Quality of Service Routing

P. Van Mieghem (Ed.), F.A. Kuipers, T. Korkmaz, M. Krunz,
M. Curado, E. Monteiro, X. Masip-Bruin, J. Solé-Pareta and S. Sánchez-López.

No Institute Given

Abstract. Constraint-based routing is an invaluable part of a full-fledged Quality of Service architecture. Unfortunately, QoS routing with multiple additive constraints is known to be a NP-complete problem. Hence, accurate constraint-based routing algorithms with a fast running time are scarce, perhaps even non-existent. The expected impact of such an efficient constraint-based routing algorithm has resulted in the proposal of numerous heuristics and a few exact QoS algorithms.
This chapter presents a thorough, concise and fair evaluation of the most important multi-constrained path selection algorithms known today. A performance evaluation of these algorithms is presented based on a complexity analysis and simulation results. Besides the routing algorithm, dynamic aspects of QoS routing are discussed: how to cope with incomplete or inaccurate topology information and (in)stability issues.

1 Introduction

The continuous demand for using multimedia applications over the Internet has triggered a spur of research on how to satisfy the Quality of Service (QoS) requirements of these applications, e.g. requirements regarding bandwidth, delay, jitter, packet loss and reliability. These efforts resulted in the proposals of several QoS-based frameworks, such as Integrated Services (Intserv) [11], Differentiated Services (Diffserv) [10] and Multi-Protocol Label Switching (MPLS) [73]. One of the key issues in providing QoS guarantees is *how to determine paths that satisfy QoS constraints*. Solving this problem is known as QoS routing or constraint-based routing.

The research community has extensively studied the QoS routing problem, resulting in many QoS routing algorithms. In this chapter, we provide an overview and performance evaluation for unicast[1] QoS routing algorithms, which try to find a path between a *source* node and a *destination* node that satisfies a set of constraints.

Routing in general involves two entities, namely the *routing protocol* and the *routing algorithm*. The *routing protocol* manages the dynamics of the routing process: capturing the state of the network and its available network resources and distributing this information throughout the network. The *routing algorithm*

[1] Multicast QoS routing faces different conceptual problems as discussed in [48]. An overview of several multicast QoS algorithms has been given in [75] and more recently in [86].

uses this information to compute paths that optimize a criterion and/or obey constraints. Current best-effort routing consists of shortest path routing that optimizes the sum over the constituent links of a single measure like hopcount or delay. QoS routing takes into account multiple QoS requirements, link dynamics, as well as the implication of the selected routes on network utilization, turning QoS routing into a notoriously challenging problem. Despite its difficulty, we argue that QoS routing is invaluable in a network architecture that needs to satisfy traffic and service requirements. For example, in the context of ATM (PNNI), QoS routing is performed by source nodes to determine suitable paths for connection requests. These connection requests specify QoS constraints that the path must obey. Since ATM is a connection-oriented technology, a path selected by PNNI will remain in use for a potentially long period of time. It is therefore important to choose a path with care. The IntServ/RSVP framework is also able to guarantee some specific QoS constraints. However, this framework relies on the underlying IP routing table to reserve its resources. As long as this routing table is not QoS-aware, paths may be assigned that cannot guarantee the constraints, which will result in blocking. In MPLS, which is a convergence of several efforts aimed at combining the best features of IP and ATM, a source node selects a path, possibly subject to QoS constraints, and uses a signaling protocol (e.g. RSVP or CR-LDP) to reserve resources along that path. In the case of DiffServ, QoS-based routes can be requested, for example, by network administrators for traffic engineering purposes. Such routes can be used conform to a certain service level agreement [91]. These examples all indicate the importance of constraint-based routing algorithms, both in ATM and IP. Depending on the frequency at which constrained paths are requested, the computational complexity of finding a path subject to multiple constraints is often a complicating but decisive factor.

To enable QoS routing, it is necessary to implement state-dependent, QoS-aware networking protocols. Examples of such protocols are PNNI [7] of the ATM Forum and the QoS-enhanced OSPF protocol [5]. For the first task in routing (i.e., the representation and dissemination of network-state information), both OSPF and PNNI use link-state routing, in which every node tries to acquire a "map" of the underlying network topology and its available resources via flooding. Despite its simplicity and reliability, flooding involves unnecessary communications and causes inefficient use of resources, particularly in the context of QoS routing that requires frequent distribution of multiple, dynamic parameters, e.g., using triggered updates [3]. Designing efficient QoS routing protocols is still an open issue that needs to be investigated further. Hereafter in Sections 2 and 3, we assume that the network-state information is temporarily static and has been distributed throughout the network and is accurately maintained at each node using QoS link-state routing protocols. Once a node possesses the network-state information, it performs the second task in QoS routing, namely computing paths based on multiple QoS constraints. In this chapter, we focus on this so-called *multi-constrained path* selection problem and consider numerous

proposed algorithms. Before giving the formal definition of the *multi-constrained path* problem, we explain the notation that is used throughout this chapter.

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 number of QoS measures (e.g., delay, hopcount, ...) is denoted by m. Each link is characterized by a 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. The QoS measure of a path can either be additive (e.g., delay, jitter, the logarithm of 1 minus the probability of packet loss), in which case the weight of that measure equals the sum of the QoS weights of the links defining that path. Or the weight of a QoS measure of a path can be the minimum(maximum) of the QoS weights along the path (e.g., available bandwidth and policy flags). Constraints on min(max) QoS measures can easily be treated by omitting all links (and possibly disconnected nodes) which do not satisfy the requested min(max) QoS constraints. We call this topology filtering. In contrast, constraints on additive QoS measures cause more difficulties. Hence, without loss of generality, we assume all QoS measures to be additive.

The basic problem considered in this chapter can be defined as follows:

Definition 1 *Multi-Constrained Path (MCP) problem:* Consider a network $G(N, E)$. Each link $(u, v) \in E$ is specified by a link weight vector with as components m additive QoS 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 a source node s to a destination node 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 that satisfies all m constraints is often referred to as a feasible path. There may be multiple different paths in the graph $G(N, E)$ that satisfy the constraints. According to **Definition 1**, any of these paths is a solution to the MCP problem. However, it might be desirable to retrieve the path with smallest length $l(P)$ from the set of feasible paths. This problem is called the *multi-constrained optimal path* problem and is formally defined as follows:

Definition 2 *Multi-Constrained Optimal Path (MCOP) problem:* Consider a network $G(N, E)$. Each link $(u, v) \in E$ is specified by a link weight vector with as components m additive QoS 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 a source node s to a destination node d such that:
(i) $w_i(P) \stackrel{def}{=} \sum_{(u,v)\in P} w_i(u, v) \leq L_i$ for $i = 1, ..., m$
(ii) $l(P) \leq l(P^*)$, $\forall P^*, P$ satisfying (i)

where $l(P)$ can be any function of the weights $w_i(P)$, $i = 1, ..., m$, provided it obeys the criteria for "length" or "distance" in vector algebra (see [80], Appendix A). Minimizing a properly chosen length function, can result in an efficient use of the network resources and/or result in a reduction of monetary cost.

In general, MCP, irrespective of path optimization, is known to be a NP-complete problem [24]. Because MCP and MCOP are NP-complete, they are considered to be intractable for large networks. Accordingly, mainly heuristics have been proposed for these problems. In Section 2, the lion's share of the published QoS algorithms is briefly described and compared based on extensive simulations. Complexity will be an important criterion for evaluating the algorithms. Complexity refers to the intrinsic minimum amount of resources needed to solve a problem or execute an algorithm. Complexity can be divided into *time* complexity and *space* complexity, but only the worst-case *computational time-complexity* and the execution time is here considered. There can be a significant difference between these complexities. Kuipers and Van Mieghem [47] demonstrate that, under certain conditions and on average, the MCP problem can be solved in polynomial time despite its worst-case NP-complete complexity. Moreover, there exist specific classes of graphs, for which the MCP problem is not NP-complete at all, e.g. if each node has only two neighbors [49].

This chapter follows the two parts structure of routing: the first two sections concentrate on the routing algorithm, while the remaining sections emphasize the routing dynamics. In Section 2 we present an overview of the most important MCP algorithms. Section 3 continues with a performance evaluation of the algorithms listed in Section 2 and based on the simulation results, deduces the fundamental concepts involved in QoS routing. The origins of incomplete or inaccurate topology state information are explained in Section 4. Section 5 provides an overview for QoS protocols and Section 6 treats stability of QoS routing. Finally, Section 7 concludes and lists open issues.

2 Overview of MC(O)P Algorithms

2.1 Jaffe's Approximate Algorithm

Jaffe [33] has presented two MCP algorithms. The first is an exact pseudo-polynomial-time algorithm with a worst-case complexity of $O(N^5 b \log Nb)$, where b is the largest weight in the graph. Because of this prohibitive complexity, only the second algorithm, coined further as Jaffe's algorithm, is discussed. Jaffe proposes to use a shortest path algorithm on a linear combination of the two link weights,

$$w(u,v) = d_1 \cdot w_1(u,v) + d_2 \cdot w_2(u,v) \tag{1}$$

where d_1 and d_2 are positive multipliers.

Each line in Figure 1 shows equilength paths with respect to (w.r.t.) the linear length definition (1). Jaffe's algorithm searches the path weight space along parallel lines specified by $w(P) = c$. As soon as this line hits a path represented by the encircled black dot, the algorithm returns this path as the shortest w.r.t. the linear length definition (1). Figure 1 also illustrates that the shortest path based on a linear combination of link weights does not necessarily reside within the constraints. Jaffe had also observed this fact and he therefore provided the following nonlinear definition for the path length

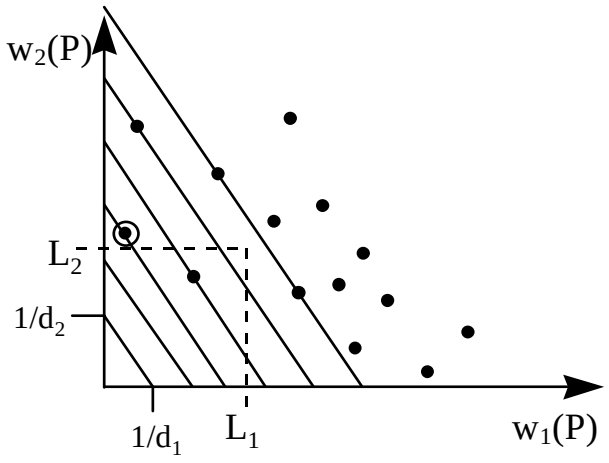

Fig. 1. Representation of the link weight vector $\boldsymbol{w}(P)$ of paths in two dimensions. Jaffe's scanning procedure first encounters the encircled node, which is the path with minimal length.

$f(P) = \max\{w_1(P), L_1\} + \max\{w_2(P), L_2\}$, whose minimization can guarantee to find a feasible path if such a path exists. However, because no simple shortest path algorithm can cope with this nonlinear length function, Jaffe approximates the nonlinear length by the linear length function (1). Andrew and Kusuma [1] generalized Jaffe's analysis to an arbitrary number of constraints m, by extending the linear length function to

$$l(P) = \sum_{i=1}^{m} d_i w_i(P) \tag{2}$$

and the nonlinear function to

$$f(P) = \sum_{i=1}^{m} \max\left(w_i(P),\, L_i\right)$$

For the simulations in Section 3 we have used $d_i = \frac{1}{L_i}$ which maximizes the volume of the solution space that can be scanned by linear equilength lines (2) subject to $w_i(P) \leq L_i$. Furthermore, we have used Dijkstra's algorithm with Fibonacci heaps, leading to a complexity for Jaffe's algorithm of $O(N \log N + mE)$.

If the returned path is not feasible, then Jaffe's algorithm stops, although the search could be continued by using different values for d_i, which might result in a feasible path. Unfortunately, in some cases, even if all possible combinations of d_i are exhausted, a feasible path may not be found using linear search. As shown in [80], an exact algorithm necessarily must use a nonlinear length function, even

though a nonlinear function cannot be minimized with a simple shortest path algorithm.

2.2 Iwata's Algorithm

Iwata *et al.* [32] proposed a polynomial-time algorithm to solve the MCP problem. The algorithm first computes one (or more) shortest path(s) based on one QoS measure and then checks if all the constraints are met. If this is not the case, the procedure is repeated with another measure until a feasible path is found or all QoS measures are examined. A similar approach has been proposed by Lee *et al.* [51]. In the simulations we only evaluate Iwata's algorithm [32].

The problem with Iwata's algorithm is that there is no guarantee that any of the shortest paths for each measure individually is close to a path within the constraints. This is illustrated in Figure 2, which shows the twenty shortest paths of a two-constraint problem applied to a random graph with 100 nodes. Only the second and third shortest path for measure 1 and the second and fourth shortest path for measure 2 lie within the constraints.

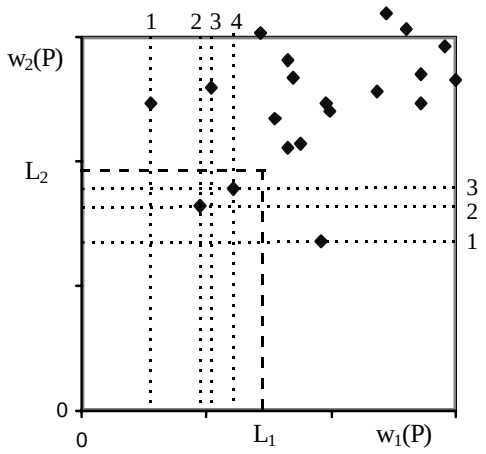

Fig. 2. Twenty shortest paths for a two-constraint problem. Each path is represented as a dot and the coordinates of each dot are its path-length for each measure individually.

In our simulations we will only consider one shortest path per QoS measure computed via Dijkstra's algorithm, leading to a complexity of $O(mN \log N + mE)$.

2.3 SAMCRA: A Self-Adaptive Multiple Constraints Routing Algorithm

SAMCRA [80] is the exact successor of TAMCRA, a Tunable Accuracy Multiple Constraints Routing Algorithm [20, 19]. TAMCRA and SAMCRA are based on three fundamental concepts: (1) a nonlinear measure for the path length, (2) a k-shortest path approach [17] and (3) the principle of non-dominated paths [30]:

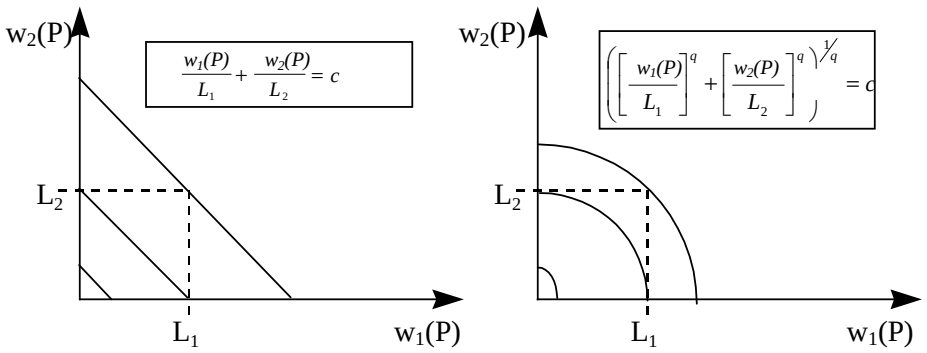

Fig. 3. Scanning procedure with (a) straight equilength lines. (b) curved equilength lines.

1. Figure 3 illustrates that the curved equilength lines of a nonlinear length function scan the constraints area in a more efficient way than the linear equilength lines of linear length definitions. The formula in Figure 3b is derived from Holder's q-vector norm [25]. Ideally, the equilength lines should perfectly match the boundaries of the constraints. Scanning the constraint area without ever selecting a solution outside the constraint area is only achieved when $q \to \infty$. Motivated by the geometry of the constraints surface in m-dimensional space, the length of a path P is defined, equivalent to Holder's q-vector norm with $q \to \infty$, as follows [20]:

$$l(P) = \max_{1 \leq i \leq m} \left(\frac{w_i(P)}{L_i} \right) \tag{3}$$

where $w_i(P) = \sum_{(u,v) \in P} w_i(u,v)$.
A solution to the MCP problem is a path whose weights are all within the constraints: $l(P) \leq 1$. Depending on the specifics of a constrained optimization problem, SAMCRA can be used with different length functions, provided they obey the criteria for length in vector algebra. Example length functions are given in [80]. The length function (3) treats all QoS measures as equally important. An important corollary of a nonlinear path length as (3) is that

the subsections of shortest paths in multiple dimensions are not necessarily shortest paths. This suggests to consider in the computation more paths than only the shortest one, leading to the k-shortest path approach.

2. The k-shortest path algorithm as presented in [17] is essentially Dijkstra's algorithm that does not stop when the destination is reached, but continues until the destination has been reached by the shortest path, the second shortest, ... , k-th shortest path. In SAMCRA the k-shortest path concept is applied to the intermediate nodes i on the path from source node s to destination node d, to keep track of multiple sub-paths from s to i. Not all sub-paths are stored, but the search-space is reduced by applying the principle of non-dominance.
3. The principle of non-dominance is the third concept in SAMCRA. A path Q is dominated by a path P if $w_i(P) \leq w_i(Q)$ for $i = 1, .., m$, with an inequality for at least one i. SAMCRA only considers non-dominated (sub)-paths. This property allows to efficiently reduce the search-space without compromising the solution. "Dominance" can be regarded as a multidimensional relaxation. The latter is a key fundament of single parameter shortest path algorithms (such as Dijkstra and Bellman-Ford).

SAMCRA and TAMCRA have a worst-case complexity of

$$O(kN \log(kN) + k^2 mE)$$

For TAMCRA the number k of paths considered during execution is fixed and hence the complexity is polynomial, while SAMCRA self-adaptively controls this k, which can grow exponentially in the worst case. Knowledge about k is crucial to the complexity of SAMCRA. One upper-bound for k is $k_{\max} = \lfloor e(N-2)! \rfloor$, which is an upper-bound on the total number of paths between a source and destination in $G(N, E)$ [81]. If the constraints/measures have a finite granularity, another upper-bound applies

$$k_{\max} = \min\left(\frac{\prod_{i=1}^{m} L_i}{\max_j(L_j)}, \lfloor e(N-2)! \rfloor\right)$$

where the constraints L_i are expressed as an integer number of a basic unit.

The self-adaptivity in k makes SAMCRA an exact MCOP algorithm: SAMCRA guarantees to find the shortest path within the constraints provided such a path exists. In this process, SAMCRA only allocates queue-space when truly needed and self-adaptively adjusts the number of stored paths k in each node. In TAMCRA the allocated queue-space is predefined via k. During the simulations with TAMCRA we chose $k = 2$, because this small value for k already produces good results. Of course a better performance is achieved when k is increased. Simulation results for different values for k can be found in [20].

2.4 Chen's Approximate Algorithm

Chen and Nahrstedt [12] provided an approximate algorithm for the MCP problem. This algorithm first transforms the MCP problem into a simpler problem

by scaling down $m-1$ (real) link weights to integer weights as follows,

$$w_i^*(u,v) = \left\lceil \frac{w_i(u,v) \cdot x_i}{L_i} \right\rceil \text{ for } i = 2, 3, \ldots, m,$$

where x_i are predefined positive integers. The simplified problem consists of finding a path P for which $w_1(P) \leq L_1$ and $w_i^*(P) \leq x_i$, $2 \leq i \leq m$. A solution to this simplified problem is also a solution to the original MCP problem, but not vice versa, because the conditions of the simplified problem are more strict. Since the simplified problem can be solved exactly, Chen and Nahrstedt have shown that *the MCP problem can be solved exact in polynomial time, when at least* $m-1$ *QoS measures have bounded integer weights.*

To solve the 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). The algorithms return a path that minimizes the first (real) weight provided that the other $m-1$ (integer) weights are within the constraints. The EBF algorithm is expected to give the better performance in terms of execution time when the graph is sparse and the number of nodes relatively large. We have chosen to implement the EBF version for our simulations.

The complexities of EDSP and EBF are $O(x_2^2 \cdots x_m^2 N^2)$ and $O(x_2 \cdots x_m NE)$, respectively. To achieve a good performance, high x_i's are needed, which makes this approach rather computationally intensive for practical purposes. By adopting the concept of non-dominance, like in SAMCRA, this algorithm could[2] reduce its search-space, resulting in a faster execution time.

2.5 Randomized Algorithm

Korkmaz and Krunz [45] have proposed a randomized heuristic for the MCP problem. The concept behind randomization is to make random decisions during the execution of an algorithm [62] so that unforeseen traps can potentially be avoided when searching for a feasible path. The proposed randomized algorithm is divided into two parts: the initialization phase and the randomized search. In the initialization phase, the algorithm computes the shortest paths from every node u to the destination node d w.r.t. each QoS measure and the linear combination of all m measures. This information will provide lower bounds for the path weight vectors of the paths from u to d. Based on the information obtained in the initialization phase, the algorithm can decide whether there is a chance of finding a feasible path or not. If so, the algorithm starts from the source node s and explores the graph using a randomized breadth-first search (BFS). In contrast to conventional BFS, which systematically discovers every node that is reachable from a source node s, the randomized BFS discovers nodes from which there is a good chance to reach a destination node d. By using the information obtained in the initialization phase, the randomized BFS can check whether this chance

[2] In Section 3 we have simulated all algorithms in their original form, without any possible improvements.

exists before discovering a node. If there is no chance, the algorithm can foresee the trap and does not explore such nodes further. We will refer to this search-space reducing technique as the look-ahead property. The look-ahead property is twofold: (1) the lower bound vectors obtained in the initialization phase are used to check whether a subpath from s to u can become a feasible path. This is a search-space reducing technique. (2) A different preference rule to extract nodes can be adopted, based on the predicted end-to-end length, i.e. the length of the subpath weight vector plus the lower bound vector. The randomized BFS continues searching by randomly selecting discovered nodes until the destination node is reached. If the randomized BFS fails in the first attempt, it is possible to execute only the randomized BFS again so that the probability of finding a feasible path can be increased.

Under the same network conditions, multiple executions of the randomized algorithm may return different paths between the same source and destination pair, providing some load-balancing. However, some applications might require the same path again. In such cases, path caching can be used [70].

The worst-case complexity of the randomized algorithm is $O(mN \log N + mE)$. For the simulations we only executed one iteration of the randomized BFS.

2.6 H_MCOP

Korkmaz and Krunz [46] also provided a heuristic called H_MCOP. This heuristic tries to find a path within the constraints by using the nonlinear path length function (3) of SAMCRA. In addition, H_MCOP tries to simultaneously minimize the weight of a single "cost" measure along the path. To achieve both objectives simultaneously, H_MCOP executes two modified versions of Dijkstra's algorithm in backward and forward directions. In the backward direction, H_MCOP uses the Dijkstra algorithm for computing the shortest paths from every node to the destination node d w.r.t. $w(u,v) = \sum_{i=1}^{m} \frac{w_i(u,v)}{L_i}$. Later on, these paths from every node u to the destination node d are used to estimate how suitable the remaining sub-paths are. In the forward direction, H_MCOP uses a modified version of Dijkstra's algorithm. This version starts from the source node s and discovers each 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 H_MCOP considers complete paths before reaching the destination, it can foresee several infeasible paths during the search. If paths seem feasible, then the algorithm can switch to explore these feasible paths based on the minimization of the single measure. Although similar to the look-ahead property, this technique only provides a preference rule for choosing paths and cannot be used as a search-space reducing technique.

The complexity of the H_MCOP algorithm is $O(N \log N + mE)$. If one deals only with the MCP problem, then H_MCOP should be stopped whenever a feasible path is found during the search in the backward direction, reducing the computational complexity. The performance of H_MCOP in finding feasible

paths can be improved by using the k-shortest path algorithm and by eliminating dominated paths.

2.7 Limited Path Heuristic

Yuan [92] presented two heuristics for the MCP problem. The first "limited granularity" heuristic has a complexity of $O(N^m E)$, whereas the second "limited path" heuristic (LPH) has a complexity of $O(k^2 NE)$, where k corresponds to the queue-size at each node. The author claims that when $k = O(N^2 \log_2 N)$, the limited path heuristic has a very high probability of finding a feasible path, provided that such a path exists. However, applying this value results in an excessive execution time.

The performance of both algorithms is comparable when $m \leq 3$, but for $m > 3$ the limited path heuristic is better than the limited granularity heuristic. Hence, we will only evaluate the limited path heuristic. Another reason for omitting an evaluation of the limited granularity heuristic is that it closely resembles the algorithm from Chen and Nahrstedt (Section 2.4).

The limited path heuristic is an extended Bellman-Ford algorithm that uses two of the fundamental concepts of TAMCRA. Both use the concept of non-dominance and maintain at most k paths per node. However, TAMCRA uses a *k-shortest* path approach, while LPH stores the first (and not necessarily shortest) k paths. Furthermore LPH does not check whether a sub-path obeys the constraints, but only at the end for the destination node. An obvious difference is that LPH uses a Bellman-Ford approach, while TAMCRA uses a Dijkstra-like search. The simulations revealed that Bellman-Ford-like implementations require more *execution time* than Dijkstra-like implementations, especially when the graphs are dense. Conform the queue-size allocated for TAMCRA, we also allocated $k = 2$ in the simulations for LPH.

2.8 A*Prune

Liu and Ramakrishnan [53] considered the problem of finding not only one but multiple (K) shortest paths satisfying the constraints. The length function used is the same as Jaffe's length function (2). Liu and Ramakrishnan 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 the simulations we took $K = 1$.

A*Prune first calculates for each QoS measure the shortest paths from the source s to all $i \in N \backslash \{s\}$ and from the destination 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 deployed by Korkmaz and Krunz [45]). After this initialization phase the algorithm proceeds in a Dijkstra-like fashion. The node with the shortest predicted end-to-end

length[3] 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.

If Q is the number of stored paths, then the worst-case complexity is $O(QN(m+h+\log Q))$, where h is the number of hops of the retrieved path. This complexity is exponential, because Q can grow exponentially with $G(N, E)$. Liu and Ramakrishnan [53] do mention that it is possible to implement a Bounded A*Prune algorithm, which runs polynomial in time at the risk of loosing exactness.

2.9 Overview of special-case QoS Routing Algorithms

Several works in the literature have aimed at addressing special yet important sub-problems in QoS routing. For example, researchers addressed QoS routing in the context of bandwidth and delay. Routing with these two measures is not NP-complete. Wang and Crowcroft [88] presented a *bandwidth-delay based routing algorithm* which simply prunes all links that do not satisfy the bandwidth constraint and then finds the shortest path w.r.t. the delay in the pruned graph. A much researched problem is the NP-complete *Restricted Shortest Path* (RSP) problem. The RSP problem only considers two measures, namely delay and cost. The problem consist of finding a path from s to d for which the delay obeys a given constraint and the cost is minimum. In the literature, the RSP problem is also studied under different names such as the delay-constrained least-cost path, minimum-cost restricted-time path, or constrained shortest path. Many heuristics have been proposed for this problem, e.g. [29, 72, 36, 28]. Several path selection algorithms based on different combinations of bandwidth, delay, and hopcount were discussed in [68] (e.g. widest-shortest path and shortest-widest path). In addition, new algorithms were proposed to find more than one feasible path w.r.t. bandwidth and delay (e.g. Maximally Disjoint Shortest and Widest Paths) [79]. Kodialam and Lakshman [41] proposed bandwidth guaranteed dynamic routing algorithms. Orda and Sprintson [69] considered pre-computation of paths with minimum hopcount and bandwidth guarantees. They also provided some approximation algorithms that take into account certain constraints during the pre-computation. Guerin and Orda [27] focussed on the impact of reserving in advance on the path selection process. They describe possible extensions to path selection algorithms in order to make them advance-reservation aware, and evaluate the added complexity introduced by these extensions. Fortz and Thorup [22] investigated how to set link weights based on previous measurements so that the shortest paths can provide better load balancing and can meet the desired QoS constraints. When there exist certain specific dependencies between the QoS measures, due to specific scheduling schemes at network routers, the path selection problem is also simplified [56]. Specifically, if Weighted Fair Queueing

[3] The length function is a linear function of all measures (2). If there are multiple sub-paths with equal predicted end-to-end length, the one with the shortest length so-far is chosen.

scheduling is being used and the constraints are on bandwidth, queueing delay, jitter, and loss, then the problem can be reduced to a standard shortest path problem by representing all the constraints in terms of bandwidth. However, although queueing delay can be formulated as a function of bandwidth, this is not the case for the propagation delay, which cannot be ignored in high-speed networks.

3 Performance Analysis of MCP Algorithms

3.1 Comparison of MCP Algorithms.

In this section we will present and discuss the simulations results for the MCP problem. The simulations consist of creating a Waxman topology [90], [81] in which the evaluated algorithms compute a path based on a set of constraints. After storing the desired results, this procedure is repeated. Waxman graphs are often chosen in simulations as topologies resembling communication networks. Moreover these graphs are easy to generate, allowing us to evaluate a large number of topologies. This last property is crucial in an algorithmic study, where it is necessary to evaluate many scenarios in order to be able to draw confident conclusions. As shown in [81], the conclusions reached for the Waxman graphs are also valid for the class of random graphs $G_p(N)$. All simulations consisted of generating 10^4 topologies. The m weights of a link were assigned independent uniformly distributed random variables in the range (0,1).

The choice of the constraints is important, because it determines how many (if any) feasible paths exist. We adopt two sets of constraints, referred to as $L1$ and $L2$:

- $L1 : L_i = w_i(P),\ i = 1, ..., m$, where P is the shortest path according to (3)
- $L2 : L_i = \max_{j=1,...,m}(w_i(SP_j)),\ i = 1, ..., m$, where SP_j is the shortest path based on the j-th measure.

The first set of constraints, denoted by $L1$, is very strict: there is only one feasible path present in the graph. The second set of constraints ($L2$) is based on the weights of the shortest paths for each QoS measure. We use Dijkstra to compute these shortest paths and for each of these m paths we store their path weight vectors. We then choose for each measure i the maximum i-th component of these m path weight vectors. With these constraints, the MCP problem can be shown to be polynomial [49]. (Iwata's algorithm can always find a feasible path with this set of constraints)

During all simulations we stored the *success rate* and the *normalized execution time*. The *success rate* of an algorithm is defined as the number of feasible paths found divided by the number of examined graphs. The *normalized execution time* of an algorithm is defined as the *execution time* of the algorithm (over all examined graphs) divided by the *execution time* of Dijkstra's algorithm.

Our simulations revealed that the Bellman-Ford-like algorithms (Chen's algorithm and the Limited Path Heuristic) consume significantly more *execution*

time than their Dijkstra-like counterparts. We therefore omitted them from the results presented in this chapter.

Figure 4 gives the *success rate* for four different topology sizes (N = 50, 100, 200 and 400), with $m = 2$. The exact algorithms SAMCRA and A*Prune always give the highest *success rate* possible. The difference in the *success rate* of the heuristics is especially noticeable when the constraints are strict. In this case Jaffe's algorithm and Iwata's algorithm perform significantly worse than the others. The only heuristic that is not affected much by strict constraints is the randomized algorithm. However, its *execution time* is comparable to that of the exact algorithms.

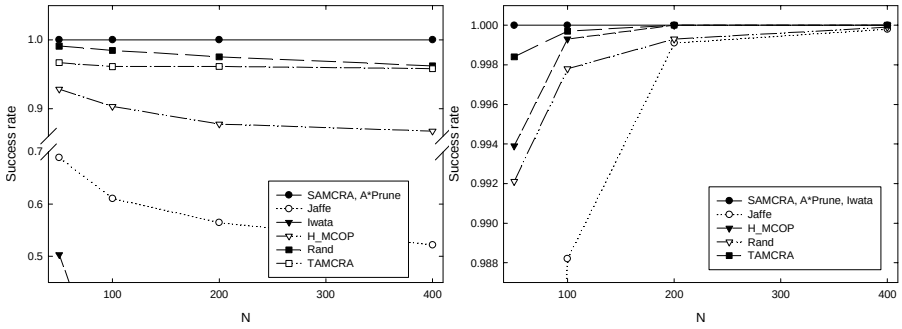

Fig. 4. The success rate for $m = 2$. The results for the set of constraints $L1$ is depicted on the left and for $L2$ on the right.

Figure 5 displays the *normalized execution time*. It is interesting to observe that the *execution time* of the exact algorithm SAMCRA does not deviate much from the polynomial time heuristics. This difference increases with the number of nodes, but an exponential growing difference is not noticeable! A first step towards understanding this phenomenon was provided by Kuipers and Van Mieghem in [47] and [49]. Furthermore, it is noticeable that when the constraints get looser, the *execution time* increases. The algorithms to which this applies, all try to minimize some length function (MCOP). When constraints get loose, this means that there will be more paths within the constraints, among which the shortest path has to be found. Searching through this larger set results in an increased *execution time*. If optimization is not strived for (MCP), then it is easier to find a feasible path under loose constraints than when constraints are strict.

We have also simulated the performance of the algorithms as a function of m (m = 2, 4, 8 and 16). The results are plotted in Figures 6 and 7. We can

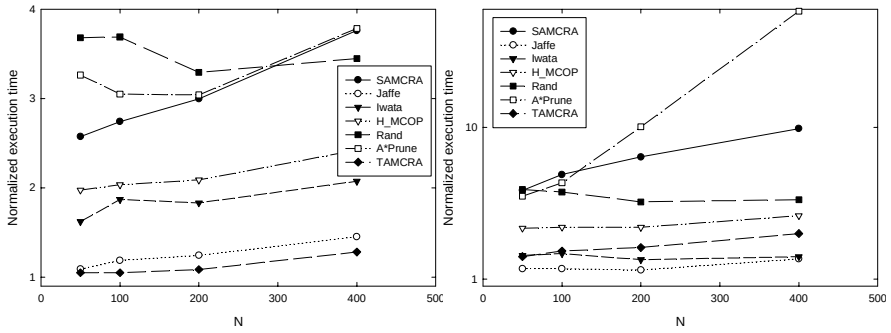

Fig. 5. The normalized execution times for $m = 2$. The results for the set of constraints $L1$ are plotted on the left and for $L2$ on the right.

see that the algorithms display a similar ranking in *success rate* as in Figure 4. All link weights are independent uniformly distributed random variables. Under independent link weights, the larger m, the larger the set of non-dominated paths to evaluate. However, at a certain threshold point (m), the constraint values will become dominant, leading to an increasing number of paths that violate the constraints and hence less paths to evaluate. This property is explained in [80]. The impact of the constraint values can also be seen by comparing the execution times in Figures 6 and 7. If the constraints are loose, then a significant difference in *execution time* is noticeable between the exact algorithms SAMCRA and A*Prune. This can be attributed to the look-ahead property of A*Prune, which can foresee whether sub-paths can lead to feasible end-to-end paths. Again, note that we do not see any NP-complete behavior in the *execution times*.

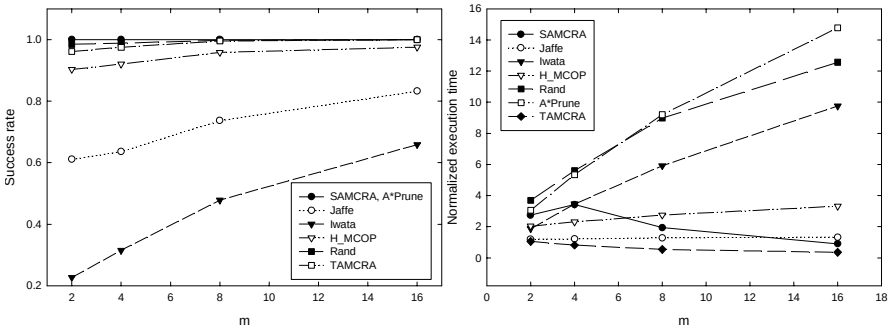

Fig. 6. The success rate and normalized execution time in a 100-node network, as a function of m, with the set of constraints $L1$.

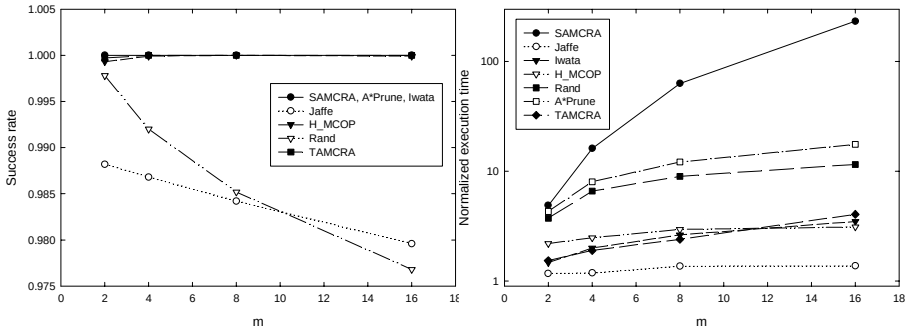

Fig. 7. The success rate and normalized execution time in a 100-node network, as a function of m, with the set of constraints $L2$.

Based on these results we can rank the heuristics according to their *success rate* and *execution time* as follows: TAMCRA, H_MCOP, Randomized algorithm, Jaffe's algorithm, Iwata's algorithm. The simulation results presented in [46] displayed a higher *success rate* for H_MCOP than for TAMCRA. This was due to a programming error, where the forward search of H_MCOP was revisiting the previously explored nodes (which is similar to using $k > 1$ in the k-shortest-paths-based algorithms). This implementation bug has now been fixed, which resulted in a better *success rate* for TAMCRA.

3.2 Summary of the Performance of MCP algorithms.

Based on the simulation results of the previous section, the strengths of these algorithms are summarized. The conclusions are only valid for the considered class of graphs, namely the Waxman graphs (and according to [81] also random graphs) with independent uniformly distributed link weights, but might also hold for other classes of graphs.

For the MCP problem, we observe that TAMCRA-like algorithms have a higher *success rate* than linear approximations and Bellman-Ford based algorithms. This higher *success rate* is attributed to the following concepts:

1. *Using a Dijkstra-like search along with a nonlinear length function*
   A nonlinear length function is a prerequisite for exactness. When the link weights are positively correlated, a linear approach may give a high *success rate* in finding feasible paths, but under different circumstances the returned path may violate the constraints by 100%.
   A Bellman-Ford-like search runs better on sparse than on dense graphs, however our simulations indicated that even on sparse graphs, the Dijkstra-like heap-optimized search runs significantly faster.

2. *Tunable accuracy through a k-shortest path functionality*
   Routing with multiple constraints may require that multiple paths be stored at a node, necessitating a k-shortest path approach.
3. *Reducing the search-space through the concept of non-dominance*
   Reducing the search-space is always desirable, because this reduces the *execution time* of an algorithm. The non-dominance principle is a strong search-space reducing technique, especially when the number of constraints is small. Note that the constraints themselves, if strict, also provide a search-space reduction, since many sub-paths will violate those constraints.
4. *Predicting the feasibility of paths (look-ahead property)*
   First calculating a path in polynomial time between the source and destination and using this information to find a feasible path between the same source and destination is especially useful when graphs become "hard to solve", i.e. N, E and m are large. This look-ahead property allows to compute lower bounds for end-to-end paths, which can be used to check the feasibility of paths. Moreover, better preference rules could be adopted to extract nodes from the queue.

The exactness of the TAMCRA-like algorithms depends on the liberty to choose k. If k is not restricted, then both MCP and MCOP problems can be solved exact, as done by SAMCRA. Although k is not restricted in SAMCRA, simulations on Waxman graphs with independent uniformly distributed random link weights show that the *execution time* of this exact algorithm increases linearly with the number of nodes, providing a scalable solution to the MC(O)P problem. If a slightly larger *execution time* is permitted, then such exact algorithms are a good option. Furthermore, simulations show that TAMCRA-like algorithms with small values of k render near-exact solutions with a Dijkstra-like complexity. For example, TAMCRA with $k = 2$ has almost the same *success rate* as the exact algorithms.

4 Influence of network dynamics on QoS routing

The QoS path selection problem has been addressed in previous sections assuming that the *exact* state of the network is known. Such an assumption is often imposed to isolate the impact of network dynamics from the path selection problem. In practice, however, network dynamics can greatly affect the accuracy of the captured and disseminated state information, resulting in some degree of uncertainty in state information.

In current networks, the routing protocol is dynamic and distributed. The dynamic behavior means that important topology changes are flooded to all nodes in the network while the distributed nature implies that all nodes in the network are equally contributing to the topology information distribution process. Since QoS is associated with resources in the nodes of the network, the QoS link weights are, in general, coupled to these available resources. As illustrated in Figure 8, we distinguish between topology changes that (1) occur

infrequently and (2) rapidly change in time. The first kind reflects topology changes due to failures and the joining/leaving of nodes. In the current Internet, only this kind of topology changes is considered. Its dynamic is relatively well understood. The key point is that the time between two 'first kind' topology changes is long compared to the time needed to flood this information over the whole network. Thus, the topology databases on which routing relies, converge rapidly with respect to the frequency of updates to the new situation and the transient period where the databases are not synchronized (which may cause routing loops), is generally small.

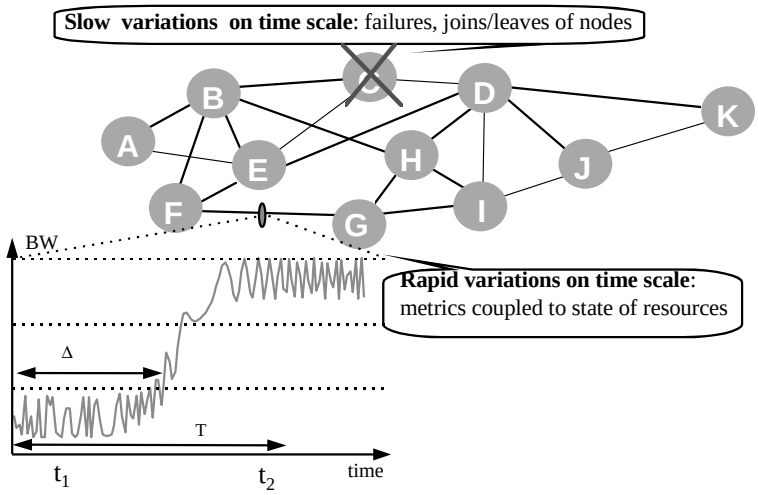

Fig. 8. Network topology changes on different time scales

The second type of rapidly varying changes are typically those related to the consumption of resources or to the traffic flowing through the network. The coupling of the QoS measures to state information seriously complicates the dynamics of flooding because the flooding convergence time T can be longer than the change rate Δ of some metric (such as available bandwidth). Figure 8 illustrates how the bandwidth BW on a link may change as a function of time. In contrast to the first kind changes where $T << \Delta$, in the second kind changes, T can be of the same order as Δ. Apart from this, the second type changes necessitates the definition of a significant change that will trigger the process of flooding. In the first kind, every change was significant enough to start the flooding. The second kind significant change may be influenced by the flooding convergence time T and is, generally, strongly related to the traffic load in (a part of) the network. An optimal update strategy for the second type changes is highly desirable. So far, unfortunately, no optimal topology update rule for

the second type changes has been published, although some partial results have appeared as outlined in Section 5.

To reduce the overhead of flooding, tree-based broadcasting mechanisms [31] are proposed where a given link state advertisement is delivered only once to every node. Tree-based broadcasting eliminates the unnecessary advertisement overhead, but it introduces a challenging problem, namely how to determine and maintain *consistent* broadcast trees throughout the network. Various tree-based broadcasting mechanisms have been proposed for this purpose (e.g., [8, 31, 9, 18]), but they all involve complex algorithms and protocols that cannot be supported with the existing TCP/IP protocol suite. Korkmaz and Krunz [43] have proposed a hybrid approach that combines the best features of flooding and tree-based broadcasting.

Besides the update rule (also called triggering policies [52]), a second source of inaccuracy is attributed to state aggregation. Most link-state routing protocols are hierarchical, whereby the state of a group of nodes (an OSPF area or a PNNI peer group) is summarized (aggregated) before being disseminated to other nodes [42, 84, 82]. While state aggregation is essential to ensuring the scalability of any QoS-aware routing protocol, it comes at the expense of *perturbing* the true state of the network.

5 Overview of dynamic QoSR proposals.

A large amount of proposals to deal with the network dynamics are discussed in this section. The multitude of the proposals and the lack of optimal solutions illustrate the challenging difficulty. Moreover, it points to a currently missing functionality in end-to-end quality assured networking.

5.1 Path Selection under Inaccurate Information

As explained in Section 4, some level of uncertainty in state information is unavoidable. To account for such uncertainty, path selection algorithms may follow a *probabilistic* approach in which link state parameters (e.g., delay, available bandwidth) are modelled as random variables (rvs) [26]. Since QoS routing has not yet been implemented in real networks, one of the difficulties lies in what distributions are appropriate for these rvs. In a number of simulation-based studies (e.g., [6, 34, 35]), a uniformly distributed link bandwidth is assumed while for the link delay, various distributions such as exponential, normal, and gamma are suggested. The exact shape of the distribution may not be a critical issue, as robust path selection algorithms require only knowledge of the statistical moments of the distribution (e.g., *mean and variance*). These statistical moments can be computed simply as follows. Each node maintains a moving average and corresponding variance for a given link state parameter. For example, the moments for the bandwidth can be updated whenever there is a change in the available bandwidth (e.g., flow is added or terminated), while the ones for the delay can be

updated whenever a packet leaves the router. In case of a high packet transmission rate, sampling can be used to update the delay parameters. Once the mean and variance are computed for each QoS metric, they can be disseminated using QoS-enhanced versions [5] of OSPF[4]. A crucial question here is when and how to advertise the mean and variance values. A triggered-based approach similar to the one in [3] or [52] can be used for this purpose.

In the case of probabilistically modelled network-state information, the objective of the path selection algorithm is to identify the *most probable* feasible path. This problem has mainly been investigated under bandwidth and/or delay constraints. The general problem at hand can be formulated as follows:

Definition: *Most-Probable Bandwidth-Delay Constrained Path (MP-BDCP) Problem*: Consider a network $G(N, E)$, where N is the set of nodes and E is the set of links. Each link $(i, j) \in E$ is associated with an available bandwidth parameter $b(i, j)$ and a delay parameter $d(i, j)$. It is assumed that the $b(i, j)$'s and $d(i, j)$'s are independent rvs. For any path P from the source node s to the destination node t, let $b(P) \stackrel{\text{def}}{=} \min\{b(i, j) \mid (i, j) \in P\}$ and $d(P) \stackrel{\text{def}}{=} \sum_{(i,j)\in P} d(i, j)$. Given a bandwidth constraint B and a delay constraint D, the problem is to find a path that is most likely to satisfy both constraints. Specifically, the problem is to find a path P^* such that for any other path P from s to t,

$$\pi_B(P^*) \geq \pi_B(P), \quad \text{and} \tag{4}$$

$$\pi_D(P^*) \geq \pi_D(P), \tag{5}$$

where $\pi_B(P) \stackrel{\text{def}}{=} \Pr[b(P) \geq B]$ and $\pi_D(P) \stackrel{\text{def}}{=} \Pr[d(P) \leq D]$.

If the $b(i, j)$'s and $d(i, j)$'s are constants, the MP-BDCP problem reduces to the familiar bandwidth-delay constrained path problem, which can be easily solved in two steps [88]: (i) prune every link (i, j) for which $b(i, j) < B$, and (ii) find the shortest path w.r.t. the delay parameter in the pruned graph. However, MP-BDCP is, in general, a hard problem. In fact, the objectives (4) and (5) of the MP-BDCP problem give rise to two separate problems: the *most-probable bandwidth constrained path* (MP-BCP) problem and the *most-probable delay constrained path* (MP-DCP) problem. We first review the studies focusing on these problems separately. We then continue our review by considering both parts of the combined MP-BDCP problem simultaneously.

MP-BCP Problem MP-BCP is a rather simple problem, and can be exactly solved by using a standard version of the Most Reliable Path (MRP) algorithm [50, 26], which associates a probability measure $\rho(i, j) \stackrel{\text{def}}{=} \Pr[b(i, j) \geq B]$

[4] The current version of OSPF considers only a single, relatively static cost metric. Apostolopoulos *et al.* [5] described a modification to OSPF that allows for disseminating multiple link parameters by exploiting the type-of-service (TOS) field in link-state advertisement (LSA) packets.

with every link (i, j). So, $\pi_B(P) = \prod_{(i,j)\in P} \rho(i, j)$. To find a path that maximizes π_B, one can assign the weight $-\log \rho(i, j)$ to each link (i, j) and then run the Dijkstra's shortest path algorithm. In [44] the authors slightly modified the Dijkstra's algorithm for solving the same problem without using logarithms. While the MP-BCP can be efficiently addressed using such exact solutions, the MP-DCP problem is, in general, shown to be NP-hard [23]. Accordingly, most research has focused on the MP-DCP problem.

MP-DCP Problem The MP-DCP problem can be considered under two different models, namely rate-based and delay-based [26]. The "rate-based" model achieves the delay bound by ensuring a minimum service rate to the traffic flow. The main advantage of this model is that the end-to-end delay bound can be mathematically represented depending on the available bandwidth on each link. So it seems one can address the MP-DCP problem by using the similar approach of the above MP-BCP problem. In spite of some similarities, however, these problems are not exactly the same due to the fact that the accumulative effect associated with the delay is not produced in the case of bandwidth. In [26] Guerin and Orda showed that the problem is, in general, intractable. Accordingly, they first considered the special cases of the problem and provided tractable solutions for these cases. They then introduced a near-optimal algorithm, named QP, for the MP-DCP problem under rate-based model. Although the rate-based model leads to some attractive solutions, it requires to add new networking mechanisms, mostly regarding using schedulers that allow rate to be strictly guaranteed along the path.

On the other hand, the "delay-based" model provides a general approach for achieving the delay bound by concatenating the local delays associated with each link along the selected path. Note that the above definition formulates the MP-DCP problem based on this general model. The MP-DCP problem is essentially an instance of the stochastic shortest path problem, which has been extensively investigated in the literature (e.g., [54, 30]). One key issue in stochastic shortest path problems, in general, is how to define the optimality of a path. Some formulations (e.g., [60, 77, 37, 71]) aim at finding the most likely shortest path. Others consider the least-expected-delay paths under interdependent or time-varying probabilistic link delays [78, 59, 76]. Cheung [15] investigated dynamic stochastic shortest path problems in which the probabilistic link weight is "realized" (i.e., becomes exactly known) once the node is visited. Several studies define path optimality in terms of maximizing a user-specified objective function (e.g., [54, 21, 61, 63, 64]). Our formulation of the MP-DCP problem in the above definition belongs to this category, where the objective is to find a path that is most likely to satisfy the given delay constraint.

Guerin and Orda [26] also considered the MP-DCP problem under the delay-based model and provided tractable solutions for some of its special cases. These cases are relatively limited, so it is desirable to find general tractable solutions which can cope with most network conditions. In [44], Korkmaz and Krunz provided two complementary (approximate) solutions for the MP-DCP problem

by employing the central limit theorem approximation and Lagrange relaxation techniques. These solutions were found to be efficient, requiring, on average, a few iterations of Dijkstra's shortest path algorithm. In [26] Guerin and Orda considered a modification of the problem, in which the goal is to partition the given end-to-end delay constraint into local link constraints. The optimal path for the new problem is, in general, different from the one for the MP-DCP problem [55]. Moreover, the solutions provided for the partitioning problem in [55] are computationally more expensive than the solutions in [44] which directly addresses the MP-DCP problem. To reduce the complexity, the authors in [26] has also considered the hierarchical structure of the underlying networks.

Lorenz and Orda has further studied the modified partitioning problem [55]. They first considered the OP (Optimal Partition) Problem and provided an exact solution to it under a particular family of probability distributions (including normal and exponential distributions), where the family selection criterion is based on having a certain convexity property. They then analyzed the OP-MP (Optimally Partitioned Most Probable Path) Problem and provided a pseudopolynomial solution using dynamic programming methods. In fact, the solution uses a modification of the Dynamic-Restricted Shortest Path Problem (D-RSP). The RSP problem is a well-known problem which aims to find the optimal path that minimizes the cost parameter among all the paths that satisfy the end-to-end delay constraint. Since the RSP Problem is NP-hard, the authors provided a pseudopolynomial solution from which a new algorithm named Dynamic-OP-MP algorithm is inferred. The main difference between the Dynamic-OP-MP algorithm and the D-RSP algorithm is the cost computation method. As in the OP Problem, the MP-OP Problem is analyzed in detail, particularly when a uniform distribution exists, generating a Uniform-OP-MP algorithm. Finally, they proposed a new approach to obtain a fully polynomial solution to deal with the OP-MP Problem. As in the last case, this approach is based on making some modifications to the D-RSP algorithm, resulting in a non-optimal approximation (named discrete solution). This solution introduces a bounded difference in terms of cost and success probability regarding the optimal solution by interchanging the cost and delay roles in the D-RSP algorithm.

MP-BDCP problem MP-BDCP belongs to the class of multi-objective optimization problems, for which a solution may not even exist (i.e., the optimal path w.r.t. π_B is not optimal w.r.t. π_D, or vice versa). To eliminate the potential conflict between the two optimization objectives, one can specify a *utility function* that relates π_B and π_D, and use this function as a basis for optimization. For example, one could maximize $\min\{\pi_B(P), \pi_D(P)\}$ or the product $\pi_B(P)\pi_D(P)$. Rather than optimizing a specific utility function, Korkmaz and Krunz [44] proposed a heuristic algorithm to compute a subset of *nearly nondominated paths* for the given bandwidth and delay constraints. Given this set of paths, a decision maker can select one of these paths according to his/her specific utility function.

5.2 Safety Based Routing.

The Safety-Based Routing (SBR) was proposed by Apostolopoulos *et al.* [6]. SBR assumes explicit routing with bandwidth constraints and on-demand path computation. The idea of SBR is to compute the probability that a path can support an incoming bandwidth request. Therefore, SBR computes the Safety (S) parameter defined as the probability that the total required bandwidth is available on the sequence of links that constitute the path. This probability can be used to classify every link, and to find the safest path, i.e. the path having the best chance for supporting total required bandwidth. Since the safety of each link is considered as independent from that of the others links in a path, the safety S of a path is the product the safeties of every link in that path. Once S has been computed it is included in the path selection process as a new link weight.

SBR uses two different routing algorithms based on combining S with the number of hops, the safest-shortest path and the shortest-safest path. The safest-shortest path algorithm selects that path with the larger safety S among the shortest paths. The shortest-safest path algorithm on the other hand, selects paths with larger safety and if more than one exists the shortest one is chosen. In addition, the SBR mechanism uses triggering policies[5] in order to reduce the signaling overhead while keeping a good routing performance.

A performance evaluation of the blocking probability shows [6] that the shortest-safest path algorithm is the most effective one for any of the triggering policies that were evaluated.

5.3 Ticket-based Distributed QoS routing.

Three algorithms are simulated in [13]: the flooding algorithm, the TBP and the shortest-path algorithm (SP). The simulation results are represented using three parameters, the success ratio, the average messages overhead and the average path cost. The simulation results in [13] show that the TBP achieves a high success ratio and low-cost feasible paths with minor overhead.

The Ticket-based Distributed QoS Routing mechanism was proposed by Chen and Nahrstedt [13]. They focus on the NP-complete delay-constrained least-cost routing (different from the one explained in Section 2.4). They propose a routing algorithm which targets to find the low-cost path, in terms of satisfying the delay constraint, by using only the available inaccurate or imprecise routing information. To achieve its purpose, initially, Chen and Nahrstedt suggest a simple imprecise state model that defines which information must be stored in every node: connectivity information, delay information, cost information and an additional state variable named delay variation which stands for the estimated maximum change of the delay information before receiving the next updating message. For simplicity reasons, the imprecise model is not applied to the connectivity and cost information. They justify this assumption saying that

[5] The most important update policies are discussed in [52].

the global routing performance is not significantly degraded. Then, a multipath distributed routing scheme, named ticket based probing is proposed. The ticket based probing sends routing messages, named probes, from a source s to a destination d. Based on the (imprecise) network state information available at the intermediate nodes, these probes are routed on a low-cost path that fulfils the delay requirements of the LSP request. Each probe carries at least one ticket in such a way that by limiting the number of tickets, the number of probes is limited as well. Moreover, since each probe searches a path, the number of searched paths is also limited by the number of tickets. In this way, the trade-off between the signalling overhead and the global routing performance may be controlled. Finally, based on this ticket based probing scheme, Chen and Nahrstedt suggest a routing algorithm to address the NP-complete delay-constrained least-cost routing problem, called Ticket Based Probing algorithm (TBP). Three algorithms are simulated in [13]: the flooding algorithm, the TBP algorithm and the shortest-path algorithm. Simulations are presented using three parameters, the success ratio, the average messages overhead and the average path cost. The results show that the TBP algorithm exhibits a high success ratio and a low-cost path satisfying the delay constraint with minor overhead while tolerating a high degree of inaccuracy in the network state information.

5.4 BYPASS based routing.

BYPASS based routing (BBR) [58] presents a different idea to solve the bandwidth blocking due to inaccurate routing information produced by a triggering policy based on either threshold based triggers or class based triggers. BBR is an explicit routing mechanism that instructs the source nodes to compute both the working route and as many paths to bypass the links (named bypass-paths) that potentially cannot cope with the incoming traffic requirements. The idea of the BBR mechanism is derived from protection switching for fast rerouting discussed in [14]. However, unlike the use of protection switching for fast rerouting, in BBR both the working and the alternative paths (bypass-paths) are computed simultaneously but not set up; they are only set up when required.

In order to decide those links that might be bypassed (named obstruct-sensitive links, OSLs), a new policy is added. This policy defines a link as OSL whenever a path setup message sent along the explicit route reaches a link with insufficient residual bandwidth. Combining the BBR mechanism and Dijkstra's algorithm, two different routing algorithms are proposed [58]: the Shortest-Obstruct-Sensitive Path (SOSP), which computes the shortest path among all the paths with the minimum number of obstruct-sensitive links, and the Obstruct-Sensitive-Shortest Path (OSSP), which computes the path that minimizes the number of obstruct-sensitive links among all the shortest paths. Once the working path is selected, BBR computes the bypass-paths that bypass those links in the working path defined as OSL. When the working path and the bypass-paths are computed, the working path setup process starts. A signaling message is sent along the explicit path included in the setup message. When a node detects that a link in the explicit path has not enough available bandwidth

to support the required bandwidth, it sends the setup signaling message over the bypass-path of this link. The bypass-paths nodes are included in the setup signaling message as well, i.e. bypass-paths are also explicitly routed.

The BBR performance is evaluated by simulation. The obtained results shown in Figure 9 exhibit the reduction obtained in the bandwidth blocking ratio of the BBR (with both SOSP and OSSP) compared to the Widest-Shortest Path (WSP) and the Safety Based Routing (Shortest-Safest-Path, SSP). These algorithms are simulated with both the Threshold and the Exponential class triggering policies. Figure 9 indicates that the SOSP algorithm is, in terms of blocking probability, the most effective.

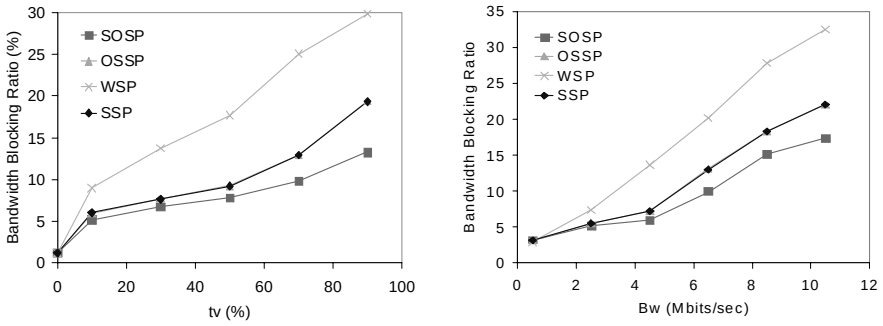

Fig. 9. Bandwidth Blocking Ratio for the threshold and the exponential class triggering policies

5.5 Path selection algorithm based on available bandwidth estimation

Anjali *et al.* [2] propose an algorithm for path selection in MPLS networks. They note that most recent QoS routing algorithms utilize the nominal available bandwidth information of the links to optimally select the path. Assuming that most of the traffic flows do not strictly use the requested bandwidth, the nominal link utilization overestimate the actual link consumption, which leads to non-efficient network resource utilization. Therefore, the network performance can be improved by a path selection process based on an accurate measurement of the actual available link bandwidth instead of the nominal value. For scalability reasons, this measurement cannot be achieved by any updating link state database process. Moreover, the authors [2] argue that due to the available bandwidth variability, a single sample cannot accurately represent the actual bandwidth availability and as a consequence routing decisions taken based on single samples are likely wrong. Since perfectly updated network state information is in general not possible, Anjali *et al.* [2] present an Available Bandwidth

Estimation Algorithm that estimates the actual available bandwidth on each link. A path is computed with a shortest widest path routing algorithm (Section 2.9) that uses these available bandwidth estimations as link weight. Finally, in order to limit the network congestion, a threshold parameter is added. Once the path has been computed, the available bandwidth on the bottleneck link of the path is computed. The threshold parameter is applied to this bottleneck value to compute a benchmark for path selection in such a way that if the bandwidth requested is larger than a certain fraction of the bottleneck link bandwidth, the incoming request is rejected.

The proposed path selection algorithm is shown [2] to perform better than the shortest path routing algorithm in terms of rejection probability, because the proposed routing algorithm based on the available bandwidth estimation algorithm has more accurate information about the actual link load and therefore can take more precise decisions.

5.6 Centralized server based QoS Routing

Unlike the previous proposals, Kim and Lee [40] do not attempt to enhance the routing process under inaccurate network state information but rather to eliminate the inaccuracy. Kim and Lee propose a centralized server based QoS routing scheme, which both eliminates the overhead due to the exchange of network state update messages and achieves higher routing performance by utilizing accurate network state information in the path selection process. Routers are clients of the route server and send routing queries for each one of the incoming requests. The route server stores and maintains two data structures, the Network Topology Data Base (NTDB), which keeps the link state information for each link in the network, and the Routing Table Cache (RTC) that stores the computed path information.

Although the main idea is derived from that suggested in [4], these new schemes differ in how the network state information is collected. Instead of collecting the link state information from the other routers, in this new approach the proposed router server updates and maintains a link state database as the paths are assigned to or return back from a certain flow. The main issues in this centralized scheme are: (1) the processing load and storage overhead required at the server, (2) the protocol overhead to exchange the router queries and the replies between the server and the remote routers that act as clients and (3) the effects produced when the server becomes either a bottleneck point or a single point of failure. Kim and Lee [40] suggest various alternatives to reduce the loads and overhead.

Two routing algorithms are used: a modification of the Dijkstra's algorithm and the Bellman-Ford algorithm with QoS extensions. Assuming the existence of a certain locality in the communication pattern, a large number of source-destination pairs are expected to be unused. Hence, a path caching approach is used to reduce the path computation overhead. The size of the RTC is controlled by two parameters: the maximum number K of entries (source-destination pairs)

in the RTC and the maximum number n of paths for each source-destination pair.

The server based QoS routing scheme is evaluated by simulation [40]. On one hand, the simulations show that a simple path caching scheme substantially reduces the path computation overhead when considering locality in the communication pattern. On the other hand, the simulations indicate that the proposed schemes perform better than the distributed QoS routing schemes with similar protocol overhead.

5.7 A localized QoS Routing approach

The main advantage of a localized approach for QoS routing as proposed by Nelakuditi *et al.* [66], is that no global network state information exchange among network nodes is needed, hence reducing the signaling overhead. The path selection is performed in the sources nodes based on their local view of the global network state. The main difficulty in implementing any localized QoS routing scheme is how the path is selected only based on the local network state information collected in the source nodes. In order to address this problem Nelakuditi *et al.* present a new adaptive proportional routing approach for localized QoS routing schemes. They propose an idealized proportional routing model, where all paths between a source-destination pair are disjoint and their bottleneck link capacities are known. In addition to this ideal model, the concept of virtual capacity of a path is introduced which provides a mathematically sound way to deal with shared link among multiple paths. The combinations of these ideas is called Virtual Capacity based routing (VCR). Their simulations [66] show how the VCR scheme adapts to traffic load changes by adjusting traffic flows to the set of predefined alternative paths. However, Nelakuditi *et al.* describe two significant difficulties related to the VCR implementation that lead them to propose an easily realizable implementation of the VCR scheme, named Proportional Sticky routing (PSR).

The PSR scheme operates in two stages: proportional flow routing and computation of flow proportions. PSR proceeds in cycles of variable lengths. During each cycle, any incoming request can be routed along a certain path selected among a set of eligible paths, which initially may include all the candidates paths. A candidate path is ineligible depending on the maximum permissible flow blocking parameter, which determines how many times this candidate path can block a request before being ineligible. When all candidate paths become ineligible a cycle terminates and all the parameters are reset to start the next cycle. An eligible path is finally selected depending on its flow proportion: the larger the flow proportion, the larger the chances for being selected.

Simulation results show that the PSR scheme is simple, stable and adaptive, and the authors [66] conclude that it is a good alternative to global QoS routing schemes.

5.8 Crankback and fast re-routing

Crankback and fast re-routing were included in the ATMF PNNI [7] to address the routing inaccuracy due to fast changes in the resources [83] and due to the information condensation [82] of the hierarchical network structure.

The establishment of a connection between two nodes A and K as shown in Figure 10, takes place in two phases. Based on the network topology reflecting a snap shot at time t_1 and flooded to the last node at $t_1 + T$, the routing algorithm (e.g. SAMCRA) computes the path from A to K subject to some QoS requirements. Subsequently, in the second phase, the needed resources along that path are installed in all nodes constituting that path. This phase is known as the 'connection prerequisite' and the network functionality that reserves resources is called signaling. The signaling operates in a hop by hop mode: it starts with the first node and proceeds further to the next node if the 'installation' is successful. Due to the rapidly changing nature of the traffic in the network, at time $t_2 > t_1$ and at a certain node I (as exemplified in Figure 10), the connection set-up process may fail because the topology situation at time t_1 may significantly differ from that at time t_2 (Figure 8). Rather than immediately blocking the path request from A to K, PNNI invokes an emergency process, called *crankback*. The idea is similar to back tracking. The failure in node I returns the previous node D with the responsibility to compute immediately an alternative path from itself towards K, in the hope that along that new path the set-up will succeed.

The crankback process consumes both much CPU-time in the nodes as control data and yet, does not guarantee a successful connection set-up. When the crankback process returns back to the source node A and this node also fails to find a path to K, the connection request is blocked or rejected and much computational effort of cranking back was in vain.

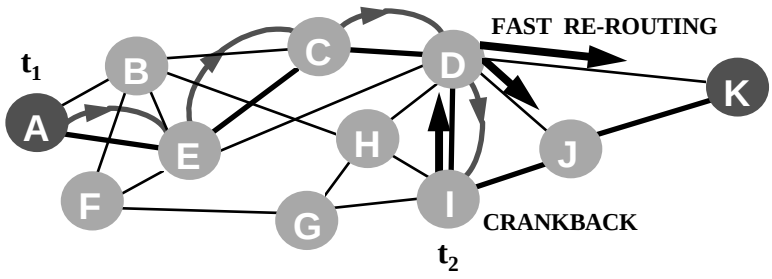

Fig. 10. Illustration of crankback and fast-rerouting.

Although the crankback process seems an interesting emergency 'exit' to prevent blocking, the efficiency certainly needs further study. For, in emergency cases due to heavy traffic, the crankback processes generate additional control

traffic possibly causing a triggering of topology flooding, and hence even more control data is created, eventually initiating a positive feedback loop with severe consequences. These arguments suggest to prevent invoking crankback as much as possible by developing a good topology update strategy.

6 Stability aspects in QoS Routing

If the topology changes as explained in Section 4 are inappropriately fast flooded (and trigger new path computations), route flapping may occur degrading the traffic performance significantly. This section outlines approaches to avoid routing instability.

Routing instabilities were already observed in the ARPANET [39]. The reasons for this routing instability were attributed to the type of link weight sampling used and the path selection algorithm. The use of instantaneous values of the link delay led to frequent changes in the metric, and the shortest paths computed were rapidly out-dated. The application of the Bellman-Ford algorithm with a dynamically varying metric instead of a static metric led to routing loops. These problems were partially overcome by using averaged values of link delay over a ten-second period and by the introduction of a link-state routing protocol as OSPF. With the constant growth of the Internet, the problem has become recurrent and other solutions had to be found.

6.1 Quantization of QoS measures and smoothing

A common approach to avoid routing instability is the advertisement of the link weights that are quantified or smoothed in some manner rather than advertising instantaneous values. This approach has two main consequences, one directly related to routing stability and the other related to routing overhead. The quantization/smoothing limits overshoots in the dynamic metric which reduces the occurrence and the amplitude of routing oscillation. Simultaneously, the distribution of an excessive amount of routing updates is avoided reducing the flooding overhead. While improving the routing stability, the quantization/smoothing of link weights damps the dynamic coupling to actual resource variations and may lower the adaptation capabilities of the routing protocol. The update strategy consists of a trade-off between routing stability and routing adaptation.

Besides quantization/smoothing of resource coupled link weights, the link weight can be evaluated on different time-scales as proposed by Vutukury and Garcia-Luna-Aceves [85]. A longer time-scale that leads to path computation and a shorter time-scale that allows for the adaptation to traffic bursts.

The techniques of metric quantization/smoothing proposed by Khanna and Zinky [39] reduce routing oscillations, but are not sufficient under adverse circumstances (high loads or bursty traffic) in packet switched networks. When the link weights are distributed, the information may already be out-dated, leading to the typical problem of QoS routing under inaccurate information as discussed in Section 4.

6.2 Algorithms for load balancing

Load-balancing provides ways of utilizing multiple-paths between a source and a destination, which may avoid routing oscillations. There are approaches for load balancing in best-effort networks and in QoS-aware networks. Load balancing including QoS can be done per class, per flow or per traffic aggregate (best-effort and QoS flows).

Load balancing in best effort networks A simple approach of load balancing in best-effort networks is to use alternate paths when congestion rises as in the algorithm Shortest Path First with Emergency Exits (SPF-EE) [87]. This strategy prevents the excessive congestion of the current path because it deviates traffic to an alternate path when congestion starts to rise, and thus avoids routing oscillations. First, the next-hops on the shortest path to all the destinations in the network are determined. Subsequently, the next-hop on the alternate path (the emergency exit) is added to the routing table. The emergency exit is the first neighbor in the link-state database that is not the next-hop of the shortest path tree nor the final destination. The emergency exit is only used when the queue length exceeds a configured threshold. With this approach two objectives are achieved: the pre-computation of alternate paths allows for traffic distribution over those paths when congestion occurs and the routing update period is increased due to the limitation of traffic fluctuations.

As an alternative to single shortest path algorithms as SPF-EE, Vutukury and Garcia-Luna-Aceves [85] introduce multiple paths of unequal cost to the same destination. The algorithm proposed by these authors finds near-optimal multiple paths for the same destination based on a delay metric. The algorithm is twofold: it uses information about end-to-end delay to compute multiple paths between each source-destination pair, and local delay information to adjust routing parameters on the previously defined alternate paths. This short scale metric determines the next hop from the list of multiple next-hops that were computed based on the larger scale metric.

Even tough the proposals described above permit load balancing and avoid routing oscillations, they do not take into consideration the requirements of the different types of traffic. This problem has been addressed by some proposals within a connection-oriented context.

Load balancing supporting QoS Nahrstedt and Chen [65] conceived a combination of routing and scheduling algorithms to address the coexistence of QoS and best-effort traffic flows. In their approach, traffic with QoS guarantees is deviated from paths congested with best-effort traffic in order to guarantee the QoS requirements of QoS flows and to avoid resource starvation of best-effort flows. The paths for QoS flows are computed by a bandwidth-constrained source-routing algorithm and the paths for best-effort flows are computed using max-min fair routing. The authors also address the problem of inaccurate information that arises with the use of stale routing information due to the insufficient frequency of routing updates or to dimension of the network. As was

stated above, inaccurate information is a major contributor to routing instability. To cope with inaccurate information, besides keeping the values of available residual bandwidth (RB) on the link, the estimation on the variation of RB is also kept (ERBV). These two values define an interval (RB-ERBV, RB+ERBV) where the residual bandwidth on the next period will be. The routing algorithm of QoS flows will find a path between a source and a destination that maximizes the probability of having enough available bandwidth to accommodate the new flow.

Ma and Steenkiste [57] proposed another routing strategy that addresses inter-class resource sharing. The objective of their proposal is also to avoid starvation of best-effort traffic on the presence of QoS flows. The strategy comprises two algorithms: one to route best-effort traffic and the other to route QoS traffic. The routing decisions are based on a metric that enables dynamic bandwidth sharing between traffic classes, particularly, sending QoS traffic through links that are less-congested with best-effort traffic. The metric used for path computation is called virtual residual bandwidth (VRB). The value of the VRB can be above or below the actual residual bandwidth depending on the level of congestion on the link due to best-effort traffic. The algorithm uses the Max-Min Fair Share Rate to evaluate the degree of congestion [38]. If the link is more (less) congested with best-effort traffic than the other links on the network, VRB is smaller (higher) than the actual residual bandwidth. When the link has a small amount of best-effort flows, VRB will be high and the link will be interesting for QoS flows. The paths for best-effort traffic are computed based on the Max-Min Fair Rate for a new connection.

Shaikh *et al.* [74] present a hybrid approach to QoS routing that takes the characteristics of flows into account to avoid instability. The resources in the network are dynamically shared between short-lived (mice) and long-lived (elephants) flows. The paths for long-lived flows are dynamically chosen, based on the load level in the network, while the paths for short flows are statically precomputed. Since dynamic routing is only used for long-lived flows, the protocol overhead is limited. At the same time, the duration of these flows avoids successive path computations which is beneficial for the stability. The path selection algorithm computes widest-shortest paths that can accommodate the needs of the flow in terms of bandwidth. This approach is similar to the one used by Vutukury *et al.* described above.

While the above strategies are aimed at connection-oriented networks, the algorithm Enhanced Bandwidth-inversion Shortest-Path [89] has been proposed for hop-by-hop QoS routing in Differentiated Services networks. This proposal is based on a Widest-Shortest Path algorithm that takes into account the hopcount. The hopcount is included in the cost function in order to avoid oscillations due to the increased number of flows sent over the widest-path. This approach is similar to the one presented by Shaikh *et al.* [74] , but instead of making traffic differentiation per flow, it uses class-based differentiation.

Hop-by-hop QoS routing strategy (UC-QoSR) was developed in [67] for networks where traffic differentiation is class-based. This strategy extends the OSPF

routing protocol to dynamically select paths adequate for each traffic class according to a QoS metric that evaluates the impact of the degradation of delay and loss at each router on application performance. The UC-QoSR strategy comprises a set of mechanisms in order to avoid routing instability. Load balancing is embedded in the strategy, since the traffic of all classes is spread over available paths. The link weights are smoothed by using a moving average of its instantaneous values. The prioritizing of routing messages is used to avoid instability due to stale routing information. Combined with these procedures, the UC-QoSR strategy uses a mechanism named class-pinning, that controls the path shifting frequency of all traffic classes. With this mechanism, a new path is used only if significantly better than the path that is currently used by that class [16].

7 Summary and Discussion

Once a suitable *QoS routing protocol* is available and each node in the network has an up to date view of the network, the challenging task in QoS routing is to find a path subject to multiple constraints. The algorithms proposed for the *multi-constrained (optimal) path* problem are discussed and their performance via simulations in the class of Waxman graphs with independent uniformly distributed link weights is evaluated. Table 1 displays the worst-case complexities of the algorithms discussed in Section 2.

Algorithm	Worst-case complexity
Jaffe's algorithm	$O(N \log N + mE)$
Iwata's algorithm	$O(mN \log N + mE)$
SAMCRA, TAMCRA	$O(kN \log(kN) + k^2 mE)$
EDSP, EBF	$O(x_2^2 \cdots x_m^2 N^2)$, $O(x_2 \cdots x_m NE)$
Randomized algorithm	$O(mN \log N + mE)$
H_MCOP	$O(N \log N + mE)$
LPH	$O(k^2 NE)$
A*Prune	$O(QN(m + N + \log h))$

Table 1. Worst-case complexities of QoS routing algorithms.

The simulation results show that the worst-case complexities of Table 1 should be interpreted with care. For instance, the actual execution time of H_MCOP will always be longer than that of Jaffe's algorithm under the same conditions. In general, the simulation results indicate that TAMCRA-like algorithms that use a k-shortest path algorithm and a nonlinear length function while eliminating dominated paths and possibly applying other search-space reducing techniques such as look-ahead perform best. The performance and complexity of TAMCRA-like algorithms is easily adjusted by controlling the value of k. When

k is not restricted, TAMCRA-like algorithms as SAMCRA lead to exact solutions. In the class of Waxman or random graphs with uniformly distributed link weights, simulations suggest that the execution times of such exact algorithms increase almost linearly with the number of nodes in $G(N, E)$, contrary to the expected exponential (NP) increase.

The study reveals that the exact algorithm SAMCRA (and likewise TAMCRA) can be extended with the look-ahead property. The combination of the four powerful concepts (non-linear definition of length, k-shortest paths, dominance and look-ahead) into one algorithm makes SAMCRAv2 the current most efficient exact QoS routing algorithm.

The second part of this chapter discussed the dynamics of QoS routing, mainly QoS routing without complete topology information and the stability of QoS routing are addressed. A probabilistic approach is discussed to incorporate the complex dynamic network processes. While the study of QoS routing algorithms has received due attention, the routing dynamics and the behavior of the QoS routing protocol deserve increased efforts because these complex processes are insufficiently understood. As a result, a commonly accepted QoS routing protocol is a still missing functionality in today's communication networks.

List of Open Issues

- Determining for which graphs and link weight structures the MC(O)P is not NP-complete.
- A detailed and fair comparison of the proposed dynamic aspects of QoS routing proposals. Usually, authors propose an idea and choose a few simulations to show the superiority of their approach compared to other proposals.
- Designing efficient QoS routing protocols.
- Aiming for an optimized QoS routing protocol.
- The deployment of QoS routing for DiffServ.
- Combined approaches of QoS routing and QoS signaling.
- QoS multicast routing.
- QoS routing implications on layer 2 technologies.
- QoS routing in Adhoc networks.

References

1. L.H. Andrew and A.A.N. Kusuma, "Generalized Analysis of a QoS-aware routing algorithm", Proc. of IEEE GLOBECOM 1998, Piscataway, NJ, USA, vol. 1, pp. 1-6, 1998.
2. T. Anjali, C. Scoglio, J. de Oliveira, L.C. Chen, I.F. Akyldiz, J.A. Smith, G. Uhl, A. Sciuto, "A New Path Selection Algorithm for MPLS Networks Based on Available Bandwidth Estimation", Proc. of QofIS 2002, pp.205-214, Zurich, Switzerland, October 2002.
3. G. Apostolopoulos, R. Guerin, S. Kamat and S.K. Tripathi, "Quality of Service Based Routing: A performance perspective", Proc. of ACM SIGCOMM '98, Vancouver, British Columbia, Canada, pp. 17-28, August/September, 1998.

4. G. Apostolopoulos, R. Guerin, S. Kamat, S.K. Tripathi, "Server Based QoS Routing", Proc. of IEEE Globecom 1999, 1999.
5. G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda and T. Przygienda, "QoS Routing Mechanisms and OSPF Extensions", RFC 2676, August 1999.
6. G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Improving QoS routing performance under inaccurate link state information", Proc. of the 16th International Teletraffic Congress (ITC '16), Edinburgh, United Kingdom, June 7-11, 1999.
7. The ATM Forum, "Private Network-to-Network Interface Specification Version 1.1 (PNNI 1.1)", af-pnni-0055.002, April 2002.
8. E. Basturk and P. Stirpe, "A hybrid spanning tree algorithm for efficient topology distribution in PNNI", Proc. of the 1st IEEE International Conference on ATM (ICATM '98), pages 385–394, 1998.
9. B. Bellur and R.G. Ogier, "A reliable, efficient topology broadcast protocol for dynamic networks", Proc. of IEEE INFOCOM'99, volume 1, pages 178 –186, 1999.
10. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
11. R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.
12. S. Chen and K. Nahrstedt, "On finding multi-constrained paths", Proc. of ICC '98, New York, pp. 874-879, 1998.
13. S. Chen and K. Nahrstedt, "Distributed QoS Routing with Imprecise State Information", Proc. of 7th IEEE International Conference of Computer, Communications and Networks, Lafayette, LA, pp. 614-621, October 1998.
14. T.M. Chen and T.H. Oh, "Reliable Services in MPLS", IEEE Communications Magazine, pp. 58-62, 1999.
15. R.K. Cheung, "Iterative methods for dynamic stochastic shortest path problems", Naval Research Logistics, 45:769–789, 1998.
16. M. Curado, O. Reis, J. Brito, G. Quadros, E. Monteiro, "Stability and Scalability Issues in Hop-by-Hop Class-based Routing", Proceedings of the 2nd International Workshop on QoS in Multiservice IP Networks (QoS-IP2003), Milano, Italy, February 24-26, 2003.
17. E.I. Chong, S. Maddila and S. Morley, "On Finding Single-Source Single-Destination k Shortest Paths", J. Computing and Information, 1995, special issue ICCI'95, pp. 40-47.
18. Y.K. Dalal and R.M. Metcalfe, "Reverse path forwarding of broadcast packets", Communications of the ACM, 21:1040–1048, December 1978.
19. H. De Neve and P. Van Mieghem, "A multiple quality of service routing algorithm for PNNI", Proc. of IEEE ATM workshop, Fairfax, May 26-29, 1998, pp. 324-328.
20. H. De Neve and P. Van Mieghem, "TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm", Computer Communications, 2000, vol. 23, pp. 667-679.
21. A. Eiger, P.B. Mirchandani, and H. Soroush, "Path preferences and optimal paths in probabilistic networks", Transportation Science, 19(1):75–84, February 1985.
22. B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights", Proc. of IEEE INFOCOM 2000, vol. 2, pp. 519-528, 2000.
23. H. Frank, "Shortest paths in probabilistic graphs", Oper. Res., 17:583–599, 1969.
24. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
25. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 1st ed., North Oxford Academic, Oxford, 1983.

26. R. Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms", IEEE/ACM Transactions on Networking, 7(3):350 –364, June 1999.
27. R. Guerin and A. Orda, "Networks with advance reservations: The routing perspective", Proc. of IEEE INFOCOM 2000, Israel, March 26-30, 2000.
28. L. Guo and I. Matta, "Search space reduction in QoS routing", Proc. of the 19th III Int. Conference on Distributed Computing Systems, III, May 1999, pp. 142-149.
29. R. Hassin, "Approximation schemes for the restricted shortest path problem", Mathematics of Operations Research, 17(1):36–42, 1992.
30. M.I. Henig, "The shortest path problem with two objective functions", European J. of Operational Research, 1985, vol. 25, pp. 281-291.
31. P.A. Humblet and S.R. Soloway, "Topology broadcast algorithms", Computer Networks and ISDN Systems, 16:179–186, 1988/89.
32. A. Iwata, R. Izmailov, D.-S. Lee, B. Sengupta, G. Ramamurthy and H. Suzuki, "ATM Routing Algorithms with Multiple QoS Requirements for Multimedia Internetworking", IEICE Transactions and Communications E79-B, no. 8, pp. 999-1006, 1996.
33. J.M. Jaffe, "Algorithms for finding paths with multiple constraints", Networks 14, pp. 95-116, 1984.
34. Y. Jia, I. Nikolaidis, and P. Gburzynski, "Multiple path routing in networks with inaccurate link state information", IEEE ICC, volume 8, pages 2583–2587. IEEE, 2001.
35. W. Jianxin, W. Weiping, C. Jianer, and C. Songqiao, "A randomized QoS routing algorithm on networks with inaccurate link-state information", Proc. of the International Conference on Communication Technology (WCC - ICCT 2000), volume 2, pages 1617–1622. IEEE, 2000.
36. A. Juttner, B. Szviatovszki, I. Mecs and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem", Proc. of IEEE INFOCOM 2001, volume 2, pages 859–868. IEEE, April 2001.
37. J. Kamburowski, "A note on the stochastic shortest route problem", Operations Research, 33(3):696–698, May-June 1985.
38. S. Keshav, *An Engineering Approach to Computer Networking: ATM networks, the Internet, and the Telephone Network*, Addison-Wesley, 1997
39. A. Khanna and J. Zinky, "The Revised ARPANET Routing Metric", Proc. of SIGCOMM'89, 1989.
40. S. Kim and M. Lee, "Server Based QoS Routing with Implicit network State Updates", Proc. of IEEE Globecom 2001, vol.4, pp. 2182-2187, San Antonio, Texas, November 2001.
41. M. Kodialam and T.V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration", Proc. of IEEE INFOCOM 2000, pp. 902-911, 2000.
42. T. Korkmaz and M. Krunz, "Source-oriented topology aggregation with multiple QoS parameters in hierarchical networks", The ACM Transactions on Modeling and Computer Simulation (TOMACS), 10(4):295–325, Oct. 2000.
43. T. Korkmaz and M. Krunz, "Hybrid flooding and tree-based broadcasting for reliable and efficient link-state dissemination", Proc. of IEEE GLOBECOM '02 Conference - High-Speed Networks Symposium, Nov. 2002.
44. T. Korkmaz and M. Krunz, "Bandwidth-delay constrained path selection under inaccurate state information", to appear in IEEE/ACM Transactions on Networking, 2003.
45. T. Korkmaz and M. Krunz, "A randomized algorithm for finding a path subject to multiple QoS requirements", Computer Networks, vol. 36, pp. 251-268, 2001.

46. T. Korkmaz and M. Krunz, "Multi-Constrained Optimal Path Selection", Proc. of IEEE INFOCOM 2001.
47. F.A. Kuipers and P. Van Mieghem, "QoS routing: Average Complexity and Hopcount in m Dimensions", Proc. of Second COST 263 International Workshop, QofIS 2001, Coimbra, Portugal, pp. 110-126, September 24-26, 2001.
48. F.A. Kuipers and P. Van Mieghem, "MAMCRA: A Constrained-Based Multicast Routing Algorithm", Computer Communications, vol. 25/8, pp. 801-810, May 2002.
49. F. A. Kuipers and P. Van Mieghem, "The Impact of Correlated Link Weights on QoS Routing", to appear in Proc. of IEEE INFOCOM 2003.
50. E.L. Lawler, *Combinatorial Optimization: networks and matroids*, New York: Holt, Rinehart and Winston, 1976.
51. W.C. Lee, M.G. Hluchyi and P.A. Humblet, "Routing Subject to Quality of Service Constraints in Integrated Communication Networks", IEEE Network, pp. 46-55, July/August, 1995.
52. B. Lekovic and P. Van Mieghem, "Link State Update Policies for Quality of Service Routing", Proc. of IEEE Eighth Symposium on Communications and Vehicular Technology in the Benelux (SCVT2001), Delft, The Netherlands, pp. 123-128, October 18 2001.
53. G. Liu and K.G. Ramakrishnan, "A*Prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints", Proc. of IEEE INFOCOM 2001.
54. R.P. Loui, "Optimal paths in graphs with stochastic or multidimensional weights", Communications of ACM, 26(9):670–676, 1983.
55. 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.
56. Q. Ma and P. Steenkiste, "Quality-of-Service Routing with Performance Guarantees", Proc. of 4th Int. IFIP Workshop on QoS, May 1997.
57. Q. Ma and P. Steenkiste, "Supporting Dynamic Inter-Class Resource Sharing: A Multi-Class QoS Routing Algorithm", Proc. of IEEE INFOCOM 1999.
58. X. Masip, S. Sánchez, J. Solé and J. Domingo, "A QoS Routing Mechanism for Reducing Inaccuracy Effects", Proc. of QoS-IP, Milán, Italy, February 2003.
59. E.D. Miller-Hooks and H.S. Mahmassani, "Least expected time paths in stochastic, time-varying transportation networks", Transportation Science, 34(2):198–215, May 2000.
60. P.B. Mirchandani, "Shortest distance and reliability of probabilistic networks", Comput. & Ops. Res., 3:347–355, 1976.
61. P.B. Mirchandani and H. Soroush, "Optimal paths in probabilistic networks: A case with temporal preferences", Comput. and Operations Research, 12(4):365–381, 1985.
62. R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
63. I. Murthy and S. Sarkar, "Exact algorithms for the stochastic shortest path problem with a decreasing deadline utility function", European Journal of Operational Research, 103:209–229, 1997.
64. I. Murthy and S. Sarkar, "Stochastic shortest path problems with piecewise-linear concave utility functions", Management Science, 44(11):S125—S136, Nov. 1998.
65. K. Nahrstedt and S. Chen, "Coexistence of QoS and Best Effort Flows - Routing and Scheduling", Proc. of 10th IEEE Tyrrhenian International Workshop on Digital Communications: Multimedia Communications, Ischia, Italy, September, 1998.

66. S. Nelakuditi, Z. Zhang, R.P. Tsang, "Adaptive Proportional Routing: A Localized QoS Routing Approach", Proc. of IEEE INFOCOM 2000, pp. 1566-1575.
67. M. Oliveira, J. Brito, B. Melo, G. Quadros, E. Monteiro, "Quality of Service Routing in the Differentiated Services Framework", Proceedings of SPIE's International Symposium on Voice, Video, and Data Communications (Internet III: Quality of Service and Future Directions), Boston, Massachusetts, USA, November 5-8, 2000.
68. A. Orda, "Routing with End-to-End QoS Guarantees in Broadband Networks", IEEE/ACM Transactions on Networking, vol. 7, no. 3, pp. 365-374, 1999.
69. A. Orda and A. Sprintson, "QoS routing: the precomputation perspective", Proc. of IEEE INFOCOM 2000, pp. 128-136, 2000.
70. M. Peyravian and A.D. Kshemkalyani, "Network path caching: Issues, algorithms and a simulation study", Performance Evaluation, vol. 20, no. 8, pp. 605-614, 1997.
71. G.H. Polychronopoulos and J.N. Tsitsiklis, "Stochastic shortest path problems with recourse", Operations Research Letters, 10:329–334, August 1991.
72. D.S. Reeves and H.F. Salama, "A distributed algorithm for delay-constrained unicast routing", IEEE/ACM Transactions on Networking, 8(2):239–250, April 2000.
73. E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.
74. A. Shaikh, J. Rexford and K. Shin, "Load-Sensitive Routing of Long-Lived IP Flows", Proc. of ACM SIGCOMM'99.
75. H.F. Salama, D.S. Reeves and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks", IEEE JSAC, 15(3), pp. 332-345, April 1997.
76. S. Sen, R. Pillai, S. Joshi, and A.K. Rathi, "A mean-variance model for route guidance in advanced traveler information systems", Transportation Science, 35(1):37–49, Feb. 2001.
77. C.E. Sigal, A.A.B. Pritsker, and J.J. Solberg, "Stochastic shortest route problem", Operations Research, 28(5):1122–1129, Sept.-Oct. 1980.
78. R.A. Sivakumar and R. Batta, "The variance-constrained shortest path problem", Transportation Science, 28(4):309–316, Nov. 1994.
79. N. Taft-Plotkin, B. Bellur and R. Ogier, "Quality-of-Service routing using maximally disjoint paths", Proc. of the Seventh International Workshop on Quality of Service (IWQoS'99), London, England, pp. 119-128, May/June, 1999.
80. P. Van Mieghem, H. De Neve and F.A. Kuipers, "Hop-by-Hop Quality of Service Routing", Computer Networks, vol. 37/3-4, pp. 407-423, October 2001.
81. P. Van Mieghem, "Paths in the simple Random Graph and the Waxman Graph", Probability in the Engineering and Informational Sciences (PEIS), vol. 15, pp. 535-555, 2001.
82. P. Van Mieghem, "Topology Information Condensation in Hierarchical Networks", Computer Networks, Vol.. 31, no. 20, pp. 2115-2137, November, 1999.
83. P. Van Mieghem and H. De Neve, 1998, "Aspects of Quality of Service Routing", Proc. of SPIE'98, Nov. 1-6, Boston (USA), 3529A-05, 1998.
84. P. Van Mieghem, "Estimation of an optimal PNNI topology", Proc. of IEEE ATM Workshop, pp. 570–577, 1997.
85. S. Vutukury and J.J. Garcia-Luna-Aceves, "A Simple Approximation to Minimum-Delay Routing", Proc. of ACM SIGCOMM'99.
86. B. Wang and J.C. Hou, "Multicast routing and its QoS extension: problems, algorithms, and protocols", IEEE Network, vol. 14, no. 1, pp. 22-36, Jan.-Feb 2000.
87. Z. Wang and J. Crowcroft, "Shortest Path First with Emergency Exits", Proc. of SIGCOMM'90, Philadelphia, USA, September 1990.

88. Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications", IEEE JSAC, vol. 14, no. 7, pp. 1228-1234, September, 1996.
89. J. Wang and K. Nahrstedt, "Hop-by-hop Routing Algorithms for Premium-class Traffic in Diffserv Networks", Proc. of IEEE INFOCOM 2002.
90. B.M. Waxman, "Routing of multipoint connections", IEEE JSAC, 6(9):1617-1622, december 1998.
91. X. Xiao and L.M. Ni, "Internet QoS: A big picture", IEEE Network, vol. 13, no. 2, pp. 8-18, March-April 1999.
92. X. Yuan, "Heuristic Algorithms for Multiconstrained Quality-of-Service Routing", IEEE/ACM Transactions on Networking, vol. 10, no. 2, April 2002.