A Unified Framework for Multipath Routing for Unicast and Multicast Traffic

Tuna Güven[†], Richard J. La[†], Mark A. Shayman,[†] and Bobby Bhattacharjee[‡]

***Abstract*— We study the problem of load balancing the traffic from a set of unicast and multicast sessions. The problem is formulated as an optimization problem. However, we assume that the gradient of the network cost function is *not* available and needs to be estimated. Multiple paths are provided between a source and a destination using application-layer overlay. We propose a novel algorithm that is based on what is known as simultaneous perturbation stochastic approximation and utilizes only noisy measurements collected and reported to the sources, using an overlay architecture.**

We consider three network models that reflect different sets of assumptions regarding multicast capabilities of the network. Using an analytical model we first prove the almost sure convergence of the algorithm to a corresponding optimal solution under each network model considered in this paper with decreasing step sizes. Then, we establish the weak convergence (or convergence in distribution) with a fixed step size. In addition, we investigate the benefits acquired from implementing additional multicast capabilities by studying the relative performance of our algorithm under the three network models.

I. Introduction

Multicast traffic over the Internet is growing steadily with increasing number of demanding applications including Internet broadcasting, video conferencing, data stream applications, distributions, and exchange of large data sets by geographically distributed scientists working in collaboration. Ideally many of these applications require certain rate guarantees, and providing such guarantees demands that the network be utilized more efficiently than with current approaches to satisfy the rate requirements. Traffic mapping (or load balancing) is one particular method to carry out traffic engineering, which deals with the problem of assigning traffic load onto pre-established paths to meet certain requirements [1].

Many major Internet Service Providers (ISPs) are in various stages of increasing their network capacity and node connectivity [22]. A higher level of network connectivity typically provides multiple paths between source-destination pairs, and offers an opportunity to better utilize network resources through load balancing. The focus of this paper is to investigate the potential benefits of load balancing both unicast and multicast traffic within a single domain and to propose a practical routing algorithm for carrying out such load balancing, using only noisy network measurements.

Although the issue of distributed routing using multiple paths is an old problem with extensive literature on it (e.g., [5], [9]), there is a limited amount of existing work on multipath *multicast* routing. Park and Shin [18] propose a scheme that creates multiple trees between a source and a set of destinations and splits the traffic optimally among the trees. However, the proposed solution covers only a single source case. In addition, it assumes the existence of the gradient of an analytical cost function and that the cost function is strictly convex and continuously differentiable. As discussed in [7], in practice it may be difficult, if not impossible, to precisely define accurate analytical cost functions due to the dynamic nature of networks. Further, even when an analytical cost function exists, it may not be differentiable everywhere. As we will show in this paper, these assumptions can be relaxed considerably.

In another set of work [17], [24] solutions based on network coding are proposed. Even though they approach the problem under a more general architecture, these solutions suffer from the limitations inherited from network coding. Network coding relies on an unrealistic assumption that the network is lossless as long as the *average* link rates do not exceed the link capacities. Furthermore, a packet loss can be much more costly when network coding is employed because it can potentially affect the decoding of a large number of other packets. In addition, any event that changes the min-cut max-flow value between a source and a receiver requires the code to be updated at every node simultaneously. This brings about considerable complexity and demands a high level of coordination and synchronism among the nodes. Besides, similar to earlier efforts, these solutions also assume that there is only one multicast session in the network.

In this paper we propose a distributed optimal routing algorithm that balances the load along multiple paths for multiple unicast and multicast sessions. Our *measurement-based* algorithm does not assume the existence of the gradient of an analytical cost function and is a natural generalization of the *unicast* routing algorithms proposed in [7], [11]. To the best of our knowledge this is the first attempt to address the issue of (optimal) multi-path routing with *multiple* multicast sessions in a *distributed* manner, while relying only on (local) network measurements.

In order to study the performance of the proposed algorithm and to understand and quantify the benefits of implementing additional multicast capabilities in the underlying IP network, we consider three different network models with gradually increasing network capabilities; first, we look at the problem under the traditional network model without any IP multicast functionality. In this model multiple paths are provided using a limited number of (application-layer) overlay nodes that are used as relay nodes. In the second model, we allow IP multicast and load balancing of multicast traffic is carried out utilizing

[†] – Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742, {tguven,hyongla,shayman}@eng.umd.edu

[‡] – Department of Computer Science, University of Maryland, College Park, MD 20742, bobby@cs.umd.edu

multiple multicast trees available for each source. These trees are rooted either at the source or at the overlay nodes that act as surrogate sources for traffic forwarded by the source. The routers are assumed to be capable of copying and forwarding multicast packets onto downstream branches. Finally, in the third model we replace the routers in the second model with "smart" routers capable of forwarding multicast packets onto each downstream branch at a *different* rate. This model enables our algorithm to exercise finer control over rate allocation than in the second model. These models together provide us with a general framework that helps us identify functionalities beyond basic operations (e.g., store-and-forward) that are essential for the performance gain from load balancing.

It is worth noting that the finer control available in the third model comes at the price of additional intelligence needed at the routers. This is due to the fact that in order to ensure the delivery of distinct packets to the receivers a source needs to maintain careful bookkeeping of all the packets forwarded to each receiver so that every packet is forwarded to each receiver and delivery of duplicate packets is minimized. For the same reasons, an intermediate IP router must be able to identify the set of intended receivers for *each* multicast packet. We will show that this problem of potentially complicated bookkeeping at the multicast sources and the core overlay nodes can be avoided by employing a family of source codes called Digital Fountain codes [16] (subsection II-B).

Our proposed algorithm is based on *simultaneous perturbation stochastic approximation* (SPSA). We consider two scenarios: We first consider the case of decreasing step sizes. We show that under a set of mild conditions the algorithm converges with probability 1 (w. p. 1) to an optimal solution that minimizes the network cost with decreasing step sizes. In practice, a policy that employs decreasing step sizes may lose the ability to react in a timely manner to changes in network conditions once the step sizes become small. As a result, the step sizes need to be reset, for example after a certain period of time, to ensure that the algorithm stays responsive to network changes. An alternative to this is to use a constant or fixed step size. Although we are not able to prove the same almost sure (a.s.) convergence with a fixed step size, we show that when the step size is sufficiently small, the algorithm converges to a small neighborhood around the set of optimal points. In subsection VII-B we demonstrate using simulation results that the performance of a fixed step size algorithm is comparable to that of a policy with decreasing step sizes. Some preliminary results of this paper have been reported in [12].

The rest of the paper is organized as follows: In Section II we introduce the basic setup of the problem and the Digital Fountain Codes that are used to solve the bookkeeping problem mentioned earlier. Section III describes the three network models used for analysis and performance evaluation, which is followed by a description of the optimization framework and the proposed algorithm in Section IV. We establish the convergence properties of the proposed algorithm in Section V. Section VI discusses the implementation issues and simulation results are provided in Section VII. We conclude in Section VIII.

II. Background & Set-up

Consider a network that consists of a set of unidirectional links $\mathcal{L} = \{1, \ldots, \mathrm{L}\}$. Let $\mathcal{S} = \{1, \ldots, \mathrm{S}\}$ be the set of source nodes. Each source node is associated with a session that can be either a unicast or a multicast session.[1] We use D^s to denote the set of destination nodes for the source $s \in \mathcal{S}$. Each source needs to deliver packets to every destination $d \in D^s$ at a fixed rate r^s. In this paper we study the scenario where the sources can make use of pre-installed application-layer overlay nodes that can be used to provide the sources with alternate paths in addition to the default path provided by the underlying IP routing protocol.

We are interested in designing a load balancing algorithm that can utilize multiple paths available between sources and destination nodes and optimize the network performance according to a given network cost function. In order to capture the performance improvement vs. cost trade-off with increasing network capabilities, we consider three different network models (described in Section III). These models differ in the set of assumptions we impose on the capability of the underlying network. We study the relative performance of these systems and some of the properties of their operating points.

In the remainder of this section we first explain the overlay architecture that is used to create multiple paths between a source node and a destination node. Here a destination node refers to a unicast destination node or a multicast receiver node. Then, we explain how some of the bookkeeping issues that arise in our problem can be handled using a special family of codes, called Digital Fountain codes.

A. General routing framework - overlay architecture

In standard application-layer overlay networks overlay nodes are selected among the end-hosts that are possibly located under different administrative domains (see [6]). In this paper we only consider the case of a single administrative domain. An *overlay node* in our architecture is a router or an end host with a simple device (e.g., a network processor) attached to it. We call a network router with such a device a *core* overlay node and an end host overlay node an *edge* overlay node. In this paper we assume that all source nodes are edge overlay nodes.

A source node can create an alternate path to a destination node using one of core overlay nodes as follows: The source node first attaches an additional IP header to the packet with the IP address of the selected overlay node as the destination address[2] and then forwards the packet to the overlay node using the underlying routing protocol. When the overlay node receives the packet, it first strips from the packet the extra IP header used by the application overlay and then forwards it to the destination node utilizing the underlying routing protocol. This is shown in Fig. 1.

This can be viewed as a form of loose source routing in the sense that the source node can exercise a certain level of

[1] For simplicity each source is assumed to have a single session. However, the results also apply to the case where there may be multiple sessions originating at a single source node.

[2] This is true for the multicast case as well. The outer IP header address will be the IP address of the overlay node while the inner IP address will be the multicast address.

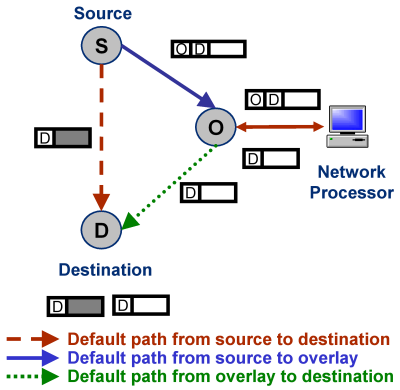

Fig. 1. Proposed overlay architecture for multi-path routing.

route selection for individual packets. Also, we assume that at most one overlay node is used along any path. In principle a source node can forward any fraction of packets to a destination node through any of the available core overlay nodes, creating multiple paths to the destination node. Note that this approach does not require any changes to the underlying IP routing protocol.

We denote the set of *core* overlay nodes by $\mathcal{O}$ and the set of overlay nodes in $\mathcal{O}$ used to create *alternative* paths between a source $s \in \mathcal{S}$ and its destination node(s) in D^s by $O_c^s \subseteq \mathcal{O}$. Since every source node is also an (edge) overlay node, the set of overlay nodes utilized by a source $s \in \mathcal{S}$ is given by $O^s := O_c^s \cup \{s\}$. We denote the number of paths available between a source s and its destination node(s) by N_s, which is given by $|O^s|$, i.e., the cardinality of O^s. Define $N = \sum_{s\in\mathcal{S}}(N_s \cdot |D^s|)$.

B. Digital Fountain codes

As discussed in Section I, when a source s forwards packets to a destination d, without any special coding it must ensure that the destination receives all distinct packets necessary for recovering the message. When different sets of packets are forwarded to different destinations using two or more overlay nodes, the source must keep track of the packets forwarded along different paths so that every destination receives all necessary packets. As a result, this requires complicated bookkeeping at the multicast sources and the core overlay nodes. We propose the adoption of Digital Fountain codes to solve this problem.

The original application area of Digital Fountain codes [16], [20] is the reliable transmission of data over the Internet as an alternative to the unicast TCP/IP retransmissions. Fountain codes are rateless in the sense that the number of encoded packets that can be generated from the source message is potentially limitless; the number of encoded packets to be generated can be determined on the fly. Regardless of the statistics of the erasure events on the channel, one can send as many encoded packets as needed in order for the decoder to recover the source data. The input and output symbols can be bits or more generally binary vectors of arbitrary length. Each output symbol is generated by a (binary) addition of some randomly selected input symbols. The number of input symbols to be added is determined according to some fixed degree distribution. It is assumed that each output symbol is tagged with information describing which input symbols are used to generate it, for example, in the packet header.

A decoding algorithm for a Fountain code is an algorithm that recovers the original K input symbols from *any* set of M output symbols with a high probability. For good Fountain codes the value of M is very close to K and the decoding time is approximately linear in K. An example is the Raptor codes [20] that has linear time encoders and decoders for which the probability of decoding failure converges to zero polynomially fast in the number of input symbols. For instance, for $K =$ 64,536 and $M =$ 65,552, i.e., with a redundancy of 1.5 percent, it is shown in [20] that the error probability is upper bounded by 1.71×10^{-14}. In practice, however, Fountain codes introduce approximately 5 percent overheads.

In our formulation we assume that a source first divides the traffic into blocks of, say, K symbols and applies a form of Digital Fountain code (e.g,. Raptor code) to generate encoded output symbols that are forwarded to the destinations. Here the block size will be constrained by the buffer size at the source. Every receiver can recover the K source symbols in each block from any M distinct encoded symbols. Given that number of distinct encoded packets can be limitless for Fountain codes, the source node can forward distinct set of encoded packets to each overlay node. Hence, overlay nodes can be viewed as content delivery servers that store *distinct* portions of the original content to be distributed. It is worth to mention that our model is different from traditional content delivery schemes making use of Fountain coding, such as [4]. First, *only* end-host nodes are assumed to serve as overlays in these schemes. Second, the overlay nodes are assumed to have *overlapping* portions of the original content which is natural when peer-to-peer applications are considered. On the contrary, we purposefully make sure that each *core* overlay have a distinct set of encoded content by making use of the Fountain coding. This way we make sure that each destination node receives distinct set of encoded packets along each path available to it, *i.e.*, the possibility of receiving multiple copies of the same encoded packet is eliminated. This guarantees each receiver to successfully decode the data received as long as destinations receives packets at a sufficient rate. Therefore, the source node does not require any bookkeeping as long as it sends *distinct* packets along each path at sufficient rates. This allows us to formulate our problem as one of assigning packet forwarding rates to available paths for each destination, subject to a constraint that the aggregate rate at which the destination receives packets exceeds certain threshold. This threshold depends on the demand rate r^s as well as the efficiency of the coding scheme.

III. Network Models

In this section we describe three different models that we consider for performance evaluation in increasing order of network capabilities: For each $s \in \mathcal{S}$ and $d \in D^s$ let $x_{o,d}^s$ be the rate at which the source node s sends packets to destination d through an overlay node $o \in O^s$.[3] Also, define x_o^s to be the total

[3]When a source s uses a default path or a multicast tree rooted at itself to deliver some of packets to the receivers, we say that the source uses itself as an overlay node to forward the packets, although strictly speaking no processing by any overlay node is involved for these packets.

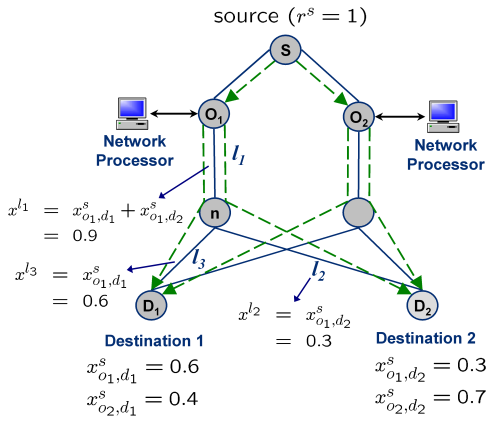

Fig. 2. Network model-I (NM-I). Each dotted line represents a unicast session.

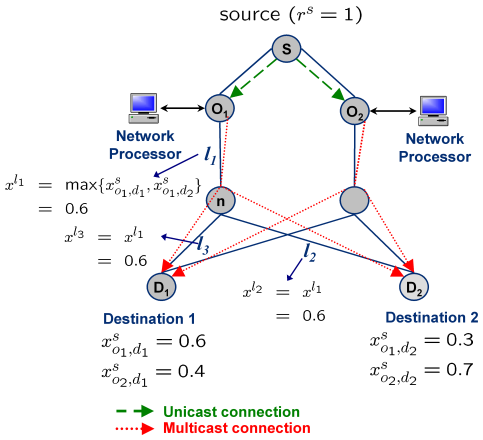

Fig. 3. Network model-II (NM-II).

rate at which an overlay node o receives packets from source s. In a unicast case this is simply the rate at which packets are forwarded to the destination through the overlay node, while in the case of a multicast session, it depends on the capability of the underlying network and the implementation as will be explained shortly.

As mentioned in the previous subsection, the adoption of a Digital Fountain code reduces our problem to that of rate assignment $\boldsymbol{x} = (x^s_{o,d}, s \in \mathcal{S}, o \in O^s, d \in D^s)$, which is the focus of the remainder of this paper. We assume that the overlay nodes can copy packets. Hence, the sources need to deliver only a single copy of the packets to an overlay node. The overlay node acts as a surrogate source for these packets and is responsible for sending a copy of the packets it receives to each destination. Under this assumption, the rate x^s_o to an overlay node o is given by $x^s_o = \max_{d \in D^s} x^s_{o,d}$, and depending on the assumed network model and the assigned rates, some or all of the packets forwarded to the overlay node are relayed to the destinations.

A. Network Model-I

The first network model we consider represents a unicast only IP network, where routers do not possess IP multicast functionality. We assume that packets are encoded using a Digital Fountain code at the source. A source node first forwards the encoded packets to overlay nodes at the required rate, and the overlay nodes create a unicast session for *each* destination and forward packets at the specified rate $x^s_{o,d}$. Hence, a source node and overlay nodes need to maintain multiple unicast sessions in case of a multicast session with more than one destination. This is shown in Fig. 2.

Let $V^{n_1}_{n_2} \subset \mathcal{L}$ be the set of links in the default path from node n_1 to node n_2. Given a rate assignment $\boldsymbol{x}$, the link loads $x^l, l \in \mathcal{L}$, under this network model are given by

$$x^l = \sum_{s \in S} \left(\sum_{o \in O^s_c : l \in V^s_o} x^s_o + \sum_{o \in O^s} \left(\sum_{d \in D^s : l \in V^o_d} x^s_{o,d} \right) \right) \quad (1)$$

We refer to this model as NM-I.

B. Network Model-II

Under Network Model-II the routers are IP multicast capable. We assume that each overlay node $o \in O^s$ creates a separate multicast tree $\mathcal{MT}^s_o$ rooted at itself for forwarding packets from the source s, using an intradomain multicast algorithm (e.g., DVMRP [23]). However, we assume that IP multicast routers are only capable of copying and forwarding packets. Hence, every packet forwarded to an overlay node by a source node s is relayed to *all* destinations in D^s. As a result, the rate at which destination nodes receive packets from an overlay node is the same (assuming no packet losses) and is given by $x^s_o = \max_{d \in D^s} x^s_{o,d}$. This is shown in Fig. 3. Clearly, this may cause a receiver to receive packets at a rate larger than the intended rate. However, as we will show shortly, our algorithm exploits this property through measurements and attempts to eliminate such redundancy. In fact, at the optimal operating point $\boldsymbol{x}^\star$ we have $x^s_{o,d}{}^\star = x^{s\star}_o$ for all $d \in D^s$.

Under this model the load of link l can be written as

$$x^l = \sum_{s \in \mathcal{S}} \left(\sum_{o \in O^s_c : l \in V^s_o} x^s_o + \sum_{o \in O^s : l \in T^s_o} x^s_o \right) \quad (2)$$

where T^s_o is the set of links in the multicast tree $\mathcal{MT}^s_o$ in the case of a multicast session or the set of links in the default path in the case of a unicast session, *i.e.*, $T^s_o = V^o_d$. This model is referred to as NM-II.

C. Network Model-III

In this model, in addition to the IP multicast capability we also assume that each router is capable of forwarding packets onto each branch at a different rate. We refer to these routers as "smart" routers to distinguish them from the routers used in NM-II. This is shown in Fig. 4. Under this model a source s can select the individual rates $x^s_{o,d}$ independently for each destination, and packets will be forwarded to a destination $d \in D^s$ at the intended rate $x^s_{o,d}$ (as opposed to $\max_{d' \in D^s} x^s_{o,d'}$ under NM-II).[4] This allows the network operator more flexibility and fine-grained control in rate assignment and to better exploit

[4]This assumes that there are no packet losses along the path.

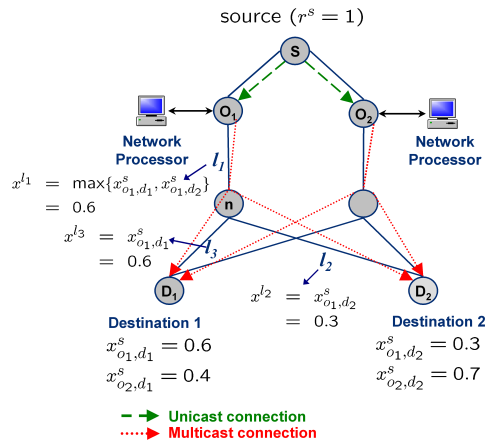

Fig. 4. Network model-III (NM-III).

the existence of multiple paths through overlay nodes, while making use of multicast nature of the traffic at the same time.

The link rates under this model are given by

$$x^l = \sum_{s\in S}\left(\sum_{o\in O^s_c: l\in V^s_o} x^s_o + \sum_{o\in O^s}\max_{d\in D^s: l\in \hat{V}^o_d} x^s_{o,d}\right). \quad (3)$$

Here $\hat{V}^o_d$ denotes the set of links along the path from overlay node o to destination d. In the case of a multicast session, this is the set of links in the multicast tree, which may be different from the default path provided by the underlying routing protocol. We will refer to this model as NM-III.

Due to the large gap between the level of intelligence currently available at the IP routers and the required intelligence assumed in this model, it is unlikely that this type of network will be available in the near future. However, we consider this model for comparison purposes, and compare the performance of the other models against that of this somewhat ideal model.

As mentioned before, under these models overlay nodes can be viewed as content delivery servers that store distinct portions of the original content to be distributed. Our goal is to design a *unified* load balancing algorithm that minimizes the total network cost by distributing the traffic load among multiple available paths, under *all* three network models. However, since the link loads depend on the assumed network model, the desired operating point as well as the aggregate network cost are also determined by the assumed network model. This will allow us to quantify the additional net benefit we can acquire from placing increasing network capabilities and to carry out the performance vs. cost trade-off analysis.

IV. Optimization Framework and the Proposed Algorithm

We formulate the problem of rate assignment as an optimization problem, where the objective function is the sum of link costs. A link cost is a function of the total rate traversing the link x^l and is given by $C_l(x^l), l \in \mathcal{L}$. These link cost functions are assumed to be convex but need not be differentiable. The optimization problem can be stated as follows:

$$\min_{\boldsymbol{x}} C(\boldsymbol{x}) = \textstyle\sum_{l\in\mathcal{L}} C_l(x^l) \quad (4)$$

$$\text{s.t. } \textstyle\sum_{o\in O_s} x^s_{o,d} = r^s + \epsilon^s, \forall s \in \mathcal{S}, d \in D^s \quad (5)$$

$$x^s_{o,d} \geq \nu, \ \forall s \in \mathcal{S}, o \in O^s, d \in D^s \quad (6)$$

where r^s is the assumed traffic rate of source s, ν is an arbitrarily small positive constant,[5] and ϵ^s is the additional rate required by the coding scheme for a receiver to successfully decode the incoming encoded data.

Our problem in (4) can be viewed as a natural generalization of the problem studied in [7], [11]. Indeed, the algorithm proposed in [11] is a special case of the algorithm described in this paper with *only unicast* sessions under NM-I described in subsection III-A. There are two major differences between the current problem and that investigated in [7], [11]. First, in this paper the resulting link loads $x^l, l \in \mathcal{L}$, from the rate assignment and hence the overall network cost depend on the adopted network model. Consequently, the performance of the system depends on the assumed capability of the underlying network. Second, the lack of differentiability of the cost function with respect to the rate assignment $\boldsymbol{x}$ raises some technical issues as will be clear in the analysis.

A. Simultaneous perturbation stochastic approximation

The optimization problem in (4) can be solved using a Stochastic Approximation (SA) technique. SA is a recursive procedure for finding the root(s) of equations using noisy measurements, and is useful for finding extrema of functions [3], [21]. The general constrained SA is similar to the well-known gradient projection algorithm, in which at each iteration $k = 0, 1, \ldots$, the variables are updated based on the gradient. In SA, however, the gradient vector $\nabla C(k)$ is replaced by its approximation $\hat{\boldsymbol{g}}(k)$. The approximation is often obtained through measurements of the cost $C(\boldsymbol{x})$ around $\boldsymbol{x}(k)$. Under appropriate conditions, $\boldsymbol{x}(k)$ can be shown to converge to a solution of (4) almost surely (a.s.) as will be shown in subsection V-B.

One particular method used for gradient estimation is called *Simultaneous Perturbation* (SP). When SP is employed, all elements of $\boldsymbol{x}(k)$ are randomly perturbed simultaneously to obtain two measurements, $y(\boldsymbol{x}(k) + \xi(k)\boldsymbol{\Delta}(k))$ and $y(\boldsymbol{x}(k) - \xi(k)\boldsymbol{\Delta}(k))$ [21]. Here $\xi(k)$ is some positive scalar, and $\boldsymbol{\Delta}(k) = (\Delta_1(k), \ldots, \Delta_m(k))$ is a random perturbation vector generated by the SP method and needs to satisfy certain conditions. The i-th component of the gradient approximation $\hat{\boldsymbol{g}}(k)$ is computed from these two measurements according to

$$\hat{g}_i(k) = \frac{y(\boldsymbol{x}(k) + \xi(k)\boldsymbol{\Delta}(k)) - y(\boldsymbol{x}(k) - \xi(k)\boldsymbol{\Delta}(k))}{2\xi(k)\Delta_i(k)}, \quad i = 1, \ldots, m. \quad (7)$$

SA algorithms that use SP for gradient estimation are called Simultaneous Perturbation Stochastic Approximation (SPSA). As shown in [11], [21], SPSA has significant advantages over SA algorithms employing traditional gradient estimation methods such as Finite Difference (FD).

[5]For instance, some of the control packets may be routed along different paths available between the source and destination nodes.

B. Proposed routing algorithm

The decision variable $\boldsymbol{x}$ in the optimization problem in (4) - (6) is a collection of rate assignments of the sources, i.e., $\boldsymbol{x} = (\boldsymbol{x_s}, s \in \mathcal{S})$, where $\boldsymbol{x_s} = (x^s_{o,d}, 0 \in O^s, d \in D^s)$. Moreover, the constraints given in (5) and (6) comprise separate constraints for each source that are independent of others. Therefore, the problem can be naturally decomposed into several coupled subproblems, one for each source.

Let Θ_s denote the set of feasible rate assignments for source s that satisfy the constraints in (5) - (6) and $\Pi_{\Theta_s}[\boldsymbol{\zeta}]$ the projection of a vector $\boldsymbol{\zeta}$ onto the feasible set Θ_s using the Euclidean norm. We denote the set of links utilized by source s's packets by L^s. Clearly, this set L^s depends on the assumed network model and is given by $\{V^s_o \cup V^o_d : o \in O^s, d \in D^s\}$ for NM-I, $\{V^s_o \cup T^s_o : o \in O^s\}$ for NM-II, and $\{V^s_o \cup \hat{V}^o_d : o \in O^s, d \in D^s\}$ for NM-III.

In order to find a solution to (4) we propose the following SPSA-based algorithm to be run at each source node in a distributed manner: At time $k = 0, 1, \ldots$, each source s updates its rate vector $\boldsymbol{x_s}(k)$ according to

$$\boldsymbol{x_s}(k+1) = \Pi_{\Theta_s}[\boldsymbol{x_s}(k) - a_s(k)\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)] \tag{8}$$

where $a_s(k) > 0$ is the step size, and $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ is an approximation to either the gradient vector $\boldsymbol{\nabla} C_s(k) = (\partial C(\boldsymbol{x}(k))/\partial x^s_{o,d}, o \in O^s, d \in D^s)$ if $C(x(k))$ is differentiable or a subgradient vector $sg(x)$ otherwise [13].

In our proposed algorithm we compute $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ according to

$$\begin{aligned}\hat{g}_{s,i}(k) &= \frac{N_s}{N_s - 1}\frac{y_s(\Pi_\Theta[\boldsymbol{x}(k) + \Xi(k)\boldsymbol{\Delta}(k)]) - y_s(\boldsymbol{x}(k))}{\xi_s(k)\Delta_{s,i}(k)} \\ &= \frac{N_s}{N_s - 1}\frac{(C^+_s(k) + \mu^+_s(k)) - (C^-_s(k) + \mu^-_s(k))}{\xi_s(k)\Delta_{s,i}(k)}, \\ &\qquad i = 1, \ldots, N_s \cdot |D^s|,\end{aligned} \tag{9}$$

where $\boldsymbol{\Delta}(k) = (\boldsymbol{\Delta_s}(k),\ s \in S)$ is an $N \times 1$ vector, $\boldsymbol{\Delta_s}(k)$ is the random perturbation vector generated by source s at iteration k, $\Xi(k)$ is an $N \times N$ diagonal matrix composed of block diagonal entries $\{\Xi_s(k) := \xi_s(k) \cdot I_s, s \in \mathcal{S}\}$ with $\xi_s(k) > 0$ and I_s being the $(N_s \cdot |D^s|) \times (N_s \cdot |D^s|)$ identity matrix. The parameters $\xi_s(k)$, $\boldsymbol{\Delta_s}(k)$ and $a_s(k)$ are pre-programmed at the sources and satisfy assumptions to be stated in the following section. An example of these parameters that satisfy the assumptions is provided in subsection VII-A.

We denote by $y_s(\boldsymbol{x})$ the noisy measurements of the *partial* network cost $\Lambda_s(\boldsymbol{x}) := \sum_{l \in L^s} C_l(x^l)$ obtained with a rate assignment vector $\boldsymbol{x}$. The variables $C^-_s(k)$ and $C^+_s(k)$ denote $\Lambda_s(\boldsymbol{x}(k))$ and $\Lambda_s(\Pi_\Theta[\boldsymbol{x}(k) + \Xi(k)\boldsymbol{\Delta}(k)])$, respectively, and $\mu^+_s(k)$ and $\mu^-_s(k)$ represent the measurement noises due to stochastic nature of traffic and/or potential lack of synchronization of the algorithms at the sources and will be modeled as random variables (rvs). For simplicity of analysis, we assume that the distribution of the noise $\frac{\mu^+_s(k) - \mu^-_s(k)}{\Delta_{s,i}(k)}$ in the approximation is conditionally independent of the perturbation vector $\boldsymbol{\Delta}_s(k)$ selected by the source throughout this paper. In addition, we assume that a source keeps drawing a new perturbation vector until $\Pi_{\Theta_s}[\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)] \neq \boldsymbol{x_s}(k)$. The pseudo code of the proposed algorithm is provided in Table I.

TABLE I

PSEUDO CODE OF THE PROPOSED ALGORITHM

STEP 0: (Initialization)
At each source $s \in S$, set k =1, and initialize $\boldsymbol{x_s}(k)$, $a_s(k)$ and $\xi_s(k)$.

STEP 1: (First measurement)
Obtain a noisy measurement of partial network cost.

$$\mathcal{Y}^-_s := y_s(\boldsymbol{x}(k)) = \sum_{l \in L^s} C_l(x^l) + \mu^-_s(k)$$

STEP 2: (Random perturbation)
Compute a random perturbation vector $\Delta_s(k)$. Let

$$\boldsymbol{x_s}(k) = \Pi_\Theta[\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)]$$

STEP 3: (Second measurement)
Obtain a noisy measurement of partial network cost (after perturbation).

$$\mathcal{Y}^+_s := y_s(\boldsymbol{x}(k)) = \sum_{l \in L^s} C_l(x^l) + \mu^+_s(k)$$

STEP 4: (Estimation of subgradient)
Estimate the subgradient of the cost $C_s(k)$ according to

$$\hat{g}_{s,i}(k) = \frac{N_s}{N_s - 1}\frac{\mathcal{Y}^+_s - \mathcal{Y}^-_s}{\xi_s(k)\Delta_{s,i}(k)}, \quad i = 1, \ldots, N_s \cdot |D^s|,$$

STEP 5: (Update of rate assignment)
Update the rate assignment using the subgradient estimate according to

$$\boldsymbol{x_s}(k+1) = \Pi_{\Theta_s}[\boldsymbol{x_s}(k) - a_s(k)\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)]$$

STEP 6:
$k \leftarrow k + 1$. Go to **STEP 1**.

There are several differences between our proposed algorithm and a standard SPSA algorithm. First, the gradient approximation in (9) differs from the standard SA; each source uses only *partial* cost information (*i.e.,* summation of the costs of the links in L^s) as opposed to the total network cost, which is the summation of the costs of all the links in the network. This is to minimize the communication overhead stemming from the exchange of link cost information to the sources. In addition, the noise terms observed by the sources are allowed to be different. Second, while $\xi(k)$ is a positive scalar in standard SA, in our case $\Xi(k)$ is an $N \times N$ diagonal matrix. This allows the possibility of having different $\xi_s(k)$ values at different sources. Third, there is an extra multiplicative factor $\frac{N_s}{N_s - 1}$ in (9) compared to the standard SA. This is due to the projection of the perturbed rate vector $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ onto the feasible set Θ_s for all $s \in S$ using L_2 projection when calculating $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$. This is explained in more detail in Appendix III.

Our algorithm in (8) does not assume that the sources have the same step sizes $a_s(k)$ at each iteration. This permits a certain level of asynchronous operation among the sources. For example, this allows a scenario where the sources start the algorithm at different times (without losing the convergence property shown in the following section). However, we assume that sources update their rates once every iteration after they start the algorithm. This assumption is reasonable in our case, for at each iteration sources should make use of the collected information that is already available. This is, however, not to say that the updates take place simultaneously. The errors due to this lack of synchronization are assumed to be included in the measurement errors $\mu^\pm_s(k)$ in (9).

V. Convergence of the Proposed Algorithm

In this section we establish the convergence of the proposed algorithm in (8) and (9). We first consider the case where the step sizes $\{a_s(k), k = 1, 2, \ldots\}$ are decreasing and establish the a.s. convergence of the algorithm to a solution of the optimization problem in (4) - (6). Then, we study the case of a fixed or constant step size, i.e., $a_s(k) = a$ for all $s \in \mathcal{S}$ and $k = 0, 1, \ldots$, and demonstrate the weak convergence of the algorithm to a small neighborhood of a solution.

A. Decreasing step size case

In this subsection we establish the a.s. convergence (or convergence w. p. 1) of (8) under all three network models described in Section III.

The following definition is borrowed from convex analysis [13], and further details can be found in [19].

Definition 1: Suppose that $\boldsymbol{h}$ is a real-valued convex function on $\mathbb{R}^r$. A vector $\boldsymbol{sg}(\boldsymbol{x})$ is said to be a subgradient of $\boldsymbol{h}$ at a point $\boldsymbol{x}$ if $\boldsymbol{h}(\boldsymbol{z}) \geq \boldsymbol{h}(\boldsymbol{x}) + (\boldsymbol{z} - \boldsymbol{x})^T \boldsymbol{sg}(\boldsymbol{x})$ for all $\boldsymbol{z} \in \mathbb{R}^r$. The set of all subgradients of $\boldsymbol{h}$ at $\boldsymbol{x}$ is called the subdifferential of $\boldsymbol{h}$ at $\boldsymbol{x}$ and is denoted by $\partial \boldsymbol{h}(\boldsymbol{x})$.

In order to establish the convergence of our algorithm based on SPSA, we assume that the following conditions hold:

A1. The link cost functions $C_l\left(x^l\right)$ are convex for all $l \in L$, but are not necessarily differentiable. The subdifferential of C at $\boldsymbol{x}$ is denoted by $\partial C(\boldsymbol{x})$ and is bounded for all $\boldsymbol{x} \in \Theta$, where Θ is the feasible set of $\boldsymbol{x}$, i.e., $\boldsymbol{x_s} \in \Theta_s$ for all $s \in \mathcal{S}$.

A2. The perturbation terms $\Delta_{s,i}(k)$ are (i) mutually independent with zero mean for all $s \in S$ and $i \in \{1, 2, \cdots, N_s \cdot |D^s|\}$, (ii) uniformly bounded by some constant $\alpha < \infty$ with support on finite discrete sets, (iii) independent of $(\boldsymbol{x}(n), n = 0, 1, \cdots, k)$, and (iv) $\mathbf{E}\left[(\Delta_{s,i}(k))^{-1}\right]$, $\mathbf{E}\left[(\Delta_{s,i}(k))^{-2}\right]$ are bounded for all k.

A3. $\mathbf{E}\left[{\mu_s^{(\pm)}}^2(k)|\boldsymbol{\Delta}(k), \mathcal{F}_k\right]$ are bounded and $\mathbf{E}\left[\mu_s^+(k) - \mu_s^-(k)|\boldsymbol{\Delta}(k), \mathcal{F}_k\right] = 0$ a.s. for all k, where $\mathcal{F}_k$ is the σ-field generated by $\{\boldsymbol{x}(0), \cdots, \boldsymbol{x}(k)\}$.

A4. (i) $\sum_{k=0}^{\infty} a_s(k) = \infty$, (ii) $a_s(k) \to 0$ as $k \to \infty$, (iii) $\sum_{k=0}^{\infty} \left(\frac{a_s(k)}{\xi_s(k)}\right)^2 < \infty$, (iv) $\xi_s(k) \to 0$ as $k \to \infty$, and (v) $\lim_{k\to\infty} \left(\frac{\xi_s(k)}{\xi_{s'}(k)}\right) = 1$ for all $s, s' \in \mathcal{S}$.

A5. Define $\hat{a}(k) := \max_{s\in S} a_s(k)$. (i) $\sum_{k=0}^{\infty}(\hat{a}(k) - a_s(k)) < \infty$ for all $s \in \mathcal{S}$, and (ii) $\lim_{k\to\infty} \frac{a_s(k)}{\hat{a}(k)} = 1$.

The main result of this subsection is given by the following theorem.

Theorem 1: Under Assumptions A1 - A5, the sequence $\boldsymbol{x}(k) = (\boldsymbol{x_s}(k),\, s \in S)$ generated by the algorithm given by (8) converges to a point in the set of solutions of (4) - (6) w. p. 1 under each of the three network models with link loads defined by (1) - (3), starting from any initial rate assignment $(\boldsymbol{x_s}(0), s \in \mathcal{S}) \in \Theta$.

Proof: We provide a proof only under NM-I with link loads given by (1) in Appendix I. The proof for the other two models follows from the fact that the necessary convexity of the objective function can be established in a similar manner. ■

One thing to note here is that the proposed algorithm does not require any modifications for its convergence under different network models. This allows us to compare different network models using the same optimal routing algorithm and quantify the benefits obtained from increasing multicast capabilities. For the same reason the underlying IP network can be upgraded gradually without requiring any changes to our algorithm.

1) A property of NM-II: As mentioned in subsection III-B, under NM-II all destinations receive packets at the same rate $\max_{d\in D^s} x_{o,d}^s = x_o^s$ from overlay node $o \in O^s$ because the multicast routers are assumed to be capable only of copying and forwarding packets. Based on this observation one can trivially prove the following corollary.

Corollary 1: Let $\boldsymbol{x}^\star$ be a feasible solution of (4) to which the a.s. convergence in Theorem 1 takes place under NM-II with link loads defined by (2). Then, for all $s \in \mathcal{S}$ and $o \in O^s$, we have

$$x_{o,d}^{s\star} = x_o^{s\star} \quad \text{for all} \quad d \in D^s.$$

This simple result tells us that under NM-II the optimization problem in (4) - (6) can be reduced to that of finding the optimal rate assignments $(x_o^{s\star}, s \in \mathcal{S}, o \in O^s)$ to the overlay nodes. This is because the rates to individual receivers from an overlay node are the same as the rate from the source node to the overlay node. Therefore, the optimization problem in (4) - (6) can be rewritten as the following simpler optimization problem under NM-II.

$$\min_{\boldsymbol{x}} C(\boldsymbol{x}) = \min_{\boldsymbol{x}} \sum_{l\in\mathcal{L}} C_l(x^l) \tag{10}$$

$$\text{s. t.} \sum_{o\in O_s} x_o^s = r^s, \ \forall\ s \in S \tag{11}$$

where, with a little abuse of notation, $\boldsymbol{x} = (x_o^s, s \in \mathcal{S}, o \in O^s)$. When the number of receivers is large, this leads to much lower computational requirement at the sources and faster convergence as will be demonstrated in Section VII.

Note that in (11) the term ϵ^s is removed from (5). This is due to the fact that, at a feasible solution, a source node can deliver packets to the overlay nodes that simply forward the packets to all destinations. As a result, there is no issue of bookkeeping at the source and ϵ^s can be set to zero. We refer to this variation of NM-II as NM-IIb in the rest of the paper.

B. Constant step size case

In the previous subsection we have assumed decreasing step sizes $\{a_s(k), k = 1, 2, \ldots\}$ that satisfy the conditions in Assumption A4 and A5. Although it is possible to adopt decreasing step sizes at the sources, a constant step size may be preferred for practical reasons. For instance, when decreasing step sizes are employed, once the step sizes become small, the response of the algorithm to a sudden network state change becomes slow, which is undesirable, and the step sizes need to be reset. This will require additional mechanism and decision maker that monitor the network for any significant change and reset the step sizes at the sources when necessary.

In this subsection we consider the case where the step sizes at the sources are fixed at$a_s(k) = a > 0$ and $\xi_s(k) = \xi$ for all $s \in \mathcal{S}$ and $k = 1, 2, \ldots$. Throughout this subsection we assume that $\xi := \frac{a \cdot \nu}{2\alpha}$, where α is the uniform bound on the perturbation terms $\Delta_{s,i}$ in Assumption A2. Although we are not able to establish the same a.s. convergence to a solution with constant step sizes, as shown in [15] under certain conditions constant step size SA algorithms can achieve weak convergence to a neighborhood of the solution set. Since the performance near the set of solutions is comparable to that of a solution in our problem, a constant step size policy is expected to perform reasonably well, which is supported by our simulation results in Section VII.

In this subsection we are interested in the system behavior when the step size is sufficiently small. To this end, we consider the following parametric scenario: Let $\{a_n, n = 1, 2, \ldots\}$ and $\{q_n, n = 1, 2, \ldots\}$ be a decreasing sequence of positive real numbers with $a_n \to 0$ and a nondecreasing sequence of non-negative integers, respectively.

Fix $n = 1, 2, \ldots$. We rewrite the update rule in (8) for all sources in the following compact form:

$$\begin{aligned} \boldsymbol{x}^n(k+1) &= \Pi_\Theta[\boldsymbol{x}^n(k) - a_n \hat{\boldsymbol{g}}^{\boldsymbol{n}}(k)] \\ &= \boldsymbol{x}^n(k) - a_n \hat{\boldsymbol{g}}^{\boldsymbol{n}}(k) + \boldsymbol{\upsilon}^n(k) \ , \end{aligned}$$

where $\boldsymbol{\upsilon}^n(k)$ is the reflection term to keep $\boldsymbol{x}^n(k+1)$ in the feasible set Θ.[6]

Let us define the following piece-wise constant continuous-time processes:

$$\boldsymbol{X}^n(t) = \begin{cases} a_n \sum_{i=q_n}^{\lfloor t/a_n \rfloor + q_n - 1} \boldsymbol{x}^n(i) \ , & \text{for } t \geq 0 \\ -a_n \sum_{i=\lfloor t/a_n \rfloor + q_n}^{q_n - 1} \boldsymbol{x}^n(i) \ , & \text{for } t < 0 \end{cases}$$

The process $\boldsymbol{\Upsilon}^n(t)$ is defined similarly with $\boldsymbol{\upsilon}^n(k)$ in place of $\boldsymbol{x}^n(k)$.

Before we state the main result of this subsection we impose the following additional assumption on $\hat{\boldsymbol{g}}(k), k = 0, 1, \ldots$.

A6. $\{\hat{\boldsymbol{g}}^n(k); n, k\}$ is uniformly integrable.

The main result of this subsection is now given by the following theorem.

Theorem 2: Suppose that Assumptions A1-A3 and A6 hold. Then, for each subsequence of $\{\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot), \quad n \in \mathbb{N} := \{1, 2, \ldots\}\}$, there exists a further subsequence $\{\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot), \quad n \in K\}$ and a process $(\boldsymbol{X}(\cdot), \boldsymbol{\Upsilon}(\cdot))$ such that the following distributional convergence (denoted by $\Rightarrow_n$) takes place:

$$(\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot)) \Rightarrow_n (\boldsymbol{X}(\cdot), \boldsymbol{\Upsilon}(\cdot)) \ ,$$

where $\dot{\boldsymbol{X}} \in \partial C(\boldsymbol{X}) + \boldsymbol{\Upsilon}, \quad \boldsymbol{\Upsilon} \in -V(\boldsymbol{X}(t))$ for almost all ω (in the sample space Ω of the underlying probability space) and t. Here $V(\boldsymbol{x})$ denotes the convex cone generated by the set of outward normals defined in Appendix I.

Furthermore, for any $\delta > 0$, the fraction of time that $\boldsymbol{X}^n(a_n q_n + \cdot)$ spends in $N_\delta(S_\Theta)$ on [0, T] goes to one (in probability) as $n \to 0$ and $T \to \infty$, where S_Θ is the set of stationary points defined in Appendix I and $N_\delta(S_\Theta) := \{\boldsymbol{x} \in \Theta : \inf_{\boldsymbol{x}^\star \in S_\Theta} ||\boldsymbol{x} - \boldsymbol{x}^\star|| \leq \delta\}$.

Proof: The proof of Theorem 2 is given in Appendix II. ■

Theorem 2 states that for sufficiently small step sizes a, as $k \to \infty$, the rate assignment $\boldsymbol{x}(k)$ oscillates around a small neighborhood of the set of optimal solutions.

VI. Implementation Issues

A. Traffic Monitoring

In our framework, traffic monitoring is handled by an overlay architecture [14]: Each link in the network is mapped to the closest overlay node with a tie-breaking rule that gives a unique mapping. Each overlay node periodically polls the links that it is responsible for, processes the data and forwards necessary link cost information to the sources that utilize the links it manages. This eliminates the need for each source to probe the links it uses separately. Moreover, this makes it possible for the overlay nodes to consolidate the link cost information before forwarding it to the sources; the cost function in (4) is the sum of link cost functions. Thus, if the overlay nodes have the knowledge of the set L^s of the links used by each source s, each overlay node can first compute the *sum* of the link costs over the links in L^s it handles and report only the sum to the source. This will consequently minimize the overhead introduced by collection and dissemination of the link state information. In Section VII, we compute an upper bound on the required overhead for network monitoring and show that it is minimal compared to network capacity.

B. Traffic Filtering

In a scenario where a multicast source does not employ Digital Fountain coding, (e.g., NM-IIb algorithm) one has to give special care while splitting the traffic at the source node to avoid the well-known reordering problem, especially for the TCP traffic. The algorithm proposed in this paper calculates the rates at which traffic should be distributed among the alternative paths without requiring or specifying the exact paths that a particular packet should follow. Therefore, any existing filtering scheme that minimizes the reordering problem can be used for this purpose. A possible solution based on hash functions is presented in [7].

VII. Simulation Results

In this section we evaluate the performance of our proposed algorithm and validate its convergence. In addition, we compare the performance of our algorithm under the three network models to quantify the benefits with increasing level of intelligence at the routers, as well as that of DVMRP for comparison.

We developed a packet level discrete-event simulator for our study. For the optimization problem, the link cost function is selected to be $\left(x^l/c^l\right)^2$, where c^l is the link capacity and x^l is the link rate as defined in Section III. This cost function is employed for its simplicity and ease of visualization. However, any other cost function that satisfies Assumption A1 can be used. In all simulations, the measurement or sampling period is set to one second. Therefore, source nodes can update their rates at most approximately every two seconds since

[6]We use a superscript n to denote the dependency on the step size a_n.

two measurements are needed for estimating the (sub)gradient vector according to (9). For simplicity we set the source coding overhead/redundancy ϵ^s to zero. Experiments are conducted with an intradomain network topology shown in Fig. 5. This is a close approximation of Sprint backbone topology as reported in [22].

Each link has a capacity of 20 Mbps. Packet size is selected to be 500 bytes. There are three multicast sources and $\mathcal{S} = \{1, 9, 22\}$. Each source has 18 receivers. The receiver sets are given by D^1 = {2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 15, 16, 19, 20, 21, 22, 23, 25}, D^9 = {1, 2, 3, 4, 6, 7, 8, 10, 11, 13, 15, 16, 17, 18, 21, 22, 23, 24} and D^{22} = {1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 15, 16, 17, 20, 21, 23, 26}. Nodes 10 and 23 are selected as additional core overlay nodes. This gives $O^1 = \{1, 10, 23\}$, $O^9 = \{9, 10, 23\}$ and $O^{22} = \{22, 10, 23\}$, and each source-destination pair has three paths[7]. Each source generates Poisson traffic with an average rate of 10 Mbps.

Under all network models the rates along alternate routes through core overlay nodes are initially set to zero, i.e., $x^s_{o,d}(0) = 0$ if $o \neq s$ and $x^s_{s,d}(0) = r^s$. Hence, under NM-I model, the algorithm starts with traditional unicast routing to each destination, while under NM-II, NM-IIb and NM-III all the traffic is initially routed through the multicast tree rooted at the source and is gradually shifted to alternative trees rooted at core overlay nodes 10 and 23 when desired.

Throughout this section we only plot a single sample path out of ten independent runs starting with different random seeds. The other sample paths are similar. We also compute the optimal cost values under NM-II and NM-III, using a MATLAB optimization package. The optimal values under NM-II and NM-III are 12.5875 and 11.7612, respectively. As expected the optimal value under NM-III is smaller than under NM-II due to assumed additional capabilities of the routers. However, interestingly the difference is rather small in this example. Hence, the additional complexity of having smart routers capable of forwarding packets onto each branch at a different rate offers only a marginal benefit in this case[8].

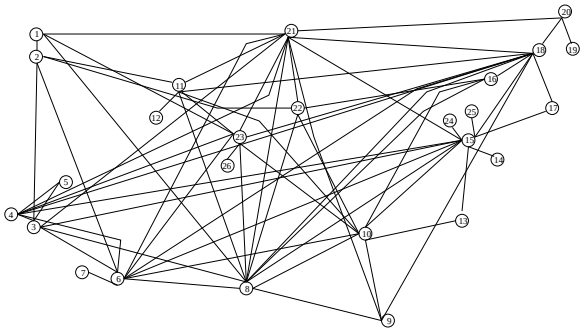

Fig. 5. Network topology 2

Let us briefly comment on the required overhead for this example. There are 136 unidirectional links with a capacity of 20 Mbps. Suppose that the number of links assigned to each overlay node is roughly the same. Since there are 5 overlay nodes – 3 sources and 2 core overlay nodes, each overlay node manages approximately 20 to 50 links. As link cost information is consolidated by the overlay nodes and only one packet is sent by an overlay node to each source, the traffic carrying the cost information from the overlay nodes to the sources will be minimal. Moreover, it is reasonable to assume that the concentration of packets carrying link cost measurements tends to increase towards the overlay nodes, causing higher overhead on the links around them. We consider the worst scenario where all the measurement packets for an overlay node arrive on the same link. In this case, assuming a measurement packet size of 100 bytes and taking the upper end of 50 links per overlay node, the total traffic due to the measurement packets on an incoming link is approximately 50 links $\times$ 1 measurement/link/second $\times$ 800 bits/measurement or 0.2 percent of the link capacity. Hence, the overhead due to the measurements is not significant.

A. Decreasing step size

In this subsection we adopt a decreasing step size policy, whose parameters satisfy Assumptions A3-A5. In particular, we select $a_s(k) = 20/(k+100)^{0.602}$ and $\xi_s(k) = 50/(k^{0.101})$. Fig. 6 shows the evolution of network cost and packet loss rate. The perturbation terms $\Delta_{s,i}(k)$ are given by mutually independent Bernoulli random variables that take on values 1 and -1 with equal probability, i.e., $P(\Delta_{s,i}(k) = 1) = P(\Delta_{s,i}(k) = -1) = 0.5$.

It is clear from the achieved network cost and packet loss rate that our algorithm considerably outperforms DVMRP under both NM-II and NM-III by distributing the traffic among three available multicast trees. However, as shown in the figures, while our algorithm can reduce the network cost somewhat under NM-I, it cannot eliminate packet losses and incurs a higher cost than DVMRP. This poor performance is due to the lack of multicast functionality; since we cannot create multicast trees, independent unicast sessions are set up to each destination from the sources and overlay nodes, ignoring the multicast nature of the traffic. This requires generating a separate copy of the same packet for each destination, even when these packets traverse many common links, and leads to severe link stress in the network. However, Fig. 6 suggests that NM-I still outperforms the traditional single path unicast routing.

From Fig. 6(b), we find that the single path routing experiences a packet loss rate of 70,000 packets/sec, roughly 52 percent of all packets, whereas NM-I reduces it to nearly 30,000 packets/sec, approximately 22 percent of all packets. Computing the fraction of packets dropped under NM-II and NM-III is much more difficult because multiple copies of packets can be generated from a single copy of packets at intermediate nodes.

Fig. 6 shows that our algorithm converges faster under NM-IIb than under any other model. This is due to the fact that we only need to optimize the overlay rates x^s_o instead of individual receiver rates $x^s_{o,d}$. Hence, the number of parameters that need to be updated in each iteration is much smaller than the other cases (9 versus 162). Finally, we see that three paths

[7]Since both of the core nodes are also destination nodes for all three sources, under NM-I there are only two paths from the sources to these nodes.

[8]This is not to say that the same holds in all cases as the difference depends on the topology and source-destination pair selections as well as their traffic demand. Indeed, we suspect that when there is a severe bottleneck along a path from an overlay node to a destination that can be congested under NM-II, then the optimal cost under NM-II can be larger than that under NM-III.

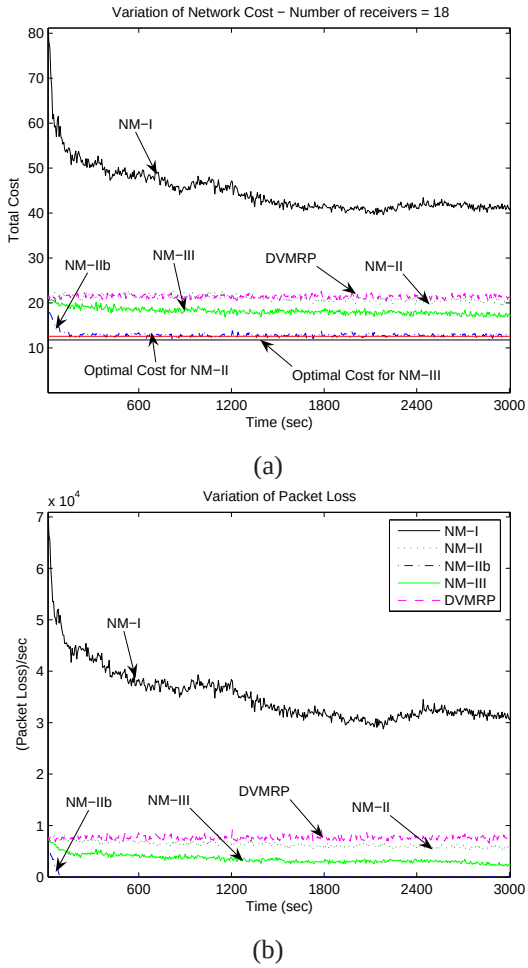

Fig. 6. Network costs and packet losses - Poisson traffic source. (a) Network cost, (b) packet losses.

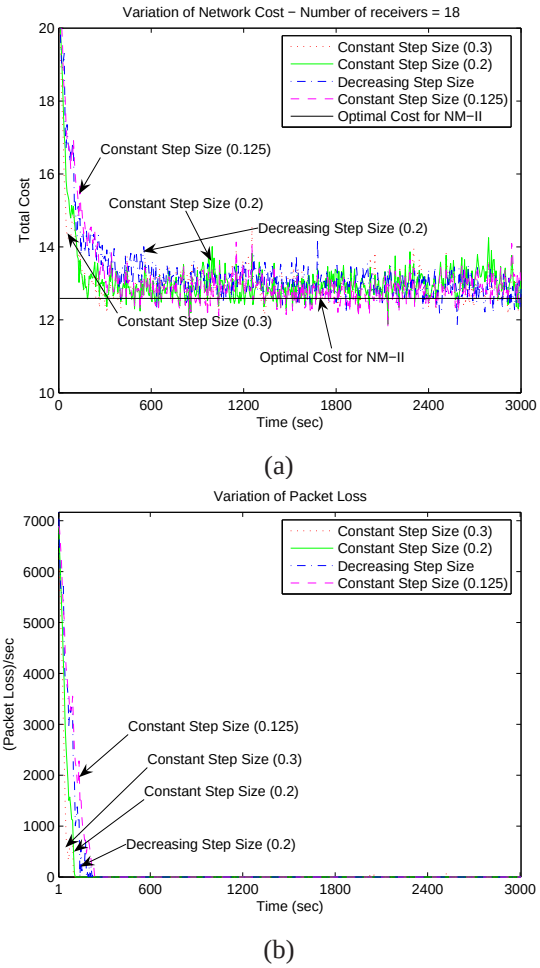

Fig. 7. Network cost and packet drops with fixed step sizes and a decreasing step size.

per receiver is sufficient to successfully distribute the traffic even when the DVMRP algorithm cannot eliminate the packet losses.

B. Constant step size

In this subsection we fix the step size $a_s(k) = a$ for all $s \in \mathcal{S}$ and $k = 0, 1, \ldots$, and compare the system performance for different values of a and against the decreasing step size case. Fig. 7 plots the evolution of network cost and packet losses under NM-IIb for different values of step sizes. The values in parentheses denote the step size. The figure clearly shows that when the step size is fixed the network cost oscillates slightly above the optimal value. Moreover, the initial convergence rate is better for several values of the fixed step size as the algorithm is able to reach a small neighborhood of the optimal operating points faster than with decreasing step sizes. Since this neighborhood is small and a constant step size policy is more robust in the presence of network fluctuations and can track dynamical changes, this result suggests that the algorithm with a well chosen fixed step size may be preferred in practice.

VIII. Conclusion

We studied the issue of multi-path routing of both unicast and multicast traffic where the gradient of network cost is not available and needs to be estimated using noisy measurements. We proposed an optimal routing algorithm based on the idea of simultaneous perturbation stochastic approximation with provable convergence properties. Three different network models (NM-I, II, and III) were considered to evaluate its performance and to quantify the benefits of additional functionality/intelligence in the underlying IP network. In NM-III we established a routing framework that generalizes the multiple distribution trees to a more general multiple path scenario where each destination can receive packets at a different rate from a multicast tree. The need for complicated bookkeeping at the sources and intermediate IP routers is handled by employing Digital Fountain coding. We proved the convergence properties of the proposed algorithm both with decreasing step sizes and with a fixed step size. Further, our simulation studies show that while basic IP multicast functionality in NM-II is crucial for better performance, additional functionalities introduced in NM-III provide only marginal benefits in relation to required complexity.

References

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. RFC 3272, 2002.

[2] D. Bertsekas, A. Nedic, and A.E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[3] J. R. Blum. Multidimensional stochastic approximation methods. *Ann. Math. Stat.*, 25:737–744, 1954.

[4] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *ACM SIGCOMM*, 2002.

[5] C. G. Cassandras, M. Vasmi Abidi, and D. Towsley. Distributed routing with on-line marginal delay estimation. *IEEE Transactions on Communications*, 38(3):348–359, 1990.

[6] Y. Chu, S. G. Rao, and Hui Zhang. A case for end system multicast. In *ACM SIGMETRICS*, 2000.

[7] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Anchorage, Alaska, April 2001.

[8] G. B. Folland. *Real Analysis: Modern Techniques and Their Applications*. John Wiley & Sons, 1984.

[9] R. G. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, 25(1):73–85, 1977.

[10] G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford Science Publications, 2nd edition, 1998.

[11] T. Güven, C. Kommareddy, R. J. La, M. A. Shayman, and B. Bhattacharjee. Measurement based optimal multi-path routing. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Hong Kong, March 2004.

[12] T. Guven, R. J. La, M. A. Shayman, and B. Bhattacharjee. Measurement based multipath multicast. In *IEEE Infocom Global Internet Symposium (GI)*, March 2005.

[13] Y. He, M. C. Fu, and S. I. Marcus. Convergence of simultaneous perturbation stochastic approximation for nondifferentiable optimization. *IEEE Transactions on Automatic Control*, 48:1459–1463, 2003.

[14] C. Kommareddy, T. Güven, B. Bhattacharjee, R. J. La, and M. A. Shayman. Intradomain overlays: Architecture and applications. Tech. Rep. UMIACS-TR# 2003-70, available at http://www.cs.umd.edu/Library/TRs/CS-TR-4501/CS-TR-4501.pdf.

[15] H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications, 2nd ed.* Springer-Verlag, 2003.

[16] D. J. C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[17] T. Noguchi, T. Matsuda, and M. Yamamoto. Performance evaluation of new multicast architecture with network coding. *IEICE Trans. Commun*, E86-B:1788–1795, 2003.

[18] K. Park and Y. Shin. Uncapacitated point-to-multipoint network flow problem and its application to multicasting in telecommunication networks. *European Journal of Operational Research*, 147:405–417, 2003.

[19] R. T. Rockafellar. *Convex Analysis*. Princeton, NJ: Princeton Univ. Press, 1970.

[20] A. Shokrollahi. Raptor codes. preprint 2003.

[21] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automat. Contr.*, 37:332–341, 1992.

[22] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.

[23] D. Waitzman, C. Partridge, and S. Steering. Distance vector multicast routing protocol. RFC 1075, 1998.

[24] Y. Zhu, B. Li, and J. Guo. Multicast with network coding in application-layer overlay networks. *IEEE Journal on Selected Areas in Communications*, 22:107–120, 2004.

APPENDIX I
PROOF OF THEOREM 1

The basic approach we follow in the proof is similar to that in [15, pp. 127-131] and [11]. A main difference between the proof in [11] and the current proof stems from the fact that the cost function is no longer continuously differentiable with respect to the rate assignment although the convexity is preserved. This is because of the max operator in the link cost functions in (1) - (3). Here we will show that the same convergence holds even when the cost function is not differentiable, borrowing tools from the convex analysis and the concept of subgradient.

Collecting the terms of (8) for all sources, we have the following update rule for the system:

$$\boldsymbol{x}(k+1) = \Pi_\Theta[\boldsymbol{x}(k) - \boldsymbol{A}(k)\hat{\boldsymbol{g}}(k)] \ , \tag{12}$$

where $\hat{\boldsymbol{g}}(k) := (\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k), s \in \mathcal{S})$, $\boldsymbol{A}(k)$ is an $N \times N$ diagonal matrix with the diagonal entries $\boldsymbol{A}_{ii}(k)$ given by the step sizes $a_s(k)$ of corresponding sources. This is different from the standard stochastic approximation algorithms that have a scalar step size.

In the rest of the proof we closely follow the same outline in [15] with the gradient $\boldsymbol{\nabla} C(\boldsymbol{x})$ replaced by a subgradient $\boldsymbol{sg}(\boldsymbol{x})$. Unless stated otherwise $\boldsymbol{E}_k(\cdot)$ represents $\mathbf{E}[\cdot|\mathcal{F}_k]$. Rewrite (12) in the following form

$$\begin{aligned}
&\boldsymbol{x}(k+1) = \Pi_\Theta[\boldsymbol{x}(k) - \boldsymbol{A}(k)\hat{\boldsymbol{g}}(k)] \\
&= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k), \\
&= \boldsymbol{v}(k) + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)
\end{aligned} \tag{13}$$

where

$$\begin{aligned}
\boldsymbol{v}(k) &= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)], \\
\boldsymbol{\varrho}(k) &= \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)) - \hat{\boldsymbol{g}}(k), \ \boldsymbol{b}(k) = \boldsymbol{sg}(\boldsymbol{x}(k)) - \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)), \\
\boldsymbol{\tau}(k) &= \Pi_\Theta[\boldsymbol{v}^\gamma(k)] - \boldsymbol{v}^\gamma(k), \\
\boldsymbol{v}^\gamma(k) &= \boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k) + \boldsymbol{b}(k)]I(k), \\
\boldsymbol{\phi}(k) &= \boldsymbol{v}^\gamma(k) - \boldsymbol{v}(k) + \big(\Pi_\Theta[\boldsymbol{v}(k)] - \boldsymbol{x}(k)\big)\big(1 - I(k)\big),
\end{aligned}$$

$I(k)$ is an indicator function of the event $\{||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \leq \gamma(k)/2\}$, and $\{\gamma(k), k = 0, 1, \ldots\}$ is a sequence of positive real numbers such that (i) $\gamma(k) \to 0$ and (ii) $||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)|| \leq \gamma(k)/2$ for all but a finite number of k w. p. 1. The following lemma is used to show the existence of such a sequence.

Lemma 1: Under Assumptions A1 - A5, the bias and error terms $\boldsymbol{b}(k)$ and $\boldsymbol{\varrho}(k)$, respectively, satisfy the following:

(a) $\boldsymbol{b}(k) \to 0$ w. p. 1 , and

(b) $\sum_{k=0}^{\infty} \boldsymbol{E}_k(||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)||^2) < \infty$ w. p. 1.

Proof: The proof is provided in Appendix III. ■

Now we prove the existence of the sequence $\{\gamma(k), k = 0, 1, \ldots\}$ that satisfies the given conditions. Since $\sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i)$, $j \geq k$, is a martingale and $\sum_{i=k}^{\infty} \boldsymbol{E}_i(||\boldsymbol{A}(i)\boldsymbol{\varrho}(i)||^2) < \infty$ from Lemma 1, by the Doob-Kolmogorov inequality ([10, p. 309]), for every $\epsilon > 0$, we have

$$P\Big(\sup_{j \geq k} \Big|\Big| \sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i) \Big|\Big| \geq \epsilon\Big) \leq \frac{1}{\epsilon^2} \sum_{i=k}^{\infty} \boldsymbol{E}_i(||\boldsymbol{A}(i)\boldsymbol{\varrho}(i)||^2) \ .$$

Since the right-hand side of the equation goes to zero as $k \to \infty$, it follows that

$$\lim_{k \to \infty} P\Big(\sup_{j \geq k} \Big|\Big| \sum_{i=k}^{j} \boldsymbol{A}(i)\boldsymbol{\varrho}(i) \Big|\Big| \geq \epsilon\Big) = 0 \ ,$$

and the existence of the sequence $\{\gamma(k), k = 0, 1, \ldots\}$ is guaranteed.

Let $\boldsymbol{X^0}(t)$ be the following continuous time interpolation of $\boldsymbol{x}(k)$.

$$\boldsymbol{X^0}(t) := \frac{t_{k+1} - t}{\hat{a}(k)} \boldsymbol{x}(k) + \frac{t - t_k}{\hat{a}(k)} \boldsymbol{x}(k+1) \ , \ t \in [t_k, t_{k+1}) \tag{14}$$

where $t_k = \sum_{i=0}^{k-1} \hat{a}(i)$.[9] We will show that the trajectory of this continuous time process follows that of the differential inclusion to be defined shortly, and the path of this differential inclusion will be shown to converge to a point in the set of solutions of the optimization problem we consider. Let

$$\boldsymbol{B^0}(t_k) := \sum_{i=0}^{k-1} \boldsymbol{A}(i)\boldsymbol{b}(i), \quad \boldsymbol{M^0}(t_k) := \sum_{i=0}^{k-1} \boldsymbol{A}(i)\boldsymbol{\varrho}(i),$$
$$\mathbf{T^0}(t_k) := \sum_{i=0}^{k-1} \boldsymbol{\tau}(i), \qquad \boldsymbol{\Phi^0}(t_k) := \sum_{i=0}^{k-1} \boldsymbol{\phi}(i)$$

and the corresponding piecewise linear interpolations for $t \in (t_k, t_{k+1})$ are defined similarly as in (14).

We rewrite $\boldsymbol{X^0}(t)$ using the relation (13) as follows.

$$\begin{aligned}
&\boldsymbol{X^0}(t)\\
&= \frac{t_{k+1}-t}{\hat{a}(k)}\boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\Big(\boldsymbol{x}(k) + \boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k))\\
&\qquad +\boldsymbol{\varrho}(k) + \boldsymbol{b}(k)] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\Big)\\
&= \boldsymbol{x}(k) + \frac{t-t_k}{\hat{a}(k)}\Big(\boldsymbol{A}(k)\big[-\boldsymbol{sg}(\boldsymbol{x}(k))\\
&\qquad +\boldsymbol{\varrho}(k) + \boldsymbol{b}(k)\big] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\Big)\\
&= \boldsymbol{x}(0) + \sum_{i=0}^{k-1}\Big(\boldsymbol{A}(i)[-\boldsymbol{sg}(\boldsymbol{x}(i)) + \boldsymbol{b}(i) + \boldsymbol{\varrho}(i)]\\
&\quad +\boldsymbol{\tau}(i) + \boldsymbol{\phi}(i)\Big) + \frac{t-t_k}{\hat{a}(k)}\Big(\boldsymbol{A}(k)[-\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{\varrho}(k)\\
&\quad +\boldsymbol{b}(k)] + \boldsymbol{\tau}(k) + \boldsymbol{\phi}(k)\Big)\\
&= \boldsymbol{x}(0) + \boldsymbol{B^0}(t) + \boldsymbol{M^0}(t) - \sum_{i=0}^{k-1}\hat{a}(i)\boldsymbol{sg}(\boldsymbol{x}(i))\\
&\quad -(t-t_k)\boldsymbol{sg}(\boldsymbol{x}(k)) + \boldsymbol{Z^0}(t) + \mathbf{T^0}(t) + \boldsymbol{\Phi^0}(t)\\
&= \boldsymbol{x}(0) + \boldsymbol{B^0}(t) + \boldsymbol{M^0}(t) + \boldsymbol{H^0}(t) + \boldsymbol{Z^0}(t)\\
&\quad +\mathbf{T^0}(t) + \boldsymbol{\Phi^0}(t)\ ,
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{Z^0}(t) &= \sum_{i=0}^{k-1}\big(\hat{a}(i) - \boldsymbol{A}(i)\big)\boldsymbol{sg}(\boldsymbol{x}(i))\\
&\quad +\frac{t-t_k}{\hat{a}(k)}\big(\hat{a}(i) - \boldsymbol{A}(i)\big)\boldsymbol{sg}(\boldsymbol{x}(k))\ ,\\
\boldsymbol{H^0}(t) &= -\int_0^t \boldsymbol{sg}(\bar{\boldsymbol{x}}(s))ds\ ,\ \text{and}\\
\bar{\boldsymbol{x}}(s) &= \boldsymbol{x}(k)\ ,\ s \in [t_k, t_{k+1})\ .
\end{aligned}$$

We are interested in the tail properties of the interpolated processes. To this end we introduce the following time shifted and centered processes.

$$\begin{aligned}
\boldsymbol{X^k}(t) \quad &:= \quad \boldsymbol{x}(k) + \boldsymbol{B^k}(t) + \boldsymbol{M^k}(t) + \boldsymbol{H^k}(t) + \boldsymbol{Z^k}(t)\\
&\quad +\mathbf{T^k}(t) + \boldsymbol{\Phi^k}(t)\ , \qquad (15)
\end{aligned}$$

where $\boldsymbol{B^k}(t) = \boldsymbol{B^0}(t_k + t) - \boldsymbol{B^0}(t_k)$, $\boldsymbol{M^k}(t)$, $\mathbf{T^k}(t)$, $\boldsymbol{\Phi^k}(t)$ and $\boldsymbol{Z^k}(t)$ are defined similarly, and $\boldsymbol{H^k}(t) = -\int_0^t \boldsymbol{sg}(\bar{\boldsymbol{x}}(t_k + s))ds$.

[9] Recall that $\hat{a}(k) = \max_{s\in\mathcal{S}} a_s(k)$.

We now show that every process on the right hand of (15) is equicontinuous and hence $\boldsymbol{X^k}(t)$ is also equicontinuous. First, from Lemma 1 and the definition of $\{I(k), k = 0, 1, \ldots\}$ one can show that there is a null set Ω_0 such that for each outcome $\omega \notin \Omega_0$, the processes $\boldsymbol{B^k}(t)$, $\boldsymbol{M^k}(t)$ and $\boldsymbol{\Phi^k}(t)$ converge to zero uniformly on finite intervals as $k \to \infty$. Second, under Assumptions A5 and A1 (in particular, $\partial C(\boldsymbol{x})$ is bounded for all $\boldsymbol{x} \in \Theta$, which implies that there exists a finite bound B on $\boldsymbol{sg}(\boldsymbol{x}(k))$) it is easy to verify that $\boldsymbol{Z^k}(t) \to 0$ as $k \to \infty$. Hence, $\boldsymbol{B^k}(t)$, $\boldsymbol{M^k}(t)$, $\boldsymbol{\Phi^k}(t)$, and $\boldsymbol{Z^k}(t)$ converge to zero uniformly on each bounded interval in $(-\infty, \infty)$ as $k \to \infty$, and thus are equicontinuous [15, p. 101]. The equicontinuity of $\boldsymbol{H^k}(t)$ follows from the same argument in the proof of Proposition 1 in [13] because of the existence of a finite bound B on $\boldsymbol{sg}(\boldsymbol{x})$ mentioned before. The equicontinuity of $\mathbf{T^k}(t)$ can shown by the same argument in the proof of Theorem 5.2.1 in [15, p. 128].

Since the processes on the right-hand side of (15) are equicontinuous, so is $\boldsymbol{X^k}(t)$. Therefore, Arzela-Ascoli Theorem [8, p. 13] tells us that, for every $\omega \notin \Omega_0$, there exists a subsequence $\{k_j, j = 1, 2, \ldots\}$ such that $\{\boldsymbol{X}^{k_j}(\omega, .), \boldsymbol{H}^{k_j}(\omega, .), \mathbf{T}^{k_j}(\omega, .)\}$ converges to some $\{\boldsymbol{X}(\omega, .), \boldsymbol{H}(\omega, .), \mathbf{T}(\omega, .)\}$ uniformly and, following the same argument in the proof of Proposition 1 in [13], we can write

$$\boldsymbol{X}(\omega, t) = \boldsymbol{X}(\omega, 0) + \boldsymbol{H}(\omega, t) + \mathbf{T}(\omega, t)\ . \qquad (16)$$

The second term on the right-hand side $\boldsymbol{H}(\omega, t) = \int_0^t \tilde{\boldsymbol{h}}(\omega, s)ds$, where $\tilde{\boldsymbol{h}}(\omega, s) \in -\partial C(\boldsymbol{X}(\omega, s))$ [13]. The third term $\boldsymbol{T}(\omega, t) = \int_0^t \tilde{\boldsymbol{\tau}}(\omega, s)ds$ with $\tilde{\boldsymbol{\tau}}(\omega, s) \in -V(\boldsymbol{X}(\omega, s))$ for almost all s [15, pp. 128-129], where $V(\boldsymbol{x})$ is the convex cone generated by the set of outward normals $\{\boldsymbol{y} : \boldsymbol{y} = \boldsymbol{\nabla q_i}(\boldsymbol{x}), \text{s.t.}\, \boldsymbol{q_i}(\boldsymbol{x}) = 0\}$ and $\boldsymbol{q_i}(\cdot)$ are the constraints of the optimization problem.

From (16) it is plain that the limit $\boldsymbol{X}(\omega, .)$ of any convergent subsequence satisfies the differential inclusion:

$$\dot{\boldsymbol{X}} \in -\partial C(\boldsymbol{X}) + \mathbf{T}\ ,\ \mathbf{T}(t) \in -V(\boldsymbol{X}(t)). \qquad (17)$$

This tells us [13] that the limit $\boldsymbol{X^k}(\omega, \cdot)$ converges to the stationary set of points of (17) in Θ w. p. 1. We denote this set of stationary points where $0 \in -\partial C(\boldsymbol{x}) + \boldsymbol{\tau}$ by S_Θ. Since $C(\cdot)$ is a convex function and Θ is a nonempty convex set, any point in S_Θ attains the minimum of $C(\cdot)$, and this completes the proof of the theorem.

Appendix II
Proof of Theorem 2

We use similar arguments used in the proof of Theorems 8.2.1 and 8.2.5 of [15, pp. 251-254 and 261-262] in our proof. First, the existence of a convergent subsequence follows from the fact that the uniform integrability of $\{\hat{\boldsymbol{g}}^n(k); n, k\}$ in Assumption A6 implies tightness of $\{\boldsymbol{X}^n(a_n q_n + \cdot), \boldsymbol{\Upsilon}^n(a_n q_n + \cdot)\}$ [15, p. 253], which is a sufficient condition for the existence of a convergent subsequence.

Following the argument in the proof of Lemma 1, for sufficiently small step size a_n we can replace the projection of

$\boldsymbol{x}^n(k) + \xi^n \boldsymbol{\Delta}(k)$ with the linear approximation in (18). Thus, from (22) we have

$$\boldsymbol{E}_k(\hat{\boldsymbol{g}}^n(k)) = \overline{\boldsymbol{sg}}(\boldsymbol{x}^n(k)) + O(a_n) + \delta^n(k)$$

First, from the argument in the proof of Lemma 1 (in particular, eq. (21)) one can show that $\delta^n(k) \to 0$ as $n \to 0$ (hence $a_n \to 0$). Second,

$$\lim_{m,\overline{m},a} \frac{1}{m} \sum_{i=\overline{m}}^{\overline{m}+m-1} O(a) = 0$$

if we take $\overline{m} = m$ and $a = m^{-2}$ with $m \to \infty$.

Now since the mapping from $\boldsymbol{x} \in \Theta$ to the subdifferential $\partial C(\boldsymbol{x})$ is upper semicontinuous, from [15, pp. 261-262], the limit process $\{\boldsymbol{X}(\cdot), \boldsymbol{\Upsilon}(\cdot)\}$ satisfies

$$\dot{\boldsymbol{X}} \in \partial C(\boldsymbol{X}) + \boldsymbol{\Upsilon}, \ \boldsymbol{\Upsilon}(t) \in -V(\boldsymbol{X}(t)) \quad \text{for almost all } \omega \text{ and } t \ .$$

Similarly as in the proof of Theorem 1, this implies that the limit $\boldsymbol{X}^{\boldsymbol{k}}(\cdot)$ converges to the stationary set of points of (17) in Θ. This completes the proof.

Appendix III
Proof of Lemma 1

Let $\bar{\boldsymbol{\Delta}}_s(k)$ be an $N \times 1$ vector, where values of entries corresponding to those of source s are $\Delta_{s,i}(k)$ and zero otherwise. Hence, $\boldsymbol{\Delta}(k) = \sum_{s \in S} \bar{\boldsymbol{\Delta}}_s(k) = (\Delta_{s,i}, s \in S, i \in 1, 2, \cdots, N_s \cdot |D^s|)$. Similarly, $\boldsymbol{u_s}$ is an $N \times 1$ vector, where the values of entries corresponding to those of source s are one and zero otherwise.

First, we show that there exists a finite K_1 such that for all $k \geq K_1$ we can rewrite $C_s^{\pm}(k)$ as

$$C_s^-(k) = \Lambda_s(\boldsymbol{x}(k)) \ \text{ and}$$
$$C_s^+(k) = \Lambda_s\left(\boldsymbol{x}(k) + \sum_{s' \in S} \xi_{s'}(k) \tilde{\boldsymbol{\Delta}}_{s'}(k)\right) \ ,$$

where

$$\tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) = \boldsymbol{\Delta}_{\boldsymbol{s'}}(k) - \frac{\sum_{j=1}^{N_{s'}} \Delta_{s',j}(k)}{N_{s'}} \boldsymbol{u_{s'}} \ .$$

Given any $N_s \times 1$ vector $\boldsymbol{\vartheta}$, the solution of the minimization problem

$$\min_{\boldsymbol{\eta}} \quad ||\boldsymbol{\vartheta} - \boldsymbol{\eta}||^2$$
$$\text{s. t.} \quad \boldsymbol{\eta}^T \boldsymbol{u} = r_s$$

is given by

$$\eta_i = \vartheta_i + \frac{r_s - \sum_{j=1}^{N_s} \vartheta_j}{N_s} \ , \tag{18}$$

where $\boldsymbol{u} = [1, 1, \cdots, 1]^T$. Obviously, if $\eta_i \geq 0$ for all i, this solution is equivalent to the L_2 projection. Here for the purpose of temporary perturbation we replace (6) with a non-negativity constraint. Thus, the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ can be calculated using (18) if

$$x_{s,i}(k) + \xi_s(k) \left(\Delta_{s,i}(k) - \frac{\sum_{j=1}^{N_s} (\Delta_{s,j}(k))}{N_s}\right) \geq 0 \ . \tag{19}$$

Recall that $\Delta_{s,i}(k)$ is bounded by α from Assumption A2. Hence, (19) holds if $\xi_s(k) \leq \frac{\min_j \{x_{s,j}(k)\}}{2\alpha}$. From (6) we know $\frac{\min_j \{x_{s,j}(k)\}}{2\alpha} \geq \frac{\epsilon}{2\alpha}$. Since $\xi_s(k) \to 0$, there exists finite K_1 such that $\xi_s(k) \leq \frac{\epsilon}{2\alpha}$ for all $k \geq K_1$. Therefore, (18) can be used to compute the projection of $\boldsymbol{x_s}(k) + \xi_s(k)\boldsymbol{\Delta_s}(k)$ for sufficiently large $k \geq K_1$.

Define for $k \geq K_1$

$$\begin{aligned} G_k^s &:= \frac{C_s^+(k) - C_s^-(k)}{\xi_s(k)} \\ &= \frac{\Lambda_s\left(\boldsymbol{x}(k) + \sum_{s' \in S} \xi_{s'}(k)\tilde{\boldsymbol{\Delta}}_{s'}(k)\right) - \Lambda_s(\boldsymbol{x}(k))}{\xi_s(k)} \\ &= \frac{\Lambda_s\left(\boldsymbol{x}(k) + \xi_s(k)\hat{\boldsymbol{\Delta}}(k)\right) - \Lambda_s(\boldsymbol{x}(k))}{\xi_s(k)}, \end{aligned}$$

where

$$\hat{\boldsymbol{\Delta}}(k) = \sum_{s' \in S} \frac{\xi_{s'}(k)}{\xi_s(k)} \tilde{\boldsymbol{\Delta}}_{s'}(k)$$

Definition 2 ([13]): Suppose that $h : \mathbb{R}^r \to \mathbb{R}$ is a real-valued convex function on $\mathbb{R}^r$. The one-sided directional derivative of h at $\boldsymbol{x}$ with respect to a vector $\boldsymbol{y}$ is defined to be

$$h'(\boldsymbol{x}; \boldsymbol{y}) = \lim_{\lambda \downarrow 0} \frac{h(\boldsymbol{x} + \lambda \boldsymbol{y}) - h(\boldsymbol{x})}{\lambda} \ .$$

For each vector $\boldsymbol{y}$ the directional derivative satisfies

$$h'(\boldsymbol{x}; \boldsymbol{y}) = \max_{sg(\boldsymbol{x}) \in \partial h(\boldsymbol{x})} sg(\boldsymbol{x})^T \boldsymbol{y} \ .$$

In other words, the directional derivative is the maximum of the inner products $< sg(\boldsymbol{x}), \ \boldsymbol{y} >$ over $\partial h(\boldsymbol{x})$. We denote the set of subgradients that achieve the maximal inner product by $\partial h_{\boldsymbol{y}}(\boldsymbol{x}) \subseteq \partial h(\boldsymbol{x})$, i.e., $\partial h_{\boldsymbol{y}}(\boldsymbol{x}) = \arg\max_{sg(\boldsymbol{x}) \in \partial h(\boldsymbol{x})} sg(\boldsymbol{x})^T \boldsymbol{y}$.

Let $\Lambda_s'(\boldsymbol{x}(k)\,; \hat{\boldsymbol{\Delta}}(k))$ (resp. $C'(\boldsymbol{x}(k)\,; \hat{\boldsymbol{\Delta}}(k))$) be the one-sided directional derivative of Λ_s (resp. C) at $\boldsymbol{x}(k)$ with respect to vector $\hat{\boldsymbol{\Delta}}(k)$. Since $\Lambda_s(\cdot)$ is convex and continuous, we can show using [13, Lemma 1] that, for any $\epsilon > 0$, there exists finite K_2 such that, for all $k \geq K_2$,

$$|\Lambda_s'(\boldsymbol{x}(k); \hat{\boldsymbol{\Delta}}(k)) - G_k^s| < \frac{\epsilon}{2} \qquad \text{w. p. 1.}$$

From above, for every $\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k)) \in \partial \Lambda_{s,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$, we have

$$\begin{aligned} \Lambda_s'(\boldsymbol{x}(k); \hat{\boldsymbol{\Delta}}(k)) &= \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \hat{\boldsymbol{\Delta}}(k) \\ &= \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s' \in S} \frac{\xi_{s'}(k)}{\xi_s(k)} \tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) \ . \end{aligned} \tag{20}$$

Since $\frac{\xi_{s'}(k)}{\xi_s(k)} \to 1$, there exists finite $K_3 \geq \max(K_1, K_2)$ such that, for all $\epsilon > 0$ and $k \geq K_3$,

$$\left| \frac{C_s^+(k) - C_s^-(k)}{\xi_s(k)} - \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s' \in S} \tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) \right| < \epsilon \quad \text{w. p. 1.} \tag{21}$$

Consequentially, for all $k \geq K_3$,

$$\begin{aligned}
&\boldsymbol{E}_k(\hat{g}_{s,m}(k)) \\
&= \frac{N_s}{N_s-1}\boldsymbol{E}_k\left(\frac{C_s^+(k) - C_s^-(k) + \mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,m}(k)}\right) \\
&= \frac{N_s}{N_s-1}\boldsymbol{E}_k\left(\frac{\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s'\in S}\tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) + \delta_1(k)}{\Delta_{s,m}(k)}\right) \\
&= \frac{N_s}{N_s-1}\left(\frac{N_s-1}{N_s}\overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k)) + O(\xi_s(k))\right) + \bar{\delta}_1(k) \\
&= \overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k)) + O(\xi_s(k)) + \bar{\delta}_1(k) \qquad (22)
\end{aligned}$$

where $\overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k))$ is the entry of $\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))$ corresponding to $\boldsymbol{x}_{s,m}(k)$,

$$\delta_1(k) = \frac{C_s^+(k) - C_s^-(k)}{\xi_s(k)} - \overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))^T \sum_{s'\in S}\tilde{\boldsymbol{\Delta}}_{\boldsymbol{s'}}(k) \ ,$$

and

$$\bar{\delta}_1(k) = \frac{N_s}{N_s-1}\boldsymbol{E}_k\left(\frac{\delta_1(k)}{\Delta_{s,m}(k)}\right) \ .$$

It is now plain to see that $\lim_{k\to\infty}\bar{\delta}_1(k) \to 0$ w. p. 1 from (21) and hence

$$\boldsymbol{E}_k(\hat{g}_{s,m}(k)) - \overline{\boldsymbol{sg}}_{s,m}(\boldsymbol{x}(k)) \to 0 \qquad \text{w. p. 1.} \qquad (23)$$

According to [2, Proposition 4.2.4, p. 232], if a real-valued function h is a sum of a set of convex functions $\{h_1, \ldots, h_n\}$, i.e., $h(\boldsymbol{x}) = \sum_{i=1}^n h_i(\boldsymbol{x})$, the subdifferential of h at $\boldsymbol{x}$ is given by

$$\begin{aligned}
&\partial h(\boldsymbol{x}) = \sum_{i=1}^n \partial h_i(\boldsymbol{x}) \\
&:= \left\{\sum_{i=1}^n sg^i(\boldsymbol{x}) : \ sg^i(\boldsymbol{x}) \in \partial h_i(\boldsymbol{x}) \text{ for all } i \in \{1,\ldots,n\}\right\}.
\end{aligned}$$

This tells us that, for each $s \in \mathcal{S}$, the subgradient $\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k))$ in (20) can be written as

$$\overline{\boldsymbol{sg}}_s(\boldsymbol{x}(k)) = \sum_{l\in L^s}\overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k)) \ ,$$

where $\overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k)) \in \partial C_{l,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$. In addition, this implies that any summation $\sum_{l\in\mathcal{L}}\overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k))$ with $\overline{\boldsymbol{sg}}^l(\boldsymbol{x}(k)) \in \partial C_{l,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$ for all $l \in \mathcal{L}$ is a subgradient in $\partial C_{\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$.

For each $l \in \mathcal{L}$, choose a subgradient $\boldsymbol{sg}^{l\star}(\boldsymbol{x}(k)) \in \partial C_{l,\hat{\boldsymbol{\Delta}}(k)}(\boldsymbol{x}(k))$ and let $\overline{\boldsymbol{sg}}_s^\star(\boldsymbol{x}(k)) = \sum_{l\in L^s}\boldsymbol{sg}^{l\star}(\boldsymbol{x}(k))$, $s \in \mathcal{S}$, and $\boldsymbol{sg}^\star(\boldsymbol{x}(k)) = \sum_{l\in\mathcal{L}}\boldsymbol{sg}^{l\star}(\boldsymbol{x}(k))$. Then, from the above argument, it is clear that $\overline{\boldsymbol{sg}}_s^\star(\boldsymbol{x}(k))$ satisfies (20) for all $s \in \mathcal{S}$. Moreover, if we denote by $s(m)$ the source corresponding to the m-th entry of $\boldsymbol{x}$, then the m-th entry of $\boldsymbol{sg}^\star(\boldsymbol{x}(k))$ equals the m-th entry of $\overline{\boldsymbol{sg}}_{s(m)}^\star(\boldsymbol{x}(k))$. Therefore, from (23) this proves claim (i) of the lemma that there exists a sequence of subgradients $\{\boldsymbol{sg}(\boldsymbol{x}(k)), k = 0, 1, \ldots\}$ such that $\boldsymbol{b}(\boldsymbol{x}(k)) := \boldsymbol{sg}(\boldsymbol{x}(k)) - \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k))$ goes to 0 w. p. 1.

From the assumption that $\mathbf{E}\left[\mu_s^+(k) - \mu_s^-(k)|\mathcal{F}_k\right] = 0$ and using the independence of $\mu_s^\pm(k)$ and $\boldsymbol{\Delta}_{\boldsymbol{s}}(k)$, we can bound the second moment of $\hat{\boldsymbol{g}}_{\boldsymbol{s}}(k)$ as follows:

$$\begin{aligned}
&\boldsymbol{E}_k((\hat{g}_{s,i}(k))^2) \\
&= \boldsymbol{E}_k\left(\left(\frac{C^+(k) - C^-(k) + \mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2\right) \qquad (24) \\
&= \boldsymbol{E}_k\left(\left(\frac{C^+(k) - C^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2 + \left(\frac{\mu_s^+(k) - \mu_s^-(k)}{\xi_s(k)\Delta_{s,i}(k)}\right)^2\right)
\end{aligned}$$

After some algebra using the bounds on $\mathbf{E}\left[(\boldsymbol{\Delta}_{\boldsymbol{s}}(k))^2\right]$, $\mathbf{E}\left[(\boldsymbol{\Delta}_{\boldsymbol{s}}(k))^{-2}\right]$, and $\mathbf{E}\left[(\mu_s^\pm(k))^2\right]$, one can show that the first term in (24) is $O(1)$ and the second term is $O(\xi_s(k)^{-2})$. Note that $\boldsymbol{E}_k(\boldsymbol{\varrho}^2(k)) = \boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)^2) - (\boldsymbol{E}_k(\hat{\boldsymbol{g}}(k)))^2$. Hence, since $\sum_{k=0}^\infty\left(\frac{a_s(k)}{\xi_s(k)}\right)^2 < \infty$ from Assumption A4, it is easy to see that $\sum_{i=k}^\infty \boldsymbol{E}_k(||\boldsymbol{A}(k)\boldsymbol{\varrho}(k)||^2) < \infty$ w. p. 1.

PLACE PHOTO HERE

Tuna Güven received the B.S.E.E. from Middle East Technical University, Ankara, Turkey in 2000, and M.S. and Ph.D. degrees in Electrical and Computer Engineering from University of Maryland, College Park, MD in 2002 and 2006, respectively. His research interests includes performance evaluation and optimization in communication networks.

PLACE PHOTO HERE

Richard J. La received his B.S.E.E. from the University of Maryland, College Park in 1994 and M.S. and Ph.D. degrees in Electrical Engineering from the University of California, Berkeley in 1997 and 2000, respectively.

From 2000 to 2001 he was with the Mathematics of Communication Networks group at Motorola Inc,. Since 2001 he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Maryland. He was a recipient of an NSF CAREER Award in 2003.

PLACE PHOTO HERE

Mark A. Shayman (M'81-SM'03) received the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, in 1981.

He served on the Faculty of Washington University, St. Louis, MO, and the University of Maryland, College Park, where he is currently Professor of Electrical and Computer Engineering.

Dr. Shayman received the Donald P. Eckman Award from the American Automatic Control Council and the Presidential Young Investigator Award from the National Science Foundation. He has served as Associate Editor of the IEEE Transactions on Automatic Control.

PLACE PHOTO HERE

Bobby Bhattacharjee is an associate professor in the Department of Computer Science at the University of Maryland, College Park. His research interests are in the design and implementaion of scalable systems, protocol security, and peer-to-peer systems. He is a member of the ACM.