Traffic Engineering Using MPLS for Best Effort Traffic

Jerapong Rojanarowan, Bernd G. Koehler, and Henry L. Owen

Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332, USA
jerapong@ece.gatech.edu, bernd.koehler@usma.edu, henry.owen@ece.gatech.edu

***Index Terms*— Best effort traffic engineering, Multiprotocol Label Switching, Stateless traffic routing**

***Abstract*— The advent of Multi-protocol Label Switching (MPLS) enables traffic engineering by introducing connection-oriented features of forwarding packets over arbitrary non-shortest paths. Our goal in this research is to improve the network utilization for best effort traffic in IP networks. By examining the best effort traffic class, we assume the large volume of research that has been conducted on traffic engineering for assured forwarding and expedited forwarding with admission control allows traffic engineering on these traffic classes. We examine a different and in some sense a "more complex" problem of traffic engineering in the remaining network bandwidth which is utilized by best effort traffic. We present a generic traffic engineering framework and four specific algorithms. This framework has two prominent features 1) it uses MPLS to encapsulate source-destination aggregate flows within a Label Switched Path (LSP), with multiple LSPs per source-destination. 2) The framework is *"stateless"*, only the topology is used to determine the traffic routing.**

I. Introduction

As the Internet becomes more popular, there is more traffic in the network. Growing network traffic can cause network congestion. Essentially network congestion may result from a shortage of bandwidth or inefficient traffic management that causes uneven traffic distribution. In particular, the forwarding paradigm in IP networks based on the destination address maps the traffic onto the shortest path whereas the capacity of the links not on the shortest path is underutilized. Therefore, there is a possibility for traffic engineering to improve the performance of IP networks.

In order to provide such capability, the basic IP forwarding paradigm of present-day IP networks must be enhanced to support traffic engineering. The advent of Multi-protocol Label Switching (MPLS) make this feasible by introducing the connection-oriented features of forwarding packets over arbitrary non-shortest paths.

A. MPLS and Traffic Engineering

In this section we describe MPLS application to traffic engineering. MPLS was introduced by the Internet Engineering Task Force (IETF) as a novel forwarding paradigm in IP networks based on labels [1]. It is aimed to accommodate IP Quality of Service (QoS), forwarding speed, and traffic engineering [2].

MPLS displaces the hop-by-hop forwarding paradigm with a label swapping forwarding paradigm. A label is a short, fixed length, locally significant identifier assigned to a packet at the ingress router of an MPLS domain corresponding to its Forwarding Equivalence Class (FEC). A FEC is considered as a group of packets that has the same path forwarding requirements. A benefit of the FEC is that it can support a wide range of forwarding granularities, ranging from per-destination to per-application [3]. MPLS directs the flows of packets along the predetermined Label Switched Paths (LSPs) across the network based on the labels.

In MPLS, there are two alternative LSP selection mechanisms: hop-by-hop and explicit routing. A hop-by-hop path is calculated based on the normal layer 3 routing information. With an explicit routing mechanism, the path is completely assigned by the originator independent of layer 3 routing. There are two methods to set up an explicit route: Constraint-based Routed Label Distribution Protocol (CR-LDP) [4] and extensions to Resource Reservation Protocol (RSVP) [5].

Traffic engineering is the process of optimization the network utilization or fulfilling some policy objectives. The limitation of traffic engineering in IP networks is due to the destination based forwarding scheme. While this mechanism is scalable, it unevenly distributes traffic using only the shortest paths which in turn leads to inefficient use of network resources. The Explicitly Routed LSP (ER-LSP) with optimization objectives is the primary tool for traffic engineering in MPLS. With ER-LSP, the ingress routers are able to control the traffic distribution in the network to meet the performance objectives. However, the exact algorithm for determining the ER-LSP is not specified in the IETF.

B. Best Effort Traffic Class

Currently IP networks can support only a single best effort service class. Recent research has begun examining how to deliver QoS for best effort traffic [6]. Many traditional applications such as file transfer, remote terminal, and electronic mail have been served sufficiently because of their elasticity. These applications can tolerate performance

variations during the presence of congestion in the network. The majority of the best effort traffic over the Internet is Transmission Control Protocol (TCP). The close-loop congestion control mechanisms in TCP regulate the transmission rate of the source. The offered load presented by a TCP connection varies with the network condition along with the complex interaction between routing, queueing and flow control. This causes the offered traffic to be dynamic and elastic in nature. Typically, the size of elastic flows is variable and exhibits a heavy-tailed distribution [7]. In addition, best effort traffic is not subject to admission control. As a result, it is very difficult to describe its aggregate bandwidth requirements. These characteristics make traffic engineering of best effort traffic challenging.

II. Best Effort Framework

One constraint in the examination of traffic engineering applied to best effort traffic is that traffic demands are not well-defined. Consequently traditional optimization techniques do not directly apply. Given this difficult constraint, this research examines how well we can do with traffic engineering on best effort traffic without a set of well-defined best effort demands.

We propose a *stateless* framework where the input to the traffic engineering algorithm is restricted to the network topology. We assume that our techniques are applied only to the remaining best effort bandwidth in a network with multiple traffic classes. The other priority classes use admission control and traffic engineering based on other more typical traffic engineering algorithms. We examine the remaining network capacity and attempt to maximize the capability to deliver best effort traffic with this limited available capacity.

The motivation for exploring this approach is the following observation: current IP routing protocols are state-independent - they utilize metrics that only depend upon the topology to calculate the shortest path. Therefore, paths do not change in response to network congestion or demand variation. Despite the non-adaptive nature of this scheme, it consistently provides an adequate level of performance over a broad range of demand patterns. One can do better, but the tradeoff is in the additional complexity required for state-sensitive routes and metrics. The objective of this work is to develop a framework that, although static and stateless, improves upon the current paradigm and provides an acceptable level of performance over a broad range of network conditions as applied to best effort traffic.

The general operation of the proposed framework consists of the following steps [8]:

1) For each source-destination pair $w \in \mathcal{W}$ a set of paths $\mathcal{P}_w$ is computed based on the network topology $\mathcal{G}(\mathcal{V}, \mathcal{A})$.
2) Each path $p \in \mathcal{P}$ is instantiated as a permanent LSP.
3) For each path, an optimal rate r_p is computed. Each path is guaranteed this rate along each link in the path through link sharing discipline such as Weighted Fair Queueing (WFQ).
4) For each path, $\mathcal{P}_w$ assigned to a source-destination pair, a fraction $\phi_p \in [0, 1]$ of the total traffic is calculated.
5) Each ingress node i of a source-destination pair $(i, j) \equiv w$ splits j-bound traffic according to the assigned path fractions ϕ_p, $\forall p \in \mathcal{P}_w$.

Barring any topology changes (e.g., link failure/recovery), path, rate, and fraction computation is only executed once, during the system initialization. A potential performance enhancement is to periodically recalculate the assigned rates and fractions based on aggregate flow measurements. We do not examine that enhancement in this paper.

The primary design criteria for the proposed framework is that the input is restricted to the network topology. The primary advantage of this restriction is inherent simplicity - there is no additional overhead for the exchange of network state. Clearly, the framework should

1) Split aggregate source-destination traffic over multiple paths.
2) Account for the elastic nature of best effort traffic.
3) Restrict the input to the network topology.
4) Provide a level of service better than that of shortest-path routing.

In fact, there are exactly two design unknowns: 1) the path set $\mathcal{P}_k$, and 2) the fractions ϕ_p, $\forall p \in \mathcal{P}_k$. The primary difference between the proposed framework and an adaptive scheme is there is no periodic exchange of network state and adjustment of path flow parameters. Instead, the objective is by choosing a "good" path set $\mathcal{P}_k$ and fractions ϕ_p, improvements over shortest-path routing can be achieved over a wide range of traffic demands. While the results may not be as good as an adaptive scheme, the payoff is in a much simpler framework. The proposed framework at present incorporates four algorithms for calculating optimal paths and flow rates.

A. Defining the Path Set

The initial step of the proposed framework is to determine the path set $\mathcal{P}_k$ for each SD pair. Paths with smaller hop-counts are more efficient in terms of bandwidth utilization. We also expect diminishing returns as the path set size grows large, especially for sparse networks. Large path sets frequently include paths with high hop-counts, and these extremely long paths contribute little to the optimal solution. They also preclude traffic on shorter paths contained within them. A small path set also has the added benefit of reducing the complexity of the rate computation problem.

Two alternative path sets are proposed. The first set consists of the k-shortest paths between source and destination nodes. The second set consists of k-disjoint paths between source and destination nodes. If there are no disjoint paths, $\mathcal{P}_k$ consists of the shortest path. If there are only $n \leq k$

disjoint paths, then only the n disjoint paths are included in $\mathcal{P}_k$.

B. Calculating Path Fractions ϕ_k and Rates r_k

Two alternative methods for calculating r_k are proposed. The first method stems from classical flow control theory and is based on a "max-min fair" rate allocation [9].

A rate allocation vector $\mathbf{R}$ is *max-min fair* if it is feasible and for each $p \in \mathcal{P}$, r_p cannot be increased without decreasing some r'_p for which $r'_p \leq r_p$. In other words, it is impossible to increase a given path flow without taking bandwidth away from a path less well off. To derive r_k, we first determine a path set $\mathcal{P}_k$ for each source-destination pair. We then calculate a max-min fair allocation to each $\{r_p : p \in \mathcal{P}\}$. Individual path rates are given by $x_p = r_p$, and the total rate assigned to each source-destination pair $k \in \mathcal{K}$ is

$$r_k = \sum_{p \in \mathcal{P}_k} r_p \tag{1}$$

The path fractions ϕ_p are calculated by

$$\phi_p = \frac{x_p}{r_k} = \frac{x_p}{\sum_{p \in \mathcal{P}_k} r_p} \tag{2}$$

The second method solves a combined rate allocation/delay minimization network optimization problem. Recall that in general, solving a multi-commodity flow problem requires a finite demand set to constrain the feasible region of the flow vector $\mathbf{x}$. In the absence of any demand constraints, the optimal solution tends to $\mathbf{x} = \mathbf{0}$. Zero flow entails zero (or minimum) delay. However, for best effort traffic source-destination demands are not well defined, and the best that can be said is that each $d_k \in [0, d_k^*]$ where d_k^* is the intrinsic demand defined as the demand in the absence of competing traffic. We propose to solve an *unconstrained* optimization problem with an objective function comprised of two components. One component minimizes the total aggregate delay, while the other is a penalty term for making source-destination flows too small. The objective function is given by

$$f(\mathbf{x}) = \sum_{(i,j) \in A} D_{ij}(\mathbf{x}) + \sum_{k \in \mathcal{K}} \frac{a}{r_k} \tag{3}$$

$$D_{ij}(\mathbf{x}) = \frac{f_{ij}}{c_{ij} - f_{ij}}, \quad f_{ij} = \sum_{p \ni (i,j)} x_p \tag{4}$$

where D_{ij} represents the delay of each link $(i,j) \in \mathcal{A}$, and $\sum_{k \in \mathcal{K}} \frac{a}{r_k}$ is a penalty term for making source-destination flow r_k too small. By adding a penalty term the optimal solution finds a balance between congestion and throughput, and at a coarse level models the behavior of a best effort network flow control - flows increase the offered load until throttled by network congestion.

We have proposed two methods for selecting the path set $\mathcal{P}_w$ and two methods for calculating the allocation rates r_w and fractions ϕ_p. This leads to four different candidate traffic engineering algorithms labeled SMM, SOP, DMM and DOP. They are summarized in Table I. We compare and contrast these four algorithms with each other, as well as the currently deployed shortest-path algorithm (SPF).

TABLE I
PROPOSED FRAMEWORK ALGORITHMS

Algorithm	Paths	Rates
SMM	shortest	max-min
SOP	shortest	optimal
DMM	disjoint	max-min
DOP	disjoint	optimal

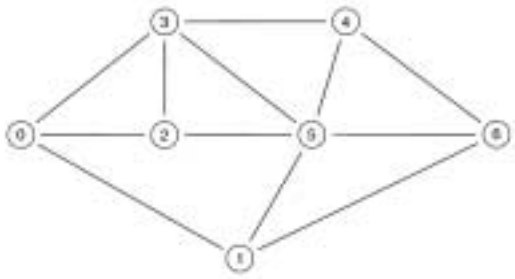

Fig. 1. Network topology

III. SIMULATION SCENARIOS

The simulations in this paper are conducted using the NS simulator with MPLS Traffic Engineering Extensions. MplsWF2Q is the modified version of Worst-case Fair WFQ (WF2Q+) [10], and it is used to guarantee that each LSP receives the assigned flow rate.

Fig. 1 shows the simple network topology used in this paper to obtain the initial results. It consists of 7 nodes and 12 bi-directional links with capacity of 45 Mbps. We have considered 2 cases: the one-demand case where node 0 generates traffic to node 6, and the all-demand case where all nodes are traffic originators. In each case, two different types of traffic, User Datagram Protocol (UDP) and TCP, are utilized with packet sizes of 1000 bytes. The traffic flows generated in each node are exponentially distributed.

We simulate each algorithm with the number of desired paths, in k-shortest and k-disjoint paths, set to 2 to 5 paths. The results of 2 and 3 paths are included in this paper.

IV. RESULTS

One can see from the results that in all cases performance improvements with best effort traffic engineering are noticeable when the number of paths is greater than or equal to 2. We define the one-demand case to be the case where node 0 generates traffic to node 6. In the one-demand case which is shown in Fig. 2, the performance improvements over SPF are considerable even when only 2 paths are used. Fig. 2 plots the maximum utilization of any link in the network on the y axis and the arrival

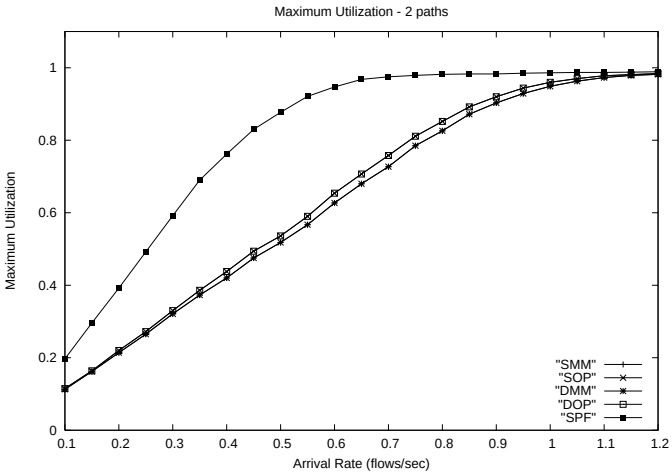

Fig. 2. Maximum network utilization for UDP traffic (one-demand): 2-path case

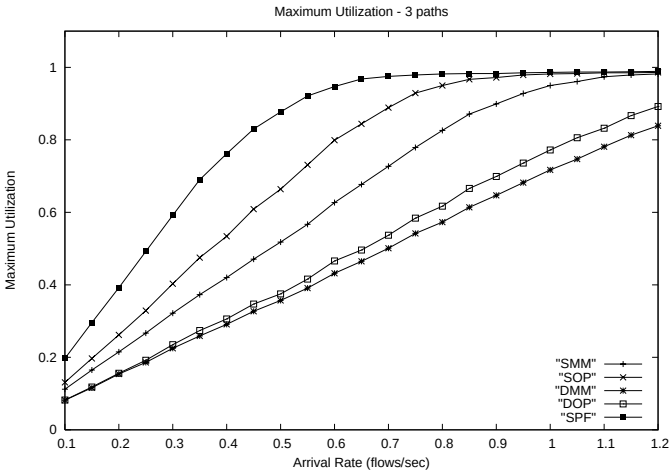

Fig. 3. Maximum network utilization for UDP traffic (one-demand): 3-path case

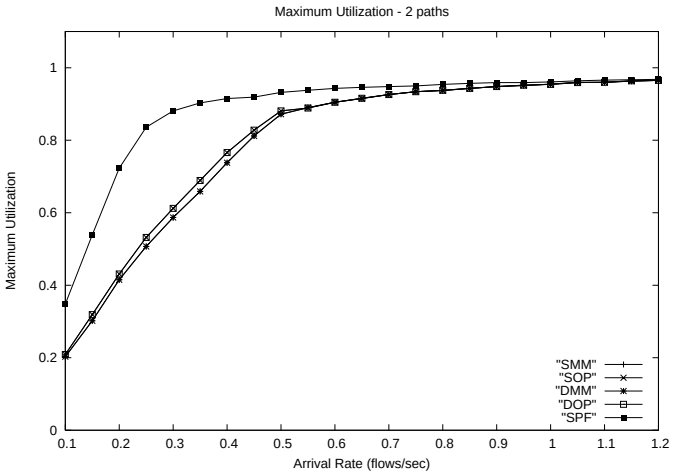

Fig. 4. Maximum network utilization for TCP traffic (one-demand): 2-path case

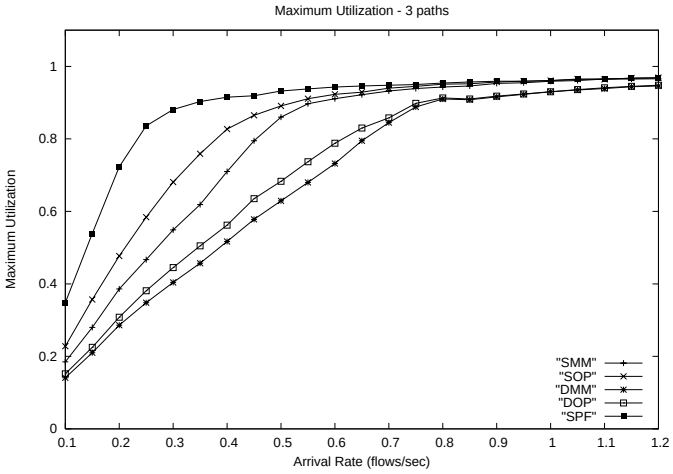

Fig. 5. Maximum network utilization for TCP traffic (one-demand): 3-path case

rate in flows per second on the x axis. Lower values of maximum utilization are better. All four algorithms have lower utilizations than standard shortest path, thus all four algorithms perform roughly the same and are all better than SPF.

From Fig. 3 which uses 3 paths instead of two, one can see that addition paths cause the four algorithms to perform a bit differently relative to each other. The DMM algorithm becomes our best choice to decrease the network utilization. One can also see that the algorithms that use k-disjoint paths (DOP and DMM) perform better than the algorithms that use k-shortest paths (SOP and SMM). This result is not surprising because a set of shortest paths has an undesirable property that multiple aggregate flows may traverse the same link. For instance, 0–1–6, 0–1–5–6, and 0–3–4–6 paths are used for 3-shortest paths and there are two of them sharing the 0–1 link. Fig. 4 and Fig. 5 repeat the same situations in Fig. 2 and Fig. 3 except that TCP traffic is used instead of UDP traffic. The conclusion is the same. Going from 2 paths (Fig. 4) to 3 paths (Fig. 5) again reduces the maximum utilization with the DMM algorithm remaining the best choice.

Although we do not show the results for the 4- and 5-path cases here, the conclusions still hold true. In particular, the results are nearly identical to the 3-path case. This is because the number of paths in this small network is very limited. For example, there are at most 3 disjoint paths from node 0 to node 6, i.e., 0–1–6, 0–2–5–6, and 0–3–4–6. At 80% of maximum utilization, DMM can support 75-80% and 150-170% of more flows than SPF for the 2- and 3-path cases, respectively.

We define the all-demand case to be when all nodes transmit to every other node. In the all-demand case, shown in Figs. 6 to 9, smaller performance gains are achieved and DMM is still the best candidate. However, we do not obtain as much of an advantage by splitting traffic over more than 2 different paths. Furthermore, all algorithms do not drastically outperform the SPF as in the one-demand case. The reason is when all nodes are generators, all paths are evenly utilized thus there are few opportunities left for

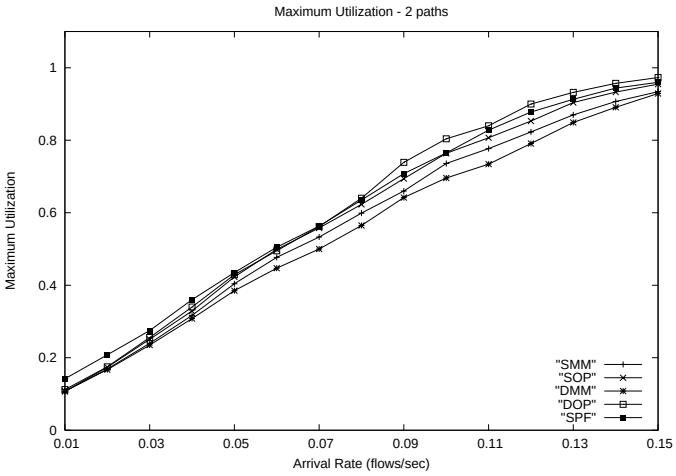

Fig. 6. Maximum network utilization for UDP traffic (all-demand): 2-path case

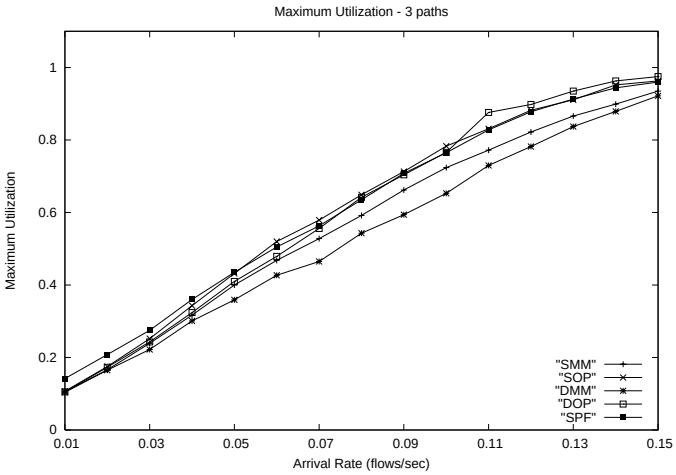

Fig. 7. Maximum network utilization for UDP traffic (all-demand): 3-path case

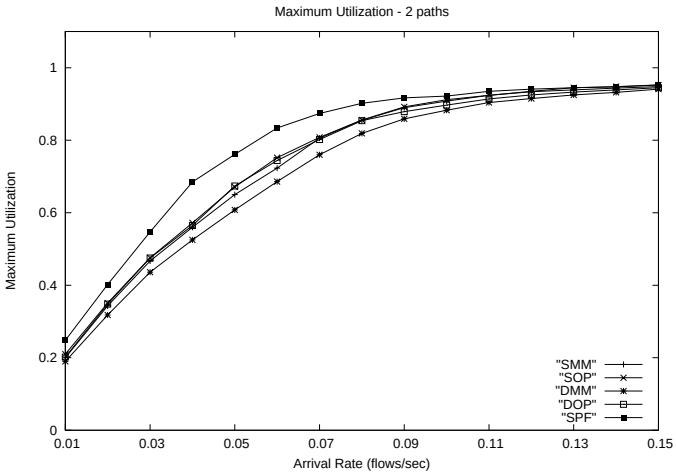

Fig. 8. Maximum network utilization for TCP traffic (all-demand): 2-path case

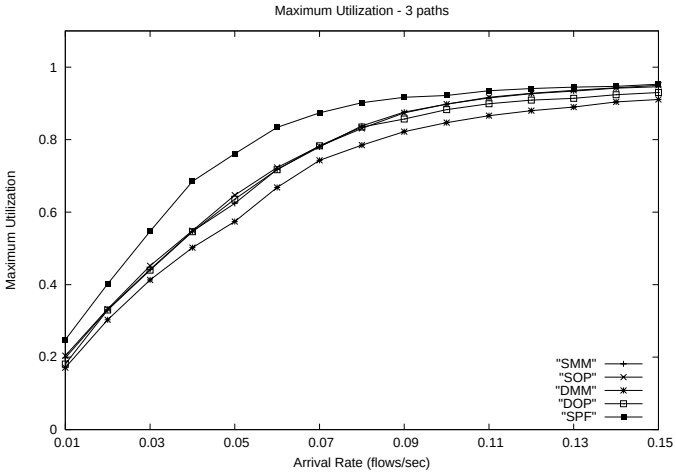

Fig. 9. Maximum network utilization for TCP traffic (all-demand): 3-path case

traffic engineering. This confirms the fact that when traffic is unevenly distributed, as in the one-demand case, traffic engineering has the potential to decrease the utilization by splitting the traffic over under-utilized paths. However, this is not the case when all nodes transmit with the same amount of traffic because traffic is rather evenly distributed. As a result, traffic engineering best effort traffic when all nodes are transmitting the same amount of traffic to all other nodes does not result in a dramatic improvement.

V. Conclusion

In this paper we have proposed a traffic engineering framework for best effort traffic. This framework is *"stateless"*, only the topology is used to determine the traffic routing. Four algorithms were investigated. The results show that there are performance improvements over SPF, especially when the traffic distribution is uneven. The DMM algorithm that incorporates disjoint paths while using max-min-fair rates is our best candidate in all cases. Our proposed framework can support both inelastic (e.g., UDP) and elastic (e.g., TCP) traffic.

We are presently examining additional new traffic engineering algorithms targeted for best effort traffic. Using these new algorithms we are comparing these more capable algorithms to our present set of algorithms to determine how to maximize the limited left over bandwidth which is available for best effort traffic. We believe that networks which use traditional traffic engineering algorithms coupled with admission control for the premium traffic classes and our algorithms with the remaining bandwidth left over for best effort traffic will better utilize limited network resources.

References

[1] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC3031, Jan. 2001.

[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," RFC2702, Jan. 2002.

[3] Y.-D. Lin, N.-B. Hsu, and R.-H. Hwang, "QoS routing granularity in MPLS networks," *IEEE Commun. Mag.*, pp. 58–65, June 2002.

[4] B. Jamoussi, *et al.*, "Constraint-based LSP setup using LDP," RFC3212, Jan. 2002.

[5] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, “RSVP-TE: Extensions to RSVP for LSP tunnels,” RFC3209, Dec. 2001.
[6] B. Wydrowski and M. Zukerman, “QoS in best-effort networks,” *IEEE Commun. Mag.*, pp. 44–49, Dec. 2002.
[7] J. W. Roberts, “Traffic theory and the internet,” *IEEE Commun. Mag.*, pp. 94–99, Jan. 2001.
[8] B. Koehler, D. Barlow, H. Owen, and J. Sokol, “Traffic engieering communication protocols for best effort traffic,” in *International Conference on Communication Systems and Networks CSN 2002*, Malaga, Spain, Sept. 2002, pp. 360–365.
[9] J. M. Jaffe, “Bottleneck flow control,” *IEEE Trans. Commun.*, vol. 29, pp. 954–962, July 1981.
[10] J. C. R. Bennett and H. Zhang, “Hierachical packet fair queueing algorithms,” *IEEE/ACM Trans. Networking*, vol. 5, pp. 675–689, Oct. 1997.