Decentralized Optimal Traffic Engineering in Connectionless Networks

Bernardo A. Movsichoff, *Student Member, IEEE,* Constantino M. Lagoa, *Member, IEEE,* and Hao Che

Abstract— **This paper addresses the problem of optimal traffic engineering in a connectionless autonomous system. Based on Nonlinear Control Theory, the approach taken in this paper provides a family of optimal adaptation laws. These laws enable each node in the network to independently distribute traffic among any given set of next-hops in an optimal way, as measured by a given global utility function of a general form. This optimal traffic distribution is achieved with minimum information exchange between neighboring nodes. Furthermore, this approach not only allows for optimal multiple forwarding paths but also enables multiple Classes of Service; e.g., classes of service defined in the DiffServ architecture. Moreover, the proposed decentralized control scheme enables optimal traffic redistribution in the case of link failures. Suboptimal control laws are also presented in an effort to reduce the computational burden imposed on the nodes of the network. Finally, an implementation of these laws with currently available technology is discussed.**

Index Terms— **Traffic Engineering, Sliding Mode Control, Mathematical Programming, Optimization**

I. Introduction

In current autonomous systems, Traffic Engineering (TE) has been an important means to optimize utilization of network resources and to avoid congestion resulting from the adoption of the shortest path routing paradigm. The most elementary form of TE is known as Equal Cost Multi-Path (ECMP), which is widely deployed in current routers. ECMP allows traffic to be evenly distributed among multiple next-hops lying in the equal-cost shortest paths, hence reducing the chance of congestion. On the other hand, more sophisticated TE strategies have mainly relied on a connection-oriented model, such as Multi-Protocol Label Switching (MPLS). The basic idea of these strategies is to use tunneled paths, not necessarily those found by a shortest path routing protocol, to forward packets. By steering traffic away from the congested shortest paths and redirecting it to the established tunnels, higher network resource utilization can be achieved. The mathematical foundation for this type of TE has been established to allow optimal distributed load balancing and rate adaptation in the presence of multiple paths and multiple CoSs; e.g., [7], [8], [9].

Manuscript received . . .
. . .
. . . . This work was supported by the National Science Foundation under grants ECS-9984260 and ANI-0125653.

Bernardo A. Movsichoff and Constantino M. Lagoa are with the Department of Electrical Engineering, The Pennsylvania State University. Email: bernardo@gandalf.ee.psu.edu; lagoa@engr.ee.psu.edu
Hao Che is with the Department of Computer Science and Engineering, University of Texas at Arlington. Email: hche@cse.uta.edu

Digital Object Identifier xx.xxxx/JSAC.2005.xxxxxx

Recently, a different line of research [3], [11], [14] argued that sophisticated TE does not necessarily have to rely on a connection-oriented model. In particular, Y. Wang, et al. [14] showed that by properly assigning link weight values based on a given traffic demand matrix, Non-equal distribution of traffic among Equal Cost Multiple shortest Paths (NECMP) can lead to a globally optimal TE solution, equivalent to the one obtained with a connection oriented solution; e.g., the MPLS based TE. Sridharam, et al. [11] further demonstrated how this idea can be implemented in an Open Shortest Path First (OSPF) domain.

The potential benefit of using the above approach as compared with the connection-oriented approach can be significant. Using this approach to allow optimal TE, there is no need for introducing new protocol stacks (such as MPLS) into the connectionless IP networks and, hence, no egress rooted $O(N)$ trees or fully meshed $O(N^2)$ connections among edge nodes need to be provisioned and maintained, where N is the number of edge nodes. This simplification significantly reduces the software upgrading, operational and management costs.

Although encouraging, the above solution has three critical weaknesses. First, the approach taken is intrinsically a centralized one. The optimal link weight assignment is based on the availability of a global traffic demand matrix. This has two important implications: (1) the approach can only respond to slow traffic demand changes [3]; (2) the approach cannot deal with link/node failures [11]. Second, this solution fails if the traffic demand matrix is not available. Third, this solution only works for the best-effort traffic. To provide optimal TE for multiple Classes of Service (CoSs), a connection-oriented approach (e.g., application of [7], [8] and [9] to a DiffServ enabled MPLS network) is still the only viable solution. The ability to provide CoS-based TE is critical simply because a key for revenue generation is the ability to provide value-added services.

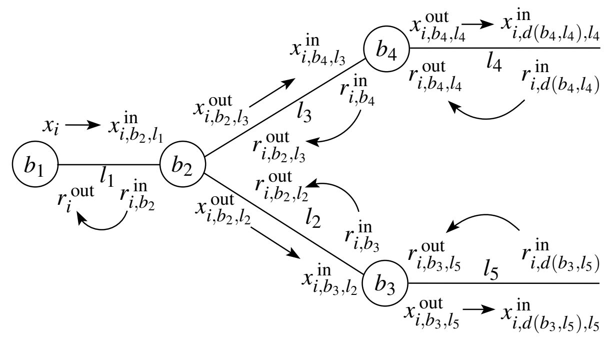
In this paper, a theoretical framework to solve the above mentioned issues is proposed. Specifically, the proposed solution allows CoS and per hop based optimal load balancing and rate adaptation among any given set of multiple paths and, in particular, among a set of equal cost multiple shortest paths. A large family of distributed control laws is obtained, which converges to the maximum of a general concave utility function. Due to the distributed nature of control, this approach can optimally redistribute traffic among multiple shortest paths and react to small time-scale traffic demand variations and link/node failures. The proposed algorithms allow for globally optimal, multiple CoS based TE, that closely mimics the optimal TE offered by a connection-oriented approach where the traffic demand matrix is known. Even without the knowledge of the traffic demand matrix, the proposed approach allows globally optimal, multiple CoS based NECMP under any given link weight assignment. Moreover, unlike a connection-oriented approach for which an additional set of mechanisms need to be implemented to provide high availability (such as fast re-routing and path protection for MPLS label switched paths) the intrinsic fast re-routing capability of the proposed approach provides high availability to a connectionless IP network by design, eliminating the need to design another set of protocols. The advantage of using local control for link failure recovery over path protection has also been shown in [10], where an algorithm to determine the set of available next hops is presented. However, the proposed method for load balancing among the nodes in these sets is either ECMP or round-robin and no algorithm for optimal redistribution of the traffic is provided.

This paper also addresses a challenging implementation issue; i.e., how to enable NECMP. Most of the existing routers only support ECMP, not NECMP. A general perception is that directly modifying data path functions to allow NECMP is costly. This leads to the proposal of an alternative approach presented in [11], which allows effectively NECMP without changing the existing ECMP load balancing paradigm. However, this approach involves nontrivial control plane function changes and substantial configuration overhead. In this paper, it is shown that, to allow NECMP, directly modifying the data plane functions can actually be much easier than modifying the control plane functions. This is based on the observation that more and more currently available routers employ fully programmable network processors in their network interface cards to perform data path functions. Furthermore, it is shown how NECMP can be programmed in a network processor by adding just a small number of instructions and two possible implementations are discussed.

The remainder of the paper is as follows: Section II presents notation and assumptions used throughout this paper. Section III provides a precise statement of the problem at hand, while Section IV presents the proposed optimal algorithms to tackle it. Section V provides an alternative algorithm that is simpler to implement. Finally, in Section VI, implementation of NECMP using currently available hardware is discussed, while Sections VII and VIII present simulation examples and final remarks respectively.

II. Preliminaries

Throughout this paper, traffic flows are assumed to be described by a fluid flow model, where the only resource taken into account is link bandwidth. In what follows, call and flow will be used interchangeably. Now, consider a network where several calls of different types are present; calls of different types will be understood as flows with different pairs of edge nodes (or equivalently source/destination nodes) and/or different CoSs. The concept of "flow/call" is defined to facilitate the design of the control laws. As it is seen in Section V-B, while call level control is required at the



b_1
b_2
b_3
b_4
l_1
l_2
l_3
l_4
l_5
$x_i \longrightarrow x^{\text{in}}_{i,b_2,l_1}$
r^{out}_i
r^{in}_{i,b_2}
$x^{\text{out}}_{i,b_2,l_3}$
$x^{\text{in}}_{i,b_4,l_3}$
r^{in}_{i,b_4}
$r^{\text{out}}_{i,b_2,l_3}$
$x^{\text{out}}_{i,b_4,l_4} \longrightarrow x^{\text{in}}_{i,d(b_4,l_4),l_4}$
$r^{\text{out}}_{i,b_4,l_4}$
$r^{\text{in}}_{i,d(b_4,l_4)}$
$r^{\text{out}}_{i,b_2,l_2}$
$x^{\text{out}}_{i,b_2,l_2}$
r^{in}_{i,b_3}
$x^{\text{in}}_{i,b_3,l_2}$
$r^{\text{out}}_{i,b_3,l_5}$
$r^{\text{in}}_{i,d(b_3,l_5)}$
$x^{\text{out}}_{i,b_3,l_5} \longrightarrow x^{\text{in}}_{i,d(b_3,l_5),l_5}$

Fig. 1. Notation

domain edge nodes, in the core nodes the control is performed solely based on the per hop information and no per call state information is required.

Assume that several paths connecting each destination node with its source node are available. The objective is to allocate resources to maximize the network's utility as measured by a given utility function, assuming that each node uses only per hop information for packet forwarding.

More precisely, consider a computer network consisting of a set of nodes $\mathscr{B}$ and a set of interconnecting links $\mathscr{L}$ and let c_l be the capacity of link $l \in \mathscr{L}$. Moreover, let $\mathscr{L}_b$ denote the set of links connected to node $b \in \mathscr{B}$ and $\mathscr{L}_{b,i} \subseteq \mathscr{L}_b$ denote the set of outgoing links for calls of type i. Also, let n be the number of types of calls utilizing the network and let $\mathscr{I}_b$ be the set of types of calls visiting node b. Given calls of type i, the quantity x_i denotes the data rate allocated to that type and vector

$$\mathbf{x} \doteq [x_1, x_2, \ldots, x_n]^T \in \mathbf{R}^n$$

contains the data rates allocated to all the calls sharing the network. Furthermore, given node $b \in \mathscr{B}$, $x^{\text{in}}_{i,b,l}$ denotes the data rate of calls of type i arriving to node b through link $l \in \mathscr{L}_b$. In a similar fashion, $x^{\text{out}}_{i,b,l}$ denotes the data rate of calls of type i departing node b through link $l \in \mathscr{L}_b$. Also, the vectors $\mathbf{x}_{\text{in}}$ and $\mathbf{x}_{\text{out}}$ stand for the vectors containing all such arriving and departing data rates respectively. This notation is exemplified in Fig. 1 for the simple case of a single ingress (source) node; i.e., node b_1, together with some additional quantities that are later used in the control laws.

Now, given any node $b \in \mathscr{B}$ and any link $l \in \mathscr{L}_b$, let $d(b,l)$ denote the downstream node; i.e., the node connected to node b through link l. Note that given any two nodes and their (lossless) connecting link, it holds that

$$x^{\text{out}}_{i,b,l} = x^{\text{in}}_{i,d(b,l),l}.$$

Also, note that $x_i = x^{\text{in}}_{i,\tilde{b},l}$ for some $\tilde{b} \in B$ and some $l \in \mathscr{L}_{\tilde{b}}$. For example in Fig. 1, the set of links connected to node b_2 is $\mathscr{L}_{b_2} = \{l_1, l_2, l_3\}$ while $b_4 = d(b_2, l_3)$ is the downstream node of node b_2 through link l_3. Furthermore, $x^{\text{in}}_{i,b_4,l_3} = x^{\text{out}}_{i,b_2,l_3}$; i.e.,

$$x^{\text{out}}_{i,b_2,l_3} = x^{\text{in}}_{i,d(b_2,l_3),l_3}.$$

Now, given calls of type i, the corresponding data rate x_i is determined at source i and multiple paths are available at each node of the network. More precisely, each node partitions incoming traffic into the available links by striving to conserve the flow for each call type (i.e., aims at no losses) and to avoid link congestion. In Fig. 1, for example, node b_2 tries to satisfy

$$x^{\text{in}}_{i,b_2,l_1} = x^{\text{out}}_{i,b_2,l_2} + x^{\text{out}}_{i,b_2,l_3}.$$

Satisfaction of this type of constraints is indicated by the quantity $r^{\text{in}}_{i,b}$, computed at each node $b \in \mathscr{B}$. In this example, r^{in}_{i,b_2} will be assigned a value of 1 if the incoming data rate is greater than the allocated outgoing rate and -1 if it is less.

Moreover, given the lossless property of the links, each node also pursues satisfaction of flow conservation constraints at each one of the downstream nodes; e.g., in Fig. 1 node b_2 tries to enforce the following flow constraints

$$x^{\text{in}}_{i,b_3,l_2} = x^{\text{out}}_{i,b_3,l_5} \quad \text{and} \quad x^{\text{in}}_{i,b_4,l_3} = x^{\text{out}}_{i,b_4,l_4}.$$ [1]

To enforce these constraints, nodes b_3 and b_4 use the computed quantities r^{in}_{i,b_3} and r^{in}_{i,b_4} as explained above. This is the same information needed at b_2 to satisfy the very same constraints. For this reason, the quantities $r^{\text{out}}_{i,b,l}$, for each $b \in \mathscr{B}$, and each $l \in \mathscr{L}_{b,i}$, are equal to the quantities $r^{\text{in}}_{i,d(b,l)}$ fed back from the downstream nodes. In the present example this amounts to take

$$r^{\text{out}}_{i,b_2,l_2} = r^{\text{in}}_{i,b_3} \quad \text{and} \quad r^{\text{out}}_{i,b_2,l_3} = r^{\text{in}}_{i,b_4}.$$

Finally, let cg_l denotes the congestion information about link $l \in \mathscr{L}$; i.e.,

$$cg_l \doteq \begin{cases} 0 & \text{if link } l \text{ is congested} \\ 1 & \text{otherwise} \end{cases}$$

III. Problem Statement

The objective of this paper is to solve the problem of maximizing utility functions of the form

$$U(\mathbf{x}) \doteq \sum_{i=i}^{n} f_i(x_i)$$

subject to link capacity constraints, CoS requirements and flow conservation constraints, both through nodes and links. The functions $f_i(\cdot)$ are assumed to be differentiable concave functions increasing in their arguments; i.e., with non-negative partial derivatives. The reason for using concave utility functions is twofold: First, from a mathematical point of view, any local minimum of the utility function is a global minimum which implies that one will not get "trapped" at a non-optimal traffic allocation. Second, a concave utility function will result in a "fair" traffic allocation in the sense that one will not starve any of the calls. In fact, one will get a larger "reward" by increasing the data rate of calls with lower data rates.

Two different but similar approaches are proposed. The first one admits an optimal solution at the cost of significant complexity, while the second one is much simpler but, in general, only quasi-optimal.

Calls of type $i = 1,2,\ldots,s$ are assumed, without loss of generality, to be of the assured forwarding CoS category (AF); calls of this CoS category are assigned a target rate that has to be achieved in an average sense; i.e., $x_i = \Lambda_i$ for a given Λ_i. On the other hand, calls of type $i = s+1, s+2, \ldots, n$ are assumed to be of the best effort CoS category (BE); i.e., calls of this CoS category utilize whatever resources are left. Other more general CoS categories with lower and/or upper bounds on the desired data rates can be addressed (see, for example, [8]), but due to space constraints only AF and BE categories are considered here.

Here, AF refers to traffic flows that have a long term average target rate similar to the ones defined in the DiffServ architecture. However, as it will shortly become apparent, it is of a more general nature since the control on the target rate for AF is performed only at the edge nodes and is transparent to the core nodes. This salient feature of the approach presented in this paper allows for the implementation of the control laws along with the DiffServ compliant CoSs. Namely, while edge nodes implement DiffServ Code Point (DSCP) marking, rate control and other DiffServ traffic conditioning mechanisms, core routers simply map packets with different DSCPs to different output queues, where some queue management algorithm and CoS-based scheduling discipline are implemented.

More precisely, given the assumptions and requirements above, the problem of optimal resource allocation can be formulated as the following optimization problem

$$\max_{\mathbf{x},\, \mathbf{x}_{\text{in}},\, \mathbf{x}_{\text{out}}} U(\mathbf{x})$$

subject to link capacity constraints[2]

$$\sum_{i=1}^{n} x^{\text{in}}_{i,b,l} + x^{\text{out}}_{i,b,l} - c_l \le 0 \qquad \forall l \in \mathscr{L}_b \text{ and each } b \in \mathscr{B}$$

flow conservation constraints at each node

$$\sum_{l \in \mathscr{L}_b} x^{\text{out}}_{i,b,l} - \sum_{l \in \mathscr{L}_b} x^{\text{in}}_{i,b,l} = 0 \qquad \forall b \in \mathscr{B}, i \in \mathscr{I}_b$$

flow conservation constraints for each link[3]

$$x^{\text{out}}_{i,b,l} - x^{\text{in}}_{i,d(b,l),l} = 0 \qquad \forall l \in \mathscr{L}; l \in \mathscr{L}_b$$

the AF requirements

$$x_i - \Lambda_i = 0 \qquad i = 1,2,\ldots,s$$

and non-negativity of all the data rates, $x_i \ge 0$, $x^{\text{in}}_{i,b,l} \ge 0$ and $x^{\text{out}}_{i,b,l} \ge 0$ for all i, all $l \in \mathscr{L}$ and all $b \in \mathscr{B}$. The optimization problem above constitutes a convex problem and can be easily solved if global information is available. However, global information on fast timescale events, as required in the above formulation, is not generally available. This paper aims at providing decentralized adaptation laws that converge to the solution of this problem.

IV. A First Family of Decentralized Laws

This section presents a family of adaptation laws that converge to the optimal of the convex problem posed above. Also, an alternative set of simpler control laws is presented that is optimal for a large subset of operating conditions.

[1]Both b_3 and b_4 might have more than just one outgoing link for call of type i. For simplicity this is not the case in Fig. 1.

[2]Note that the formulation presented in this paper allows for the existence of bidirectional links.

[3]Since lossless links are assumed, this constraints will be automatically satisfied.

A. Data Rate Adaptation

Consider the following adaptation law to be used at each source node $b_{si} \in \mathscr{B}$: For $i = 1, 2, \ldots, s$; i.e., AF calls, let

$$\dot{x}_i = z_i\big(t, cg_l(t), r_i^{\text{out}}(t)\big) \times \times \left[\frac{\partial f_i}{\partial x_i}\bigg|_{\mathbf{x}_i} - \alpha cg_l(t) - \beta_i r_i(t) - \beta_i^{\text{out}} r_i^{\text{out}}(t) \right],$$

where

$$r_i(t) = \begin{cases} 1 & \text{if } x_i > \Lambda_i \\ -1 & \text{if } x_i < \Lambda_i \end{cases}$$

and

$$r_i^{\text{out}}(t) = \begin{cases} 1 & \text{if } x_i > \sum\limits_{l \in \mathscr{L}_{d(b_{si},l)}} x_{i,d(b_{si},l),l}^{\text{out}} \\ -1 & \text{if } x_i < \sum\limits_{l \in \mathscr{L}_{d(b_{si},l)}} x_{i,d(b_{si},l),l}^{\text{out}} \end{cases},$$

where $d(b_{si}, l)$ is the only node connected to the corresponding source node b_{si}; i.e., $\mathscr{L}_{b_{si}} = \{l\}$. For $i = s+1, s+2, \ldots, n$; i.e., BE calls, let

$$\dot{x}_i = z_i\big(t, cg_l(t), r_i^{\text{out}}(t)\big) \left[\frac{\partial f_i}{\partial x_i}\bigg|_{\mathbf{x}_i} - \alpha cg_l(t) - \beta_i^{\text{out}} r_i^{\text{out}}(t) \right],$$

where r_i^{out} is defined as above. Also, for each node $b \in \mathscr{B}$, all $i \in \mathscr{I}_b$, and all $l \in \mathscr{L}_b$ let

$$\dot{x}_{i,b,l}^{\text{out}} = z_{i,b}\big(t, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big) \times \times \left[\beta_{i,b}^{\text{in}} r_{i,b}^{\text{in}}(t) - \beta_{i,b,l}^{\text{out}} r_{i,b,l}^{\text{out}}(t) - \alpha cg_l(t) \right],$$

where

$$r_{i,b}^{\text{in}}(t) = \begin{cases} 1 & \text{if } \sum\limits_{l \in \mathscr{L}_b} x_{i,b,l}^{\text{in}} > \sum\limits_{l \in \mathscr{L}_b} x_{i,b,l}^{\text{out}} \\ -1 & \text{if } \sum\limits_{l \in \mathscr{L}_b} x_{i,b,l}^{\text{in}} < \sum\limits_{l \in \mathscr{L}_b} x_{i,b,l}^{\text{out}} \end{cases} \tag{1}$$

and

$$r_{i,b,l}^{\text{out}}(t) = \begin{cases} 1 & \text{if } \sum\limits_{l \in \mathscr{L}_{d(b,l)}} x_{i,d(b,l),l}^{\text{in}} > \sum\limits_{l \in \mathscr{L}_{d(b,l)}} x_{i,d(b,l),l}^{\text{out}} \\ -1 & \text{if } \sum\limits_{l \in \mathscr{L}_{d(b,l)}} x_{i,d(b,l),l}^{\text{in}} < \sum\limits_{l \in \mathscr{L}_{d(b,l)}} x_{i,d(b,l),l}^{\text{out}} \end{cases}, \tag{2}$$

where $d(b, l)$ is the node connected to node b through link l. That is, $r_{i,b,l}^{\text{out}}$ consists of the quantity $r_{i,d(b,l)}^{\text{in}}$ fed back from $d(b, l)$ to b, as shown in Fig. 1. Recall also that $x_{i,b,l}^{\text{out}} = x_{i,d(b,l),l}^{\text{in}}$ as explained in Section II. Note that this control law is CoS agnostic and the CoS dependent control is performed only at the ingress nodes.

Given the adaptation laws above, the data rates are then forced to be greater than or equal to zero. More precisely, if any of the data rates above is zero, then the corresponding derivative is taken as the maximum between zero and the expression given above.

Note in the adaptation laws above, the terms $r_{i,b}^{\text{in}}$ are used to enforce flow conservation at node b, while the quantities r_i^{out} and $r_{i,b,l}^{\text{out}}$ are used to enforce flow conservation at the downstream nodes $d(b, l)$.

All the functions z and the scalars α and β are design parameters that have to be determined to achieve convergence and provide an acceptable transient behavior. The following theorem establishes conditions for these choices and provides the main result of this paper; i.e., the control laws above are optimal.

Theorem 1: Let $\zeta > 0$ and $\gamma > 0$ be given (arbitrarily small) constants. Also, let $z_i\big(t, cg_l(t), r_i^{\text{out}}(t)\big)$ and $z_{i,b}\big(t, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big)$ be scalar functions continuous in t for any functions $cg_l(t) \in [0, 1]$, $r_{i,b}^{\text{in}}(t) \in [-1, 1]$ and $r_i^{\text{out}}(t) \in [-1, 1]$, satisfying

$$z_i\big(t, cg_l(t), r_i^{\text{out}}(t)\big) > \zeta$$
$$z_{i,b}\big(t, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big) > \zeta$$

for all $t > 0$.

Furthermore, let $\beta_i > 0$, $\beta = \beta_i^{\text{out}} = \beta_{i,b,l}^{\text{out}} = \beta_{i,b}^{\text{in}} > 0$ and $\alpha > 0$, be constants satisfying the following inequalities

$$\alpha > \left| \frac{\partial f_i}{\partial x_i} \right|_{x_i=0} \qquad \beta_i > \left| \frac{\partial f_i}{\partial x_i} \right|_{x_i=0} \qquad \beta > \left| \frac{\partial f_i}{\partial x_i} \right|_{x_i=0}$$

for all $l \in \mathscr{L}$, all $b \in \mathscr{B}$ and all $i = 1, 2, \ldots, n$.

Then the control laws presented above converge to the maximum of the utility function

$$U(\mathbf{x}) \doteq \sum_{i=1}^{n} f_i(x_i),$$

subject to the network's link capacity constraints, CoS requirements, flow conservation constraints and non-negativity of all the data rates.

Proof: See Appendix I ■

V. Percentage Adaptation

The control laws provided in Section IV require each node to measure both incoming and outgoing data rates for each visiting type of calls. In light of this issue, this section provides an alternative set of control laws that require much less information than the previous ones. Instead of adapting the data rate being sent through each link, the adaptation is performed on the percentage of incoming traffic that has to be allocated to each outgoing link.

Define $p_{i,b,l}$ as the percentage of incoming traffic of type i at node b that is routed along each available outgoing link l; i.e.,

$$x_{i,b,l}^{\text{out}}(t) = p_{i,b,l}(t) \sum_{\substack{\tilde{l} \in \mathscr{L}_b \\ \tilde{l} \notin \mathscr{L}_{b,i}}} x_{i,b,\tilde{l}}^{\text{in}}(t) \qquad b \in \mathscr{B};\ l \in \mathscr{L}_{b,i},$$

Now, let the percentage of incoming traffic routed along link $l \in \mathscr{L}_{b,i}$ be governed by the adaptation law

$$\dot{p}_{i,b,l} = z_{i,b}\big(t, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big) \times \times \Big(\dot{x}_{i,b,l}^{\text{out}} \sum_{\tilde{l} \in \mathscr{L}_{b,i};\, \tilde{l} \neq l} p_{i,b,\tilde{l}} - p_{i,b,l} \sum_{\tilde{l} \in \mathscr{L}_{b,i};\, \tilde{l} \neq l} \dot{x}_{i,b,\tilde{l}}^{\text{out}} \Big),$$

for all $b \in \mathscr{B}$, $l \in \mathscr{L}_{b,i}$, where

$$\dot{x}_{i,b,l}^{\text{out}} = \left[\beta_{i,b}^{\text{in}} r_{i,b}^{\text{in}}(t) - \beta_{i,b,l}^{\text{out}} r_{i,b,l}^{\text{out}}(t) - \alpha cg_l(t) \right],$$

cg_l is the bottleneck information as defined in Section II and $r_{i,b}^{\text{in}}$, $r_{i,b,l}^{\text{out}}$ are given by (1) and (2) respectively. These laws are derived directly from the control laws for the data rates presented in Section IV. Note that the laws above do not require the measurement of data rates if the values of $r_{i,b}^{\text{in}}$ and $r_{i,b,l}^{\text{out}}$ are available by some other means.

The following proposition shows that under some conditions these laws are indeed optimal.

Proposition 2: Assume that all data rates are always strictly positive; i.e., there exists $\varepsilon > 0$ such that $x_i(t) > \varepsilon$, $x_{i,b,l}^{\text{in}}(t) > \varepsilon$ and $x_{i,b,l}^{\text{out}}(t) > \varepsilon$ for all i, $t \geq 0$, all $l \in \mathscr{L}$ and all $b \in \mathscr{B}$.

Then the percentage adaptation laws above converge to this optimal solution.

Proof: See Appendix II ■

A. Practical Computation of $r_{i,b}^{\text{in}}$ and $r_{i,b,l}^{\text{out}}$.

Although optimal, the laws above require access to data rate information for the computation of $r_{i,b}^{\text{in}}$ and $r_{i,b,l}^{\text{out}}$. Hence, an alternative (empirical) way to compute this information is presented.

Note that when implementing percentage adaptation, the aggregate incoming data rate is always larger than or equal to the aggregate outgoing rate; i.e.,

$$\sum_{l \in \mathscr{L}_b} x_{i,b,l}^{\text{in}} \geq \sum_{l \in \mathscr{L}_{b,i}} x_{i,b,l}^{\text{out}}. \tag{3}$$

Hence, it is not necessary for $r_{i,b}^{\text{in}}$ in (1) to assume the value -1; i.e., in this case only $r_{i,b}^{\text{in}} = 0, 1$ is needed. Note also, that $r_{i,b}^{\text{in}} = 1$ when there is some type of congestion, either in the connected links or further downstream. That is, (3) is a strict inequality. This prompts the following computation for $r_{i,b}^{\text{in}}$

$$r_{i,b}^{\text{in}} = \wedge_{l \in \mathscr{L}_{b,i}} \{ \vee \{ cg_l, r_{i,b,l}^{\text{out}} \} \},$$

where the symbols $\wedge$ and $\vee$ stand for logical AND and OR operations respectively and the quantity $r_{i,b,l}^{\text{out}}$ is received from the downstream node $d(b,l)$; i.e.,

$$r_{i,b,l}^{\text{out}} = r_{i,d(b,l)}^{\text{in}}$$

for each link $l \in \mathscr{L}_{b,i}$. With this expression, $r_{i,b}^{\text{in}}$ will only be 1 if all the available paths are congested and 0 otherwise. Intuitively, for each available downstream node $d(b,l)$, the OR operation yields 1 (congestion) if either link l is congested or there is congestion further downstream $d(b,l)$. The AND operation on the other hand, takes all downstream nodes $d(b,l)$, for $l \in \mathscr{L}_{b,i}$, and indicates congestion only if node b sees congestion in all downstream paths.

In the general case, if any optimal data rate is zero, the percentages obtained by these means might exceed one or become negative, although they will always add up to one. This issue is addressed by means of a normalization procedure that is explained in Section VII along with the problem of discretization of the continuous-time adaptation laws. These optimal adaptation laws, together with the empirical expression for the computation of $r_{i,b}^{\text{in}}$ and $r_{i,b,l}^{\text{out}}$ lead to a tractable control law that will approximately mimic the behavior of the optimal ones.

B. Routing as a Function of the Destination Node

By carefully looking at the way of computing $r_{i,b}^{\text{in}}$ and $r_{i,b,l}^{\text{out}}$ presented in the previous section, it can be seen that the computational burden at each of the nodes can be further reduced.

Indeed, consider a node b, and two types of calls i and j arriving at b which have the same destination node and share the same resources. Now, the formulas proposed in the previous section imply that

$$r_{i,b}^{\text{in}} = r_{j,b}^{\text{in}} \quad \text{and} \quad r_{i,b,l}^{\text{out}} = r_{j,b,l}^{\text{out}}.$$

Hence, if the initial conditions are the same then

$$p_{i,b,l}(t) = p_{j,b,l}(t)$$

and, hence, there is no need to independently adapt the percentages for these two call types.

Therefore, if several call types have the same destination address and share the same downstream resources, it is not necessary to independently adapt the percentages. The same percentages can then be used for all of them. More precisely, given a node b let the set of next hops for calls with destination D, $\mathscr{L}_b^D$, replace the set of next hops per type $\mathscr{L}_{b,i}$. Similarly, let per destination percentages $p_{b,l}^D$ take the place of per type percentages $p_{i,b,l}$. Then, node b needs only to adapt per destination percentages $p_{b,l}^D$, $l \in \mathscr{L}_b^D$; i.e., per type information *no longer* needs to be maintained. This implies that although the number of control laws is N^2, where N is the number of edge nodes, the distinct control laws a core router has to run is N. This scaling property allows the current solution to scale to very large autonomous systems. As it shall be seen in the following section, this property will also allow efficient implementation of the associated data plane functions.

C. Robustness With Respect to Failures

A salient feature of the laws proposed is that, once implemented, the resulting network will be robust with respect to link/node failures. In other words, after a small modification discussed below, these laws will automatically reroute traffic away from the nodes/links that have failed. The decentralized nature of the laws allows for traffic rerouting to be done by the nodes adjacent to the failure and without any change of the control law parameters.

Indeed, this can be accomplished by the following procedure: Upon detection of a link or node failure in an adjacent node or a connected link, each node performs an update of the set $\mathscr{L}_b^D$; i.e., it updates the set of available next-hops for calls with destination D. Once the update is performed, same adaptation laws are used with the new routing table. Given the decentralized nature of the laws, it can be seen that they are optimal for the "new" network configuration and will provide the desired traffic allocation; i.e., traffic is rerouted away from the failed components and a new optimal steady state allocation will be achieved.

VI. Implementing the Adaptive Routing Algorithm

As explained in the previous section, the practical implementation of the adaptation laws involves an adaptation law running at each core router for each egress node. The control law dynamically assigns a percentage $p^D_{b,l}$ of traffic of calls with destination D to be routed to the equal-cost next-hop $d(b,l)$. Since the existing routing protocol software already enables the binding between equal-cost shortest paths and the corresponding next-hops in the forwarding table, no change to the existing control plane functions is required, except for the addition of the software to implement the percentage adaptation laws, a mechanism to exchange congestion information with the neighboring routers, and a mechanism to hook up the software for the percentage adaptation laws with any link/node failure detection mechanism that is available in the router. Therefore, the remaining key issue is how to enable NECMP in the data plane.

Upon each update, $p^D_{b,l}$ is downloaded from the control plane to the data plane to facilitate NECMP. This can be easily done by sending an applet from the control plane to the network processor in the network interface card. Two possible schemes for implementing NECMP are discussed. Depending on the actual approach in use, the network processor caches $p^D_{b,l}$ in an off-chip dual port memory or a Ternary CAM (TCAM) coprocessor. In what follows, a brief review of the existing approaches to implement ECMP is presented, followed by the two proposed approaches to implement NECMP.

As explained in [4], there are two types of ECMP algorithms by design; i.e., disruptive and non-disruptive. For a non-disruptive algorithm, if no change to the set of next-hops occurs, the path a flow takes remains the same. On the other hand, for a disruptive algorithm, even if no change to the set of next-hops occurs, the path a flow takes may still change over time. An example of disruptive algorithms is to send packets to different equal-cost next-hops in a round robin fashion. Obviously, using disruptive algorithms for ECMP can result in out-of-order packet forwarding which is undesirable. Non-disruptive algorithms include hash-threshold and modulo-N algorithms (see [4]), and highest random weight algorithm (see [12]). Among these three algorithms, only hash-threshold algorithm generally applies to both ECMP and NECMP. Hence, in the sequel only this algorithm is considered.

The hash-threshold algorithm works as follows: The router first selects a key by performing a hash (e.g., CRC16) over the packet header fields that identify a flow; e.g., IP source and destination address pair. The M next-hops $d(b,l)$, $l \in \mathscr{L}^D_b$, are assigned non-overlapping regions in the key space, with the sum of the regions covering the entire key space. Here, M denotes the cardinality of the set $\mathscr{L}^D_b \subseteq \mathscr{L}_b$ and $\mathscr{L}^D_b$ denotes the set of links connecting node b with the next-hops $d(b,l)$ available for calls with destination D. The router uses the key to determine which region and thus which next-hop to use. Assuming the output of the hash function is uniformly distributed, the size of region corresponding to the next-hop $d(b,l)$ divided by the size of the key space is then equal to $p^D_{b,l}$, $l \in \mathscr{L}^D_b$. As a special case, for ECMP, all the regions have the same size, which is one M-th of the size of the key space.

Programming a network processor for hash key generation is common in support of both layer 2 switching and layer 3 address to layer 2 address mapping. For example, a 48-bit destination MAC address to an output port mapping requires that a hash key be generated from the 48-bit destination MAC address. A RISC-core (i.e., a micro-engine in a network processor, such as INTEL IXP series) needs about five instructions to generate such a hash key. Moreover, the newest generation network processors generally offer an on-chip unit which provides for CRC operations for popular industry standard polynomial values, such as, CRC8, CRC16 and CRC32. This further reduces the number of instructions to just one. Therefore, the hash key generation should not be a concern.

Now the only question left is how to map a hash key to a region and then to an equal-cost next-hop. To this end, two approaches are proposed. The first approach involves using an off-chip SRAM or DRAM and allocating a memory block with address space equal to the hash key space size, with each memory address location containing a next-hop value (more precisely, an output port value) corresponding to the region into which the memory address falls. With this solution, a direct hash key indexing will return the equal-cost next-hop, resulting in deterministic, single memory access performance.

As an example of the above solution consider the following: Suppose CRC8 is used to generate a 8-bit hash key from the IP source and destination addresses. This allows 256 different values in the key space. Consequently, 256 memory entries need to be allocated for call types destined to each egress node. Further assume there are $N = 256$ edge nodes in the routing domain (a reasonably large number given that an OSPF domain generally contains no more than a thousand nodes). So a total number of $256 \times 256 = 65536$ memory entries need to be allocated. Now if the memory width is 32 bits, which is large enough to hold an output port value, the total memory space required is 0.26MB, which is very small. A legitimate concern is that upon an $p^D_{b,l}$ update, theoretically, all 65536 entries may need to be updated in the very worst-case. This will have implication on the lookup performance. So for this solution, a dual-port memory is preferred. With a dual-port memory, database update and lookup can be performed simultaneously, although with a small chance of incorrect return of the next-hop values. Another concern is that the key space size directly affects the granularity of control. Assume there are five next-hops on average, each region then has $256/5 = 50$ levels.

If the performance is highly sensitive to the granularity of control, the following approach is preferred. Here, a TCAM coprocessor is employed to do the hash key to next-hop or output port mapping. The regions are stored in the TCAM and the corresponding next-hop values are stored in an associated memory. A region matched by a hash key will result in the corresponding output port values in the associated memory to be returned to the network processor. This approach is possible since most of today's network processors are fully integrated with TCAM coprocessors for fast packet classification. Since each TCAM memory cell can take on three states; i.e., 0, 1, and X (i.e., "don't care"), a region can be expressed in TCAM memory using just several memory slots (for details on how a range/region is expressed in a TCAM, please refer to [1]). For example, in a 16-bit key space, region $[256, 512]$ can be expressed using just two TCAM entries; i.e.,

```
0000  0001  XXXX  XXXX
0000  0010  0000  0000,
```

and region $[6000, 6063]$ can be expressed in just three TCAM entries; i.e.,

```
0001  0111  0111  XXXX
0001  0111  100X  XXXX
0001  0111  1010  XXXX.
```

Now, assume five TCAM entries are required on average to express a region, and that each TCAM memory width is 64 bits. Also assume the other parameters are the same as the previous example. Then the total TCAM memory required is

$$5(\text{TCAM entries}) \times 5(\text{regions}) \times \\ \times 8(\text{bytes of TCAM width}) \times 256(\text{edge nodes}) = 0.4\,\text{Mb}.$$

This generally constitutes only a small fraction of the total TCAM memory, given that the maximum TCAM memory size for today's TCAM coprocessors has reached 18 Mb. In general, a TCAM coprocessor can only be configured to support a dozen or so number of tables. Hence, it is impractical to allocate 256 tables in the TCAM coprocessor for all the egress nodes. A solution to this is to allocate a single TCAM table for all the egress nodes. To distinguish regions for different egress nodes, 8 bits out of a 64 bits TCAM table entry can be used to encode/identify egress nodes. This allows up to 56-bit hash key size and hence highly fine granular control. Deterministic, single clock cycle lookup performance is guaranteed. Moreover, since all the TCAM coprocessors have a CPU interface, the database update can be performed in parallel with the lookup process. Based on [15], database update algorithms can be developed such that the update process poses zero negative impact on the region matching process.

For both external dual-port memory and TCAM based solutions, each memory access or per next-hop lookup takes only two or three instructions to pass a search key composed of a hash key and an encoded egress node number to the memory and read the returned output port value. Note that the egress node number can be obtained as part of the IP forwarding table lookup, meaning that each prefix in the forwarding table is mapped to an egress node number, in addition to a next-hop. In summary, the NECMP costs about four to ten instructions. Given that the total instruction budget for a network processor supporting multi-gigabit line rate is on the order of 10^2 to 10^3, the added overhead is quite small. Therefore, it can be concluded that the modification of the data plane functions to enable NECMP is a viable solution. The required effort to do so is much smaller than modifying the control plane functions as proposed in [11]. Moreover, the present solution is more straightforward.

VII. Simulation Examples

In this section, some simulations exemplifying the behavior of the proposed algorithms are presented. In particular, it is shown, that in the presence of multiple CoSs, the algorithms succeed at distributing traffic in such a way as to maximize a given utility function. These simulations also test the behavior under discretization of the control laws, delays in the propagation of congestion information and link failures.

A. Discretization and Adaptive Oscillation Reduction

In a real network, the adaptation laws provided in this paper can not be utilized as they were presented. Instead, a discrete-time version has to be implemented. Following the work in [8], [9] this is accomplished through the following forward rule approximations

$$x_i^d\left[(k+1)t_d\right] = x_i^d[kt_d] + t_d\dot{x}_i(kt_d)$$
$$x_{i,b,l}^{d^{\text{out}}}\left[(k+1)t_d\right] = x_{i,b,l}^{d^{\text{out}}}[kt_d] + t_d\dot{x}_{i,b,l}^{\text{out}}(kt_d)$$

where $k \in \mathbf{Z}^{0+}$ and t_d is the integration step.

The following proposition establishes under what conditions this approach will lead to a successful realization of the discrete-time version of the adaptation laws.

Proposition 3: Let $\mathbf{x}(t)$ and $\mathbf{x}_{\text{out}}(t)$ be the trajectories obtained using the control laws in Section IV and let $\mathbf{x}^d(t)$ and $\mathbf{x}_{\text{out}}^d(t)$ be the corresponding discrete-time trajectories obtained using the discretization algorithm above.

Given any time interval $[t_0, t_1]$ assume that the vector $\mathbf{X}^T(t) \doteq \left[\mathbf{x}^T(t) \quad \mathbf{x}_{\text{out}}^T(t)\right]$ is bounded in infinity norm for each t in this interval. Then, given any constant $\varepsilon > 0$, there exists $\delta > 0$ such that, if $t_d z_i\big(t, cg_l(t), r_i^{\text{out}}(t)\big) < \delta$ and $t_d z_{i,b}\big(t, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big) < \delta$, for all $t > 0$,

$$\left\|\mathbf{X}(t) - \mathbf{X}^d(t)\right\| < \varepsilon \quad \forall t \in [t_0, t_1],$$

where $\mathbf{X}^d(t)$ is the discrete-time counterpart of $\mathbf{X}(t)$.

Proof: Direct application of result 2 in Filippov ([2], chapter 2, page 95). ■

That is, provided that the values of the discretization step t_d and/or the functions $z_i(\cdot)$ and $z_{i,b}(\cdot)$ are chosen to be sufficiently small, the discretization above will provide a family of discrete-time control laws that will closely resemble their continuous-time counterpart.

Now, based on this result, the discretization for the percentage adaptation laws is performed in a similar way; i.e.,

$$\hat{p}_{i,b,l}^d\left[(k+1)t_d\right] = \hat{p}_{i,b,l}^d[kt_d] + t_d\frac{d\hat{p}_{i,b,l}(kt_d)}{dt}.$$

Moreover, the following normalization forces all the percentages to add up to one and to lie in $[0, 1]$

$$\hat{p}_{i,b,l} = \max\left\{\min\{\hat{p}_{i,b,l}^d, 1\}, 0\right\} \quad ; \quad \hat{p}_{i,b,l} = \frac{\hat{p}_{i,b,l}}{\sum\limits_{l \in \mathscr{L}_b} \hat{p}_{i,b,l}^d}.$$

The above discretization and the existence of delays in the propagation of congestion information perturb the behavior of the system away from the ideal one. Due to this fact the adaptation reduction scheme presented in [8] is used to mitigate this phenomenon.

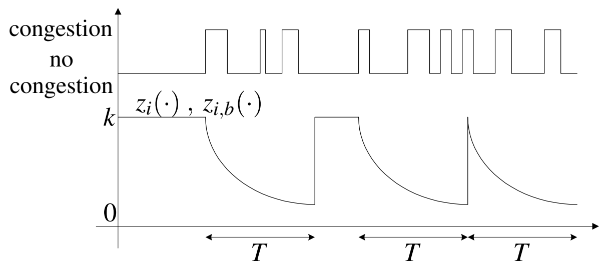


Fig. 2. Example of a scaling function $z_i(\cdot)$

TABLE I
ROUTING DECISIONS BY DESTINATION NODE

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
e_2	b_2,b_7	e_2	—	—	—	—	b_2,b_8	b_2
e_4	b_2,b_7	b_7,b_8	b_4	e_4	—	—	b_8	b_3,b_4
e_5	b_2,b_7	b_7,b_8	—	—	e_5	—	b_5	b_5,b_7
e_6	b_7	b_1,b_7,b_8	b_4,b_8	b_8	b_7	e_6	b_6	b_5,b_7

More precisely, let $T > 0$ and $k > 0$ be given, such that

$$z_i\big(0, cg_l(t), r_i^{\text{out}}(t)\big) = z_{i,b}\big(0, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big) = k.$$

Then if at time $t = t_0$ congestion is detected on link l for some call i let

$$z_i\big(0, cg_l(t), r_i^{\text{out}}(t)\big) = z_{i,b}\big(0, cg_l(t), r_{i,b}^{\text{in}}(t), r_i^{\text{out}}(t)\big) = \omega(t - t_0),$$

where $\omega\colon [0,T] \to [\zeta, k]$ is some decreasing function. This procedure is repeated again at $t \geq t_0 + T$. Figure 2 depicts the desired behavior. The trade-off, however, is that convergence time will be increased. Applying this procedure enables one to mitigate oscillations while maintaining the ability to respond to sudden changes in the network; e.g., link failures.

B. Simulation Setup

The simulations presented in this section use the network topology shown in Fig. 3. The delays, link capacities as well as source-destination pairs are indicated therein. This is the same network used in previous papers (e.g., [8], [9]) and was originally used in [6].

Multiple paths are allowed between the $n = 8$ pairs of source and destination nodes (S_i/D_i) considered in these simulations, but as mentioned before, each node only uses a per hop knowledge of the paths available. In particular, as far as the sending nodes are concerned there is only one data rate, x_i, to be determined in order to maximize the network's utility. Furthermore, these routing decisions are made solely based on the final destination of the incoming calls, without regard to their source, as shown in Table I. Note that the routing decisions in the table do not include all possible paths towards the destination node. For example, b_7 routes traffic with destination e_2 through nodes b_2 and b_8 but it does not use node b_5 to get to b_8. In other words, only a subset of all possible paths is used in this example and the ones chosen are not necessarily the shortest ones. The choice in these examples was made in order to highlight the features of the control laws, specially robustness with respect to link failures.

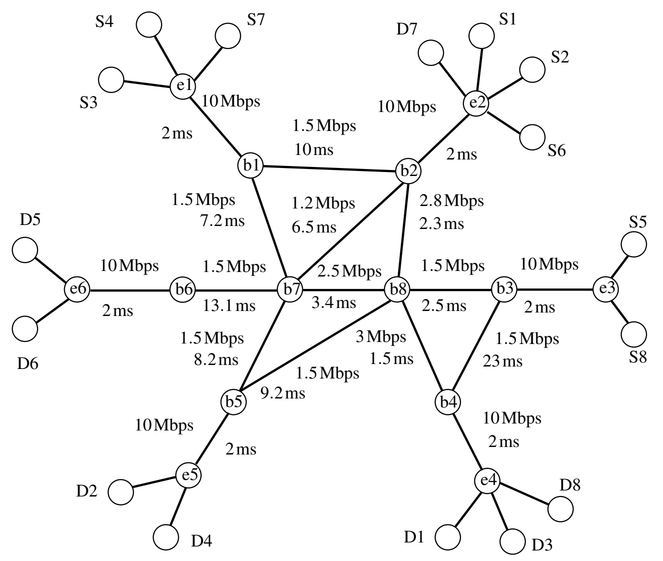


Fig. 3. Topology of the network

The objective throughout these simulations is to maximize the utility function $U(\mathbf{x})$ given by

$$U(\mathbf{x}) = \sum_{i=1}^{8} f_i(x_i),$$

where

$$f_i(x_i) = \log(x_i + 0.5) \quad i = 1, \ldots, 8,$$

shown to provide proportional fairness; e.g., see [5].

The following choice for the oscillation reducing function was found to provide a good tradeoff between speed of convergence and oscillation reduction: $\omega(t)\colon [0,15] \to [0.625, 0.1316]$, where

$$\omega(t) = 0.5(0.25 + 0.75^t).$$

The idea behind this choice is to avoid large discontinuities at the resetting instants, since they produce undesired spikes. On the other hand, the separation of these instants determines how fast a response to changing networks conditions will be obtained and also affects the speed of convergence.

Call types $i = 3$ and $i = 5$ were assumed to be of the AF class with target rates $\Lambda_3 = 1\,\text{Mbps}$ and $\Lambda_5 = 1.15\,\text{Mbps}$ respectively, while all the other types of calls are assumed to belong to BE traffic. A particular version of the AF adaptation law was utilized: the data rates x_3 and x_5 were held constant at the desired Λ_i value; this is equivalent to taking $\beta_3 = \beta_5 \to \infty$. Finally, the step size was chosen as $t_d = 5\,\text{ms}$ and in order to test the robustness to events such as link failures, the link connecting nodes b_7 and b_8 was opened at time $t = 120\,\text{s}$.

C. Data Rate Adaptation

For this example, the adaptation laws in Section IV were implemented. The values of the parameters not specified above were chosen to satisfy Theorem 1 as $\alpha = 6$ and $\beta_i^{\text{out}} = \beta_{i,b}^{\text{in}} = \beta_{i,b,l}^{\text{out}} = 4$. Finally, recall that routing is performed as stated in Table I so that comparison with percentage adaptation is meaningful.

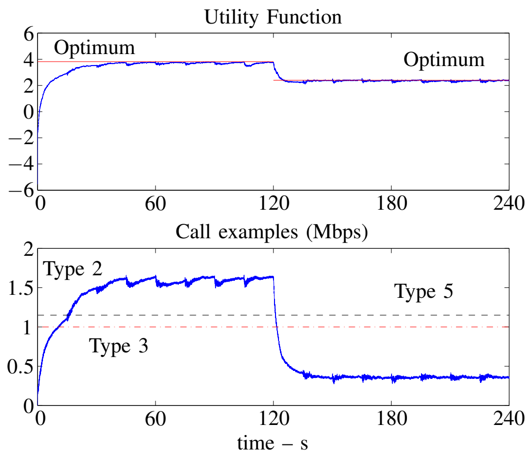


Fig. 4. Data rate adaptation

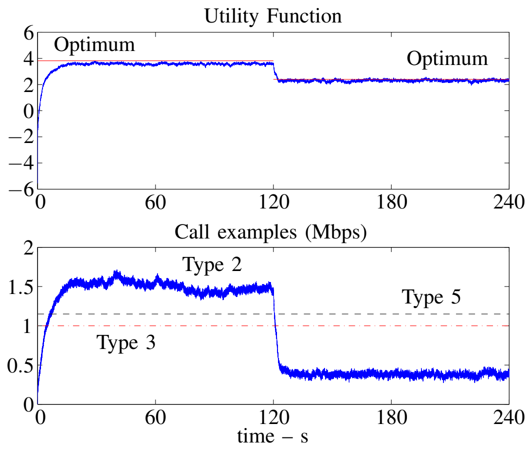


Fig. 5. Data rate adaptation without oscillation reduction

Figure 4 shows the obtained utility function together with a representative data rate trajectory for BE calls of type $i = 2$, and the AF calls types $i = 3$ and $i = 5$. It can be seen that the utility function converges in average to a value close to the optimal one. Furthermore, it exhibits milder oscillations when compared to the case of no oscillation reduction; i.e., constant $z(\cdot)$ (shown in Fig. 5). Note also, that convergence is somewhat faster without oscillation reduction. A more thorough empirical study of this behavior can be found in [8]. Although the control laws used there differ from those presented in this paper, the general observed behavior is similar.

Finally, it can be seen that these control laws excel at the task of dealing with link failures. Indeed, upon failure of the link connecting nodes b_7 and b_8 at time $t = 120$, the network rapidly reacts by appropriately re-routing traffic away from the failure and in this way steering the network towards its new optimal point of operation.

D. Percentage Adaptation

Now, the control laws in Section V are considered, where the adaptation is performed on the percentage of traffic to be carried by each link. The parameters of the adaptation laws were chosen as $\alpha = 4$ and $\beta_i^{\text{out}} = \beta_{i,b}^{\text{in}} = \beta_{i,b,l}^{\text{out}} = 6$, also satisfying the conditions set forth in Theorem 1.

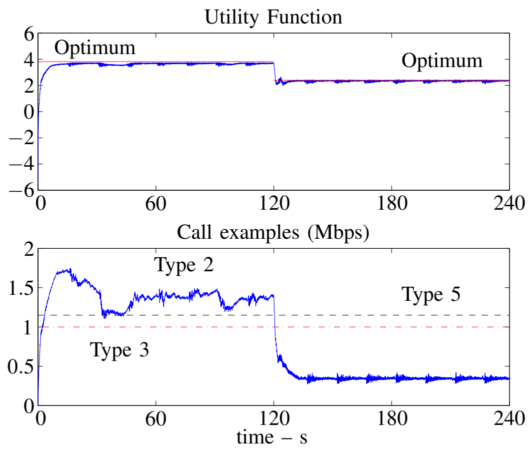


Fig. 6. Percentage Adaptation

It can be observed in Fig. 6 that although not optimal, the network achieves an utilization close to the optimal one. Moreover, the utility function is barely distinguishable from that of data rate adaptation (Fig. 4), which in the ideal case is known to be optimal. This is the case even after the link connecting nodes b_7 and b_8 fails, where convergence to a value close to the new optimum is obtained.

Overall, the behavior of these adaptation laws is comparable to that of the discrete-time version of the laws in Section IV, although much less information is used.

E. On The Choice of Parameters

To conclude these examples, some remarks on the choice of the "free" parameters of the control laws are presented.

The only requirement for the values of α, β^{in}, and β^{out} is that they satisfy the conditions set forth in Theorem 1. However, increasing the value of these parameters introduces undesired oscillation in the data rates. Hence, a small value for these parameters is preferable. Also, our simulations showed that, for the percentage adaptation laws, a value of β^{in} and β^{out} greater than α helps in the satisfaction of the AF constraints.

The overall behavior of the data rate trajectories for the control laws presented in this paper is very similar to the ones in [8], where a thorough empirical analysis of the effect of the parameters on the behavior of this type of adaptation laws is presented. The reader is referred to it for more details.

VIII. Conclusion

In this paper, a "large" family of adaptation laws for optimal rate adaptation and load sharing in a connectionless network was presented. Although these laws achieve optimal network utilization, the computational burden could be high. Therefore, additional empirical laws were presented to cope with this problem. Simulations show that their behavior mimics the optimal and optimality is proven for the case where all data rates are strictly positive.

The results presented in this paper are just a first step in the implementation of a completely decentralized mechanism for traffic engineering. There are several issues that need further study. In particular, effort should be put in the development of procedures for "optimal" design of the parameters of the control laws provided in this paper.

APPENDIX I
PROOF OF THEOREM 1

In this Appendix, the main steps of the proof of Theorem 1 are presented. The proof closely follows the one in [13]. Note that, given the form of the constraints, the adaptation laws presented in Section IV can be represented in the following form

$$\dot{\mathbf{X}} = \mathbf{Z}(t,\mathbf{X})\big[\nabla U(\mathbf{X}) - \mathbf{H}(\mathbf{X})\,\mathbf{v}(\mathbf{X})\big],$$

where $\nabla U(\cdot)$ denotes the gradient of the function $U(\cdot)$, $\mathbf{H}(\cdot)$ is the following matrix

$$\mathbf{H}(\cdot) \doteq \nabla \mathbf{h}(\cdot) \doteq \big[\nabla h_1(\cdot)\;\nabla h_2(\cdot)\;\cdots\;\nabla h_L(\cdot)\big],$$

$\mathbf{Z}(t,\mathbf{X})$ is a positive definite matrix with the individual z functions in the diagonal, h_i are the constraints of the optimization problem, L is the number of such constraints and $\mathbf{v}(\cdot) = \big[v_1(\cdot), v_2(\cdot), \ldots, v_L(\cdot)\big]^T$ is a L-dimensional vector. Also, define the modified utility function as

$$\widehat{U}(\mathbf{X}) \doteq U(\mathbf{X}) - \Xi(\mathbf{X}),$$

where

$$\Xi(\mathbf{X}) \doteq \big[h_1(\mathbf{X}), h_2(\mathbf{X}), \ldots, h_L(\mathbf{X})\big]\mathbf{v}(\mathbf{X}).$$

Lemma 4: The function $\widehat{U}(\mathbf{X})$ does not decrease along the trajectories.

Proof: If during the motion of the system, a sliding mode does not occur

$$\begin{aligned}\frac{d\widehat{U}}{dt} &= \big[\nabla U - \mathbf{H}(\mathbf{X})\mathbf{v}(\mathbf{X})\big]^T \dot{X} \\ &= \big[\nabla U - \mathbf{H}(\mathbf{X})\mathbf{v}(\mathbf{X})\big]^T \mathbf{Z}(t,\mathbf{X})\big[\nabla U - \mathbf{H}(\mathbf{X})\mathbf{v}(\mathbf{X})\big] \geq 0,\end{aligned}$$

since the matrix $\mathbf{Z}(t,\mathbf{X})$ is positive definite.

Now, assume that a sliding mode occurs in the intersection of some surfaces $h_k(\mathbf{X})$; i.e., $h_k(\mathbf{X}) = 0$, $k \in \mathscr{I}$. Let $\mathbf{H}_1(\mathbf{X})$ be the matrix whose columns consist on $\nabla h_k(\mathbf{X})$ for $k \in \mathscr{I}$ (in the same order as in $\mathbf{H}(\mathbf{X})$). Furthermore, let $\mathbf{H}_2(\mathbf{X})$ be the matrix whose columns are $\nabla h_k(\mathbf{X})$ for $k \notin \mathscr{I}$ (again in the same order as in $\mathbf{H}(\mathbf{X})$) and partition vector $\mathbf{v}$ into $\mathbf{v}_1$ and $\mathbf{v}_2$ accordingly. Then, it holds that

$$\frac{d\mathbf{H}_1}{dt} = \mathbf{H}_1^T \dot{\mathbf{X}} = 0.$$

Solving this equation for $\mathbf{v}_1$ the following equivalent control is obtained

$$\mathbf{v}_{1,\text{eq}} = \left(\mathbf{H_1}^T\mathbf{Z}\mathbf{H_1}\right)^{-1}\left(\mathbf{H_1}^T\mathbf{Z}\nabla U - \mathbf{H_1}^T\mathbf{Z}\mathbf{H_2}\mathbf{v}_2\right)$$

and the sliding motion satisfies the following differential equation

$$\dot{\mathbf{x}} = \sqrt{\mathbf{Z}}\mathbf{P}\sqrt{\mathbf{Z}}\big(\nabla U - \mathbf{H_2}\mathbf{v}_2\big),$$

where $\mathbf{P}$ is the projection matrix

$$\mathbf{P} \doteq I - \sqrt{\mathbf{Z}}\mathbf{H_1}\left(\mathbf{H_1}^T\sqrt{\mathbf{Z}}\sqrt{\mathbf{Z}}\mathbf{H_1}\right)^{-1}\mathbf{H_1}^T\sqrt{\mathbf{Z}}.$$

Hence,

$$\begin{aligned}\frac{d\widehat{U}}{dt} &= \big(\nabla U - \mathbf{H_2}\mathbf{v}_2\big)^T\sqrt{\mathbf{Z}}\mathbf{P}\sqrt{\mathbf{Z}}\big(\nabla U - \mathbf{H_2}\mathbf{v}_2\big) \\ &= \left\|\mathbf{P}\sqrt{\mathbf{Z}}\left(\nabla U - \mathbf{H_2}\mathbf{v}_2\right)\right\|^2 \geq 0.\end{aligned}$$

Therefore, $\widehat{U}(\mathbf{X})$ does not decrease along the trajectories. ■

Lemma 5: The time derivative of $\widehat{U}(t)$ is zero only when $\dot{\mathbf{X}} = 0$.

Proof: If a sliding mode does not occur it holds that

$$\frac{d\widehat{U}}{dt} = \big[\nabla U - \mathbf{H}\mathbf{v}\big]^T\mathbf{Z}\big[\nabla U - \mathbf{H}\mathbf{v}\big]$$

and since $\mathbf{Z}$ is positive definite

$$\frac{d\widehat{U}}{dt} = 0 \Leftrightarrow \nabla U - \mathbf{H}\mathbf{v} = 0 \Leftrightarrow \mathbf{Z}\big[\nabla U - \mathbf{H}\mathbf{v}\big] = 0 \Leftrightarrow \dot{\mathbf{X}} = 0.$$

Now, assume that a sliding mode occurs in the intersection of the surfaces $h_k(\mathbf{X}) = 0$, for $k \in \mathscr{I}$. In this case,

$$\begin{aligned}\frac{d\widehat{U}}{dt} = \left\|\mathbf{P}\sqrt{\mathbf{Z}}\big(\nabla U - \mathbf{H_2}\mathbf{v}_2\big)\right\|^2 = 0 \Leftrightarrow \mathbf{P}\sqrt{\mathbf{Z}}\big(\nabla U - \mathbf{H_2}\mathbf{v}_2\big) = 0 \\ \Leftrightarrow \sqrt{\mathbf{Z}}\mathbf{P}\sqrt{\mathbf{Z}}\big(\nabla U - \mathbf{H_2}\mathbf{v}_2\big) = 0 \Leftrightarrow \dot{\mathbf{X}} = 0.\end{aligned}$$

Therefore, $d\widehat{U}/dt = 0$ if and only if $\dot{\mathbf{X}} = 0$. ■

Theorem 6 (Utkin,1992 [13]): The maximum of $\widehat{U}$ coincides with the maximum of U; i.e., $U(\mathbf{X}^*)$ if and only if

$$\nabla U(\mathbf{X}^*) = \mathbf{H}\mathbf{v}_{\text{eq}}$$

at least in one point $\mathbf{X}^*$ and $\mathbf{v}_{\text{eq}}$ belongs to the convex hull of all vectors $\mathbf{v}$.

Lemma 7: The set of stationary points of $\widehat{U}$ coincide with the set of maximum points of $\widehat{U}$.

Proof: Let $\mathbf{X}_0$ be stationary point of $\widehat{U}$. Then,

$$\begin{aligned}\frac{\widehat{U}(\mathbf{X}_0)}{dt} = 0 \Leftrightarrow \dot{\mathbf{X}}|_{\mathbf{X}_0} = 0 \Leftrightarrow \nabla U(\mathbf{X}_0) - \mathbf{H}(\mathbf{X}_0)\mathbf{v}(\mathbf{X}_0) = 0 \\ \Leftrightarrow \mathbf{X}_0 \text{ is a KKT point} \Leftrightarrow \mathbf{X}_0 = \arg\max U = \arg\max \widehat{U}\end{aligned}$$

■

Proof of Theorem 1

Given the conditions on α, β_i and β and the fact that the functions f_i are increasing concave functions, it holds that at the solution of the optimization problem the KKT conditions are satisfied; i.e.,

$$\nabla U(\mathbf{X}^*) - \mathbf{H}\mathbf{v}_{\text{eq}} = 0$$

and $\mathbf{v}_{\text{eq}}$ belongs to the convex hull of all possible values of vector $\mathbf{v}$, so that Theorem 6 applies. Then, according to [13](Theorem 2, page 231) the control laws in Section IV converge to the maximum of the utility function $U(\mathbf{X})$. ■

APPENDIX II
SKETCH OF THE PROOF OF PROPOSITION 2

Let $p_{i,b,l}$ be defined as

$$p_{i,b,l}(t) = \frac{x^{\text{out}}_{i,b,l}(t)}{\sum\limits_{\tilde{l}\in\mathscr{L}_{b,i}} x^{\text{out}}_{i,b,\tilde{l}}(t)} \qquad l \in \mathscr{L}_{b,i}$$

Straightforward computation of the time derivative of $p_{i,b,l}(t,\mathbf{x}_{\text{out}})$ yields

$$\frac{d\,p_{i,b,l}(t)}{dt} = \frac{\dot{x}^{\text{out}}_{i,b,l}(t) \sum\limits_{\tilde{l}\in\mathscr{L}_{b,i};\,\tilde{l}\neq l} p_{i,b,\tilde{l}}(t) - p_{i,b,l}(t) \sum\limits_{\tilde{l}\in\mathscr{L}_{b,i};\,\tilde{l}\neq l} \dot{x}^{\text{out}}_{i,b,\tilde{l}}(t)}{\sum\limits_{l\in\mathscr{L}_{b,i}} x^{\text{out}}_{i,b,l}(t)},$$

where the derivatives $\dot{x}^{\text{out}}_{i,b,\tilde{l}}$ is computed according to the optimal control laws given in Section IV. Now, if all data rates are strictly positive, the following positive multiplication factor can be introduced

$$z_{i,b}\big(t, cg_l(t), r^{\text{in}}_{i,b}(t), r^{\text{out}}_{i}(t)\big) \sum_{l\in\mathscr{L}_{b,i}} x^{\text{out}}_{i,b,l}(t).$$

This factor has as only effect to change the adaptation speed but it does not affect the steady state behavior of these laws, since it is strictly positive.

Proceeding in this way and dropping function arguments for notational convenience, the proposed percentage adaptation laws are obtained

$$\dot{p}_{i,b,l} = z_{i,b}\Big(\dot{x}^{\text{out}}_{i,b,l} \sum_{\tilde{l}\in\mathscr{L}_{b,i};\,\tilde{l}\neq l} p_{i,b,\tilde{l}} - p_{i,b,l} \sum_{\tilde{l}\in\mathscr{L}_{b,i};\,\tilde{l}\neq l} \dot{x}^{\text{out}}_{i,b,\tilde{l}}\Big).$$

Therefore, these laws are equivalent to the convergent data rate adaptation laws, for the case of strictly positive data rates and thus they also converge to the optimal data rate of the optimization problem at hand. ■

REFERENCES

[1] H. Che, “Encoded range matching for a TCAM coprocessor.” [Online]. Available: http://crystal.uta.edu/~hche/PUBLICATIONS/publication.htm

[2] V. V. Filippov, *Basic Topological Structures of Ordinary Differential Equations*, ser. Mathematics and Its Applications. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998, vol. 432.

[3] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights,” in *Proc. IEEE INFOCOM’2000*, vol. 2, Tel Aviv, Israel, Mar. 2000, pp. 519–528.

[4] C. Hoops, “Analysis of an equal-cost multi-path algorithm,” IETF RFC 2992, Nov. 2000.

[5] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, “Rate control in communication networks: Shadow prices, proportional fairness and stability,” *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[6] R. J. La and V. Anantharam, “Charge-sensitive TCP and rate control in the Internet,” in *Proc. IEEE INFOCOM’2000*, vol. 3, Tel Aviv, Israel, Mar. 2000, pp. 1166–1175.

[7] C. M. Lagoa and H. Che, “Decentralized optimal traffic engineering in the Internet,” *SIGCOMM Comput. Commun. Rev.*, Oct. 2000.

[8] C. M. Lagoa, H. Che, and B. A. Movsichoff, “Adaptive control algorithms for decentralized optimal traffic engineering in the Internet,” 2004, to appear in *IEEE/ACM Trans. Networking*. [Online]. Available: http://eeilserv.ee.psu.edu/lagoa/papers/lcm2004.pdf

[9] B. A. Movsichoff, C. M. Lagoa, and H. Che, “A sliding mode approach to traffic engineering in computer networks,” in *Advances in Communication Control Networks*, ser. Lecture Notes in Control and Information Sciences (LCNCIS), C. T. Abdallah, J. N. Chiasson, and S. Tarbouriech, Eds. Springer-Verlag, 2004, to be published.

[10] G. Schollmeier, J. Charzinsky, A. Kirstadter, C. Reichert, K. J. Schrodi, Y. Glickman, and C. Winkler, “Improving the resilience in IP networks,” in *Proc. IEEE Workshop on High Performance Swithching and Routing*, Jun. 2003, pp. 91–96.

[11] A. Sridharam, R. Guérin, and C. Diot, “Achieving near optimal traffic engineering solutions for current OSPF/IS-IS networks,” in *Proc. IEEE INFOCOM’2003*, vol. 2, San Francisco, CA, USA, Mar./Apr. 2003, pp. 1167–1177.

[12] D. G. Thaler and C. V. Ravishankar, “Using name-based mappings to increase hit rates,” *IEEE/ACM Trans. Networking*, vol. 6, pp. 1–14, Feb. 1998.

[13] V. I. Utkin, *Sliding Modes in Control and Optimization*, ser. Communications and Control Engineering Series. Berlin, Heidelberg: Spriger-Verlag, 1992, vol. 66, no. 1.

[14] Y. Wang, Z. Wang, and L. Zhang, “Internet traffic engineering without full mesh overlaying,” in *Proc. IEEE INFOCOM’2001*, vol. 1, Anchorage, AK, USA, Apr. 2001, pp. 565–571.

[15] Z. Wang, H. Che, M. Kumar, and S. Das, “CoPTUA: Consistent TCAM policy table update with zero impact on data path processing,” *IEEE Trans. Comput.*, conditionally accepted with minor revisions. [Online]. Available: http://crystal.uta.edu/~hche/PUBLICATIONS/publication.htm

PLACE
PHOTO
HERE

Bernardo A. Movsichoff (SM ’04) obtained his Ingeniero Electrónico degree from the Universidad de Buenos Aires, Argentina in December 1996. Since Fall 1999 he is at The Pennsylvania State University, where he is a Research Assistant working towards a Ph.D. degree in Electrical Engineering. His research interests include control and identification of uncertain systems and the application of control theory and optimization to computer networks.

PLACE
PHOTO
HERE

Constantino M. Lagoa (M ’98) got his B.S. and M.Sc. degrees from the Instituto Superior Técnico, Technical University of Lisbon, Portugal in 1991 and 1994 respectively and his Ph.D. degree from the University of Wisconsin at Madison in 1998. He joined the Electrical Engineering Department of The Pennsylvania State University in August 1998, where he currently holds the position of Associate Professor. He has a wide range of research interests including robust control, controller design under risk specifications, control of computer networks and discrete event dynamical systems. In 2000 he received the NSF CAREER award for his proposal on system design under risk constraints.

PLACE
PHOTO
HERE

Hao Che received the B.S. degree from Nanjing University, Nanjing, China, the M.S. degree in physics from the University of Texas at Arlington, TX, in 1994, and Ph.D. degree in electrical engineering from the University of Texas at Austin, TX, in 1998. He was an Assistant Professor of Electrical Engineering at the Pennsylvania State University, University Park, PA, from 1998 to 2001, and a System Architect with Santera Systems, Inc., Plano, TX, from 2000 to 2002. Since September 2002, he has been an Assistant Professor of Computer Science and Engineering at the University of Texas at Arlington, TX. His current research interests include network architecture and design, network resource management, multiservice switching architecture, and network processor design.