA: A tunable accuracy multiple constraints routing algorithm. *Computer Communications*, 23:667–679, 2000.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[4] L. Guo and I. Matta. Search space reduction in QoS routing. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, pages 142 – 149. IEEE, May 1999.

[5] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[6] J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[7] T. Korkmaz and M. Krunz. Multi-constrained optimal path selection. In *Proceedings of the INFOCOM 2001 Conference*, volume 2, pages 834–843, Anchorage, Alaska, April 2001. IEEE.

[8] T. Korkmaz and M. Krunz. A randomized algorithm for finding a path subject to multiple QoS constraints. *Computer Networks*, 36(2-3):251–268, 2001.

[9] G. Liu and K.G. Ramakrishnan. A*Prune: an algorithm for finding k shortest paths subject to multiple constraints. In *Proceedings of the INFOCOM 2001 Conference*, pages 743–749, Anchorage, Alaska, April 2001. IEEE.

[10] P. Van Mieghem, H. De Neve and F.A. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3-4):407–423, October 2001.

[11] X. Yuan. Heuristic algorithms for multiconstrained quality-of-service routing. *IEEE/ACM Transactions on Networking*, 10(2):244–256, 2002.

Biographies

Fernando Kuipers (F.A.Kuipers@its.tudelft.nl): received the M.Sc. degree in Electrical Engineering at the Delft University of Technology in 2000. Currently he is working towards his Ph.D. degree in the Network Architectures and Services group at the same faculty. He is also a member of the DIOC (Delft interdisciplinary research center) for the Design and Management of Infrastructures, where he is taking part in the Telecommunications project. His Ph.D. work focuses on the algorithmic aspects and complexity of unicast Quality of Service routing.

Turgay Korkmaz (korkmaz@cs.utsa.edu): received his Ph.D. degree from Electrical and Computer Engineering at University of Arizona in December 2001. He is currently an Assistant Professor in Computer Science at the University of Texas at San Antonio. His research interests include QoS-based routing, multi-constrained path selection, efficient dissemination of network-state information, topology aggregation, and performance evaluation of QoS-based routing protocols.

Marwan Krunz [M] (krunz@ece.arizona.edu): received his Ph.D. degree in Electrical Engineering from Michigan State University, Michigan, in 1995. From 1995 to 1997, he was a Postdoctoral Research Associate with the Department of Computer Science and the Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park. In January 1997, he joined the University of Arizona, where he is currently an Associate Professor of Electrical and Computer Engineering. Dr. Krunz's research interests lie in the field of computer networks, especially in its performance and traffic control aspects. His recent work has focused on the provisioning of quality of service (QoS) over wireless links, QoS routing, traffic modeling, bandwidth allocation, video-on-demand systems, and power-aware protocols for ad hoc networks. He has published more than 50 journal articles and refereed conference papers. Dr. Krunz is a recipient of the National Science Foundation CAREER Award (1998-2002). He currently serves on the editorial board for the IEEE/ACM Transactions on Networking, the Computer Communications Journal, and the IEEE Communications Interactive Magazine. He is a guest co-editor of a special issue on Hot Interconnects (IEEE Micro, Jan. 2002). Dr. Krunz was the Technical Program Co-chair for the 9th Hot Interconnects Symposium (Stanford University, August 2001). He has served and continues to serve on the executive and technical program committees of many international conferences. He served as a reviewer and a panelist for NSF proposals, and is a consultant for several corporations in the telecommunications industry.

Piet Van Mieghem (P.VanMieghem@its.tudelft.nl): is professor at the Delft University of Technology with a chair in telecommunication networks and chairman of the basic unit Network Architectures and Services (NAS). His main theme of the research are new Internet-like architectures for future, broadband and QoS-aware networks. Professor Van Mieghem received a Master's and Ph.D. in Electrical Engineering from the K.U. Leuven (Belgium) in 1987 and 1991, respectively. Before joining Delft, he worked at the Interuniversity Micro Electronic Center (IMEC) from 1987 to 1991. From 1992 to 1993, he was a visiting scientist at MIT in the department of Electrical Engineering. During 1993 to 1998, he was a member of the Alcatel Corporate Research Center in Antwerp where he was engaged in performance analysis of ATM systems and in network architectural concepts of both ATM networks (PNNI) and the Internet.