OpIATE: Optimization Integrated Adaptive Traffic Engineering*

Systems and Information Engineering Technical Report: SIE-020001, Revised 11/12/02

Kaushik Sinha and Stephen D. Patek
Department of Systems and Information Engineering
University of Virginia
{ks7r, patek}@virginia.edu

November 13, 2002

Abstract

The advent of Multiprotocol Label Switching (MPLS), a connection oriented technology, allows packets in datagram networks to be sent along explicit (not necessarily shortest) paths. This provides scope for the development and more importantly, implementation of traffic engineering schemes that can implicitly or explicitly optimize a network wide performance criterion. We propose a decentralized, optimization integrated adaptive traffic engineering scheme called OpIATE which seeks to adaptively balance the dynamic traffic load experienced in a service provider's MPLS enabled intra-autonomous system(AS). A key contribution of this paper is to explore the role of non-intrusive feedback parameters, such as ingress-egress flow allocation, to steer sources in a network towards a shared global perspective in terms of network routing. The OpIATE framework depends on the edge nodes in a network having the capability to solve complex flow allocation problems and does not require participation from the interior nodes of a network. Further, no assumptions on scheduling or buffering mechanisms are made. We present an analytical model which establishes the asymptotic convergence of OpIATE flow allocation updates to a globally optimal solution. Finally, we present packet-level simulation results that illustrate the application of OpIATE for different traffic scenarios and show favorable comparison with other adaptive multipath routing schemes like MATE.

1 Introduction

1.1 Motivation

The explosive growth of the Internet in the last decade raises the question of how to accommodate the ever growing demand for network services. Often, especially in the core of the Internet, the answer has been to install more bandwidth [15]. However, in today's competitive environment such "overprovisioning" is not an economically viable long term solution. With the advent of high-bandwidth applications and near-universal

*This work is supported in part by grants from the National Science Foundation: ECS-9875688 (CAREER) and ANI-9903001.

access, service providers need to approach the task of mapping traffic onto physical topologies in a fundamentally different way. It is thus necessary to develop mechanisms that efficiently allocate resources and regulate network traffic. The goal of Internet traffic engineering, as defined by the Internet Engineering Task Force, is precisely that: to control the traffic flow through a network, offering services according to specific consumer requirements while still utilizing network resources in an efficient and cost effective manner. Traffic engineering (TE) is also necessitated by the nature of routing algorithms (such as OSPF or IS-IS) in use on the Internet today. These shortest path based routing algorithms and variants thereof like ECMP [12] are static in nature and hence are incapable of dealing with the dynamic nature of offered traffic in a network. This leads to congestion when the capacity of the shortest path to a destination is exceeded or when shortest paths from various sources coincide on a particular link creating bottlenecks. Thus, the local optimization approach of shortest path routing where each source node chooses a path that is best from its perspective, concomitant with its inability to accommodate administrative policy or capacity constraints while making routing decisions, results in networkwide sub-optimal outcomes (in terms of network utilization and levels of congestion).

Dynamically changing link costs so as to approximate optimal shortest path routing for different estimated or forecasted demand matrices has been proposed in [9, 22]. However, [9] reports the problem of finding an optimal setting of link costs for an arbitrary network to be NP hard. Also, dynamic metric manipulation based TE does not seem to be a scalable solution. This is because core networks are becoming progressively more thickly meshed and more redundant and it would be difficult to ensure that an adjustment in one part of the network does not cause a cascade effect in another part. In [19, 21] the drawbacks of shortest path routing are addressed by using alternate paths to route traffic around points of congestion or network failures. However this does not provide a basis to prevent the onset of congestion and is instead a reactive form of dealing with the congestion. Further, alternate routing could conceivably result in the oscillation of “hot spots” in a network. The preceding discussion is intended to show that an opportunity exists to steer all sources in an intra-AS from an uncoordinated myopic local outlook/action framework towards a situation where all users share a similar world view in terms of traffic routing. An initiative in this direction is Constraint Based-Routing (CBR). CBR has been proposed to enable route calculation subject to a set of constraints imposed by the network (bandwidth, delay etc) or by administrative policies. An attractive feature of CBR is that by viewing network routing as a constrained optimization problem, the route calculation procedure may be used to optimize a global performance

criterion. It must be pointed out that routes computed through CBR are not necessarily shortest paths and hence, the capability for explicit routing is required. Explicit routing in IP networks leads to an undesirable increase in packet overhead as a consequence of the requirement of attaching to the packet the network layer addresses of each node in the explicit path. The development of path oriented technologies like MPLS [23] have made CBR [1] feasible, and as such was first outlined in [2]. MPLS technology brings connection-oriented forwarding techniques together with the Internet's routing protocols by establishing a virtual connection between two points in an IP network. Thus, in MPLS, a path also known as a Label Switched Path (LSP) is identified by a sequence of labels and as in virtual-circuit packet switching, a packet is forwarded along the LSP by swapping labels. Consequently, explicit routing in MPLS eliminates the need for additional packet overhead.

Optimization based routing algorithms have been suggested before [7, 10, 18, 5, 11]. These schemes seek to obtain routing solutions by solving flow optimization problems through centralized or distributed means. Drawbacks to these procedures are that (1) even though there is some implicit cooperation among the sources (by agreeing to use the same routing algorithm) there is none when it comes to allowing each source in a network to gradually "learn" the bandwidth requirements of other sources and (2) the communication requirements in [10, 18] where there is need for per link information estimation and dissemination results in the complexity of the scheme being extended to the core of the network. The former drawback results in sources admitting levels of traffic without giving other sources an opportunity to optimally reroute theirs (in a gradual manner) and avoid potential route oscillation. Recently, a distributed asynchronous optimization based algorithm called MATE [8] has been devised that seeks to avoid network congestion by adaptively balancing load using feedback information about path congestion. MATE works by implicitly solving the routing optimization model using a construct called the first derivative path length (dynamically estimated from delay information obtained by selectively probing LSPs available to each source in the network).

1.2 Overview

The guiding objective of this paper is the development of an adaptive traffic engineering algorithm that does not suffer from the drawbacks of shortest path based routing algorithms (and variants thereof), minimizes network

[1] For a discussion of CBR issues related to MPLS domains and references to other pertinent CBR literature the reader is referred to [1].

congestion based on the measure of offered traffic load and can be applied with minimal changes to an existing service provider network. The context of application for this algorithm is Intra-AS routing where LSPs have already been laid out and is intended for traffic not requiring any form of bandwidth reservation (and best-effort in that sense). The contributions of this paper are (1) the development of a methodology that relies on a deliberate smooth application of optimal traffic mapping to ensure the graceful transition between an uncoordinated myopic outlook to a situation where all sources share a similar world view in terms of traffic routing and (2) an enhanced understanding of the role of non-intrusive feedback information for traffic engineering. Non-intrusive feedback information in the context of the paper is defined to be feedback that requires minimal participation from links making up the network i.e., individual links are not required to collect and send feedback information. This condition is to ensure that complexity is restricted to the edges of the network. A second condition for non-intrusive feedback is, as the name suggests, that it is not obtained through intrusive means (like probing for instance as in MATE). This second condition ensures that the algorithms developed do not inadvertently affect network performance by precipitating congestion.

1.3 Paper organization

The remainder of this paper is organized as follows. In Section 2, we discuss the design of OpIATE and include a description of the underlying optimization model. This is followed by Section 3 where we present the basic scheme where each source in the network calls upon the OpIATE algorithm at synchronized periods of time, makes flow updates and sends out feedback to the other sources sharing the network. This section also states a theoretical result establishing the convergence of the basic synchronous scheme to the optimal solution of the global routing problem. In Section 4, we discuss various issues in developing a packet-level implementation of OpIATE. We present simulation results in Section 5, experimenting with the effect of the variation in simulation parameters on metrics like packet loss rate and aggregate utilization. In Section 6, we summarize the paper and outline directions for future research.

2 Problem Formulation

We adopt an optimization model similar to that in MATE. The service provider network is modeled by a set L of unidirectional links indexed $1, 2, \ldots, \bar{L}$, where $\bar{L}$ denotes the total number of links in the network. Adopting a fluid model perspective, the traffic between any two points in this network is viewed as a continuous stream of bits, measured in Megabits per second (Mbps). Further, nodes at the edge of the network serve as origin or destination points for traffic and are designated as ingress nodes and egress nodes respectively. The remaining nodes in the network are referred to as core nodes. Each node acting as an ingress node can send traffic to one or more nodes designated as its egress nodes. Further, each such pair is referred to as an ingress-egress (IE) node pair. Thus, in general, the service provider network is shared by a set S of IE node pairs, indexed $1, 2, \ldots, \bar{S}$, where $\bar{S}$ denotes the total number of IE pairs under consideration. Each IE pair s has a set P_s of LSPs available to it. Note that, no two (distinct) IE pairs use the same LSP, even though some of their LSPs may have common links. At any moment in time an IE pair s has a total transmission rate of u_s and distributes x_{sp} amount of it on LSP $p \in P_s$ such that $\sum_{p \in P_s} x_{sp} = u_s$. Let $x_s = (x_{sp})_{p \in P_s}$ be the vector of flow rates on each of the LSPs of source s, and let $x = (x_s)_{s \in S}$ be the vector of all rates. Thus, the flow rate on a link $l \in L$, $\phi_l \triangleq \sum_{s \in S} \sum_{p \in P_s : l \in p} x_{sp}$ is the sum of the source rates of all LSPs that traverse that particular link l. Further, we assume that:

(A1) All IE pairs $s \in S$ are aware of the network topology (i.e, IE pairs, link capacities).

(A2) Associated with each link l is C_l the link capacity (in Mbps) and a cost $f_l(\phi_l)$ which is a function of the link flow ϕ_l. The link cost function we would like to minimize is delay. This is taken to be proportional to the expected waiting time in an M/M/1 queue with arrival rate ϕ_l and service rate C_l and is given by $f_l(\phi_l) = \frac{\phi_l}{(C_l - \phi_l)}$.

(A3) Each IE pair s has a desired flow level it would like to be able to achieve equal to d_s. We denote $d = (d_s)_{s \in S}$ to be the vector of desired flow levels. We assume that d is ϵ-*routable* in the sense that there exists a flow vector x which satisfies the desired flow of all IE pairs and is such that $\phi_l \leq \epsilon C_l, \quad \forall l \in L$, where $\epsilon \in (0, 1)$ is a parameter selected by the network administrator that determines the maximum utilization of any link.

In practice, the desired flow level, d_s, would equal the average offered load for the IE pair s, and as a result of its time varying nature (due to time of day effects) would need to be estimated (cf. Section 4).

Our objective is to minimize network congestion by optimally allocating flow on each LSP in the network according to the following non-linear program.

$$\textbf{NLP-I:} \qquad \min_{x} \quad f(x) = \sum_{l} f_l(\phi_l) \tag{1}$$

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

$$\phi_l \leq \epsilon C_l, \quad \forall l \ \in L \tag{3}$$

$$x_{sp} \geq 0, \quad \forall p \ \in P_s, \quad s \in S \tag{4}$$

$$\text{where} \quad \phi_l = \sum_{s \in S} \sum_{p \in P_s \,:\, l \in p} x_{sp}, \quad \forall\, l \in L.$$

Remarks:

1. **NLP-I** will be referred to sometimes as the **Global Problem**.

2. a flow vector x is called a feasible flow vector if it satisfies equations (2),(3),(4). Further, the feasible set defined by these equations is a convex and closed subset of $\Re^n$.

3. the optimal flow vector which is a solution to **NLP-I**, is denoted by x^*. From the ϵ-*routability* assumption (A3) we have that there exists at least one feasible solution. From (2)- (4) the feasible set is compact. Thus, from the continuity of the objective function f, an optimal solution exists.

Following [5], the partial derivative of f with respect to x_{sp} is:

$$\frac{\partial f(x)}{\partial x_{sp}} = \sum_{l \in p} f_l'(\phi_l) \tag{5}$$

Here, $f_l^{'}(\phi_l)$ will be interpreted as the first derivative length of link l, and $\partial f(x)/\partial x_{sp}$ as the first derivative length of LSP p. For **NLP-I** without capacity constraints, the following condition is necessary for a path flow

vector x^* to be optimal (cf.[6] Chapter 2).

$$x^* > 0 \quad \Longrightarrow \quad \frac{\partial f(x^*)}{\partial x_{sp'}} \geq \frac{\partial f(x^*)}{\partial x_{sp}}, \quad \forall p' \in P_s \tag{6}$$

Thus, a positive path flow is optimal only on paths with minimum first derivative length. This is a sufficient condition for optimality when the functions f_l are convex. The gradient projection algorithm is a standard technique to solve the constrained optimization problem **NLP-I** [6]. The problem is solved by iterative adjustment of x in a direction opposite to the gradient and subsequent projection onto the space of feasible solutions. Thus, for each source s, each iteration involves an adjustment of the form $x_s(k+1) = [x_s(k) - \gamma \nabla f_s(k)]^+$ where γ is a stepsize factor chosen sufficiently small, $\nabla f_s(k)$ is a vector whose (s,p)-th element is the first derivative lengths $\partial f / \partial x_{sp}(k)$ of all LSPs $p \in P_s$ at the k-th iteration and $x_s(k) = (x_{sp}(k))_{p \in P_s}$ is s's rate vector. Finally, $[x]^+$ projects its argument back to the space of the feasible solutions to **NLP-I**. The algorithm terminates when $\| x_s(k+1) - x_s(k) \| < \epsilon$ for some predefined ϵ. Thus, the projection algorithm can be carried out independently by each IE pair, $s \in S$ without explicit transmission of flow allocation information to other IE pairs in the network. In MATE [8] a distributed implementation of this algorithm is achieved by each IE pair forming an estimate of the first derivative length by probing the set of LSPs available to it.

3 Optimization Integrated Adaptive Traffic Engineering (OpIATE)

In this section, we present an idealized version of OpIATE (the "basic" scheme), which seeks to optimally allocate flow along multiple paths with non-intrusive feedback. To set the stage for this discussion, we make the following assumptions (in addition to (A1)–(A3)).

(A4) IE pairs $s \in S$ do not have apriori access to the desired flow levels $(d_{s'})$ of other IE pairs $s' \neq s$. Rather, each IE pair $s \in S$ periodically advertises its total allocation u_s so that all other IE pairs may plan accordingly.

(A5) Each path $p \in P_s,\ s \in S$ has an arc that is not shared by any other path.

(A6) There exists a solution x^* to the global problem **NLP-I** such that $\phi_l^* = \sum_{s \in S} \sum_{p \in P_s\,:\, l \in p} x_{sp}^* < \epsilon C_l$ for

each link $l \in L$. In other words, we assume that there is an optimal solution for which the total consumption of each link is strictly less than the maximum utilization level set by the network administrator.

Assumption (A4) serves to define the information pattern (feedback) that distinguishes OpIATE from other optimization-based load balancing schemes, such as MATE. Rather than collecting detailed load statistics from the core of the network OpIATE operates on edge-information, namely the total allocation of each IE pair. Assumption (A5) serves as a sufficient condition to ensure that the solution to the global problem **NLP-I** is unique (cf. Lemma 1 in Appendix A). Note that if uniqueness is clear for some other reason, say for topological reasons, then (A5) can be dropped. Assumption (A6) ensures that the solution x^* is continuous as a function of demand levels $(d_s)_{s \in S}$ in a neighborhood around the given desired flow amounts (cf. Lemma 2). Again, if this continuity can be assured for some other reason, then (A6) may be dropped.

3.1 OpIATE Algorithm - Synchronous Version

Algorithm for IE pair s: *At iteration $k = \{0, 1, 2, \dots, \}$,*

1. *IE pair s receives the current total transmission rates being achieved by the other IE pairs in the network. Let this information be summarized in the form of the vector v_s^k,*

$$v_s^k = (u_1^k, \dots, u_{s-1}^k, d_s, u_{s+1}^k, \dots, u_{\bar{S}}^k). \tag{7}$$

2. *As an intermediate step, s solves the following nonlinear program, which can be interpreted as s's local*

impression of the global problem, **NLP-I**.

$$\textbf{NLP-II(s)}: \quad \min_{x} \quad f(x) = \sum_{l} f_l(\phi_l)$$

$$\text{subject to} \quad \sum_{p \in P_i} x_{ip} = u_i^k, \quad \forall i \in S, i \neq s$$

$$\phi_l \leq \epsilon C_l, \quad \forall l \in L$$

$$\sum_{p \in P_s} x_{sp} = d_s$$

$$x_{sp} \geq 0, \quad \forall p \in P_s, s \in S$$

$$\text{where} \quad \phi_l = \sum_{s} \sum_{p \in P_s : l \in p} x_{sp}, \quad \forall l \in L.$$

Let $\tilde{x}_s^k$ denote the vector of optimal flow rates for source s onto its set of available paths P_s.

3. *Next, s computes a new transmission rate for each LSP as a weighted average of x_s^k and $\tilde{x}_s^k$:*

$$x_s^{k+1} = (1-\alpha)x_s^k + \alpha\tilde{x}_s^k, \quad \alpha \in (0,1), \tag{8}$$

where α is an "aggressiveness" parameter selected by the network administrator.

4. *Finally, s sends the value of its newly computed total transmission rate u_s^{k+1}, to all other sources in the network and repeats steps (1) through (4) at the next iteration of the algorithm where*

$$u_s^{k+1} = (1-\alpha)u_s^k + \alpha d_s \tag{9}$$

Remarks:

1. In solving **NLP-II(s)** at each iteration, IE pair $s \in S$ computes an optimal allocation of its own desired flow level (d_s) in a manner that minimizes network congestion while simultaneously respecting the total flow rates being achieved (currently) by other IE pairs in the network.

2. To accommodate slow variations in the desired flow vector d or variations in available bandwidth, each

IE pair s acts in an incremental fashion by computing its actual flow update as a convex combination of the current flow vector with the flow vector suggested by the optimal solution to **NLP-II(s)**. With $\alpha \approx 1$, flow updates in OpIATE are very close to the optimal solution of **NLP-II(s)** and all IE pairs quickly learn the desired flow levels of all other IE pairs. On the other hand, since the **NLP-II(s)** model reflects the local impression that IE pair s has about the flow requirements of all other IE pairs, the resulting global updates across all IE pairs can result in packet loss (due to conflicting impressions about the availability of bandwidth). Thus, for stable operation of the scheme, we recommend modest values of α, say $\alpha \in [0.01, 0.2]$, depending on both what kind of convergence rate is desired and the rate at which fluctuations in d are perceived. This format of rate adjustment ensures stability (by avoiding route oscillations) and results in the incremental flow update being realized almost immediately on link flows.

3. Each iteration of the synchronous OpIATE algorithm requires the solution of $\bar{S}$ **NLP-II** problems, each as complex as **NLP-I**, so the scheme is certainly very computationally expensive. In practice OpIATE would be implemented in the control plane (or above) and OpIATE updates would be computed as often as possible subject to the computational infrastructure at each ingress router. On the other hand, the motivation behind OpIATE is to provide an *adaptive* mechanism that responds to unknown and slowly varying network operating conditions. So we envision that OpIATE would be called on a relatively long time-scale, say every 2 min. Moreover, **NLP-I** and **NLP-II** can be solved efficiently by commonly available software packages such as `fmincon` in the optimization toolbox of MATLAB.

In Appendix A, we show that that the synchronous version of OpIATE converges asymptotically to the solution of the global problem, as given in the following proposition.

Proposition *Under assumptions (A1)–(A6), for any initial distribution of flow, we have*

$$x_s^k \longrightarrow x_s^*, \qquad \text{as } k \to \infty, \tag{10}$$

for all $s \in S$, where x_s^* denotes the part of the global solution x^* that relates to the IE pair s.

4 OpIATE Implementation

The previous section defined the OpIATE algorithm for a bufferless fluid system where each IE pair knows its desired flow level exactly. To evaluate the benefits of OpIATE in real networks, we have extended the basic scheme to account for packetized transmission, buffering, and the fact that desired flow levels are not precisely known and have to be estimated. We outline below some of the issues that we had to address in developing the packetized version of OpIATE.

1. OpIATE requires IE pairs be able to measure traffic rates at the ingress as well as keep track of traffic rates assigned to various LSPs. For the numerical evaluation of Section 5 we used exponential averaging to estimate the desired flow level(s) of the IE pair(s) inhabiting an ingress node. Following the terminology in [17], we denote t_i^k and l_i^k to be the arrival time and length of the k^{th} packet for IE pair i. Thus, the estimated desired flow level for an IE pair i, d_i, is updated every time a new packet is received using the equation,

$$d_i^{new} = (1 - e^{-T_i^k/K})\frac{l_i^k}{T_i^k} + e^{-T_i^k/K} d_i^{old},$$

   where $T_i^k = t_i^k - t_i^{k-1}$ and K is a constant. [17] highlights several issues involved in choosing a value of K in $e^{-T/K}$ and suggest using values between 100 and 500 ms[2].

2. OpIATE requires ingress nodes to be able to split traffic across the LSPs available for each IE pair $s \in S$. For illustration purposes, suppose that IE pair s has two available LSPs, has a desired flow $d_s = 10$ Mbps, and that OpIATE recommends sending 6 Mbps on one LSP, 2 Mbps on the other and dropping 2 Mbps (a 60-20-20 split). This split is done through randomization where, whenever a packet arrives, a uniform random number U is generated in the interval $[0, 1]$. If $U \in [0, 0.6)$ the packet is assigned to the first LSP, if $U \in [0.6, 0.8)$ it is assigned to the other LSP, finally if $U \in [0.8, 1.0]$ the packet is dropped. The advantage of randomization is that it is easier to implement than explicit metering. Of course, random splitting of TCP flows would be disruptive in practice, and a "binning" approach similar to that developed in [8] would be desirable.

3. OpIATE depends on being able to classify flows between IE pairs, and this can be done in different ways.

[2]For simulating OpIATE, the value of K was set to 100ms for all IE pairs in the network.

Some studies, for example [13], define flows based solely on IP source and destination addresses along with a timeout value. Others have a more restrictive definition that include protocol and port numbers. OpIATE is intended for traffic without specific performance requirements thus having a coarse-grained definition allows the a reduction in the number of flows being optimized. Finally, packet classification is also needed in the context of MPLS; therefore this per flow classification for OpIATE may not be an extra cost.

4. The OpIATE framework presented Section 2 does not account for cross-traffic. Cross-traffic is defined to be traffic traversing LSPs that has not been measured (and allocated) by IE pairs. In other words, IE pairs have no control over such traffic. Thus, in a network setting, the presence of continued cross-traffic would necessitate measurement and feedback from nodes constituting the core of the network to the various IE pairs sharing the network. The cross-traffic could then be incorporated into the optimization problem by incorporating a capacity reduction for the concerned links, and the convergence analysis (Refer Appendix A) for OpIATE would still hold.

5. In practice, we may also be interested in operating OpIATE in an asynchronous fashion across all IE pairs. This could be done by defining a basic update period t_0 and have IE pairs randomly choose asynchronous update periods uniformly in the interval $[t_0 - \gamma t_0, t_0 + \gamma t_0]$ where γ represents the asynchronicity parameter. Note that once the initial asynchronous update period is chosen, an IE pair could decide to stick to this initial choice or randomly choose a different update period from one iteration to the next.

5 Simulation Experiments and Results

This section describes the set of simulations conducted in the ns2 [14] packet-level discrete event simulator and is intended to compare the performance of OpIATE with MATE. Since simulation code and packet size distributions for the MATE algorithm are proprietary, it was not possible to simulate MATE. As a result, these simulations are only intended for making qualitative comparisons between the two schemes. The topology in Figure 1 is what will be henceforth referred to as the MATE topology. For simulation purposes nodes labeled 0, 1 and 2 are considered to be ingress nodes. The egress nodes for these ingress nodes are labeled 9, 10 and 11, respectively.

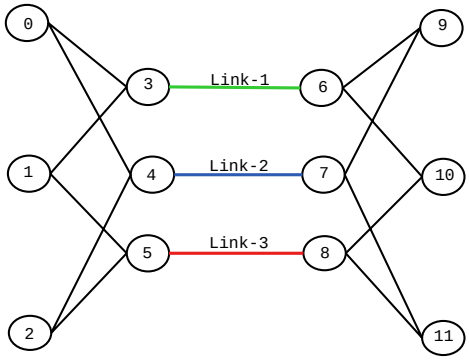


Figure 1: The MATE Topology. Each link in the topology has a capacity of 45 Mbps.

All links in the topology have a capacity of 45 Mbps. Other salient details of the original MATE experiment, for this topology, are as follows [3]. The duration of each MATE simulation is 7200 sec. There are 32 Poisson sources sending traffic into each ingress node with an average packet size[4] of 257 bytes and an average interarrival time of 3290 μs. Poisson cross-traffic enters the network at each of the three intermediate nodes 3, 4 and 5 and exits at nodes 6, 7 and 8 respectively. The average inter-arrival time for these sources is 2056 μs. The cross traffic dynamics for the duration ($[0, 7200]$ sec) of the original MATE simulation are outlined next. If we define the ratio of the average traffic (in Mbps) on a link to that link's capacity to be the *load* on that particular link then for link 1, the offered load due to cross-traffic is 0.333 in the interval $[0, 7200]$ sec. This corresponds to a total of 15 active Poisson sources. Similarly, for link 2, as a result of an increase in the number of Poisson sources from 15 to 30, the offered load due to cross-traffic is 0.333 within $[0, 3600)$ sec and 0.67 in $[3600, 7200]$ sec. Finally for link 3, the load offered by cross-traffic is 0.77 in the interval $[0, 1800)$ sec and 0.44 from $[1800, 7200]$ sec, as a result of the decrease in the number of Poisson sources from 35 to 20.

Table 1 shows the optimal load distribution required in the time intervals $[0, 1800)$ sec, $[1800, 3600)$ sec and $[3600, 7200]$ sec. Here, λ_i indicates offered load on the ith link. For instance, Table 1 indicates that for link 3 in the interval between $[1800, 3600)$ sec the aggregate load should be 0.8, meaning that the cumulative traffic from cross-traffic sources as well as that from the IE pairs 1 and 2 use up 80 $\%$ of the capacity of link 3. The initial average loading also reflects the fact that initially link 1 carries traffic from both IE pairs 0 and 1. This is due to

[3] The authors are thankful to Dr. Cheng Jin for providing this data.

[4] The original MATE experiments for this topology use an empirical packet size distribution with the average packet size being 257 bytes. In this paper, *all* packets are set to be 257 bytes in length unless otherwise specified.

link 1 being a part of the shortest path for both these IE pairs and helps explain the heavily loaded state of link 1. The load on link 2 is a result of IE pair 2 using the shortest path comprising link 2 to route its offered traffic.

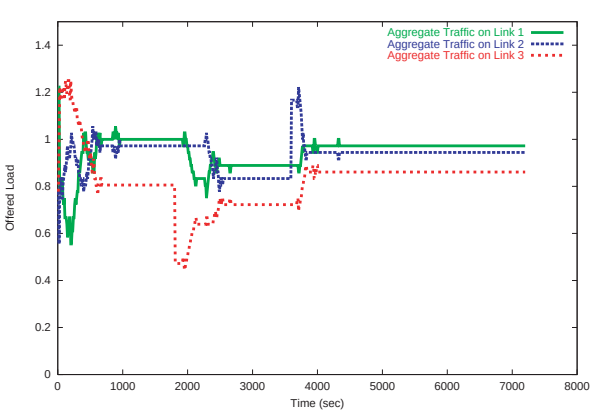
Link	Initial Load	Optimal Load Distribution		
i	λ_i	$t \in [0, 1800)$ sec	$t \in [1800, 3600)$ sec	$t \in [3600, 7200]$ sec
1	1.21	0.925	0.82	0.93
2	0.77	0.925	0.82	0.93
3	0.77	0.925	0.80	0.91

Table 1: Optimal Load Distribution for MATE experiment.

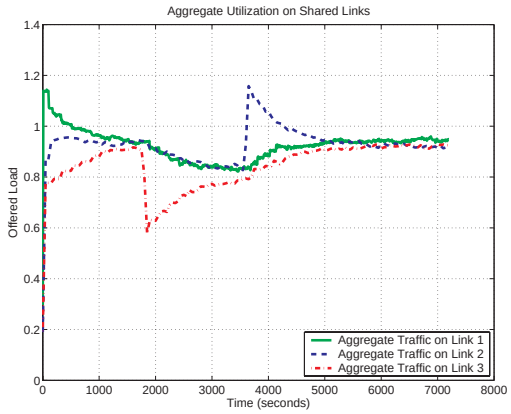
5.1 Comparison of OpIATE and MATE

Figure 2 compares the aggregate load distribution (average link utilization including cross-traffic) and packet loss rate (the ratio of the number of packets dropped within a particular time interval to that of the total number arriving within that same time interval, for a particular link) on links 1, 2 and 3 for the MATE algorithm[5] vs OPIATE. Further, the propagation delay simulated on all links was 10 ms. The MATE algorithm's initializing action [24] of splitting flow equally on the various paths available to a particular IE pair results in a high degree of packet loss for all three ingress nodes. In contrast, OpIATE smoothly varies the load (Figure 2(b)) and, as seen in Figure 2(d), only shared link 1 suffers packet loss. For $t_0 = 120$ sec, the value of the "aggressiveness" parameter $\alpha = 0.2$ and the asynchronicity parameter $\gamma = 0.5$ this loss rate is reduced to zero within 10 min of operation. This leads to the conclusion that operating OpIATE in an asynchronous mode is a viable alternative to MATE and gives comparable performance without the flip flopping of load observed for the MATE transient (Figure2(a)). In the next set of experiments, we show the operation of the OpIATE algorithm, in the MATE topology, for two different traffic scenarios. One where the network is subjected to a sharp rise (and drop) in input traffic and another where the input traffic is slowly varying. The cross-traffic dynamics in the interval [0, 7200] sec remain identical to the earlier experiment comparing OpIATE to MATE. Finally, these experiments were conducted in an asynchronous setting with the values of packet size, propagation delay and asynchronous update periods remaining the same as before.

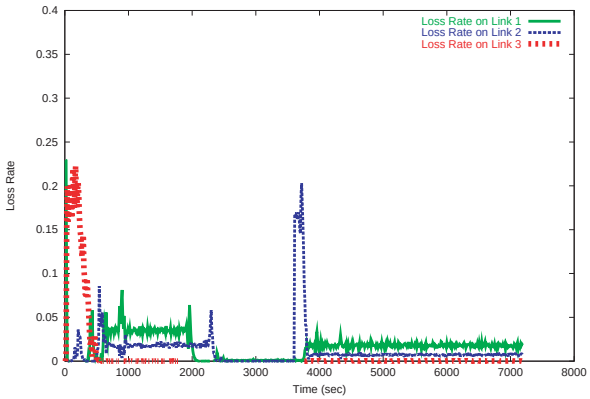
[5] Aggregate load distribution and loss profile plots for the MATE algorithm have been reproduced from [8].



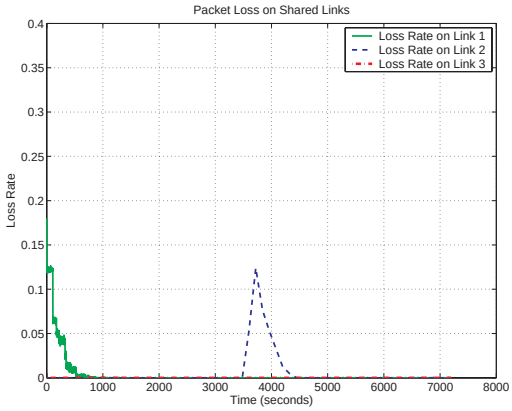
(a) MATE load distribution



(b) OpIATE load distribution



(c) MATE loss profile

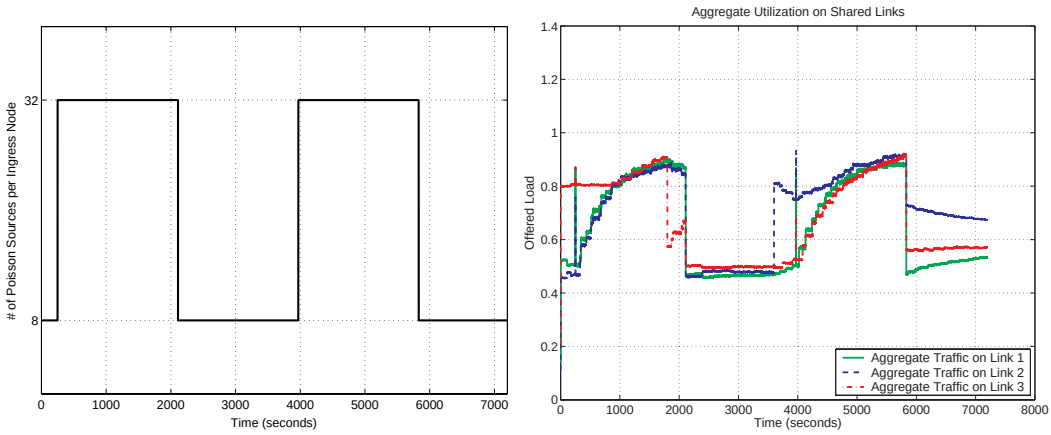


(d) OpIATE loss profile

Figure 2: Comparison of MATE vs OpIATE under Poisson Traffic

5.2 OpIATE's Impulse Response

The performance of OpIATE is tested when there is a sharp impulse, in terms of offered load, to the MATE network. The impulse hits the MATE network at $t = 250$ sec with the average Poisson arrival rate rising sharply from 5 Mbps to 20 Mbps, achieved by a sudden increase in the number of Poisson sources (from 8 to 32). The average arrival rate stays constant over the next 31 min when there is a sharp drop in traffic from 20 Mbps to 5 Mbps. The arrival rate stays at this level for the next 31 min when there is another sharp impulse rise in traffic followed by a period of stability and an impulse drop in traffic. This traffic scenario is illustrated in Figure 3(a).



(a) Input Signal

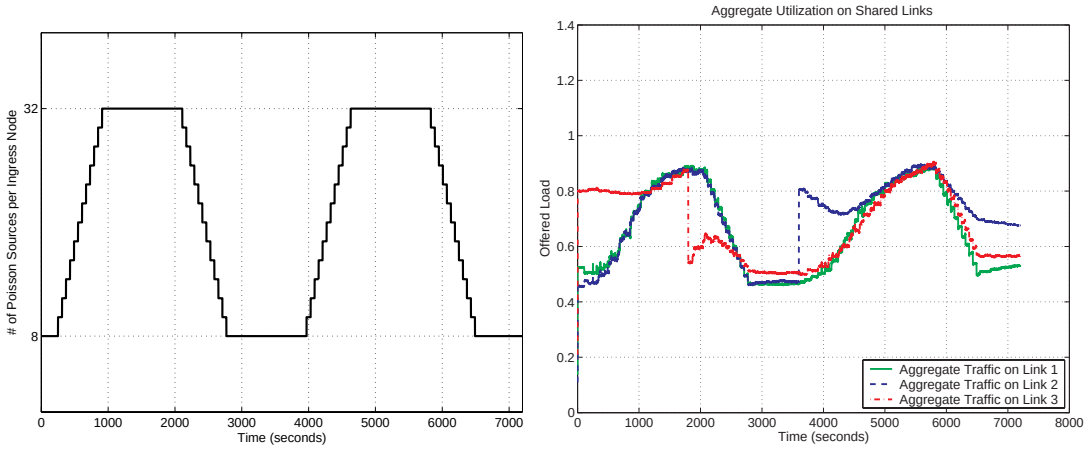
(b) Load Distribution Response

Figure 3: Impulse traffic regime. MATE topology asynchronous updates $t_0 = 120$ sec, $\alpha = 0.2$

The cross-traffic dynamics, asynchronous update periods for $t_0 = 120$ sec, $\alpha = 0.2$ and $\gamma = 0.5$ are as before. As can be seen in Figure 3(b) OpIATE is successful in distributing load optimally with no packet loss on shared links, in an impulse traffic generation scenario. Also of note is the smoothening effect of the incremental flow adjustment method on the sharp impulse in traffic load. This is of course accompanied by packet loss at the ingress nodes which has not been plotted here. Finally, we point out the rise (and drop) of load on the shared links consistent with the description of the cross-traffic dynamics at 3600 sec (for link 2) and 1800 sec (for link 3).

5.3 OpIATE's Response to Slowly Varying Load

The performance of OpIATE is tested when there is a gradual ramp up and ramp down of offered load to the MATE network. As can be seen in Figure 4(a), the ramp up starts at $t = 250$ sec with 2 sources being added every minute to an ingress node until a total of 32 sources are reached at $t = 910$ sec. Thus the average arrival rate is increased gradually from 5 Mbps to 20 Mbps. Poisson traffic is sustained at the 20Mbps average arrival rate for a period of 20 min. At the end of this period there is a gradual ramp down in traffic where 2 sources are taken off-line every minute until a total of 8 sources are left transmitting at $t = 2770$ sec. Again, there is a 20 min period where the average Poisson arrival rate is sustained at 5 Mbps. After which there is another ramp up- ramp



(a) Input Signal

(b) Load Distribution Response

Figure 4: Uniform varying traffic regime. MATE topology asynchronous updates $t_0 = 120$ sec, $\alpha = 0.2$

down period. The cross-traffic dynamics, asynchronous update periods for $t_0 = 120$sec and $\gamma = 0.5$ remain as before. With a gradual variation in offered load, one sees a gradual adaptation in the load distribution in Figure 4(b) with no accompanying packet loss on the shared links.

6 Conclusions

To summarize this paper, we have developed a decentralized adaptive traffic engineering algorithm, OpIATE, based on optimization principles, which is intended for an Intra-autonomous system carrying traffic without specific performance requirements. The asymptotic convergence of the basic scheme (for a fluid model with synchronous updates) to the optimal solution of the global optimal routing problem has been established. By design, OpIATE does not suffer from the drawbacks of conventional shortest-path routing and requires minimal changes to an existing network. We have benchmarked OpIATE's performance against another dynamic traffic engineering algorithm, MATE, in packet-level simulations. In OpIATE, edge-based feedback regarding total IE flow allocation leads to a shared global perspective, missing in other schemes, resulting in more effective traffic engineering. Finally, OpIATE highlights the potential of using non-intrusive feedback in the design of effective traffic engineering schemes.

6.1 Future Work

The packet-level OpIATE algorithm developed in Section 3 represents an initial prototype, and the related simulation results in Section 5 serve as a proof of concept. We expect that the design would have to evolve before it is operationally deployable. In the following, we list a number of issues that should be addressed in future work.

First, the OpIATE algorithm relies on the application of an incremental amount of the optimal solution to **NLP-II(s)**. In practice, for each IE pair $s \in S$ solving **NLP-II** problems for large networks might be too complex to solve frequently with great precision. On the other hand the **NLP-II** problems solved from one instant to the next generally do not vary wildly. Thus, it should be possible to use the solution to the **NLP-II** problem in the most recent iteration to "hot-start" the optimization process for the next iteration. Alternatively, one could address the complexity issue by not solving the **NLP-II** model all the way to optimality. Rather, an optimization routine could be run for a fixed (small) number of iterations and the resulting intermediate solution applied. Another related issue would be to consider the possibility of dynamically adjusting α in response to perceived performance.

The effect of OpIATE on TCP traffic has not been shown. OpIATE makes no special provisions with regard to packet reordering and, even though traffic engineering literature (see [3] for instance) suggests relegating such functionality to the destination nodes, TCP is especially sensitive to paths shifting on short time scales. In this regard, a hybrid packet forwarding scheme could be developed in which a default LSP would be chosen for TCP traffic, and one would modify the switching ratios for UDP traffic so that the resulting traffic allocations on the LSPs is consistent with the optimal ratios suggested by the solution to **NLP-II**. Clearly the effectiveness of this scheme would depend on the relative percentage of TCP vs UDP traffic constituting the offered load.

An explicit assumption in OpIATE is that there is some degree of slack capacity available in the network and that desired flow levels do not stretch these capacity constraints. In a situation where there is insufficient capacity to handle the desired flow levels, other factors, like fairness to flows, need to be considered. Future work in this area could address this situation by either extending the basic OpIATE algorithm or by providing for a seamless transition from OpIATE to algorithm(s)(for example [20]) that do indeed take care of these issues. In a similar vein, the issue of loss of feedback information needs to be addressed.

Finally, more comprehensive experimentation with OpIATE would be essential before practical deployment. It would be instructive to explore the effects of other kinds of input traffic like the Discrete Autoregressive (DAR)

traffic and Pareto traffic as well as the effect of large propagation delays on the OpIATE scheme. We expect that delay in the feedback loop should have a negligible impact as long as the length of the synchronous/asynchronous update cycle is larger than the maximum time it takes a flow update to reach an IE pair. Also, it is expected that in the future, testing of the OpIATE algorithm will be carried out in larger, more complex topologies, with more than one IE pair at an ingress node.

References

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus,"Requirements for Traffic Engineering Over MPLS," *RFC 2702,* September 1999.

[2] D. Awduche, "MPLS and Traffic Engineering in IP Networks," *IEEE Communications Magazine,* 37(12), December 1999.

[3] D. Awduche, A. Chiu, I. Widjaja and X. Xiao,"Overview and Principles of Internet Traffic Engineering," *RFC 3272,* September 1999.

[4] D.P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, 1989.

[5] D.P. Bertsekas and R. Gallager *Data Networks*, Prentice Hall, Second Edition 1992.

[6] D.P. Bertsekas, *Nonlinear Programming,* Athena Scientific, Belmont, MA 1995.

[7] D.G. Cantor, "Optimal Routing in a Packet Switched Computer Network,"*IEEE Transactions on Computers,* Vol. C-23(10), October-1974.

[8] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," *In Infocom 2001*.

[9] B. Fortz and M. Thorup,"Internet traffic engineering by optimizing OSPF weights,"*Proceedings of Infocom 2000,* Tel-Aviv, Israel, March 2000.

[10] R.G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Transactions on Communications,* 25(1):73-85, January 1977.

[11] S.H. Low and D.E. Lapsley,“Optimization Flow Control-I: Basic Algorithm and Convergence,” *IEEE/ACM Transactions on Networking.* 7(6):861-875, December 1999.

[12] J. Moy,“OSPF Version 2,” *RFC 2178,* July 1997.

[13] P. Newman, T. Lyon and G. Minshall, “Flow labeled IP: A Connectionless Approach to ATM,” *Proceedings IEEE INFOCOM 96,* March 1996.

[14] UCB/LBNL/VINT network simulator - ns (version 2.1b8a).

[15] A. Odlyzko, “The Economics of the Internet: Utility, utilization, pricing, and Quality of Service,” *Technical Report, AT & T Research Lab,* 1998.

[16] K. Sinha, *OpIATE: Optimization Integrated Adaptive Traffic Engineering,* Masters Thesis of the Department of Systems and Information Engineering, November 2002 (forthcoming).

[17] I. Stoica, S. Shenker and H. Zhang, “Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks,” *SIGCOMM-98,* Vancouver, B.C.

[18] J.N. Tsitsiklis and D.P. Bertsekas, “Distributed Asynchronous Optimal Routing in Data Networks,” *IEEE/ACM Transactions on Automatic Control,* 31(4):325-332, April 1986.

[19] Z. Wang and J. Crowcroft, “Shortest Path First with Emergency Exits,” *Proceedings of ACM SIGCOMM,* 166-176, 1990.

[20] J. Wang, S. D. Patek, H. Wang and J. Liebeherr, “Traffic Engineering with AIMD in MPLS networks,” *Proceedings of PfHSN 2002*, Berlin, Germany. April 2002.

[21] S. D. Patek, R. Venkateswaran and J. Liebeherr, “Simple Alternate Routing for Differentiated Services Networks,”*Computer Networks,* 37(4):447-466, November 2001.

[22] M. A. Rodrigues and K. G. Ramakrishnan, “Optimal routing in shortest-path networks,” *ITS’94,* Brazil.

[23] E. Rosen, A. Viswanathan and R. Callon, “Multiprotocol Label Switching Architecture,” *RFC 3031,* January 2001.

[24] I. Widjaja and A. Elwalid, “Internet Draft for MATE: MPLS Adaptive Traffic Engineering,” *draft-widjaja-mpls-mate-00.txt*.

A Analysis of the Basic Scheme

A.1 Properties of the Solution to NLP-I

We begin by establishing a sufficient condition for unique solutions to **NLP-I**.

Lemma 1 *If the desired flow levels are ϵ-routable and if there exists a link $l_p \in L$ which is unique to p for each path $p \in P_s$ for all IE pairs $s \in S$ (cf. Assumptions (A3) and (A5)), then there exists a unique solution x^* to* **NLP-I**.

Proof of Lemma 1: The existence of an optimal solution is ensured by ϵ-routability (which gives rise to a non-empty compact feasible set) along with the continuity of the objective function. Uniqueness stems from the fact, under assumption (A5), that the objective function is strictly convex. To see this, note that the objective function can be expressed as

$$f(x) = \sum_{s \in S} \sum_{p \in P_s} \frac{x_{sp}}{(C_{l_p} - x_{sp})} + \sum_{\text{all other links, } l} \frac{\phi_l}{C_l - \phi_l}, \tag{11}$$

where $\phi_l = \sum_{s \in S} \sum_{p \in P_s \,:\, l \in p} x_{sp}$. The first term above is strictly convex as a function of the aggregate flow vector x, whereas as the second term is convex. ■

We now establish a technical property associated with the (unique) optimal solution x^*.

Lemma 2 *If x^* is a solution to $\mathbf{NLP-I}$ which is such that $\phi_l^* = \sum_{s \in S} \sum_{p \in P_s \,:\, l \in p} x_{sp}^* < \epsilon C_l$ (cf. Assumption (A6)), then the gradients of all of the active constraints of* **NLP-I** *are linearly independent.*

Proof of Lemma 2: By assumption, the only active constraints are those that account desired flow levels d_s for each source $s \in S$:

$$h_s(x) \equiv \sum_{p \in P_s} x_{sp} - d_s = 0, \qquad \forall\, s \in S.$$

Thus, $\nabla h_s(x)$ is a vector of zeros and ones, where the ones correspond precisely to the elements of x that describe flow on the paths $p \in P_s$ available to the IE-pair s. ■

A.2 Convergence of the Basic Scheme

In this subsection, we establish the following general result about the basic (fluid, synchronous) implementation of OpIATE.

Proposition 1 *Under assumptions (A1)–(A6), for any initial distribution of flow, we have*

$$x_s^k \longrightarrow x_s^*, \qquad \text{as } k \to \infty, \tag{12}$$

for all $s \in S$, where x_s^* denotes the part of the global solution x^* that relates to the IE pair s.

We begin the analysis by introducing (for convenience) more concise statements of the optimization models **NLP-I** and **NLP-II(s)**. First, note that the conservation of flow constraints of Equation (2), can be rewritten in the form $h(x) = 0$ where, $h = (h_s)_{s \in S}$ and $h_s(x) = \sum_{p \in P_s} x_{sp} - d_s$ for all $s \in S$. Similarly, the capacity constraints of Equation (3), can be rewritten in the form $g(x) \leq 0$ where, $g = (g_1, \ldots, g_{\bar{L}})$ and $g_l(x) = \phi_l - \epsilon C_l$ for all $l \in L$. Thus, **NLP-I** can then be rewritten as:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) \leq 0. \end{aligned} \tag{13}$$

In a similar fashion, **NLP-II(s)** can be rewritten in concise form as:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & h(x) = r_s^k \\ & g(x) \leq 0, \end{aligned} \tag{14}$$

where

$$r_s^k = v_s^k - d, \tag{15}$$

where d is the vector of desired flow levels for all IE pairs and v_s^k, defined in Equation (7), is the approximation to d that s maintains based on feedback information from all other IE pairs. **NLP-II(s)** will sometimes be referred to as the modified version of the global problem associated with IE pair s. We can rewrite $\tilde{x}_s^k$, the solution to **NLP-II(s)** at iteration k as,

$$\tilde{x}_s^k(r_s^k) = \arg_s \min_{\substack{h(x)=r_s^k \\ g(x)\leq 0}} f(x), \tag{16}$$

where "$\arg_s \min$" refers to the minimizing argument relative to the IE pair s, i.e. the part of the (unique) optimal solution that describes the optimal allocation of s-flow onto its LSPs.

The following lemma implies that under the update rule of Equation (9) the total allocation v_s^k of each IE pair s converges to the desired flow level d_s.

Lemma 3 *Under assumptions (A1)–(A6), for any initial allocation of flows,*

$$u_s^k \longrightarrow d_s, \qquad \text{as } k \to \infty, \tag{17}$$

for all $s \in S$.

Proof: From the update rule of Equation (9), it is clear that the total allocation of IE pair s changes in each iteration k according to

$$u_s^{k+1} = (1-\alpha)u_s^k + \alpha d_s, \qquad \forall\ s \in S, \quad k = 0, 1, \ldots.$$

Thus, $u_s^{k+1} - d_s = (1-\alpha)(u_s^k - d_s)$, which implies that

$$|u_s^{k+1} - d_s| \leq (1-\alpha)|u_s^k - d_s| \qquad \forall s \in S.$$

Consequently, Equation (9) is a pseudocontraction with modulus $(1-\alpha) \in (0,1)$ and fixed point d_s. As a result (cf. [4], Chapter 3.1, Proposition 1.2), there are no other fixed points, and $u_s^k \to d_s$ as $k \to \infty$ for all $s \in S$. ■

The following corollary shows that the equality constraint of **NLP-II(s)** converges to that of **NLP-I** as $k \to \infty$ (compare Equation (13) to Equations (14) and (15)).

Corollary 1 *Under assumptions (A1)–(A6), for any initial allocation of flows,*

$$r_s^k \longrightarrow 0, \qquad \textit{as } k \to \infty, \tag{18}$$

for all $s \in S$.

Proof: This follows from the definition of r_s^k in Equation (15). Note that the i^{th} element of r_s^k can be expressed as

$$r_{s,i}^k = \begin{cases} u_i^k - d_i & \text{if } i \neq s, i \in \{1, \ldots, S\}, \\ 0 & \text{if } i = s. \end{cases} \tag{19}$$

Thus, Lemma 3 implies that $r_s^k \to 0$ as $k \to \infty$. ■

The next lemma shows that as $k \to \infty$ (i.e. as $r_s^k \to 0$), the solution to the **NLP-II(s)** optimization model for each IE pair $s \in S$ converges to the optimal solution to the global problem **NLP-I**.

Lemma 4 *Under assumptions (A1)–(A6), for any initial allocation of flows,*

$$\tilde{x}_s^k \longrightarrow x^*, \qquad \textit{as } k \to \infty, \tag{20}$$

for all $s \in S$.

Proof: Let $z(r_s^k)$ denote the unique optimal solution to **NLP-II(s)**, where the notation acknowledges that the solution only depends explicitly upon the feedback information available to s in the form of r_s^k. Thus, $\lim_{k\to\infty} \tilde{x}_s^k = \lim_{k\to\infty} z(r_s^k)$. Lemma 2 implies that the solution x^* of the global problem **NLP-I** is regular using the terminology of [6]. This, along with the fact that the objective function and constraints are twice continuously differentiable over the set of feasible solutions, implies that **NLP-I** exhibits the following continuity property (cf. Propositions 3.2.3 and 3.3.3 in [6]):

$$\lim_{k\to\infty} z(r_s^k) = z(\lim_{k\to\infty} r_s^k).$$

Thus,

$$\begin{aligned} \lim_{k\to\infty} \tilde{x}_s^k &= z(\lim_{k\to\infty} r_s^k) \\ &= z(0) \\ &= x_s^*, \end{aligned}$$

where the second equality follows from Corollary 1 and the third inequality follows from the definition of **NLP-I.** ■

Proof of Proposition 1 Let $x^k = (x_1^k, x_2^k, \ldots x_{\bar{S}}^k)$ denote the vector of flow allocation decisions of all IE pairs $s \in S$ (made synchronously in the k^{th} stage of the basic OpIATE scheme). Note that, according to the update rule of Equation (9), x^{k+1} can be expressed as

$$x^{k+1} = (1-\alpha)x^k + \alpha\tilde{x}^k, \tag{21}$$

where $\tilde{x}^k = (\tilde{x}_1^k, \tilde{x}_2^k, \ldots \tilde{x}_{\bar{S}}^k)$ is the vector of **NLP-II(s)** solutions associated with the feedback information available to all IE pairs $s \in S$. We will show that the convergence of $\tilde{x}_s^k$ to x_s^* (cf. Lemma 4) implies that $x^k \to x^*$ as $k \to \infty$.

To begin, note that by subtracting x^* from both sides of Equation (21) we get

$$x^{k+1} - x^* = (1-\alpha)(x^k - x^*) + \alpha(\tilde{x}^k - x^*),$$

which implies that

$$\|x^{k+1} - x^*\| \leq (1-\alpha)\|(x^k - x^*)\| + \alpha\|(\tilde{x}^k - x^*)\| \tag{22}$$

Now let us define

$$y^k = \|x^k - x^*\|, \tag{23}$$

$$w^k = \|\tilde{x}^k - x^*\|, \tag{24}$$

for $k = 0, 1, \ldots$. Note that the inequality of Equation (22) can be expressed alternatively as

$$y^{k+1} \leq \quad (1-\alpha)y^k + \alpha w^k. \tag{25}$$

Note also that the proof of Proposition 1 is complete once we show that $y^k \to 0$ as $k \to \infty$.

To this end, let us define the sequence $\{\xi^k\}_{k=0}^{\infty}$ such that,

$$\xi^{k+1} = (1-\alpha)\xi^k + \alpha w^k, \quad k = 0, 1, 2, \ldots \tag{26}$$

$$\xi^0 = \|x^0 - x^*\| \tag{27}$$

The remainder of the proof is to show (1) that $0 \leq y^k \leq \xi^k$ for all $k = 0, 1, \ldots$ and (2) that $\xi^k \to 0$ as $k \to \infty$.

Claim 1

$$0 \leq y^k \leq \xi^k, \qquad \forall k = 0, 1, 2, \ldots. \tag{28}$$

Proof: We proceed by induction. Note that the statement of the proposition is true for $k = 0$, by definition. Now suppose that the relation holds true for some k, i.e., $\xi^k \geq y^k \geq 0$. Then, from the definitions of y^{k+1} and ξ^{k+1}, along with the inequality of Equation (22), we have

$$\begin{aligned} 0 \quad \leq \quad y^{k+1} \quad &\leq \quad (1-\alpha)y^k + \alpha w^k, \\ &\leq \quad (1-\alpha)\xi^k + \alpha w^k, \\ &\leq \quad \xi^{k+1}, \end{aligned}$$

where the second inequality makes use of the induction hypothesis. ■

Claim 2 *Consider a stable first-order difference equation $z_{k+1} = \beta z_k + q_k$, where $0 < \beta < 1$ and $q_k \geq 0$ for all $k = 0, 1, \ldots$. Given that $q_k \to 0$ as $k \to \infty$, then $z_k \to 0$ as $k \to \infty$. In particular, ξ^k (as defined in Equation* (26)*) converges to zero.*

Proof: Let us consider the general result first. Note that by recursive substitution of the dynamical equation

$$\begin{aligned} z_{k+1} &= \beta(\beta z_{k-1} + q_{k-1}), \\ &= \beta^2 z_{k-1} + \beta q_{k-1} + q_k, \\ &\vdots \\ &= \beta^{k-s+1}[\beta^s z_0 + \beta^{s-1} q_0 + \beta^{s-2} q_1 + \ldots + q_{s-1}] + \sum_{i=0}^{k-s} \beta^i q_{s-i}. \end{aligned}$$

Define $\overline{M_0} = \max\{(\sup_k q_k), z_0\}$ and $\overline{M_s} = \sup_{k \geq s} q_k$. Note that $\overline{M_s} \to 0$ as $s \to \infty$. Therefore,

$$\begin{aligned} z_{k+1} &\leq \beta^{k-s+1}[\beta^s z_0 + \beta^{s-1} q_0 + \beta^{s-2} q_1 + \ldots + q_{s-1}] + \overline{M_s} \sum_{i=0}^{k-s} \beta^i, \\ &\leq \beta^{k-s+1}[\beta^s z_0 + \beta^{s-1} q_0 + \beta^{s-2} q_1 + \ldots + q_{s-1}] + \overline{M_s} \frac{1-\beta^{k-s+1}}{1-\beta}, \\ &\leq \beta^{k-s+1} \overline{M_0}[\beta^s + \beta^{s-1} + \ldots + 1] + \overline{M_s} \frac{1-\beta^{k-s+1}}{1-\beta}, \\ &\leq \beta^{k-s+1} \overline{M_0} \frac{1-\beta^{s+1}}{1-\beta} + \overline{M_s} \frac{1-\beta^{k-s+1}}{1-\beta}, \\ &\leq \beta^{k-s+1} \frac{\overline{M_0}}{1-\beta} + \frac{\overline{M_s}}{1-\beta}. \end{aligned}$$

Note that by choosing s large enough we can make the second term above arbitrarily small (since $\overline{M_s} \to 0$ as $s \to \infty$). Then, by choosing k large enough (greater than s), we can make the first term above arbitrarily small, and we see that $z_k \to 0$ as $k \to \infty$.

Finally, observe the recursion (Equation (26)) that defines ξ^k, is a stable first-order linear difference equation, as above, with a non-negative forcing function that converges to zero (thanks to Lemma 4). ■