ELSEVIER

Computer Networks 40 (2002) 149–165

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Comparative analysis of path computation techniques for MPLS traffic engineering

Gargi Banerjee [*], Deepinder Sidhu

Department of Computer Science and Electrical Engineering, Maryland Center for Telecommunications Research, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA

Abstract

We consider the problem of computing traffic engineered paths for requests having bandwidth and delay requirements, when these requests arrive in the network independent of one another. Providing bandwidth guarantees to applications has been important in networks offering service differentiation. With the increase in the number of real-time applications in the Internet, provision of delay guarantees is also receiving much attention. This necessitates the development of sophisticated path selection algorithms which deviate from the shortest-path routing philosophy in traditional IP networks. While these algorithms perform well from the perspective of satisfying application requirements, they often do not take into account long term effects on the network state. One of the major concerns of a service provider is to run the network at maximum utilization while reducing network costs and preventing congestion in the network. For this reason, providers are looking at traffic engineering (TE) to automate path selection procedures and to maintain network loading at an optimal level. In this paper we propose two TE path selection algorithms that consider the application's delay–bandwidth requirements as well as the TE constraints on the network. We compare the proposed algorithms to existing path computation solutions and present results that show that by considering these additional constraints, improvement is achieved in terms of reduction in request blocking probability, reduction in network costs and load distribution.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Traffic engineering; Path computation; Multi-constrained path; QoS; MPLS

1. Introduction

The exponential growth of the Internet has led to the increasing importance of network management and control functions. It is evident today that adding more bandwidth to networks is not the solution to all congestion problems. At the same time, more and more providers are showing interest in making revenues from offering differentiation of services in their networks. This requirement has increased the importance of gaining control over networks via automated traffic engineering (TE). The most common TE objectives are to reduce congestion, improve network utilization, satisfy diversified requirements and thus lead to an increase of revenue.

[*] Corresponding author. Tel.: +1-410-455-3063/2860.
E-mail addresses: gargi@mctr.umbc.edu (G. Banerjee), sidhu@umbc.edu (D. Sidhu).

One of the most important applications of MPLS networks will be in TE [2]. MPLS traffic engineering (MPLS-TE) enables a MPLS backbone to expand upon TE capabilities by routing flows across the network based on the resources the flow requires and those currently available. Since the essence of TE is mapping traffic flows onto a physical topology, it implies that at the heart of MPLS-TE resides the problem of path computation.

The Internet Protocol (IP), and the architecture of the Internet itself, is based on the simple concept of best-effort service and makes no guarantees about when data will arrive, or how much it can deliver. Though the simplicity and scalability of IP has made it highly popular, it does not provide a wide range of services. This has not been a problem in the past with most of the applications being of types Web, email and file transfer. However, as more and more applications connect to the Internet, the nature of service required by them is becoming increasingly varied. Applications now require more predictability from the networks and in turn are potential sources of revenue to network service providers. Thus, in order to increase revenue earnings, the providers need to traffic engineer their networks to help them run at maximum utilization. For this reason a number of QoS and TE mechanisms have evolved over the years to satisfy the variety of needs of applications as well as that of service providers.

QoS based path computation has assumed a lot of importance over the last couple of years. In the simplest terms, to offer QoS means to provide consistent, predictable data delivery service. Customers or users of a network may need several types of QoS guarantees from the network and they specify these requirements in the form of service level agreements (SLA). QoS path computation then translates to finding paths through the network such that these requirements are satisfied. While previous path computation algorithms were mainly hop-by-hop distributed algorithms optimizing static metrics (e.g., link cost, length, etc.), the new generation of path algorithms is moving towards source routing schemes that also take into account dynamic metrics. A few examples of these dynamic metrics are available bandwidth on links, link reliability, link load, jitter, packet loss, etc. In addition, due to the varied QoS requirements of today's applications, path computation algorithms may need to work with multiple QoS metrics. Path computation based on a single QoS metric is studied in [1,6,13,16]. Bandwidth is the most common QoS metric considered in these algorithms. A more difficult task is to solve the problem of computing paths when multiple constraints need to be satisfied. Path computation based on multiple QoS constraints has been studied in [4,9, 10,17,18]. In this case, applications express their service requirements not only in terms of the bandwidth required, but also in terms of maximum delay tolerated, delay variation, reliability, etc. Among the multiple QoS metrics, the principal ones are bandwidth and delay. Most applications express their requirements either in terms of the minimum/average bandwidth required or in terms of the minimum bandwidth and the maximum acceptable end-to-end delay. We refer to the above problems as the *bandwidth constrained* and the *delay–bandwidth constrained* path computation problems, respectively.

QoS schemes essentially provide differentiation of services. They also provide different degradation of performance for different traffic when traffic load is heavy. Under low-load conditions however best-effort schemes work just as well. Thus the question arises as to whether we can maintain the network always in a state of optimized load and avoid congestion. This is the motivation for TE. TE helps in maintaining networks in a well-balanced state. This in turn makes it easier to satisfy the QoS requirements of flows routed across it. Thus TE in ISP networks, and providing quality of service guarantees to customers that use the network, are closely related to each other. Some of the common TE objectives are to reduce the blocking probability in a network, reduce congestion on links, reduce network costs, etc. These objectives translate into new restrictions on the path selected and TE path computation algorithms should satisfy these objectives in order to maintain consistently high network utilization. For example, if the QoS metrics considered are delay and bandwidth and the TE objective in a provider's network is to minimize network costs, then the problem translates to finding the minimum cost delay–bandwidth constrained path. We denote the QoS requirements specified by applications by the general term *Q-metrics*. Without loss of generality, we call the problem of finding paths that satisfy the Q-metric constraints and as well maintain the additional TE objectives, the *TE-Q-metrics constrained* path computation problem. If the application specified Q-metrics include only bandwidth, then we call it the *TE-bandwidth constrained* problem. Otherwise, if the application specified Q-metrics are delay and bandwidth then we call it the *TE-delay–bandwidth constrained* problem.

In this paper, we present two TE path computation heuristics, which solve the TE-bandwidth (TE-B) and the TE-delay–bandwidth (TE-DB) constrained problems respectively. Both these algorithms attempt to maintain the three TE objectives of (1) increasing network revenue, (2) limiting network costs and (3) distributing network load. We compare the performance of these two algorithms with other existing competitive algorithms that use the same application specified constraints. We show that by considering the additional TE constraints, both algorithms achieve considerable performance benefits and provide more complete TE solutions. The rest of the paper is organized as follows. Section 2 introduces concepts in TE based path computation and presents the proposed path computation algorithms. Section 3 reports simulation results and performance analysis of the proposed algorithms and Section 4 concludes the paper.

2. TE-Q-metrics constrained path computation

Routing models can be broadly classified as online routing (i.e., requests arrive one by one and future requests are not known) or offline routing (i.e., all demands are known a priori and requests can be routed accordingly). There also exists an intermediate model in which (some statistical knowledge about the requests may be available, e.g., the average arrival rate for a node pair. The advantage of offline routing, is that pre-provisioning of resources can be done so that the maximum number of requests can be routed. However, with the advent of services that permit dynamic and frequent requests for capacity change, online routing of requests has assumed a great importance.

Online routing algorithms need to be pretty fast since they compute route(s) for every request arrival in the system. The time complexity of a routing algorithm is often determined by the nature and the number of path metrics considered. Commonly path metrics can be divided into three classes: additive, multiplicative and concave. Bandwidth is an example of a concave metric, whereas delay, cost, etc. are additive metrics. More background on path metrics can be found in [19]. Problems dealing with one or more concave metrics and one additive/multiplicative problem can be solved in polynomial time. However, the problem of computing optimal routes subject to constraints of two or more additive and/or multiplicative metrics is known as the *multiple constrained path selection* (MCP) [14]. The MCP decision problem is known to be NP-complete [7]. Related to MCP, but with a slightly different objective is the *restricted shortest-path* (RSP) problem. In RSP, the path is required to satisfy one constraint while being optimal with respect to another parameter. Any solution to RSP applies to the MCP problem. RSP is also known to be NP-complete [14]. The difference between the MCP and the RSP problem is that MCP does not optimize the values of any parameter. Instead, it searches for a feasible path that satisfies both constraints.

In this paper, we assume requests arrive online and no assumption is made about future requests. We assume that all applications specify their service requirements in terms of two QoS metrics: bandwidth and delay. Two forms of QoS requests are considered entering the network:

1. (A, B, Bw) where A: source node, B: destination node and Bw: minimum bandwidth the application requires.
2. (A, B, Bw, D) where A: source node, B: destination node, Bw: minimum bandwidth the application requires and D: maximum end-to-end delay the application can tolerate.

We assume that the requested bandwidth units are reserved along path $P_{A\text{–}B}$ by some TE signaling mechanism like RSVP or CR-LDP [3,11] and are available for the application. We also assume that the available QoS information is accurate.

2.1. TE objectives

The TE objectives considered in this paper are to minimize blocking of future requests (and thereby earn more revenue), minimize the overall cost of paths and distribute the loading on paths. We now take a closer look at these TE objectives.

2.1.1. Reducing blocking of requests

Reducing the blocking probability of requests in a network is a crucial TE objective. In [13], the authors present the intuition that the blocking probability of requests can be reduced, if the total available flow in the network is maximized. The assumption is that the set of transmitting source–destination (src–dest) node pairs is known. The total amount of available flow between a src–dest pair is known as the *maximum flow* (max-flow) [5] between the pair of nodes. Associated with each max-flow is a set of *minimum cut links* (min-cut) [5]. The max-flow of a particular src–dest pair decreases whenever the capacity of any of the links in its min-cut for that pair decreases infinitesimally. When a request R is routed over a path P between a particular src–dest pair s–d, the max-flow between s–d always decreases. However the request may also go over links that belong to the min-cut set of other src–dest pairs. Thus routing this request R over P may also decrease the max-flow between other src–dest pairs. The approach attempts to route R over a path, which maximizes the available flow between other src–dest pairs. The problem is NP-hard and the authors propose a heuristic algorithm that gives reasonably good solutions. For every request of Bw units of bandwidth between a pair of src–dest nodes s–d, the heuristic algorithm assigns a weight to each link in the network. This weight, which we call the *max-flow reduction weight*, is proportional to the amount of flow it will reduce between all other src–dest pairs, if the flow of Bw units is routed across it. The larger amount of flow that a link reduces, the higher its assigned weight is. A shortest-path algorithm based on the new weights then returns the minimum flow-blocking path. In this paper, we achieve the TE objective of reducing the number of blocked requests by adopting the above approach of routing requests along the least flow-blocking paths. If $f(i,j)$ denotes the max-flow reduction weight of a link $l(i,j)$, and P is a path, then we define the max-flow reduction weight of path P as

$$\text{path\_flow\_reduction}(P) = \sum f(i,j), \quad \forall l(i,j) \in P.$$

Note that if the bandwidth demands in the requests are much smaller compared to the capacity of the links in the network, the max-flow values and the min-cut links do not change much. In that respect max-flow and min-cut are quasi-static metrics in such a network.

2.1.2. Minimizing network cost

In this paper, we assume that network usage costs are solely dependent on the cost of the paths being used. Thus, the TE objective of minimizing network costs directly translates to the problem of minimizing the path cost. Path cost is a static metric and is expressed as the sum of its component link costs. If $u(i,j)$ denotes link cost for link $l(i,j)$ and P is a path, then we define cost of P as

$$\text{path\_cost}(P) = \sum u(i,j), \quad \forall l(i,j) \in P.$$

Static TE metrics help in maintaining network stability under high-load conditions. Path cost is often directly related to path length. Motivation for using the static path length as a TE constraint is derived from the findings reported in [16] that show that algorithms that optimize path length perform better under high-load conditions than algorithms that do not.

2.1.3. Distributing network load

The TE objective of distribution of network load is a difficult objective to meet since it does not directly translate to optimizing a path metric. However, we note that distribution of network load is a dynamic function and hence depends on some dynamic path metric [16] shows that balancing network load by using a dynamic link metric works well, especially under low-load conditions.

2.1.3.1. Path criticality. We now define a TE metric, path criticality, which we propose to use for load distribution. The idea behind our metric is that if there exists highly loaded links in a network then these links should be avoided during selection of paths for requests. These heavily loaded links are called *critical links*. All critical links in a path contribute towards the criticality of the path.

We can rely on existing TE infrastructure to provide the loading information on links. The basic assumption is that most networks run link state Interior Gateway Protocols (IGP) for routing purposes. [12] suggests simple extensions to IGPs by which dynamic TE information like reserved link bandwidth can be periodically fed-back to the source nodes. This information is used for identification of heavily loaded links. We define link loading of a link $l(i,j)$ as

$$\text{link\_load}_{(l)} = (\text{Reserved } Bw \text{ on } l/\text{Total reservable } Bw \text{ on } l) \times 100.$$

A critical link is now defined as a link which has its load running above a threshold percentage U. Criticality of link l is defined as

$$\begin{aligned}\text{link\_critical}_{(l)} &= 0, && \text{if link\_load}_{(l)} \leqslant U,\\ &= f(\text{link\_load}_{(l)}), && \text{otherwise,}\end{aligned}$$

where f: monotonically increasing function of $\text{link\_load}_{(l)}$.

A critical path is one which has critical component links. The more heavily a path is loaded, the more critical it is. Also the higher the number of critical component links on a path, the more critical it is. From these ideas, we now define criticality of a path P as

$$\text{path\_critical}(P) = \sum \text{link\_critical}_{(l)}, \quad \forall l \in P.$$

Intuitively, thus to distribute path loading one would select the least critical paths. Criticality is an additive metric.

Consider the example network in Fig. 1. The numbers beside the links represent the loading on the corresponding links. Let the threshold U be fixed at 28. The dark lines then represent the critical links. By our definition, the three paths ranked in ascending order of path criticality are A–B–D–E, A–D–E and A–C–D–E. Note that by definition, path criticality differs from path load. Path load is a concave metric and is defined as the maximum link load on its component links. In this sense, criticality is a more fine-tuned metric as compared to load since it depends on the loading on all component links. The difference between the two metrics can be seen in Fig. 1, where all the three paths have the same path load but differ in path criticality. We later show through our experiments that the overall distribution of load attained by using the path criticality metric is much better than using the path load metric.

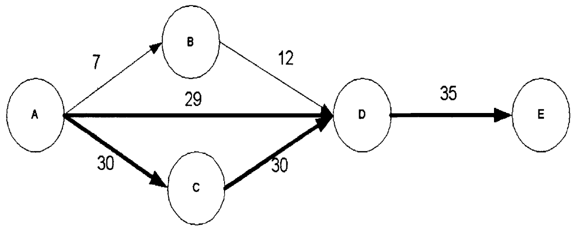


Fig. 1. Example network showing critical links.

2.2. TE-Q-metrics problem statement

We now formally define our TE-Q-metrics constrained path computation problem. Given a directed network G of N nodes and L links, each link $l \in L$ has a pre-assigned non-negative link delay parameter $d(l)$ and a non-negative link cost parameter $c(l)$. The constraints placed on a path P, are derived from the application specified Q-metrics. In this paper, the Q-metrics considered are the minimum bandwidth requirement (Bw) and the maximum end-to-end delay (D) between a given source s and destination d. Let us denote the available bandwidth and the delay on path P as bandwidth(P) and delay(P) respectively. Additionally, the path has to satisfy the modified TE objectives of minimizing flow reduction, minimizing cost and minimizing path criticality in the network. Thus, for any path P, the problem statement can be written as

(a) minimize path_flow_reduction(P),
(b) minimize path_cost(P),
(c) minimize path_critical(P),

subject to the constraints:

1. delay$(P) \leqslant D$,
2. bandwidth$(P) \geqslant Bw$.

The above formulation of the *TE-Q-metric* constrained problem has multiple objective functions and constraints, of which only one of them (i.e., bandwidth) is a concave path metric and the other four, are additive metrics. Thus, it is a variation of the RSP problem and is NP-complete. We now attempt to investigate suitable heuristics that find good solutions to the above problem.

2.3. Single QoS constraint

When application's QoS requirements are specified only in terms of bandwidth, the TE-Q-metric problem reduces to the TE-B constrained problem. Before we present our heuristic to solve this problem, we discuss some existing QoS path computation algorithms that take into account the bandwidth constraint. We also highlight some of the reasons why these existing solutions fail to qualify as an overall TE solution.

The minimum interference routing algorithm (MIRA) [13] heuristic was proposed to find a path between a pair of nodes that blocks the smallest available max-flow between all other src–dest nodes. MIRA's basic intuition is to avoid routing requests on links that reduce the max-flow between other src–dest pairs. The interested reader is referenced to [13] for details. Note that none of the static constraints (e.g., path length, path cost) is considered in MIRA. While at low to moderate network load this might be acceptable, longer paths selected at high-load conditions may cause instability by occupying more resources in an already overloaded network. In addition, with this type of algorithm, the path lengths can become long enough to make the path practically unusable. MIRA also does not take into account the current load condition on the paths. In an n-node network with m links and integral link capacities in range $[1, U]$, the time complexity of the algorithm is $O(nm + n^2 \log U)$ [13] and its space complexity is $O(n)$.

The widest shortest (WID-SHORT) path algorithm [16] makes use of the possible existence of more than one shortest path in the network. If several such shortest paths exist, then the widest one (i.e., one with the maximum available bandwidth) is selected. The effectiveness of this algorithm depends on the presence of multiple shortest paths in the network. If there is always only a single shortest path, then its performance becomes identical to that of any minimum distance routing algorithm. From the TE perspective, WID-SHORT considers the path cost and path load but does not take into account the blocking of future requests. The time complexity is $O(n \log n)$ and its space complexity is $O(n)$. A similar algorithm, the shortest-widest path selects the shortest path among paths with equal amount of available bandwidth. However [16] shows that at high-load conditions the performance of the shortest-widest path algorithm deteriorates severely.

Least-critical-K-shortest (LCKS) path algorithm is proposed in this paper as an improvement to the WID-SHORT algorithm. Since the occurrence of multiple shortest paths may be rare in many networks, we propose to find the k-shortest paths between a pair of nodes. LCKS uses the proposed path criticality metric to distribute loading, rather than the path load metric. From the set of k candidate paths, it selects the least critical path. The time complexity of the algorithm is $O(n \log n + k)$ and space complexity is $O(kn)$. All the algorithms described above perform well either in terms of reducing blocking or in terms of reducing path cost/load. The objective is to look for a more complete solution that succeeds in maintaining all TE objectives. We propose the TE-B constrained algorithm as a complete solution to the TE path computation problem.

2.3.1. TE-bandwidth constrained path computation

This section presents our path computation heuristic, the TE-B constrained algorithm. TE-B uses the quasi-static, static and dynamic TE metrics of max-flow reduction, path cost and path load respectively, in addition to the application specified QoS constraint of bandwidth. Let the minimum bandwidth requirement of the application be Bw units. We assume the bandwidth demands are much smaller compared to the capacity of the network. The set of constraints and objective functions for the system are same as presented in Section 2.2, with only the delay constraint removed. As mentioned before, this is a version of the RSP problem. [9] suggests that the MCP problem is easier to solve than the RSP problem. Thus, in order to simplify the problem, we transform some of the objective functions into constraints. By our assumption, cost and max-flow are static and quasi-static metrics respectively. Since static optimization metrics can be transformed into constraints, we introduce a new path cost constraint (C) and a new max-flow reduction constraint (F). Since path criticality is a dynamic metric we retain the minimization criterion on it. Thus, we re-write the problem statement as

(a) minimize path_critical(P), subject to constraints:

1. $\text{bandwidth}(P) \geqslant Bw$,
2. $\text{path\_flow\_reduction}(P) \leqslant F$,
3. $\text{path\_cost}(P) \leqslant C$.

The next step is to define the constraints C and F, which could either be statically specified by an administrator or selected dynamically. In our design, they are defined as dynamic constraints, with C being the cost of the path with least max-flow reduction weight, and F being the max-flow reduction weight of the least cost path. This serves as loose upper bounds and the aim is to find paths whose cost and max-flow reduction weight metrics are far from the bounds C and F respectively. We now try to design a heuristic which give near-optimal solutions to the above problem. We use the definitions shown in Fig. 2 for our algorithm.

We propose a heuristic, TE-B (shown in Fig. 3a), that firsts finds a candidate set of paths that satisfy constraints 1–3 simultaneously. From this set, it then selects the least critical path. To satisfy constraint 1, links that have less than Bw units of residual bandwidth are removed. We now need to find a candidate set of paths that satisfy both constraints 2 and 3 simultaneously. Ref. [15] describes the *A*Prune* algorithm for finding k shortest-paths subject to multiple constraints. However the worst case running complexity can become exponential. Ref. [17] describes a heuristic for solving the MCP problem by finding a candidate set of k paths whose metrics are far from the constraint bounds. We adopt this approach for solving the TE-B constrained problem. The steps for computing the set $C_{A,B}$ and the candidate path set A are the most computationally expensive steps in the algorithm and governs the time complexity

$N(n, m)$	: Network of n nodes and m links
L	: Link set
S	: Given set of all src-dest pairs
θ_{ab}	: Max-flow between pair (a, b), where $(a, b) \in S$
$C_{a,b}$	: Union of all minimum cut links between (s,d), *for all* $(s, d) \in S$ *and* $(s, d) \neq (a, b)$
U	: Load Threshold
$l(i, j)$	: Link from node i to j
CN	: Union of critical links of a network N where $l(i, j)$ is a critical link, if $link\_load(l(i, j)) > U$
$d(i, j)$	: Static delay metric for $l(i, j)$
$u(i, j)$	: Static cost metric for $l(i, j)$
$r(i, j)$	: Residual/available bandwidth on $l(i, j)$

For each QoS request of the form (A, B, Bw), and $\forall\, l(i, j) \in L$, compute weights $f(i, j)$ and $w(i, j)$:

$$f(i, j) \quad : \sum_{(s,d):l(i,j)\in C_{sd}} \alpha_{sd}, \quad where,\ \alpha_{sd} = 1/\theta_{sd}$$

$$w(i, j) \quad : \beta(i, j) \quad \text{if } l(i, j) \in CN, \text{ where } \beta(i, j) \propto link\_load(l).$$

$$: 0 \quad \text{if } l(i, j) \notin CN$$

Criticality of a path P is defined as: $W(P) = \sum w(i, j),\ \forall\, l(i, j) \in P$

Fig. 2. Definitions and terminology.

```
Algorithm: For each QoS request (A, B, Bw):
    1. Compute set $C_{A,B}$ and weights f(i, j)
    2. Prune off links that have r(i, j) < Bw
    3. A = k-shortest path set computed based on constraints C and F
    4. Compute set CN and weights w(i, j)
    5. From among k paths in A, select least critical path based on W(P);
Output: TE-bandwidth constrained path
Time Complexity: $O(min(n^{2/3}, m^{1/2}) m log(n^2/m) log U + kn log(kn))$
(a)
```

```
Algorithm: For each QoS request (A, B, Bw, D):
    1. Compute set $C_{A,B}$ and weights f(i, j)
    2. Prune off links that have r(i, j) < Bw
    3. A = k-shortest path set computed based on constraints D, C and F
    4. Compute set CN and weights w(i, j)
    5. From among k paths in A, select least critical path based on W(P);
Output: TE-bandwidth constrained path
Time Complexity: $O(min(n^{2/3}, m^{1/2}) m log(n^2/m) log U + kn log(kn))$
(b)
```

Fig. 3. (a) TE-B and (b) TE-DB.

of the TE-B algorithm. In an n-node network with m links and integral link capacities in range $[1, U]$, the fastest known max-flow algorithm has a running time of $O(\min(n^{2/3}, m^{1/2})m\log(n^2/m)\log U)$ [8]. A set of paths subject to two simultaneous constraints can be computed in time $O(kn\log(kn) + k^3 2m)$ [17]. The space complexity of the algorithm is $O(kn)$. On an average, to compute a TE-B route takes 0.15–0.2 ms on a Sun SPARC Ultra Workstation ($n = 18, m = 62, k = 4$), whereas normal Dijkstra-based routing takes about 0.05–0.09 ms.

The performance of the algorithm also depends on the appropriate definition of the constraints C and F. The quality of a solution may deteriorate in case the bounds are too loose. A tighter bound improves the probability of the algorithm finding the optimal solution. We can use an iterative method detailed in [9] that starts with the loose upper bound and progressively searches for a tighter cost bound. When this method is run as a pre-cursor to the algorithm, the search space reduces resulting in better quality solutions.

2.4. Multiple QoS constraints

When application's QoS requirements are specified in terms of bandwidth and delay, the TE-Q-metric problem reduces to the TE-DB constrained problem. Before we present our solution to this problem we discuss some existing QoS path computation algorithms that take into account the multiple constraints of delay and bandwidth. We also highlight some of the reasons why these existing solutions fail to qualify as an overall TE solution.

Minimum delay (MIN-DELAY) algorithm [18] is a bandwidth–delay based routing algorithm that prunes links that do not satisfy the bandwidth requirement, and then finds the shortest path w.r.t. delay in the reduced graph. In an n-node network with m links, time complexity of MIN-DELAY is the same as that of Dijkstra's algorithm, i.e., $O(n\log n)$ and its space complexity is $O(n)$. This algorithm considers none of the TE constraints.

Tunable accurate multiple constrained routing algorithm (TAMCRA) [17] is an efficient multiple QoS routing algorithm, for solving the MCP algorithm with K constraints. Using a non-linear weight function, it finds a candidate set of k paths whose metrics are far from the constraint bounds. In an n-node network with m links, time complexity of TAMCRA is $O(kn\log(kn) + k^3Km)$ [17] where K is the number of constraints and k is the size of the candidate set. Space requirement of TAMCRA is $O(kn)$. This algorithm does not consider the TE constraints of reducing blocking of requests or that of distributing load.

2.4.1. TE-delay–bandwidth constrained path computation

This section presents our path computation heuristic, the TE-DB constrained algorithm. It is very similar to TE-B constrained problem formulation, except that we now have an additional delay constraint. We use the same set of definitions as in Fig. 2. Let the minimum bandwidth requirement of the application be Bw units and the maximum acceptable delay be D units. As explained in Section 2.3.1, let C and F represent the cost and the max-flow reduction constraints respectively. The problem statement becomes:

(a) minimize path_critical(P), subject to constraints:

1. bandwidth$(P) \geqslant Bw$,
2. delay$(P) \leqslant D$,
3. path_flow_reduction$(P) \leqslant F$,
4. path_cost$(P) \leqslant C$.

We propose a heuristic TE-DB (shown in Fig. 3b) to solve the above problem. The algorithm is similar to the algorithm described in Section 2.3.1 except that we now have constraints 2–4 that we need to satisfy simultaneously. As before, the algorithm works by first pruning off links which have less than Bw units of residual bandwidth. It then runs the k-shortest-path algorithm on the reduced graph to find a candidate set of paths that satisfy constraints 2–4. From among the candidate set the algorithm selects the least critical path. As before, max-flow computations take $O(\min(n^{2/3}, m^{1/2}) m \log(n^2/m) \log U)$, and computing a set of paths subject to three simultaneous constraints takes $O(kn \log(kn) + k^3 3m)$. TE-DB's space complexity is $O(kn)$. On an average, to compute a TE-DB route takes 0.2–0.25 ms on a Sun SPARC Ultra Workstation ($n = 18$, $m = 62$, $k = 4$).

3. Performance studies

We evaluate the performance of TE-B and TE-DB with other algorithms that consider similar QoS constraints. We compare TE-B with WID-SHORT, MIRA and LCKS. We also compare the performance of TE-DB with that of MIN-DELAY and TAMCRA. Performance comparison is done on the basis of stochastic simulations, where connection requests randomly arrive in the network. Each algorithm attempts to route these connections and the network performance (in terms of metrics discussed in Section 3.3) is measured.

3.1. Simulation model

The network model used for our experiments is shown in Fig. 4. It represents a typical ISP network and is based on the ATT backbone network with a few additional links. The dark shaded nodes represent the gateway nodes, which serve as entry and exit points for the network traffic. The remaining nodes are the backbone nodes, which carry transit traffic only. Link capacities are either moderate (OC3 links) or high (OC12 links). All links are symmetric and link weights are assigned randomly. The link delays, also assigned randomly, are of the order of a few milliseconds. In reality, link delays and weights vary largely from one provider's network to another. However, there exists a range of acceptable values for these

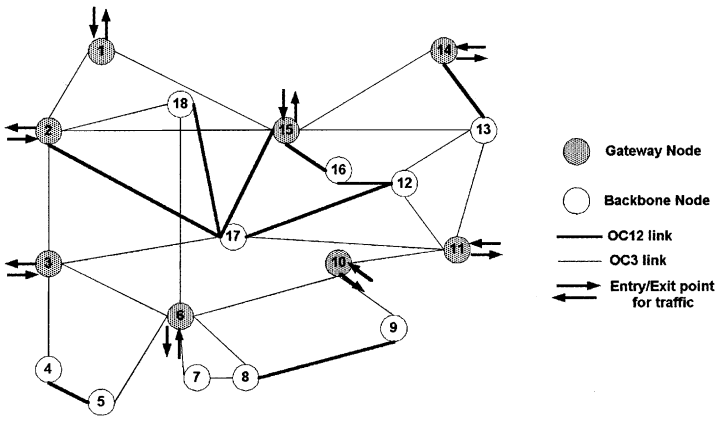


Fig. 4. ISP network.

parameters that are used in most studies (e.g., link delays for OC3 links are usually in the range of 10–100 ms). In the absence of real values, we simulate the link parameters by generating them randomly from the acceptable range of values. Experiments are repeated several times to average out the effect of randomness. In order to corroborate the results, experiments are also performed on random networks consisting of 10–40 nodes with average node degrees varying in the range of 4–6. Similar results are obtained for these random networks and hence are not shown. The parameter k determines the size of the candidate set of paths. The more candidate paths we have, better quality solutions we can attain. However the complexity of the algorithm also increases with k. Ref. [17] suggests 4–6 as a possible range of k values. We use $k = 4$ for all our experiments.

3.2. Request generation

Requests are generated by an offline utility and are of the form (Src, Dest, Bw, D, Hold-Time). The bandwidth requirement of requests varies uniformly between 1 and 200 kbits/s. The delay requirement varies uniformly between 50 and 100 ms. We assume that requests arriving in the system can either have a finite lifetime, i.e., the flows are torn down after certain holding time, or they can have infinite lifetime, i.e., flows are never torn down. Finite flows arrive in the system according to a Poisson process and the holding time is exponentially distributed.

The request generation follows two models, *uniform* and *non-uniform*. For each network, S represents the set of possible src–dest pairs between which there is traffic flow. We assume that this set S is known a priori. In the uniform model, requests are uniformly distributed between all src–dest pairs in S. This model represents networks where there is an equal proportion of traffic flowing between all end-points. In the non-uniform model, src–dest pairs could be categorized as *hot* or *cold* pairs, where hot pairs receive the majority of the routing requests. This model represents networks where a heavy portion of traffic flows between some (hot) end-points, while there is relatively less flow of traffic between other (cold) end-points. The ratio of traffic distribution between a hot pair and a cold pair and the ratio of number of hot pairs to number of cold pairs are important parameters in this model. We consider three scenarios for the non-uniform model: (a) hot and cold pairs are evenly distributed, (b) hot pairs are very high in number compared to cold pairs and (c) hot pairs are very low in number compared to cold pairs. The traffic distribution ratio between hot and cold pairs is fixed at 9:1.

3.3. Performance metrics

The evaluation of all algorithms is done from a TE perspective with the purpose of analyzing how these algorithms compare with each other in terms of blocking requests, reducing network costs and distributing network load. We define the following parameters:

- *Available max-flow:* Request blocking can be closely related to the available max-flow between src–dest pairs. More available max-flow between ingress–egress pairs indicates that more requests can be routed between them in the future, while less available max-flow indicates potential increase in blocked requests. Thus, we define the max-flow factor metric as

  $$\text{Max-flow factor} = \text{Current available max-flow between all src–dest pairs/Initial available max-flow between all src–dest pairs.}$$

- *Reducing network cost:* In order to minimize network costs the path costs should be minimized. In our experiments, the path length solely determines path cost. An algorithm that on the average returns less costly paths will be more successful in lowering network costs than algorithms, which select more expensive paths.
- *Distributing load:* To measure the distribution of load characteristics of an algorithm we examine the loading on paths. The maximum link load on the component links of a path measures path load. An algorithm, which on the average selects paths with a lower path loading, will be more

successful in distributing network load than algorithms, which select highly loaded paths.

3.4. Results

We now present results of comparisons of the path computation algorithms. Figs. 5–10 show the performance of the algorithms with respect to parameters discussed in Section 3.3. Results are shown for the network in Fig. 4. Hot and cold pairs of src–dest nodes are randomly selected and requests are generated for the uniform and the three non-uniform scenarios (a)–(c) (discussed in Section 3.2). Each algorithm tries to route these requests through the network. If a route is found, resources are reserved along it and are held as long as the request is active. Resources are freed once a request's holding time expires. A request is

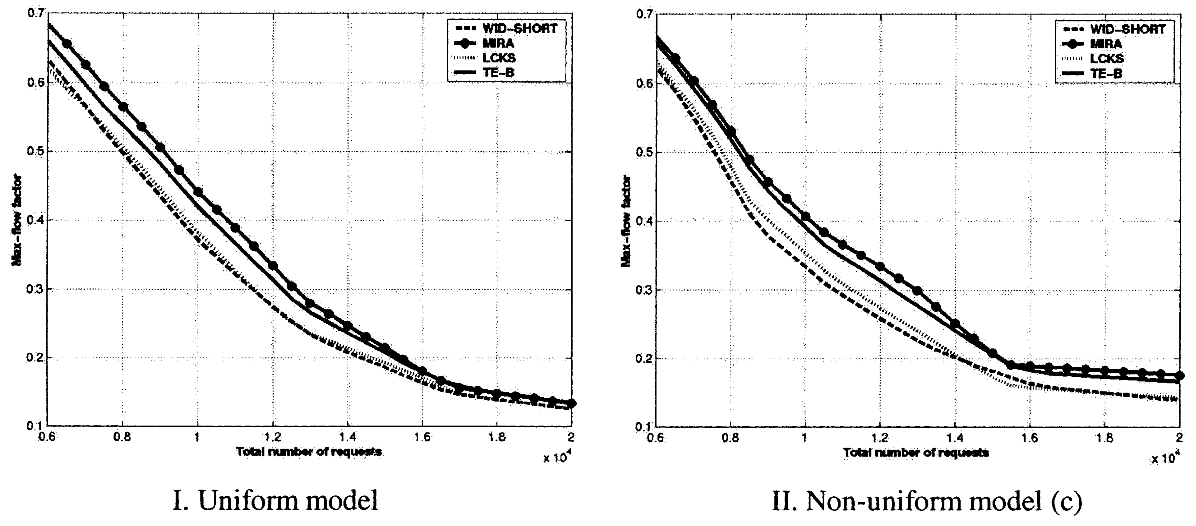


Fig. 5. Max-flow reduction in uniform/non-uniform models (1 QoS constraint).

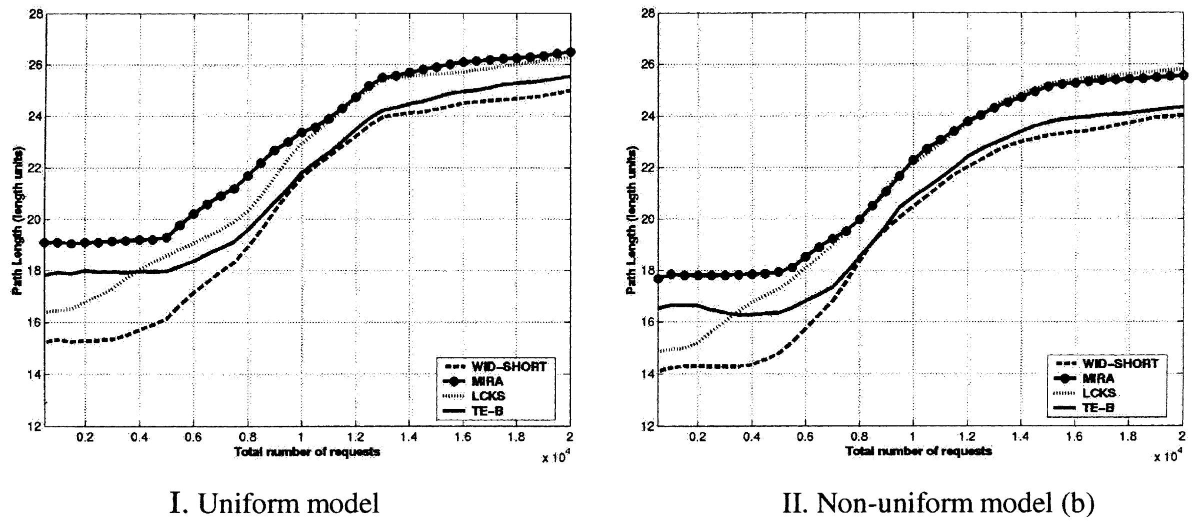
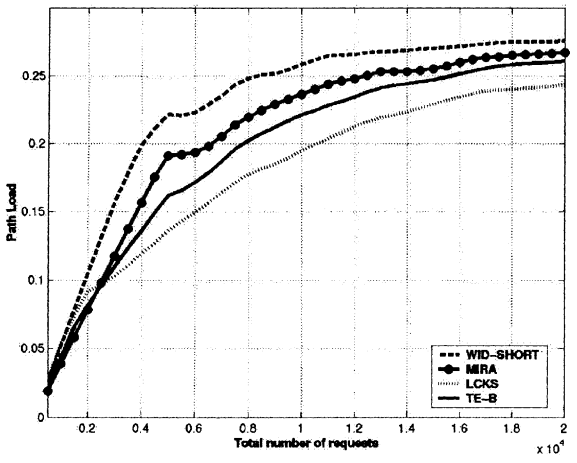
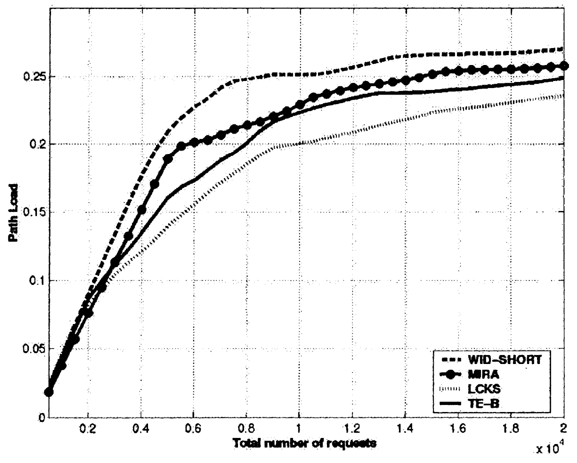


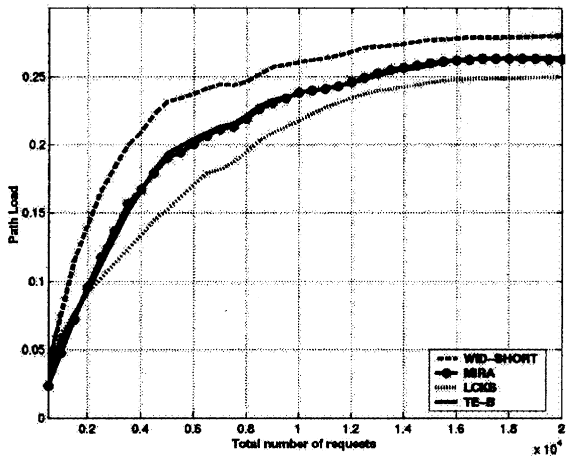
Fig. 6. Path length increase in uniform/non-uniform models (1 QoS constraint).



I. Uniform model



II. Non-uniform model (a)



III. Non-uniform model (c)

Fig. 7. Average path load in uniform/non-uniform models (1 QoS constraint).

blocked if no route can be found for it. Experiments are repeated with different src–dest pairs and different request sets and the results report the average values of the measured metrics over all the experiments. We refer to the region below 8k requests as low load, the region of 8k–14k as medium load and the region of 14k–20k requests and upwards as high load.

Figs. 5–7 present the results of experiments when only one QoS constraint, i.e., bandwidth, needs to be satisfied. All requests have infinite lifetime. The algorithms compared are MIRA, LCKS, WID-SHORT and TE-B. Results are shown for the uniform model and one or more representative scenarios of the non-uniform model. Fig. 5(I) and (II) shows the reduction in max-flow factor of the four algorithms. We see that MIRA and TE-B do consistently better than the WID-SHORT and LCKS methods (by about 10–16%). The performance of TE-B is almost always as good as that of MIRA. At low-load conditions MIRA does a little better, but with increasing loads TE-B achieves performance comparable to MIRA.

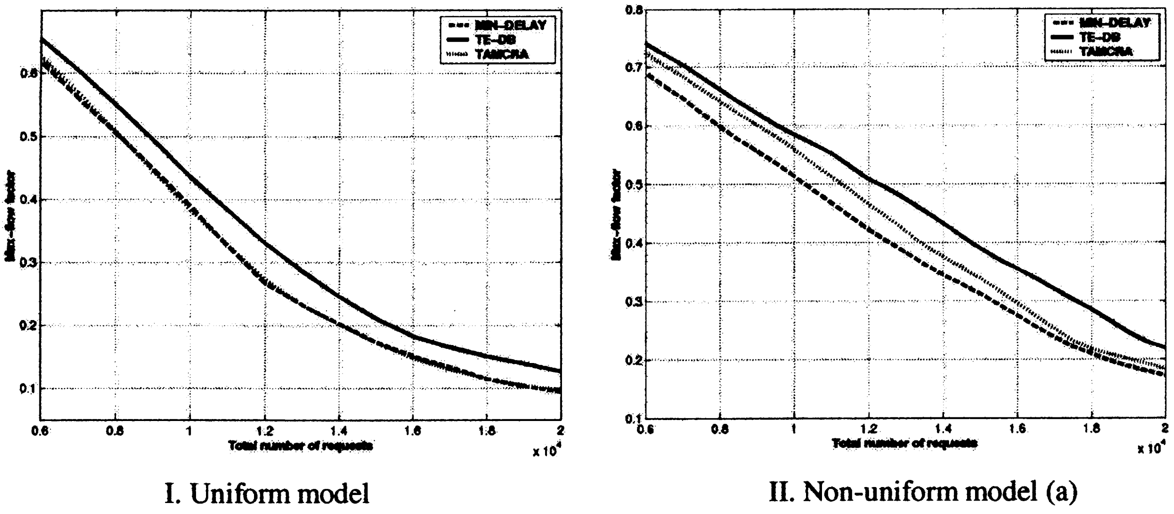


Fig. 8. Max-flow reduction in uniform/non-uniform models (2 QoS constraints).

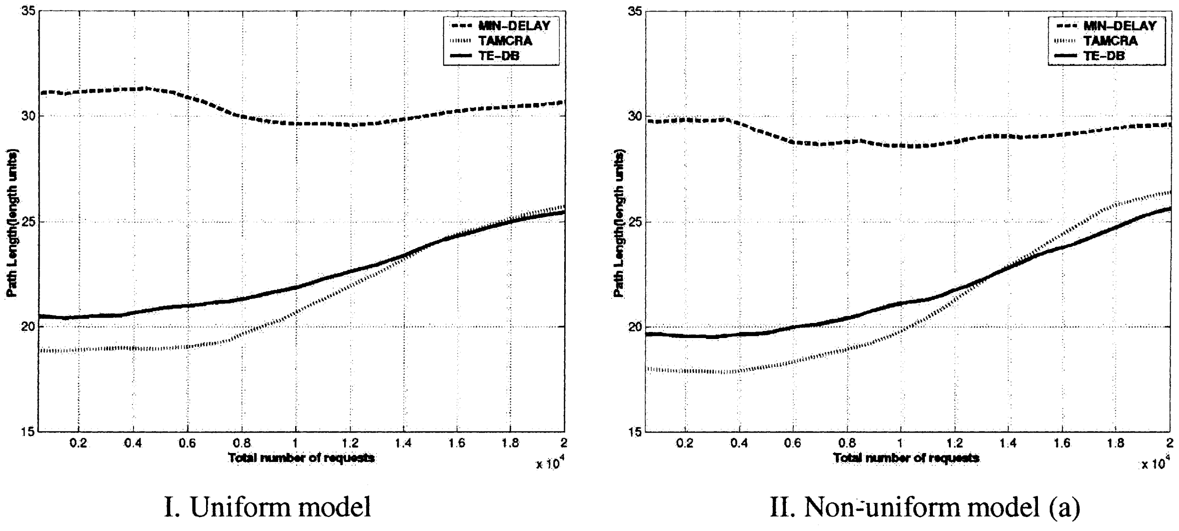
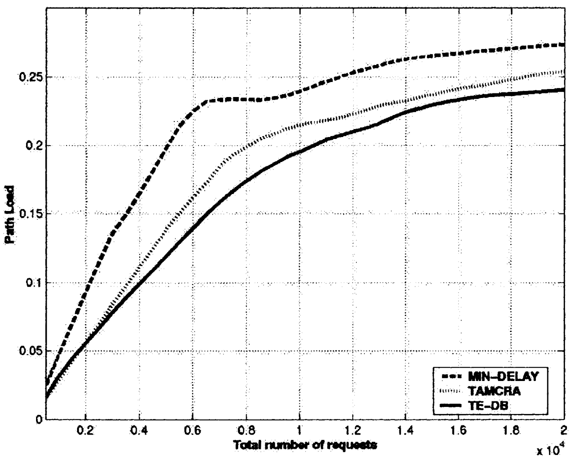
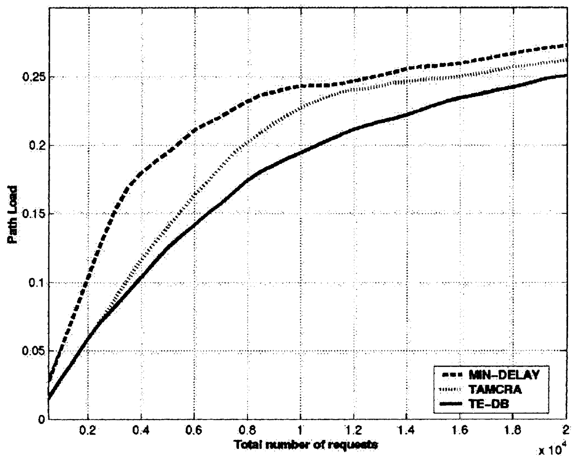


Fig. 9. Path length increase in uniform/non-uniform models (2 QoS constraints).

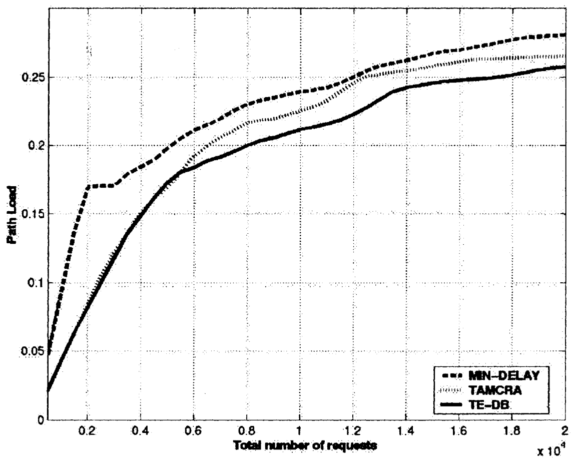
Fig. 6(I) and (II) shows the average length of paths returned by the four algorithms. Since the widest-shortest-path approach gives maximum priority to finding shortest path(s), WID-SHORT succeeds in having the lowest path costs under low-load conditions. LCKS paths are slightly longer under the same condition because it relaxes the shortest-path constraint and instead finds a set of *k*-shortest paths. MIRA returns the longest paths. At low-load conditions, both LCKS and WID-SHORT outperform TE-B and MIRA. However as the load increases on the network, TE-B returns lower cost paths than LCKS, while achieving performance close to that of WID-SHORT. At high-load conditions, on average, TE-B attains about 10–15% lower cost paths than the LCKS and MIRA algorithms.



I. Uniform model



II. Non-uniform model (a)



III. Non-uniform model (c)

Fig. 10. Average path load in uniform/non-uniform models (2 QoS constraints).

Fig. 7(I)–(III) shows the average path load for the four algorithms. We note that the total network load being the same for all algorithms, LCKS paths have the least average load. TE-B comes a close second and has lower path loading than either WID-SHORT or MIRA. The performance gains of TE-B in terms of reduction in load is about 10–20% when compared to MIRA or WID-SHORT. However, in scenario (c) of the non-uniform model, we observe that since the majority of traffic is concentrated between few hot pairs, most of the paths between them run at very high loads. In this case, since there are not many non-critical paths to off-load the traffic, load distributing does not help much in lowering the average load on paths. Thus the benefit of using a load distributing mechanism like LCKS or TE-B is much more pronounced in the uniform model and scenarios (a) and (b) of the non-uniform model. Note that though WID-SHORT and LCKS are similar algorithms, the latter achieves about 40% more reduction in average path load by using the proposed criticality metric. Thus, TE-B has an overall superior performance compared to the competitive algorithms in terms of reducing blocking, reducing network costs and distributing network load.

Figs. 8–10 present the results of experiments when both QoS constraints of bandwidth and delay have to be satisfied. The algorithms compared are MIN-DELAY, TAMCRA and TE-DB. Results are shown for the uniform model and one or more representative scenarios of the non-uniform model. All requests have infinite lifetime. Fig. 8 (I) and (II) shows the reduction in max-flow factor obtained from the three algorithms. We note that even as more requests are admitted into the system, TE-DB succeeds in maintaining the maximum available flow between transmitting src–dest pairs in the network. We note that on an average TE-DB out-performs MIN-DELAY and TAMCRA by about 15–20%.

Fig. 9(I)–(II) shows the increase in length of paths returned by the three algorithms, with increasing network load. We assume that the length of the paths serves as an adequate measure of their cost. We note that TAMCRA paths are the shortest at low-load conditions. However as the load increases, TE-DB path costs closely catch up with that of TAMCRA paths, and at highest load conditions TE-DB succeeds in maintaining the least network costs. MIN-DELAY paths are always about 40–50% more expensive than either TAMCRA or TE-DB paths.

Fig. 10(I)–(III) shows the average load on paths returned by the three algorithms. On average, the TE-DB paths have the least load under all load conditions. TE-DB attains a significant reduction in average path load, when compared to MIN-DELAY. This reduction is 60–70% at low load, 30–40% at medium load and 10–20% at high load. TE-DB paths consistently attains 5–15% lower path load than TAMCRA paths. The total network load being the same for all three algorithms, we show that TE-DB is uniformly more effective in terms of distribution of network load. However, this improvement is reduced in scenario (c) of the non-uniform model. This can be attributed to the fact that in this model majority of the requests are distributed between very few hot pairs. Thus all paths between the hot pairs are congested and there are none non-critical paths to off-load the traffic.

We now investigate a more dynamic environment, where flows are setup and torn down frequently. This environment is closer to real life scenarios, where an application requests network resources only for a limited amount of time. In this experiment, 30% of the flows are assumed to be infinitely long and the rest of them have a mean holding time of 250 s. Results are shown only for the uniform case. Figs. 11 and 12 show that as before, TE-B and TE-DB maintain a high amount of available flow in the network.

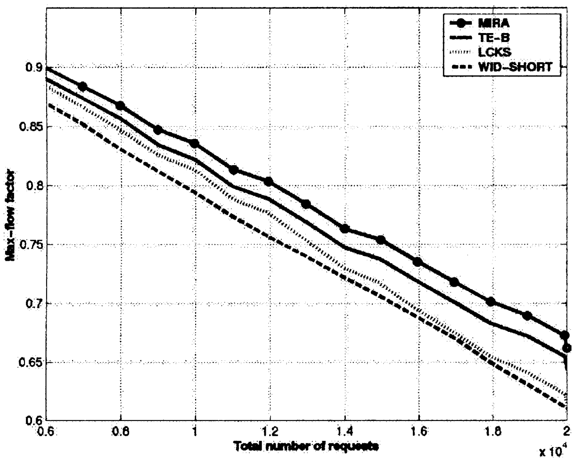


Fig. 11. Max-flow reduction in uniform model (1 QoS constraint).

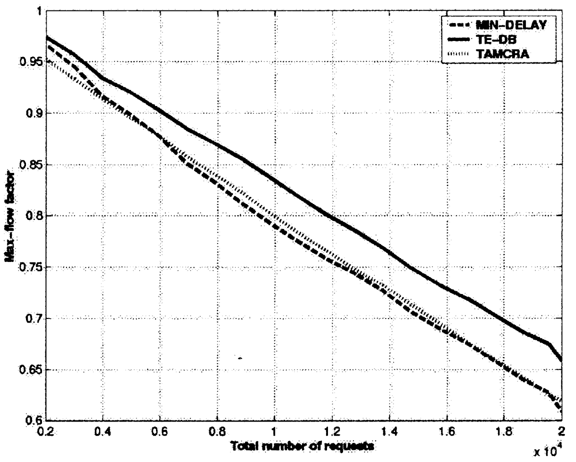


Fig. 12. Max-flow reduction uniform model (2 QoS constraints).

In the next experiment we show that the increased available flow translates to better performance. We load the network with finite-duration requests and observe the number of requests rejected by each of the algorithms. The blocking probability is measured as the ratio of the rejected requests to the total requests in the system. We conduct 10 trials and the results are shown in Figs. 13 and 14. We observe that the TE algorithms attain lower blocking probabilities, which corroborates the intuition that more requests can be admitted by maintaining a higher amount of available flow in the network.

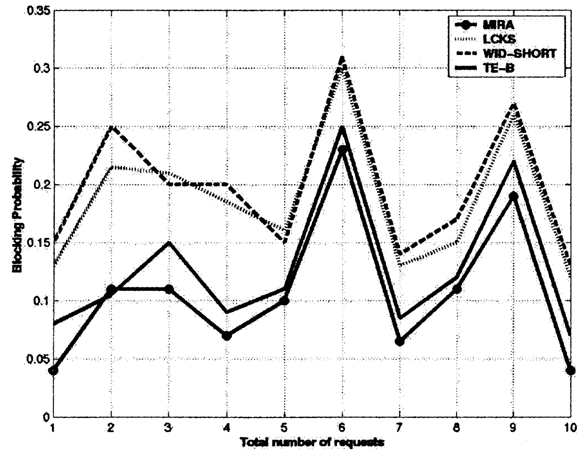


Fig. 13. Blocking probability in uniform model (1 QoS constraint).

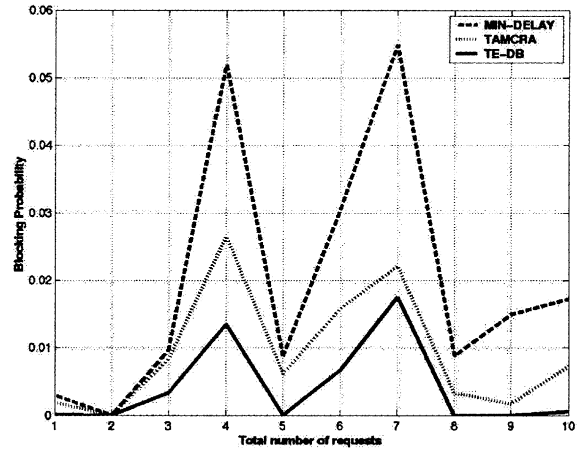


Fig. 14. Blocking probability in uniform model (2 QoS constraints).

4. Conclusion

In this paper, we propose two TE path computation algorithms, TE-B and TE-DB. The former is used when applications specify their QoS requirements only in terms of bandwidth, while the latter is used when both bandwidth and delay requirements are specified. Both these algorithms try to maintain the three TE objectives of (1) increasing network revenue, (2) limiting network costs, and (3) distributing network load. We compare the performance of these two algorithms with other existing competitive algorithms that use the same application specified constraints. We show that by considering the additional TE constraints, both TE-B and TE-DB achieve considerable performance enhancements. The other existing algorithms while performing well with respect to some metric, suffer from poor performance with respect to other metrics. However, for the TE algorithms our results show that their performance is always either the most favorable or very close to the most favorable solution with respect to all the measured metrics. We believe that the TE algorithms can be suitably deployed in provider networks since they succeed in maintaining high amount of available flow between network end-points, reducing network costs and distributing network load. TE-B and TE-DB, thus, qualify as superior TE path computation algorithms.

References

[1] G. Apostolopoulos, S. Kama, D. Williams, R. Guerin, A. Orda, T. Przygienda, QoS routing mechanisms and OSPF extensions, RFC 2676, 1999.
[2] D. Awudche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, Requirements for traffic engineering over MPLS, RFC 2702, 1999.
[3] D. Awudche, L. Berger, D. Can, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: extensions to RSVP for LSP tunnels, RFC 3209, December 2001.
[4] S. Chen, K. Nahrstedt, On finding multi-constrained paths, in: Proceedings of IEEE ICC, 1998.
[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press and McGraw Hill, 1990.

[6] B. Fortz, M. Thorup, Internet traffic engineering by optimizing OSPF weights, in: Proceedings of IEEE INFOCOM, 2000.

[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.

[8] A.V. Goldberg, S. Rao, Beyond the flow decomposition barrier, J. ACM 45 (5) (1998) 783–797.

[9] L. Guo, I. Malta, Search space reduction in QoS routing, in: Proceedings of 19th IEEE ICDCS, 1999.

[10] J. Jaffe, Algorithms for finding paths with multiple constraints, Networks 14 (1984) 95–116.

[11] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, A. Malis, Constraint-based LSP setup using LDP, RFC 3212, January 2002.

[12] D. Katz, D. Yeung, K. Kompella, Traffic engineering extensions to OSPF, Work in Progress, April 2002.

[13] M. Kodialam, T. Lakshman, Minimum interference routing with applications to MPLS traffic engineering, in: Proceedings of IEEE INFOCOM, 2000.

[14] T. Korkmaz, M. Krunz, S. Tragoudas, K. Kompella, An efficient algorithm for finding a path subject to two additive constraints, in: Proceedings of ACM SIGMETRICS, 2000.

[15] G. Liu, K.G. Ramakrishnan, A*Prune: an algorithm for finding *K* shortest paths subject to multiple constraints, in: Proceedings of IEEE INFOCOM, 2001.

[16] Q. Ma, P. Steenkiste, On path selection for traffic with bandwidth guarantees, in: Proceedings of IEEE Conference on Network Protocols, 1997.

[17] H. De Neve, P.V. Mieghem, A multiple quality of service routing algorithm for PNNI, in: Proceedings of ATM Workshop, 1998.

[18] Z. Wang, J. Crowcroft, Bandwidth–delay based routing algorithms, in: Proceedings of IEEE GLOBECOM, 1995.

[19] Z. Wang, On the complexity of quality of service routing, Inform. Process. Lett. 69 (3) (1999) 111–114.

Gargi Banerjee received the B.S. degree in Computer Science from Jadavpur University, India in 1997. She received M.S. in Computer Science from University of Maryland, Baltimore County in 2000 and is currently pursuing her Doctoral studies at the same University under the advisement of Dr. Deepinder Sidhu. Her research interests include computer and communication networks, networking protocols, routing and switching technologies, optical networking, QoS and traffic engineering solutions.

Deepinder P. Sidhu (SM'84/ACM'84) received the B.S. degree in Electrical Engineering from University of Kansas, Lawrence, and the M.S. and Ph.D. degrees in Computer Sciences and Theoretical Physics respectively from the State University of New York, Stony Brook, NY.

He is a Professor of Computer Science with the Computer Science Department of the University of Maryland, Baltimore County (UMBC) and the University of Maryland Institute for Advanced Computer Studies (UMIACS) at College Park. He is the director of Maryland Center for Telecommunications Research (MCTR) at UMBC. His current research interests are in the areas of computer networks and distributed systems. He has published over 100 papers in theoretical physics and computer science. His contributions to the determination of weak neutral current couplings of quarks, in gauge theories of fundamental particles, were cited by Steve Weinberg in his 1979 Nobel Prize acceptance speech.

Dr. Sidhu is the Editor-in-Chief of Journal of High Speed Networks. He was ACM/SIGCOMM National Lecturer for 1992–95. He is a co-founder of Protocol Development Corp. (acquired by Phoenix Techn. Ltd.) and Telenix Corp. He managed the Secure and Distributed Systems Department within the R&D Division of SDC-Burroughs (now Unisys). He is the author of a graduate level text, OSI Conformance Testing, (Englewood Cliffs, NJ: Prentice Hall, 1994).