Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Computer Networks 40 (2002) 695–709

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

MATE: multipath adaptive traffic engineering

Anwar Elwalid [a], Cheng Jin [b], Steven Low [c,*], Indra Widjaja [a]

[a] *Lucent Technologies, Bell Laboratories, 600 Mountain Avenue, 07974 Murray Hill, NJ, USA*
[b] *Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109-2122, USA*
[c] *Computer Science and Electrical Engineering, Caltech, Pasadena, CA 91125, USA*

Received 9 March 2002; accepted 12 March 2002

Responsible Editor: I.F. Akyildiz

Abstract

Destination-based forwarding in traditional IP routers has not been able to take full advantage of multiple paths that frequently exist in Internet service provider networks. As a result, the networks may not operate efficiently, especially when the traffic patterns are dynamic. This paper describes a multipath adaptive traffic engineering scheme, called MATE, which is targeted for switched networks such as multiprotocol label switching networks. The main goal of MATE is to avoid network congestion by adaptively balancing the load among multiple paths based on measurement and analysis of path congestion. MATE adopts a minimalist approach in that intermediate nodes are not required to perform traffic engineering or measurements besides forwarding packets. Moreover, MATE does not impose any particular scheduling, buffer management, or a priori traffic characterization on the nodes. This paper presents an analytical model, derives a class of MATE algorithms, and proves their convergence. Several practical design techniques to implement MATE are described. Simulation results are provided to illustrate the efficacy of MATE under various network scenarios.

Keywords: Traffic engineering; Multipath routing; Optimal routing; Asynchronous routing algorithm

1. Introduction

Internet traffic engineering is emerging as an important tool to provide fast, reliable and differentiated services. According to the Internet engineering task force (IETF), Internet traffic engineering is broadly defined as that aspect of network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks [1]. More specifically, traffic engineering often deals with effective mapping of traffic demands onto the network topology, and adaptively reconfiguring the mapping to changing network conditions. It is more general than QoS routing in the sense that traffic engineering typically aims at maximizing operational network efficiency while meeting certain constraints, whereas the main objective in QoS routing is to meet certain QoS constraints for a given source–destination traffic flow.

The emergence of multiprotocol label switching (MPLS) with its efficient support of explicit

[*] Corrersponding author.
E-mail addresses: anwar@lucent.com (A. Elwalid), chengjin@cs.caltech.edu (C. Jin), slow@caltech.edu (S. Low).

1389-1286/02/$ - see front matter © 2002 Published by Elsevier Science B.V.
PII: S1389-1286(02)00308-0

routing provides basic mechanisms for facilitating traffic engineering [9]. Explicit routing allows a particular packet stream to follow a pre-determined path rather than a path computed by hop-by-hop destination-based routing such as OSPF or IS–IS. With destination-based routing as in traditional IP network, explicit routing may be provided by attaching to each packet the network-layer address of each node along the explicit path. This approach generally incurs prohibitive overhead. In MPLS, a path (known as a label switched path or LSP) is identified by a concatenation of labels which are stored in the nodes. As in traditional virtual-circuit packet switching, a packet is forwarded along the LSP by swapping labels. Thus, support of explicit routing in MPLS does not entail additional packet header overhead.

In this paper, we propose a state-dependent traffic engineering mechanism called multipath adaptive traffic engineering (MATE). MATE assumes that several explicit LSPs (typically ranges from 2 to 5) between an ingress node and an egress node in an MPLS domain have been established using a standard protocol such as CR-LDP [6] or RSVP-TE [2], or configured manually. This is a typical setting which exists in an operational ISP network that implements MPLS. The goal of the ingress node is to distribute the traffic across the LSPs so that the loads are balanced and congestion is minimized. The traffic to be balanced by the ingress node is the aggregated flow (called traffic trunk in [7]) that shares the same destination (and possibly quality of service). Fig. 1 shows an example of a network environment where there are two ingress nodes, AI and BI, and two egress nodes, AE and BE, in an MPLS domain. MATE would be run on AI and BI to balance traffic destined to AE and BE, respectively, across the LSPs connecting AI to AE and BI to BE. Note that the LSPs connecting the two pairs may share links. In the following, we will derive adaptive MATE algorithms, discuss their implementation, and present simulation results to illustrate their performance. We will prove that it is possible to achieve stability even when ingress–egress (IE) pairs operate asynchronously and in a distributed manner.

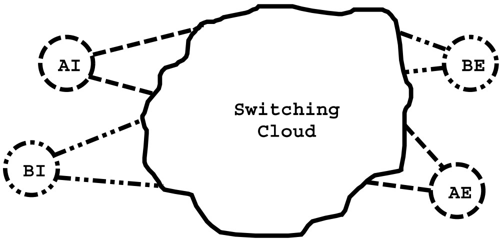


Fig. 1. A transit network running MATE.

We now comment on related work. Several researchers have proposed to add traffic engineering capabilities in traditional datagram networks using shortest path algorithms (e.g., see [5,10]). Although such schemes have been shown to improve the efficiency of the network, they suffer from several limitations including:

- load sharing cannot be accomplished among paths of different costs,
- traffic of different QoS classes follow the same route,
- traffic/policy constraints (for example, avoiding certain links for particular source–destination traffic) are not taken into account,
- modifications of link metrics to re-adjust traffic mapping tend to have network-wide effects and may cause undesirable and unanticipated traffic shifts, and
- traffic demands must be predictable and known a priori.

The combination of MPLS technology and its traffic engineering capabilities are expected to overcome these limitations. Explicit LSPs and flexible traffic assignment address the first two limitations. Constraint-based routing has been proposed to address the third limitation. Furthermore, network-wide effects can be prevented since LSPs can be pinned down. A change in LSP route limits the disturbance of the traffic for the corresponding source–destination pair. The objective of this paper is to address the final limitation.

In MPLS, traffic engineering mechanisms may be time dependent or state dependent. In a time-dependent mechanism, historical information based on seasonal variations in traffic is used to pre-program LSP layout and traffic assignment. Additionally, customer subscription or traffic projection may be used. Pre-programmed LSP layout typically changes on a relatively long time scale (e.g., diurnal). Time-dependent mechanisms do not adapt to unpredictable traffic variations or changing network conditions. An example of a time-dependent mechanism is a global centralized optimizer where the input to the system is a traffic matrix and multiclass QoS requirements as described in [8].

When there are appreciable variations in actual traffic that could not be predicted using historical information, a time-dependent mechanism may not be able to prevent significant imbalance in loading and congestion. In such a situation, a state-dependent mechanism can be used to deal with adaptive traffic assignment to the established LSPs according to the current state of the network which may be based on utilization, packet delay, packet loss, etc. In this paper, we assume that LSP layout has been determined. The focus is on load balancing traffic among multiple LSPs between an ingress node and an egress node.

The rest of the paper is organized as follows. Section 2 details the overall MATE scheme and discusses several implementation techniques, such as traffic filtering and distribution, traffic measurement, bootstrapping, etc. Section 3 presents an analytical model of MATE and proves its stability and optimality. Section 4 describes an experimental setup to verify the effectiveness of the proposed scheme. Section 5 presents the simulation results that illustrate the behavior of the algorithm in different environments. Conclusions are given in Section 6. Analytical proofs are collected in the Appendix A.

2. MATE algorithms and implementation techniques

2.1. Overview

The basic idea of MATE is as follows. The ingress node of each LSP periodically sends probe packets to estimate a congestion measure on the forward LSP from ingree to egress. The congestion measure can be delay, loss rate, or other performance metrics; see below for measurement details. Each ingress node then routes incoming traffic onto multiple paths to its egress node in a way that equalizes the *marginal* congestion measure (their derivatives). That is, traffic will be shifted from LSPs with higher marginals to LSPs with lower marginals. In equilibrium all LSPs that carry any flow will have minimum and equal marginals. As will be shown in the next section, equalizing the marginal measure minimizes the total congestion measure of the entire MPLS network.

Fig. 2 shows a functional block diagram of MATE located at an ingress node. Incoming traffic enters into a *filtering and distribution function* whose objective is to facilitate traffic shifting among the LSPs in a way that reduces the possibilities of having packets arrive at the destination out of order. The mechanism does not need to know the statistics of the traffic demands or flow state information. The *traffic engineering function* decides when and how to shift traffic among the LSPs. This is done based on LSP statistics which are obtained from the *measurement and analysis function*.

MATE operates in two phases: a monitoring phase and a load balancing phase. In the monitoring phase, probe packets are sent periodically, and congestion measure on the LSPs and their derivatives are estimated. If an appreciable and persistent change in the network state is detected, transition is made to the load balancing phase. In the load balancing phase, the algorithm continues to monitor congestion measures on the LSPs and tries to equalize their marginals. Once the measures are approximately equalized, the algorithm transits to the monitoring phase and the whole process repeats.

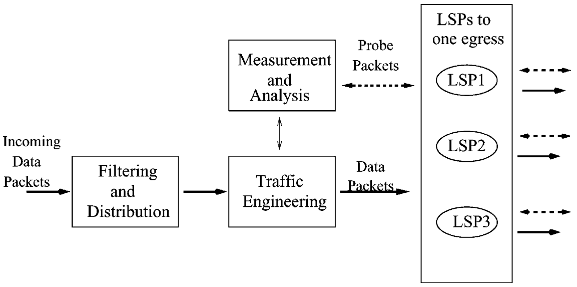


Fig. 2. MATE functions in an ingress node.

One-way LSP statistics (congestion measure) such as packet delay and packet loss are measured by transmitting probe packets periodically to the egress node which returns them back to the ingress node. Probing may be done per class, i.e., probe packets have the same type of service header information as the traffic class being engineered. Based on the information in the returning probe packets, the ingress node is able to compute the one-way LSP statistics. Estimators of LSP statistics from the probes are obtained reliably and efficiently using bootstrap resampling techniques (see below). Recent measurements in the Internet indicate little variations of aggregate traffic on links in 5-min intervals [12]. This quasi-stationarity condition where traffic statistics change relatively slowly (much longer than the round-trip delay between the ingress and egress nodes) facilitates traffic engineering and load balancing based on measurement of LSP statistics.

The derivative can be derived by measuring these statistics at different loads. Specifically, the ingress node varies the traffic on each LSP slightly, measure the statistics before and after the variation, and use the scaled difference as an estimate of marginal congestion measure. For example, let $D_s(x_s)$ be the delay on LSP s when x_s amount of traffic is routed on the LSP, and let $D_s(x_s + \epsilon)$ be the new delay when an amount ϵ of traffic is added to the LSP. Then the marginal delay $D'_s(x_s)$ is estimated as $(D_s(x_s + \epsilon) - D_s(x_s))/\epsilon$. Note that fixed propagation delay has no effect on the derivative.

Some of the features of MATE include:

- distributed adaptive load-balancing algorithm,
- end-to-end control between ingress and egress nodes,
- no new hardware or protocol requirement in intermediate nodes,
- no knowledge of traffic demand is required,
- no assumption on the scheduling or buffer management schemes at a node,
- optimization decision based on path congestion measure,
- minimal packet reordering, and
- no clock synchronization between two nodes (see Section 2.3).

2.2. Traffic filtering and distribution

The traffic filtering and distribution function first distributes the engineered traffic for a given IE pair equally among N bins, where the number of bins determines the minimum amount of the traffic that can be shifted. If the total incoming traffic to be engineered is of rate R bps, each bin would receive an amount of $r = R/N$ bps. Then, the traffic from the N bins is mapped into the M LSPs according to the MATE algorithm described in Section 2.1.

The engineered traffic can be filtered and distributed into the N bins in a number of ways. A simple method is to distribute the traffic on a per-packet basis without filtering. For example, one may distribute incoming packets at the ingress node to the bins in a round-robin fashion. Although it does not have to maintain any per-flow state, the method suffers from potentially having to reorder an excessive number of packets for a given flow. On the other extreme, one may filter the traffic on a per-flow basis (e.g., based on <source IP address, source port, destination IP address, destination port, IP protocol> tuple), and distribute the flows to the bins such that the loads are similar. Although per-flow traffic filtering and distribution preserves packet sequencing, this approach has to maintain a large number of states to keep track of each active flow.

MATE filters the incoming packets using a hash function on the IP field(s). The fields can be based on the source and destination address pair, or other combinations. A typical hash function is based on cyclic redundancy check (CRC). The purpose of the hash function is to randomize the address space to prevent hot spots. Traffic can be distributed into the N bins by applying a modulo-N operation on the hash space. Note that packet sequence for each flow is maintained with this method.

After the engineered traffic is distributed into the N bins, a second function maps each bin to the corresponding LSP according to the MATE algorithm. The rule for the second function is very simple. If LSP i is to receive twice as much traffic as LSP j, then LSP i should receive traffic from twice as many bins as LSP j. The value N should be chosen so that the smallest amount of traffic that can be shifted, which is equal to $1/N$ of the total incoming traffic, has a reasonable granularity.

2.3. Traffic measurement and analysis

MATE does not require each node to perform traffic measurement. Only the ingress and egress nodes are required to participate in the measurement process.

Our experience suggests that packet delay is a metric that can be reliably measured. The delay of a packet on an LSP can be obtained by transmitting a probe packet from the ingress node to the egress node. The probe packet is time-stamped at the ingress node at time T_1 and recorded at the egress node at time T_2. If the ingress' clock is faster than the egress' clock by T_d, then the total packet delay (i.e, queueing time, propagation time, and processing time) is $T_2 - T_1 + T_d$. A group of probe packets sent one at a time on an LSP can easily yield an estimate of the mean packet delay $E[T_2 - T_1] + T_d$. The reliability of the estimator can be evaluated by bootstrapping (see details below) to give the confidence interval for the mean delay. An important point to note is that, since we use only marginal delay, the value of T_d is not required. Therefore, clock synchronization is not necessary.

Packet loss probability is another metric that can be estimated by a group of probe packets. In general, only reasonably high packet loss rates can be reliably observed. Packet loss probability can be estimated by encoding a sequence number in the probe packet to notify the egress node how many probe packets have been transmitted by the ingress node, and another field in the probe packet to indicate how many probe packets have been received by the egress node. When a probe packet returns, the ingress node is able to estimate the one-way packet loss probability based on the number of probe packets that has been transmitted and the number that has been received. The advantage of this approach is that it is resilient to losses in the reverse direction.

The bootstrap is a powerful technique for assessing the accuracy of a parameter estimator in situations where conventional techniques are not valid [14]. Most other techniques for computing the variance of parameter estimators or for setting confidence intervals for the true parameter assume that the size of the available set of sample values is sufficiently large, so that asymptotic results (central limit theorem) can be applied. However, in many situations the sample size is necessarily limited, such is the case in traffic engineering mechanisms like MATE, where the probe packet load should not consume significant network resources. In MATE, we use the bootstrap to obtain reliable estimates of the congestion measures of the mean delay and cell loss rate from a given set of measurements obtained via the probe packets. By selecting a desirable confidence interval, we get a dynamic way of specifying the number of observations needed. This provides a built-in reliability estimator which automatically selects the required number of probe packets to send. We have found this quite useful in our implementations, in comparison with schemes where the number of probe packets is set in an ad hoc manner, and the number of probes may be too small or too large. The following is a basic procedure for computing a confidence interval:

- *Step* 0: Suppose the original sample is $X = \{x_1, x_2, \ldots, x_m\}$.
- *Step* 1: Draw a random sample of m values, with replacement, from X. This produces the bootstrap resample Y.
- *Step* 2: Calculate the mean for Y (say, μ_1).
- *Step* 3: Repeat steps 1 and 2 a large number of times to obtain n bootstrap estimates $\mu_1, \mu_2, \ldots, \mu_n$.
- *Step* 4: Sort the bootstrap estimates into increasing order $\mu(1), \ldots, \mu(n)$.
- *Step* 5: The desired $(1 - \alpha)100\%$ bootstrap confidence interval for the mean is $(\mu(q_1),\ \mu(q_2))$, where $q_1 = (n\alpha/2)$ and $q_2 = n - q_1 + 1$.

3. MATE stability

In this section we present an analytical model of MATE and prove their stability and optimality.

3.1. Model

We model a MATE network by a set L of unidirectional links. It is shared by a set S of IE node pairs, indexed $1, 2, \ldots, S$. Each of these IE pairs s has a set $P_s \subseteq 2^L$ of LSPs available to it. Note that, by definition, no two (distinct) IE pair uses the same LSP, even though some of their LSPs may share links. Hence P_s are disjoint sets.

An IE pair s has a total input traffic of rate r_s and routes x_{sp} amount of it on LSP $p \in P_s$ such that

$$\sum_{p \in P_s} x_{sp} = r_s, \quad \forall s.$$

Let $x_s = (x_{sp}, p \in P_s)$ be the rate vector of s, and let $x = (x_{sp}, p \in P_s, s \in S)$ the vector of all rates. The flow on a link $l \in L$ has a rate that is the sum of source rates on all LSPs that traverse link l:

$$x^l = \sum_{s \in S} \sum_{l \in p, p \in P_s} x_{sp}.$$

Associated with each link l is a cost $C_l(x^l)$ as a function of the link flow x^l. We assume that, for all l, $C_l(\cdot)$ is convex (and hence continuous).

Our objective is to minimize the total cost $C(x) = \sum_l C_l(x^l)$ by optimally routing the traffics on LSPs in $\bigcup_s P_s$:

$$\min_x \qquad C(x) = \sum_l C_l(x^l) \tag{1}$$

$$\text{subject to } \sum_{p \in P_s} x_{sp} = r_s, \quad \forall s \in S, \tag{2}$$

$$x_{sp} \geqslant 0, \quad \forall p \in P_s, s \in S. \tag{3}$$

A vector x is called a *feasible rate* if it satisfies (2) and (3). A feasible rate x is called *optimal* if it is a minimizer to the problem (1)–(3).

As observed in [4, Chapter 5], the derivative of the objective function with respect to x_{sp} is

$$\frac{\partial C}{\partial x_{sp}}(x) = \sum_{l \in p} C_l'(x^l).$$

We will interpret $C_l'(x^l)$ as the first derivative length of link l, and $\partial C / \partial x_{sp}(x)$ as the (first derivative) length of LSP p.

The following characterization of optimal rate is a direct consequence of the Kuhn–Tucker theorem (see also [4, Chapter 5]). It says that at optimality a pair splits its traffic only among LSPs that have the minimum (and hence equal) first derivative lengths.

Theorem 1. *The rate vector x^* is optimal if and only if, for each pair s, all LSPs $p \in P_s$ with positive flows have minimum (and equal) first derivative lengths.*

3.2. Asynchronous algorithm

A standard technique to solve the constrained optimization problem (1)–(3) is the gradient projection algorithm. In such an algorithm routing is iteratively adjusted in opposite direction of the gradient and projected onto the feasible space defined by (2) and (3). Each iteration of the algorithm takes the form

$$x(t+1) = [x(t) - \gamma \nabla C(t)]^+,$$

where $\gamma > 0$ is a stepsize and should be chosen sufficiently small, $\nabla C(t)$ is a vector whose (s,p)th element is the first derivative length $[\nabla C(t)]_{sp} = \partial C / \partial x_{sp}$ of LSP p at time t, and $[z]^+$ is the projection of a vector z onto the feasible space. The algorithm terminates when there is no appreciable change, i.e., $||x(t+1) - x(t)|| < \epsilon$ for some predefined ϵ.

Note that the above iteration can be distributively carried out by each pair s without the need to coordinate with other pairs:

$$x_s(t+1) = [x_s(t) - \gamma \nabla C_s(t)]^+, \tag{4}$$

where $x_s(t) = (x_{sp}(t), p \in P_s)$ is rate vector of s at time t, and $\nabla C_s(t) = (\partial C / \partial x_{sp}(x(t)),\ p \in P_s)$ is the vector of first derivative lengths of LSPs in P_s. However (4) is not realistic, for two reasons.

First (4) assumes all updates are synchronized. Second it assumes zero feedback delay. Specifically (4) assumes that as soon as the IE pairs have calculated a new rate vector $x(t)$, it is reflected immediately in all the link flows:

$$x^l(t) = \sum_s \sum_{l \in p, p \in P_s} x_{sp}(t) \tag{5}$$

and in all the first derivative lengths:

$$\frac{\partial C}{\partial x_{sp}}(x(t)) = \sum_{l \in p} C_l'(x^l(t)). \qquad (6)$$

Moreover all pairs s have available these new values in $\nabla C_s(t)$ for computation of the rate vector in the next period. In practice the IE pairs update their rates asynchronously and in an uncoordinated manner. Moreover the first derivative length of a LSP can only be estimated empirically by averaging several measurements over a period of time. We now extend the model to take these factors into account.

Let $T_s \subseteq \{1, 2, \dots, \}$ be a set of times at which IE pair s adjusts its rate based on its current knowledge of the (first derivative) lengths of LSPs $p \in P_s$. At a time $t \in T_s$, s calculates a new rate vector

$$x_s(t+1) = [x_s(t) - \gamma \lambda_s(t)]^+ \qquad (7)$$

and, starting from time $t+1$, splits its traffic r_s along its LSPs in P_s according to $x_s(t+1)$ until after the next update time in T_s. Here $\lambda_s(t)$ is an estimate of the first derivative length vector at time t, and is calculated as follows.

The new rates calculated by the IE pairs may be reflected in the link flows after certain delays. We model this by (cf. (5))

$$\hat{x}^l(t) = \sum_{t'=t-t_0}^{t} \sum_{s} \sum_{l \in p, p \in P_s} a_{lsp}(t', t) x_{sp}(t'), \qquad (8)$$

$$\sum_{t'=t-t_0}^{t} a_{lsp}(t', t) = 1, \quad \forall t, \forall l, s, p \in P_s. \qquad (9)$$

In the above $\hat{x}^l(t)$ represents the flow rate available at link l at time t and is an weighted average (convex sum) of past source rates $x_{sp}(t')$. The weights $a_{lsp}(t', t)$ depend on (l, s, p, t) and can be different between each source s and link l, on different LSPs p, and at different times t. This model is very general and includes in particular the following two popular types:

- Latest data only: only the latest rate $x_{sp}(\tau)$, for some (typically *unknown*) $\tau \in \{t - t_0, \dots, t\}$, is used in the measurement of $\hat{x}^l(t)$, i.e., $a_{lsp}(t', t) = 1$ if $t' = \tau$ and 0 otherwise.
- Latest average: only the average over the latest k rates is used in the measurement of $\hat{x}^l(t)$, i.e., $a_{lsp}(t', t) > 0$ for $t' = \tau - k + 1, \dots, \tau$ and 0 otherwise, for some (typically *unknown*) $\tau \in \{t - t_0, \dots, t\}$.

An IE pair s estimates the first derivative length of an LSP $p \in P_s$ by asynchronously collecting a certain number of measurements (using probe packets, see below), and forming their mean. Hence (cf. (6))

$$\lambda_{sp}(t) = \sum_{t'=t-t_0}^{t} \sum_{l \in p} b_{lsp}(t', t) C_l'(\hat{x}^l(t')), \qquad (10)$$

$$\sum_{t'=t-t_0}^{t} b_{lsp}(t', t) = 1, \quad \forall t, \forall l, s, p \in P_s. \qquad (11)$$

Again the estimate is obtained by 'averaging' over the past values of LSP lengths, and can depend on (l, s, p, t). The model is very general and include the special cases of using only the last received measurement or the average over the last k values, as discussed above. The interpretation in both cases is that the measurements $\sum_{l \in p} C_l'(\hat{x}^l(t'))$ for $t' > \tau$ have not been received by s by time t, and the measurements for $t' < \tau$ (latest data only) or for $t' \leqslant \tau - k$ (latest average) have been discarded.

This concludes the description of our algorithm model (Eqs. (7)–(11)). The model is similar to that in [13], with two differences. First their model distinguishes between the desired rate $x(t)$ as calculated by the projection algorithm and the actual realized source rate $\hat{x}(t)$. The actual rate $\hat{x}(t)$ is a convex combination of the current desired rate $x(t)$ and the previous actual rate $\hat{x}(t-1)$. This models the fact that a desired rate $x(t)$ may not be realized immediately, as in a virtual circuit network where virtual circuits may persist over several update cycles. We are however only dealing with IP datagrams and hence it is reasonable to assume that each ingress node can shift its traffic among the LSPs available to it immediately after each update. Second their model assumes that, at time t, each s has available the current first derivative lengths $\sum_{l \in p} C'(\hat{x}^l(t))$ and uses it in place of the gradient in the update algorithm. We however assume that, at time t, s may only have outdated first derivative lengths (see (10) and (11)); moreover s uses a weighted average over several past lengths in the update algorithm. This is because, in our case, s can only estimate the first derivative lengths through noisy measurement. Despite these differences, stability can be established using the same techniques.

The next result states that the algorithm converges to an optimal routing, provided the following conditions are satisfied:

C1 The cost functions $C_l(z)$ are twice continuously differentiable and convex.

C2 Their derivatives $C_l'(z)$ are Lipschitz over any bounded sets, i.e., for any bounded set $B_l \subset \Re$ there exists a constant c_l such that for all $z, z' \in B_l$, we have $|C_l'(z) - C_l'(z')| \leqslant c_l|z - z'|$.

C3 For any constant c the sets $\{z | C_l(z) \leqslant c\}$ are bounded.

C4 The time interval between updates is bounded.

Theorem 2. *Under conditions* C1–C4, *starting from any initial vector* $x(0)$, *there exists a sufficiently small stepsize* γ *such that any accumulation point of the sequence* $\{x(t)\}$ *generated by the asynchronous algorithm is optimal.*

A more careful accounting shows that the stepsize γ, and hence the speed of convergence, depends on the degree of asynchronism as expressed by the parameter t_0 defined in (8), the 'steepness' of the cost function as expressed by the Lipschitz constant in condition C2, and the size of the network. For ease of exposition, suppose the cost functions are uniformly globally Lipschitz, i.e., for all links l and all z, z', we have

$$|C_l'(z) - C_l'(z')| \leqslant L|z - z'|.$$

Theorem 3. *An upper bound in Theorem* 2 *is*

$$\gamma < \frac{1}{L(1 + \pi h \lambda (2t_0 + 1))},$$

where π *is the total number of LSPs in the network*, h *is the number of hops in the longest* (*maximum-hop*) *LSP*, λ *is the maximum number of LSPs going through a link, and* t_0, *defined in* (8), *measures the degree of asynchronism.*

The theorem suggests that the larger the degree of asynchronism measured by t_0, the smaller the stepsize (required for convergence) and hence slower the convergence.

4. Experimental methodology

In this section, we use simulations to evaluate the effectiveness of MATE. We concentrate on two network topologies: one with a single IE pair connected by multiple LSPs, and the other with multiple IE pairs where some links are shared among the LSPs from different pairs, as shown in Figs. 3 and 4. All links are identical so that the LSPs have the same bottleneck link bandwidth. Note that in the latter case, there is a considerable interaction between the pairs.

We wrote a packet level discrete-event simulator, which supports entities such as packet queues, switched LSPs, network connections. We consider

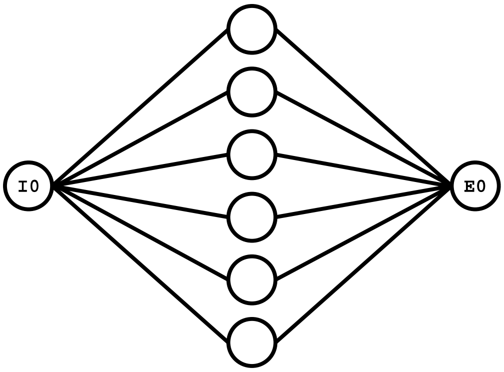


Fig. 3. Experiment network topology 1.

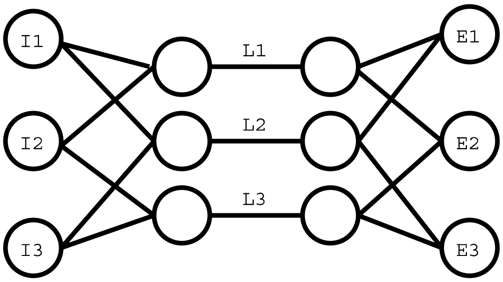


Fig. 4. Experiment network topology 2.

networking environments where the traffic conditions vary due to changes in network load (link utilization), for example, due to "rush hour" conditions, or some LSP failures, and traffic variations due to correlations and dependencies. We have two types of traffic in our simulator: engineered traffic and cross-traffic. The engineered traffic is the traffic that needs to be balanced, and the cross-traffic is the background traffic that we have no control over such as traffic on short-lived connections. We assign a lifetime to each traffic source so we are able to simulate the dynamic behavior of a network by switching on and off cross-traffic sources. We consider a traffic model which exhibits short-range dependencies, such as Poisson, and another model which can be tuned to approximate long range dependence. For the latter we use the DAR(p) process (discrete autoregressive process of order p) [11]. The parameter p determines the time-scale over which traffic dependency and correlation are exhibited. If p is 1, the process is a standard Markov process. In our experiments we set p to a value of 10; this leads to a substantial degree of correlation in the generated traces.

In each of our simulations, the engineered traffic for each pair flows from the ingress node to the egress node. The cross-traffic enters at the intermediate node and exits at egress node(s). We consider two implementations of the basic algorithm. In the first one, a small random delay is introduced before the algorithm moves from the monitoring phase to traffic engineering phase upon detection of change in traffic conditions. This damping mechanism reduces synchronization among multiple ingress nodes. In the second implementation, there is a coordination among the ingress nodes so that only one ingress node at a time enters the traffic engineering phase. This obviously requires a special coordination protocol. We omit the details in this paper.

5. Simulation results

In this section, we present simulation results that illustrate the convergence properties of MATE.

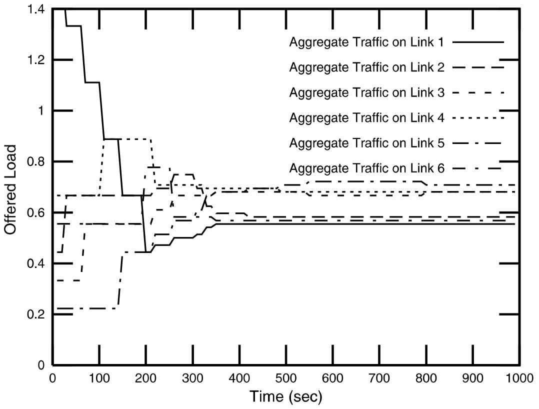


Fig. 5. Offered load under Poisson traffic for network topology 1.

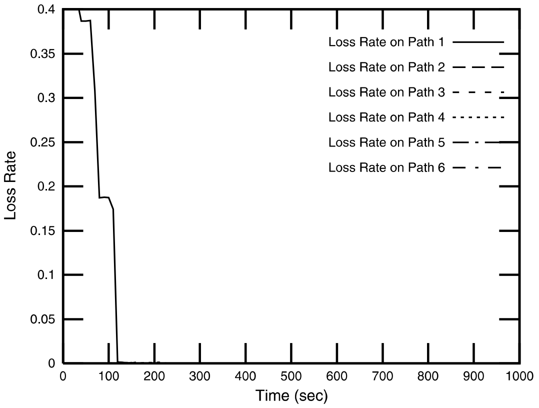


Fig. 6. Loss under Poisson traffic for network topology 1.

First we present two sets of results for the single IE pair. Figs. 5 and 6 show the results of an experiment with Poisson traffic on the network in Fig. 3. Initially, all of the engineered traffic streams are routed on one of the LSPs, and cross-traffic enter the network at the intermediate nodes connecting the ingress and egress nodes. We have an unbalanced situation with one heavily congested LSP and five lightly loaded LSPs. As shown in the plot, the algorithm is able to successfully reduce the engineered traffic from the overloaded link and distribute them to the under-utilized links. The loss curve shows clearly that the loss rate on the first

LSP dropped from 40% to a value that is too small to observe. The loss rates on the other LSPs are maintained at negligible levels throughout the simulation. The final traffic distribution converges to a steady state, where utilizations are very close on all LSPs. We observe similar behavior in Figs. 7 and 8 where the Poisson streams are replaced with DAR traffic streams. The probe traffic required in each phase of the algorithm is around 0.5% of the engineered traffic.

Figs. 9–12 show the simulation scenario for Fig. 4 under the two implementations mentioned earlier. Again the engineered traffic streams travel

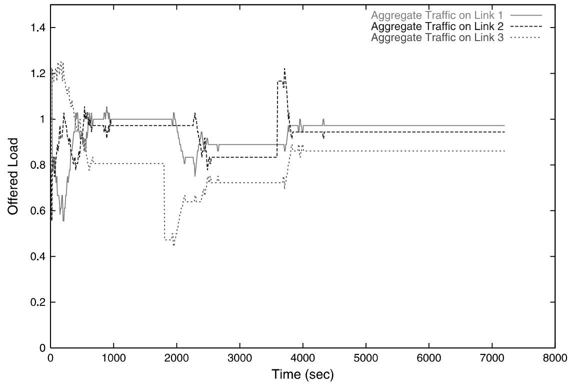


Fig. 9. Offered load under Poisson traffic for network topology 2.

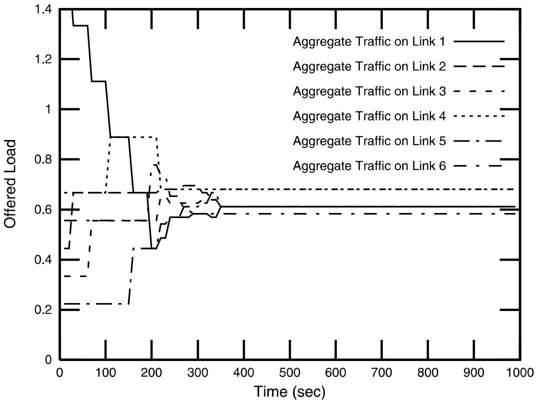


Fig. 7. Offered load under DAR traffic for network topology 1.

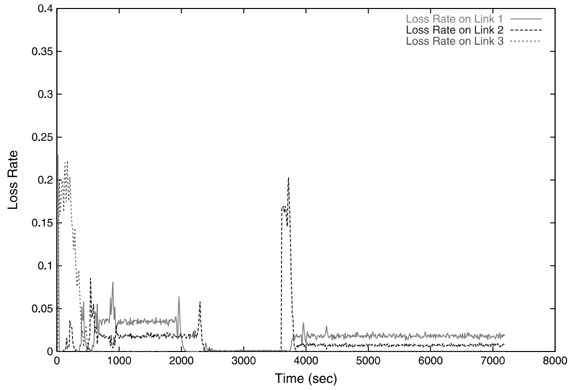


Fig. 10. Loss under Poisson traffic for network topology 2.

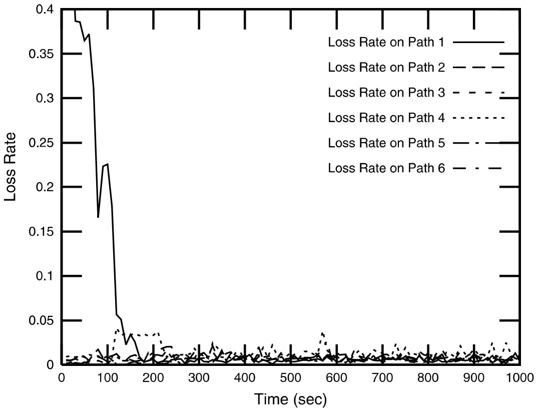


Fig. 8. Loss under DAR traffic for network topology 1.

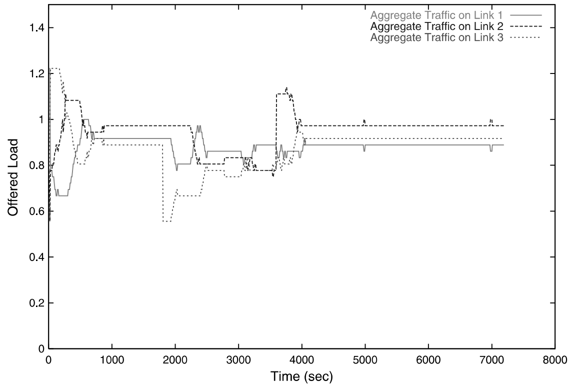


Fig. 11. Offered load under Poisson traffic with coordination for network topology 2.

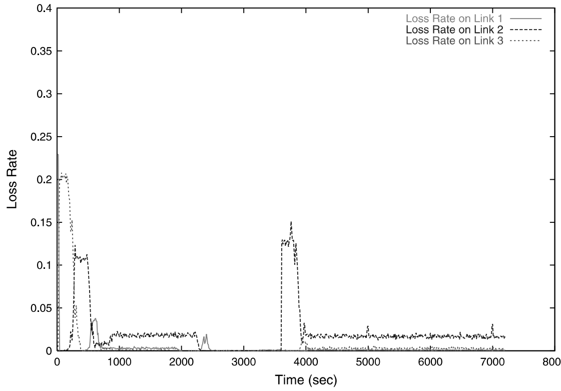


Fig. 12. Loss under Poisson traffic with coordination for network topology 2.

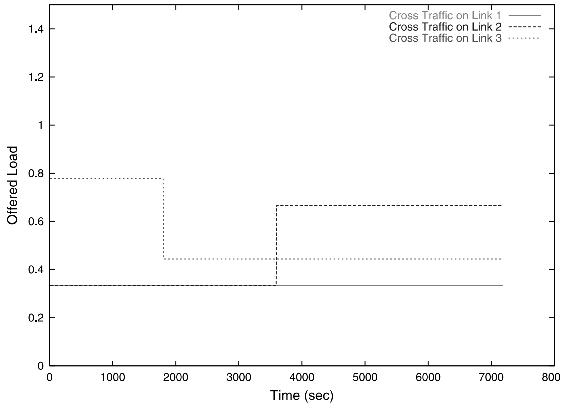


Fig. 13. Cross-traffic for network topology 2.

from the ingress node to the egress node, and the cross-traffic enters through the intermediate nodes and exit at the egress nodes. The cross-traffic dynamics are shown in Fig. 13. There is a decrease in cross-traffic on link 3 right before 2000 s and a increase in cross-traffic on link 2 around 3600 s. In order to balance traffic, the algorithms must shift traffic into link 3 and possibly out of link 2. Both implementations essentially achieve the same performance, where utilizations and loss rates on three LSPs are comparable. Figs. 14 and 15 show the same simulation with DAR traffic instead of Poisson traffic where coordination among ingress node is used.

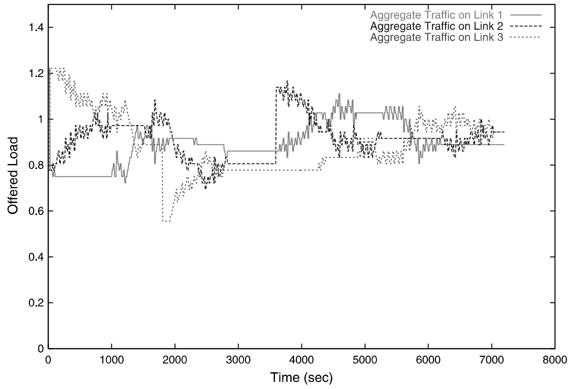


Fig. 14. Offered load under DAR traffic with coordination for network topology 2.

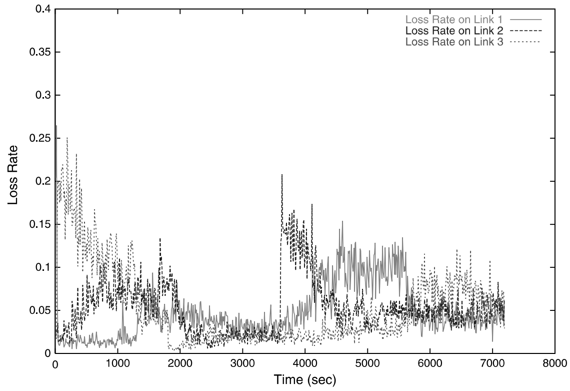


Fig. 15. Loss under DAR traffic with coordination for network topology 2.

6. Conclusion

Our focus on this paper is to apply adaptive traffic engineering to utilize network resource more efficiently and minimize congestion. We have proposed a class of algorithms called MATE, which tries to achieve these objectives using minimal assumptions through a combination of techniques such as bootstrap probe packets, which control the amount of extra traffic, and marginal delays that are easily measurable and do not require clock synchronization. Further, we prove the stability and optimality of MATE. Our simulation results show that MATE can effectively remove traffic imbalances among available LSPs. We observe that, in many cases, high packet loss rates can be significantly reduced by properly shifting some traffic to less loaded LSPs. This should benefit many applications such as TCP. For future work we will consider more realistic networking environments and examine the impact of MATE on the application level.

Appendix A

Proof of Theorem 1. Since the cost function is convex the first order optimality condition is both necessary and sufficient: x^* is optimal if and only if x^* is feasible and there exist constants ξ_s such that for all (s,p)

$$\frac{\partial C}{\partial x_{sp}}(x^*) = \sum_{l\in p} C'_l(x^{*^l}) \geqslant \xi_s \tag{A.1}$$

with equality if $x^*_{sp} > 0$. Hence all LSPs $p \in P_s$ with $x^*_{sp} > 0$ have their first derivative lengths equal to ξ_s. □

Proof of Theorem 2. Its proof is adapted from that in [13]. Let $z(t) = x(t+1) - x(t)$. Using a first order Taylor expansion for C we have for some rate vector $y(t)$ [1]

$$\begin{aligned} C(x(t+1)) &= C(x(t)) + \nabla C(x(t))z(t) + \tfrac{1}{2}z(t)\nabla^2 C(y(t))z(t) \\ &\leqslant C(x(t)) + \lambda(t)z(t) + ||\nabla C(x(t)) - \lambda(t)|| \cdot ||z(t)|| + A_1||z(t)||^2, \end{aligned} \tag{A.2}$$

where $\lambda(t) = (\lambda_s(t), s \in S)$ and the constant A_1 depends on the initial vector $x(0)$. We next show that

$$\lambda(t)z(t) \leqslant -\frac{1}{\gamma}||z(t)||^2, \tag{A.3}$$

$$||\nabla C(x(t)) - \lambda(t)|| \cdot ||z(t)|| \leqslant A_2 \sum_{t'=t-2t_0}^{t} ||z(t')||^2 \tag{A.4}$$

for some constant A_2 that depends on $x(0)$.

First, note that (A.3) holds if the following holds for all s:

$$\lambda_s(t)z_s(t) \leqslant -\frac{1}{\gamma}||z_s(t)||^2. \tag{A.5}$$

For $t \notin T_s$ (A.5) trivially holds. For $t \in T_s$ apply the projection theorem [3] to (7) to obtain

$$(x(t) - \gamma\lambda(t) - x(t+1))(x(t) - x(t+1)) \leqslant 0.$$

Rearranging terms yields (A.5).

To show (A.4) note that since all norms in $\Re^n$ are equivalent there exist constants A_3 and A_4 such that

$$\begin{aligned} ||\nabla C(t) - \lambda(t)||_2 &\leqslant A_3 \max_s \max_{p\in P_s} \left| \frac{\partial C}{\partial x_{sp}}(x(t)) - \lambda_{sp}(t) \right| \\ &\leqslant A_3 \max_s \max_{p\in P_s} \sum_{l\in p} \left| C'_l(x^l(t)) - \sum_{t'=t-t_0}^{t} b_{lsp}(t',t) C'_l(\hat{x}^l(t')) \right| \qquad \text{(A.6)} \\ &\leqslant A_4 \max_s \max_{p\in P_s} \max_{l\in p} \max_{t-t_0\leqslant t'\leqslant t} \left| C'_l(x^l(t)) - C'_l(\hat{x}^l(t')) \right| \\ &= A_4 \max_{l\in L} \max_{t-t_0\leqslant t'\leqslant t} \left| C'_l(x^l(t)) - C'_l(\hat{x}^l(t')) \right|. \qquad \text{(A.7)} \end{aligned}$$

[1] For simplicity we write xy instead of the more correct notation $x^{\mathrm{T}}y$ for the inner product of two vectors x and y. We usually use $||x||$ to denote the Euclidean norm, but sometimes $||x||_2$ for emphasis.

Let $B := \{x | C(x) \leqslant C(x(0))\}$ and $B_l := \{f | f = \sum_s \sum_{l \in p, p \in P_s} x_{sp}$, for some $x \in B\}$. In words, B is the set of rate vectors x at which the total cost $C(x)$ is no greater than the initial cost. As will be seen, provided the stepsize γ is sufficiently small, $C(x(t)) \leqslant C(x(0))$ for all t (see (A.11)). That is, B is the set of all possible rate vectors given the initial $x(0)$. (This can be made more rigorous by induction.) Then B_l is the set of all possible link flows on link l. By condition C2, we have for some constants A_5, A_6, A_2

$$\begin{aligned}
||\nabla C(t) - \lambda(t)||_2 &\leqslant A_5 \max_{l \in L} \max_{t-t_0 \leqslant t' \leqslant t} \left| x^l(t) - \hat{x}^l(t') \right| \\
&\leqslant A_5 \max_{l \in L} \max_{t-t_0 \leqslant t' \leqslant t} \sum_s \sum_{l \in p, p \in P_s} \left| x_{sp}(t) - \sum_{t''=t'-t_0}^{t'} a_{lsp}(t'', t') x_{sp}(t'') \right| &\text{(A.8)} \\
&\leqslant A_6 \max_s \max_{p \in P_s} \max_{t-t_0 \leqslant t' \leqslant t} \max_{t'-t_0 \leqslant t'' \leqslant t'} \left| x_{sp}(t) - x_{sp}(t'') \right| &\text{(A.9)} \\
&\leqslant A_6 \max_s \max_{p \in P_s} \left| x_{sp}(t) - x_{sp}(t-1) \right| + \cdots + \left| x_{sp}(t-2t_0+1) - x_{sp}(t-2t_0) \right| \\
&\leqslant A_2 \sum_{t'=t-2t_0}^{t-1} ||z(t')||_2 &\text{(A.10)}
\end{aligned}$$

Hence

$$||\nabla C(x(t)) - \lambda(t)|| \cdot ||z(t)|| \leqslant A_2 \sum_{t'=t-2t_0}^{t-1} ||z(t')|| \cdot ||z(t)|| \leqslant A_2 \sum_{t'=t-2t_0}^{t} ||z(t')||^2$$

where the last inequality follows from the fact that the convex function $\sum_i y_i y + y^2 - \sum_i y_i y$ attains its minimum of zero over $\{y_i, y | y_i \geqslant 0, y \geqslant 0\}$ at the origin. This completes the proof of (A.4).

Substituting (A.3) and (A.4) into (A.2) we have

$$C(x(t+1)) \leqslant C(x(t)) - \left(\frac{1}{\gamma} - A_1\right) ||z(t)||^2 + A_2 \sum_{t'=t-2t_0}^{t} ||z(t')||^2.$$

Summing over all t we have

$$\begin{aligned}
C(x(t+1)) &\leqslant C(x(0)) - \left(\frac{1}{\gamma} - A_1\right) \sum_{\tau=0}^{t} ||z(\tau)||^2 + A_2 \sum_{\tau=0}^{t} \sum_{t'=\tau-2t_0}^{\tau} ||z(t')||^2 \\
&\leqslant C(x(0)) - \left(\frac{1}{\gamma} - A_1 - A_2(2t_0+1)\right) \sum_{\tau=0}^{t} ||z(\tau)||^2. &\text{(A.11)}
\end{aligned}$$

Choose γ small enough such that $1/\gamma - A_1 - A_2(2t_0+1) > 0$. Since $x(t)$ is in a compact set and C is continuous, $C(x(t))$ is lower bounded. Then since $C(x(t))$ is bounded for all t we must have $\sum_{\tau=0}^{\infty} ||z(\tau)||^2 < \infty$, which implies

$$||z(t)|| \to 0 \quad \text{as } t \to \infty. \tag{A.12}$$

Substituting this into (A.10) we conclude that

$$\lambda(t) \to \nabla C(t) \quad \text{as } t \to \infty. \tag{A.13}$$

Let x^* be an accumulation point of $\{x(t)\}$. One exists since $\{x(t)\}$ is in a compact set. By (A.13) and the fact that C is continuously differentiable we have

$$\lambda(t) \to \lim_{t\to\infty} \nabla C(t) = \nabla C(x^*). \tag{A.14}$$

Since the time interval between updates is bounded, for any s, we can find a subsequence $\{x(t_k), t_k \in T_s\}$ that converges to x^*, i.e., $\lim_k x(t_k) = x^*$. Applying again the projection theorem [4] to (7) we have for any feasible x_s

$$(x_s(t_k) - \gamma\lambda_s(t_k) - x_s(t_k+1))(x_s - x_s(t_k+1)) \leqslant 0,$$
$$(z_s(t_k) + \gamma\lambda_s(t_k))(x_s - x_s(t_k+1)) \geqslant 0.$$

Taking $k \to \infty$ we have by (A.12) and (A.14) that for any feasible x_s,

$$\nabla C_s(x_s^*)(x_s - \lim_{k\to\infty} x_s(t_k+1)) \geqslant 0.$$

Since $z(t) = x(t+1) - x(t) \to 0$ by (A.12), we have $\lim_k x_s(t_k+1) = \lim_k x_s(t_k) = x_s^*$, and hence

$$\nabla C_s(x_s^*)(x_s - x_s^*) \geqslant 0$$

for any feasible x_s. Summing over all s, we have for any feasible x

$$\nabla C(x^*)(x - x^*) \geqslant 0$$

which, since C is convex, is necessary and sufficient for x^* to be optimal. □

Proof of Theorem 3. Since the cost functions C_l are globally Lipschitz uniformly in l, the constant A_1 in (A.2) equals the Lipschitz constant L. For any n-tuple z, $||z||_2 \leqslant \sqrt{n}||z||_\infty$, and hence the constant A_3 in (A.6) is π. Similarly, since $||z||_1 \leqslant n||z||_\infty$, the constant A_4 in (A.7) is πh. By Lipschitz continuity, the constant A_5 in (A.8) is $A_5 = A_4 L = \pi h L$, the constant A_6 following is $A_6 = A_5\lambda = \pi h L \lambda$. Finally, since $||z||_\infty \leqslant ||z||_2$, we have $A_2 = A_6 = \pi h L \lambda$ in (A.11). Hence from (A.11) an upper bound for the stepsize γ is

$$\gamma < \frac{1}{A_1 + A_2(2t_0+1)} = \frac{1}{L(1+\pi h\lambda(2t_0+1))}. \quad \square$$

References

[1] D. Awduche, A. Chui, A. Elwalid, I. Widjaja, X. Xiao, Overview and Principles of Internet Traffic Engineering. Available from Internet draft <draft-ietf-tewg-principles-02.txt >, May 2001.

[2] D.O. Awduche et al., RSVP-TE: Extensions to RSVP for LSP Tunnels, IETF RFC 3209, December 2001.

[3] D. Bertsekas, Nonlinear Programming, Athena Scientific, 1995.

[4] D. Bertsekas, R. Gallager, Data Networks, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1992.

[5] B. Fortz, M. Thorup, Internet traffic engineering by optimizing OSPF weights, in: Proceedings of INFOCOM'2000, Tel-Aviv, Israel, March 2000.

[6] B. Jamoussi et al., Constraint-based LSP setup using LDP, IETF RFC 3212, January 2002.

[7] T. Li, Y. Rekhter, Provider architecture for differentiated services and traffic engineering (PASTE), RFC 2430, October 1998.

[8] D. Mitra, K.G. Ramakrishnan, A case study of multiservice, multipriority traffic engineering design for data networks, in: Proceedings of the Globecom'99, December 1999.

[9] E.C. Rosen, A. Viswanathanm, R. Callon, Multiprotocol label switching architecture, IETF RFC 3031, January 2001.

[10] M.A. Rodrigues, K.G. Ramakrishnan, Optimal routing in shortest-path networks, ITS '94, Rio de Genero, Brazil.

[11] B.K. Ryu, A. Elwalid, The importance of long-range dependence of VBR video traffic in ATM traffic engineering, in: Proceedings of SIGCOMM'96, August 1996, pp. 3–14.

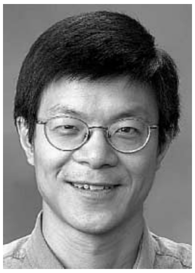
[12] K. Thompson, G.J. Miller, R. Wilder, Wide-area internet traffic patterns and characteristics, IEEE Networks 6 (6) (1997).

[13] J.N. Tsitsiklis, D.P. Bertsekas, Distributed asynchronous optimal routing in data networks, IEEE Transactions on Automatic Control 31 (4) (1986) 325–332.

[14] A.M. Zoubir, B. Boashash, The bootstrap and its applications in signal processing, IEEE Signal Processing Magazine 15 (1) (1998) 56–76.

Anwar Elwalid is with Bell Labs, Lucent Technologies, where he is a DMTS (Distinguished Member of Technical Staff). He received the BS degree in Electrical Engineering from Polytechnic Institute of New York, Brooklyn, and the Ph.D. degree in Electrical Engineering from Columbia University, New York. Since 1991 he has been with the Mathematics of Networks and Systems Research Department at Bell Labs, Murray Hill, New Jersey, where he developed theory and algorithms for network resource management and QoS support, and for the analysis and engineering of multimedia traffic. He holds several patents. His current research interests include IP and optical network architectures, traffic engineering and stochastic systems. He has been active in the Traffic Engineering and MPLS Working Groups of the IETF, and co-authored Internet Drafts and RFCs. He received best paper award from the ACM and IFIP. He has been a guest editor of the *IEEE Journal on Selected Area in Communications*, and served on the executive and technical program committees of several conferences. Dr. Elwalid is senior member of IEEE, and member of Tau Beta Pi (National Engineering Honor Society) and Sigma Xi.

Cheng Jin is a Ph.D. candidate in the Department of EECS at the University of Michigan. He received his B.Sc. in Electrical Engineering from Case Western Reserve University in 1996. His current area of research includes the placement of servers of various services inside a network and the modeling of Internet topology. He is a co-developer of the Inet topology generator.



Steven H. Low received his B.S. degree from Cornell University and Ph.D. from the University of California, Berkeley, both in electrical engineering. He was with AT&T Bell Laboratories, Murray Hill, from 1992 to 1996, and was with the University of Melbourne, Australia, from 1996 to 2000, and is now an Associate Professor at the California Institute of Technology, Pasadena. He has held visiting academic positions in the US and Hong Kong, and has consulted with companies and government in the US and Australia. He was a co-recipient of the IEEE William R. Bennett Prize Paper Award in 1997 and the 1996 R&D 100 Award. He is on the editorial board of *IEEE/ACM Transactions on Networking*. He has been a guest editor of the *IEEE Journal on Selected Area in Communications*, on the program committee of several conferences. His research interests are in the control and optimization of communications networks and protocols. His home is netlab.caltech.edu.

Indra Widjaja received the Ph.D. degree in Electrical Engineering from the University of Toronto, Toronto, Canada. From 1994 to 1997, he was an Assistant Professor of the Electrical and Computer Engineering Department of the University of Arizona. From 1997 to May 2001, he was with Fujitsu Network Communications where he was manager of systems engineering. He joined Bell Labs Research, Lucent Technologies, in May 2001. His research interests include traffic engineering, high-speed switching, and optical networking.