On-line Minimization of Call Setup Time via Load Balancing: A Stochastic Approximation Approach [1]

RAHUL SIMHA

Department of Computer Science
College of William and Mary
Williamsburg, VA 23185

simha@cs.wm.edu

JAMES F. KUROSE

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

kurose@cs.umass.edu

Abstract

With the addition of new network services, it is anticipated that the processing involved in setting up a call in a circuit-switched network or a session in a packet-switched network will vary greatly for different types of services. In this paper, we address the problem of reducing the call setup time in a circuit-switched network, or equivalently the session setup time in a packet-switched network, through the balancing of load across call processors. With a view to designing algorithms to execute on-line in a system, we formulate a stochastic optimization problem and study the use of stochastic approximation techniques. Given the distributed nature of the problem, we extend previous results obtained for a single node to the case where several nodes operate simultaneously and in an asynchronous manner. Our results include a theoretical study of convergence as well as several simulation results that compare two stochastic approximation techniques.

[1] This work was supported in part by the Office of Naval Research under Grant 00014-90-J-1293, the Defense Research Projects Agency under contract NAG2-578 and NSF grants CERDCR 8500332 and NCR-9109792.

1 Introduction

In today's telephone system the computational requirements of setting up a typical telephone call are quite well-defined and known in advance. Even so, the high system response demanded by users has prompted several research efforts in the design of the real-time operating systems that reside at the switches to handle large numbers of calls [**?**, **?**]. With new services being offered [**?**, **?**], the processing needs of calls are expected to vary greatly [**?**]. At the same time, in a packet-switching environment, there is processing associated with the assignment of virtual-circuit sessions to routes in the packet-switched network. We may anticipate that, with the addition of services such as conferencing, call-forwarding and sophisticated billing mechanisms, it is likely that the processing needs associated with each session assignment will both increase and vary among different services.

In both packet- and circuit-switching environments, we have a *network of resources* (processors) with calls (or sessions) arriving to sources at the edge of the network and seeking a route to a destination node. Typically (OOC control [**?**]), a route is selected at the originating edge of the network and a test packet is sent along the route to each of the call processors on the route; those processors then decide if the call can be assigned to that route. This procedure continues until a suitable route is found or if not, the call is rejected. The time to assign a route to a call from the moment of arrival to a source is the *call setup time.* We will use the terms *call* and *session* interchangeably, with the understanding that the techniques described in this paper apply equally well to both circuit- and packet-switched systems.

In this paper, we address the problem of minimizing the average setup time for calls arriving to a network. Our approach is to select routes carefully so that processing loads are evenly distributed around the network of processors; analogous to the similar phenomenon in distributed computer systems [**?**, **?**], the 'load balancing' of call processors then reduces the average setup time. We are particularly interested in the case where little is known about the random arrival patterns of the calls or their processing requirements and, instead, *measurements* prone to random errors must be taken from the system in order to determine load conditions. Furthermore, we focus on algorithms that execute *on-line* while the system is in operation, and which determine routing policies based on the measurements taken from the system.

The strength of our contribution is a formal theoretical treatment of a simplified version of the general setup problem. The call setup minimization problem as discussed above is cast in terms of a stochastic optimization problem with multiple, asynchronous sources of calls. We present a *decentralized stochastic optimization* algorithm for *multiple sources* and study conditions under which our algorithm converges to an optimal solution. Although our focus is on minimizing call setup times, our problem formulation and results are applicable to several other problems that have received attention in the literature including message routing, load balancing and file allocation.

Our approach is characterized by the following salient points. We seek to optimize routing policies parameterized by 'routing fractions' [?, ?, ?, ?] when the network of call processors is modeled by a network of queues. We focus on iterative gradient algorithms for the minimization of call setup time, in particular, gradient algorithms that exploit the structure of the problem. Finally, we utilize the rich and growing theory of sensitivity (derivative) estimation [?, ?, ?] in order to obtain measurements from the system, which are then used in an *on-line* manner to optimize the routing policy in a stochastic optimization procedure.

Several gradient-based approaches have been proposed in the context of packet routing and load balancing [?, ?, ?, ?, ?]; these approaches assume that sensitivity information is available through formulas obtained from a tractable model. The typical approach taken towards these problems, with temporally separate *estimation* and *optimization*, involves prescribing and solving a queueing model from which a performance metric such as the mean delay (setup time, in this case) may be computed. Following this, the relevant parameters such as the arrival and service rates are estimated from the system, and are then inserted into the queueing model. The resulting mean delay function (in terms of the routing fractions) is then solved for the optimal routing fractions. The problems with such an approach include strict distributional assumptions for tractability, the fact that no optimization is performed *during* the estimation period, and that poor estimates can result in suboptimal solutions (see [?] for a more detailed discussion).

In this paper, we focus on direct *stochastic approximation*, a procedure that interleaves estimation and optimization uniformly into a single procedure. In doing so, we also address the tradeoff between estimation and optimization. We discuss two stochastic approximation techniques, one from the literature and the other introduced in earlier work [?, ?], for augmenting deterministic algorithms into stochastic versions capable of handling measurements. Related approaches that combine estimation and optimization [?, ?] in which formulas from a model are used with estimates of the component parameters are discussed in the next section, following the problem formulation. In Section 3, we present our algorithm and theoretical results. We discuss some experimental results in Section 4 and estimation issues in sections 5 and 6 before concluding in Section 7 with a summary.

2 Problem Formulation

Calls arrive to a source at the edge of the network and an attempt is made to set up the call according to the routing policy. In this paper, we consider a particular class of routing policies, *flow-fraction* policies [?, ?, ?] in which a source associates a *routing fraction* or *routing probability* with each route it uses, and the fraction determines the percentage of incoming calls for which a setup attempt is made on that route. In practice, several routes are attempted before a call is rejected. In our formulation of the problem, we assume that the routing fractions are used for the *first* setup attempt. If the percentage of rejected first attempts is small enough, as is the case in practice, the effect on processor loads of later attempts is negligible and may be ignored. This same fact also justifies our assumption that *all* processors along a route incur processing overheads for a setup attempt along that route, even though a reject at the first processor will result in no further processing along that route. We believe that the theory developed in this paper allows this assumption to be relaxed ,albeit at the expense of increased complexity in sensitivity estimation. We note that in some networks [**?**] call setup packets cause setup computations to be performed on all processors along the attempted route.

Consider a network with C sources, each of which serves a stream of incoming calls.

- *Arrival process.* The interarrival times of the calls incident at the c-th source are independent random samples from an unknown distribution; the mean arrival rate is denoted by λ_c.

- *Call processing requirements.* A call that uses a *call processor* $r \in \{1, ..., R\}$ requires service for a period of time that is an independent random sample drawn from an unknown distribution, the mean of which is denoted by $\frac{1}{\mu_r}$. The collection of relevant parameters such as λ_c and μ_r is denoted by Θ. Processing behavior at a call processor is modeled by a queue.

- *Routing policy.* A set of routing variables known as a *local routing policy* is associated with each source and the collection of such disjoint sets, which we refer to as the *global routing policy*, is denoted by the vector x. The local routing policy for source c is denoted by $x(c) = (x_{k_{c-1}+1}, ..., x_{k_c})$. Therefore, the global routing policy is the vector $x = (x_1, ..., x_{k_C})$. Furthermore, the set of indices in the routing policy for source c is denoted by $K_c = \{k_{c-1} + 1, ..., k_c\}$; k_0 is taken to be zero. Component x_i of $x(c)$ denotes the fraction of calls to source c that are directed to one of the $|K_c|$ routes used by source c. The index 'i' only refers to the position within the vector x and does not necessarily correspond to the name of a particular route or call processor. However, we will often assume that each source has access to all routes and permit the index i to refer to routes as well, although the theoretical development in this paper is independent of this assumption. Note that this formulation allows call processors to be shared by several sources.

  We follow the conventional probabilistic implementation [**?**, **?**, **?**, **?**] of a flow-fraction policy: a source, c, using a uniform random-number generator, dispatches a call setup packet on route i with probability x_i, $i \in K_c$, thus making route i the first attempt to set up the call. Thus, over a long period the fraction of calls first attempted on route i is x_i. We note that, in some cases, a more efficient deterministic implementation of flow-fractions is possible [**?**].

  The example in Figure **??** will help explain our notation. There are three sources, one destination and nine call processors along various routes from the sources to the common destination. We number the call processors starting from the top left, working down a column. Thus the first bank of call processors has processors with service rates μ_1, μ_2 and μ_3 and we associate the rates μ_7, μ_8 and μ_9 with the third bank . The three sources serve incoming

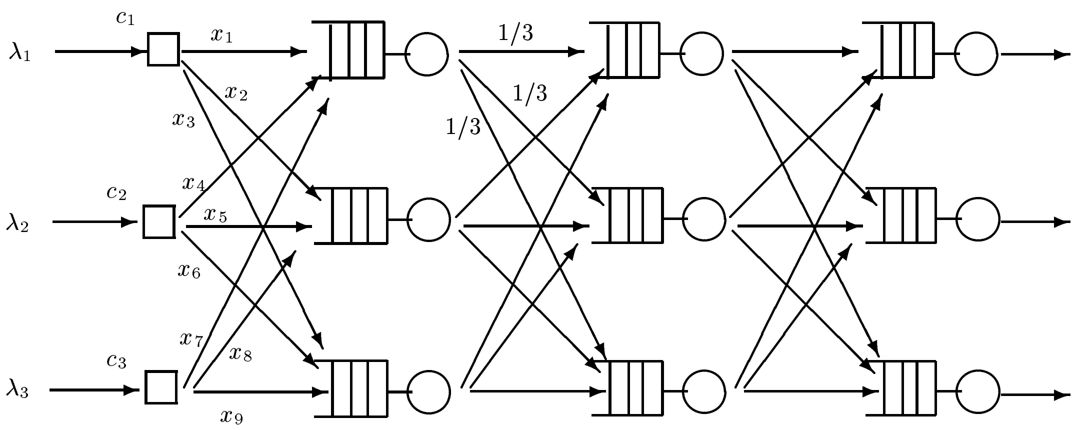

Figure 1: An example

streams of calls which arrive at rates λ_1, λ_2 and λ_3. The policies $x(1) = (x_1, x_2, x_3)$, $x(2) = (x_4, x_5, x_6)$ and $x(3) = (x_7, x_8, x_9)$ are used in directing the calls to the first bank of processors. The routing after that is determined by the fixed probabilities $(1/3, 1/3, 1/3)$ as shown in the figure. Thus, the figure shows an example of routing control performed only at the edge of the network. This is only an example; the theory extends to the more general case where routing control may be performed at intermediate nodes and also to the case in which only a few routes are used by each source.

We are now in a position to formally state the problem. Denote the expected setup time for a call by $D(x, \Theta)$ and define:

$$\begin{array}{ll} \text{Find } \min\limits_{x} & D(x, \Theta) \\ s.t. & \forall c \in \{1, ..., C\} : \sum\limits_{i=k_{c-1}+1}^{k_c} x_i = 1 \\ & \forall i \in \{1, .., k_C\} : \; c_i \leq x_i \leq b_i \end{array}$$

Since each $x(c)$ is a probability distribution, we usually have $c_i = 0$ and $b_i = 1$; however, as we will see, it is often necessary to require $0 < c_i < b_i < 1$. Define the set, S, of *feasible* global policies such that for each policy $x \in S$, the equality and inequality constraints above are satisfied. Let $S^* \subset S$ be the set of solutions to the above optimization problem. We assume that, provided a solution in S^* is found, it does not matter which of the many solutions in S^* it is. Let $D^* = D(x, \Theta)$ for $x \in S^*$.

The above problem formulation finds application in several other areas of current interest when the units of work in the system – in this case, calls – are taken to be messages, jobs or file requests. When the random characteristics of the system are known and formulas for gradient estimators can be computed, the formulation reduces to the message routing problem in [**?**, **?**]. When jobs are substituted for calls in the formulation, a simplified load balancing problem has been studied extensively in the literature in the case that each job is processed only at a single queue [**?**, **?**, **?**, **?**]. Similarly, when the units are taken to be file access requests, our problem may be reformulated as a file allocation problem [**?**, **?**]. Through the identification of our general formulation with the problems of message routing, load balancing and file allocation, our contributions are seen to extend results in these areas. We note that stochastic versions of routing [**?**] and load balancing [**?**] have been studied. In [**?**], it is assumed that formulas for gradients are available and the deterministic algorithm of [**?**] is extended to the stochastic case in this manner, referred to as the certainty-equivalence method in [**?**]. We handle stochasticity with more generality and also consider the case in which constraints are unknown and must themselves be estimated. The load balancing results in [**?**] are presented for a single controller and only for $M/M/1$ queues; the results in this paper address issues raised by multiple asynchronously operating sources, and a wider class of queueing networks both in structure and distribution. Flow-fraction routing using perturbation analysis and the algorithm in [**?**] is examined in [**?**], although no theoretical results were presented and simulations were examined in single runs; in contrast, we present a more general algorithm, a convergence result and simulation results covering a wider set of issues. Algorithmic stepsizes are considered in [**?**].

3 A Decentralized Stochastic Optimization Algorithm

3.1 Gradient Estimation

We will assume that estimates $\hat{\gamma}_r^m$ and W_r^m of the local arrival rate and local mean processing delay, and a local gradient estimate Y_r^m of the mean delay at each processor (queue) with respect to the rate of arrivals γ_r to the processor can be constructed at each processor r, using measurements made locally at the processor. Here, m denotes the number of samples used in the estimate. We say that a source c *affects* a particular call processor r if calls routed by c use r, i.e., if a change in the flow of calls at c may change the flow of incoming calls to r. Let Q_c denote the set of processors that are affected by source c. To execute each step of the iterative gradient procedure, each source c must obtain local rate, delay and derivative estimates from all processors $r \in Q_c$. The source requests this information from these processors while also specifying the number of samples, m, to be used in each estimate. As discussed in Appendix A, on receiving this information, estimates F_i^m of each partial derivative $\frac{\partial D(x,\Theta)}{\partial x_i}$ are constructed by the source. Using recently introduced queue sensitivity estimation techniques such as perturbation analysis [?] and likelihood-ratio methods [?] for Y_r^m, it is possible to obtain consistent estimators for each partial derivative, i.e., F_i^m such that $\lim_{m\to\infty} F_i^m = \frac{\partial D(x,\Theta)}{\partial x_i}$. Furthermore, these estimates are obtained with little additional overhead while observing the queues in operation. In Appendix A, we discuss the use of Infinitesimal Perturbation Analysis (IPA) [?] in obtaining F_i^m.

3.2 Algorithm

The stochastic optimization algorithm we present is an iterative algorithm in which an iteration follows a period of estimation. Each source executes a copy of the algorithm and, unless otherwise specified, executes this copy independent of the other sources. Consider a source c. For algorithmic iteration n, source c requests each processor $r \in Q_c$ to provide estimates $\hat{\gamma}_r^{L(n)}$, $W_r^{L(n)}$ and $Y_r^{L(n)}$ using $L(n)$ samples. Once this information is available at the source, the source constructs gradient estimates $F_i^{L(n)}$. We will use $X^{(n)}(c) = (X_{k_{c-1}+1}^{(n)}, \ldots, X_{k_c}^{(n)})$ to denote the current (at iteration n) routing policy at c; since the values of $X^{(n)}(c)$ will depend on previous gradient estimates, we use $X^{(n)}(c)$ instead of $x^{(n)}(c)$ to emphasize that the routing policy is a random variable. Clearly, the estimates $F_i^{L(n)}$ depend on $X^{(n)}$. We will occasionally emphasize this dependence by writing $F_i^{L(n)}(X^{(n)})$ and, where convenient, we will drop the subscript to denote a vector of such quantities. These gradient estimates are then used in an iterative optimization step to obtain the next routing policy:

$$X_i^{(n+1)} = X_i^{(n)} + a_n H_i\left(F^{L(n)}(X^{(n)}), X^{(n)}\right) \tag{1}$$

The function $H = (H_1, \ldots, H_{K_C})$ is described below in terms of each component i, following a discussion of the inequality constraints in our problem.

In our problem, the inequality constraints were defined to be $\forall i \in \{1, ..., k_C\} : \; c_i \le x_i \le b_i$. When these are imposed on the iteration variables at each step, we have $\forall i \in \{1, ..., k_C\} : \; c_i \le x_i^{(n)} \le b_i$. In the algorithms that have appeared in the literature [?, ?, ?, ?, ?], the constants c_i and b_i are usually taken to be 0 and 1 respectively to reflect the fact that the iteration variables are probabilities. In our problem, we continue to treat these variables as probabilities or fractions, although we must allow for the possibility that $0 < c_i < b_i < 1$. These inequality constraints can arise from stability conditions required at the queues. For example, consider a single source that routes calls to two processors routing policy $X(1) = (X_1, X_2)$. At each of the processors we must satisfy queue stability constraints: for $i = 1, 2$, $\lambda X_i < \mu_i$ or $X_i < \frac{\mu_i}{\lambda} = b_i$. If $b_i < 1$, due to the values of λ and μ_i (e.g., when $\lambda =$, $\mu_1 = 0.6$ and $\mu_2 = 0.6$), then we must ensure that the constraints are satisfied by the probabilities X_i. The lower bound presents a different concern: if X_i is too small then very few calls will be routed over route i and thus, as a practical matter it will be difficult to obtain sensitivity estimates. For this reason, it may be advisable to maintain $c_i > 0$. We will first assume that the constants c_i and b_i are known and later discuss the more practical case in which the upper bounds b_i are unknown (because, for example, the values of λ, μ_i and other parameters are not known) and must themselves be estimated.

The algorithm is necessarily complicated by the presence of the inequality constraints and it is here that we make a useful contribution. To understand this, we first examine the simpler case in which there are *no* inequality constraints. Note that, in this case, with only the *equality* constraint, a standard gradient algorithm may be constructed by using the well-known projection method for gradient algorithms [**?**]:

$$H_i\left(F^{L(n)}, X^{(n)}\right) = -\left(F_i^{L(n)} - \frac{1}{|K_c|}\sum_{j\in K_c} F_j^{L(n)}\right)$$

This algorithm now preserves the equality constraint $\sum_{i\in K_c} X_i^{(n)} = 1$ since the gradient step is projected onto the plane defined by the equality constraint. At this point it is important to note that there is unfortunately no such simple projection to handle *inequality* constraints. Most general algorithms for inequality constraints involve the treatment of each inequality separately, sometimes with elaborate penalty functions [**?**, **?**]. However, the nature of all the constraints in our control problem can be exploited in order to develop efficient mechanisms [**?**, **?**, **?**] to handle the inequality constraints when $c_i = 0$ and $b_i = 1$.

We now extend these results to the more general case when $0 \le c_i < b_i \le 1$. Note that the form of the iterative step remains identical to the one above with no inequality constraints, but instead the set of indices which are involved in the computation is now a subset $(A_c^{(n)})$ of the total (K_c). Consider any set $K \subseteq K_c$ and define:

$$\begin{aligned}
U_K^{(n)} &= \min_{i\in K} F_i^{L(n)} + \frac{b_i - X_i^{(n)}}{a_n} \\
L_K^{(n)} &= \max_{i\in K} F_i^{L(n)} + \frac{c_i - X_i^{(n)}}{a_n} \\
M_K^{(n)} &= \frac{1}{|K|}\sum_{i\in K} F_i^{L(n)}
\end{aligned}$$

The Algorithm:

1. First compute the set of feasible indices, $A_c^{(n)}$ as follows:

(i) Sort $F_i^{L(n)}, i \in K_c$ in ascending order and let s_m, $m = 1, ..., |K_c|$ denote the sort indices, i.e., $s_1 = j$ such that $F_j^{L(n)} = \min_{i \in K_c} F_i^{L(n)}$

(ii) $m := 1$;
done := false;

(iii) **while** $(m < k_c - 1)$ **and** (**not** done) **do**
begin
$K := \{s_m\}$;
for $i := m + 1$ **to** k_c **do**
if $L_{K \cup \{s_i\}}^{(n)} \leq M_{K \cup \{s_i\}}^{(n)} \leq U_{K \cup \{s_i\}}^{(n)}$ **then**
begin
$K := K \cup \{s_i\}$;
done:=true;
end;
$m := m + 1$;
end;

(iii) $A_c^{(n)} := K$

2. Using $A_c^{(n)}$ from above, compute $X_i^{(n+1)}$:

$$X_i^{(n+1)} = \begin{cases} X_i^{(n)} - a_n \left(F_i^{L(n)} - \frac{1}{\left|A_c^{(n)}\right|} \sum_{j \in A_c^{(n)}} F_j^{L(n)} \right), & i \in A_c^{(n)} \\ X_i^{(n)}, & i \notin A^{(n)} \end{cases} \tag{2}$$

Note that the computations for $A_c^{(n)}$ and $X_i^{(n+1)}, i \in A_c^{(n)}$ are performed at source c; thus, once the estimates are obtained the computation is distributed among the sources, which execute independently. The tighter inequality constraints, $0 \leq c_i < b_i \leq 1$, are handled in the computation of the set $A_c^{(n)}$; we now give an informal description of this set. The special case of $c_i = 0$ and $b_i = 1$ is treated elsewhere [**?**, **?**, **?**]. Consider $K \subseteq K_c$ such that when $A_c^{(n)} = K$, the result of applying equation (**??**) is a feasible vector $X^{(n+1)}(c)$. We consider what happens when $s_i \notin K$ is to be added to K (in the innermost loop, the **for** loop). We observe that, in the definitions of $U_K^{(n)}$ and $L_K^{(n)}$, the terms $F_{s_i}^{L(n)} + \frac{b_{s_i} - X_{s_i}^{(n)}}{a_n}$ and $F_{s_i}^{L(n)} + \frac{c_{s_i} - X_{s_i}^{(n)}}{a_n}$ define, for each $s_i \notin K$, an interval $[L_{K \cup \{s_i\}}^{(n)}, U_{K \cup \{s_i\}}^{(n)}]$ of feasibility for the location of the new average $M_{K \cup \{s_i\}}^{(n)}$. Provided that the new average $M_{K \cup \{s_i\}}^{(n)}$ remains within the innermost interval, the resulting vector will be feasible and hence s_i may be added to K. Stated differently, when $K = A_c^{(n)}$, $M_K^{(n)} \in [L_K^{(n)}, U_K^{(n)}] \Rightarrow X^{(n+1)} \in S$. Note that the outer loop (**while** loop) is required in order to attempt to include every pair s_m, s_i in K, in case

K is found to be empty for $m = 1$. The maintenance of feasibility may be stated more formally as follows.

LEMMA 1: Each routing policy, $X^{(n+1)}$, satisfies the inequality constraints, provided that $X^{(n)}$ is feasible, i.e. $X^{(n)} \in S \Rightarrow X^{(n+1)} \in S$.

PROOF: (See Appendix B).

3.3 Assumptions and Convergence

We are now in a position to prove a convergence result for our algorithm. For convenience of description, we use the notation $f_i(x)$ to denote $\frac{\partial D(x,\Theta)}{\partial x_i}$.

Assumptions:

*A*1 *Boundedness of derivatives.* The first and second partial derivatives of D with respect to each control variable x_i are bounded.

*A*2 *Kuhn-Tucker conditions for optimality.* The Kuhn-Tucker conditions are satisfied at every point in S^*, i.e., there exist sets $B_c, \bar{B}_c \subset K_c$ such that

(i) $B_c \cap \bar{B}_c = \phi$ and $B_c \cup \bar{B}_c = K_c$.
(ii) $\forall i, j \in B_c : f_i = f_j$
(iii) $\forall j \in \bar{B}_c : f_j > f_i$ for each $i \in B_c$.

when f_i, f_j are evaluated at $x \in S^*$.

*A*3 *Stepsize conditions.* The stepsizes satisfy the condition $\forall n : a_n = a$ for some $a > 0$.

*A*4 *Growth rate of samples.* The number of samples required per iteration satisfies $L(n) \geq c_n \log n$ for some sequence $c_n \to \infty$.

*A*5 *Strong consistency of derivative estimator.* We assume that the estimates F_i^m are strongly consistent in the following sense: $\lim_{m\to\infty} F_i^m = \frac{\partial D}{\partial x_i}$ *a.s.*

*A*6 *Existence of moment generating functions.* We assume that the moment generating functions of the interarrival times and the number in a busy period of each queue are finite in some neighborhood of zero of the generating function variable.

THEOREM 3: Under assumptions $A1 - A6$, there exists $\alpha > 0$ such that $\forall a_n = a < \alpha$, the algorithm converges to D^*, i.e.,

$$\lim_{n\to\infty} D(X^{(n)}, \Theta) = D^* \quad a.s.$$

PROOF: (See Appendix B)

3.4 Discussion

We now discuss some of the assumptions made above; in particular, to see that they are reasonable, we check to see if they hold for a network of $M/M/1$ queues. In $A1$, if the partial derivatives are continuous, as is the case with $M/M/1$ queues, and S is compact then $A1$ is satisfied. Sometimes, the set S may be forced into being compact by some suitable projection mechanism; for example, we may assume that by choosing c_i and b_i, arrival rates to the queues are bounded away from the service rates. If this is not possible then routing policies may be generated in which queue arrival rates exceed service rates causing instability, a phenomenon that may also be compounded by secondary setup attempts. Fortunately, gradients increase with queue utilization and thus, in practice, should act as penalty functions, thereby forcing stable policies. Assumption $A2$ is the standard assumption made for gradient algorithms and optimization problems [**?**, **?**, **?**, **?**], i.e., that the Kuhn-Tucker conditions are necessary at optima (see equation (7) in [**?**]). Note that convergence is in terms of the cost function; even though fluctuations in the control variables may occur, the cost function decreases and in practice fluctuations are not evident.

We distinguish between two fundamental types of *stochastic approximation techniques*. Intuitively, the 'noise' in the estimates F^m must be slowly filtered out in order to obtain convergence. Noise removal in stochastic iterations of the type in equation (**??**) is achieved using one of two stochastic approximation techniques, *stepsize control* and *sampling control* (see [**?**] for a more comprehensive discussion). In stepsize control, the standard method [**?**, **?**, **?**], the stepsizes a_n are chosen to satisfy $a_n > 0$, $a_n \to 0$ and $\sum_{n=1}^{\infty} a_n = \infty$, and $L(n)$ is taken to be a constant, $L(n) = L > 0$. In sampling control, introduced more recently [**?**, **?**], the stepsize is a small constant $a_n = a$ and the number of samples per iteration grows faster than logarithmically. The intuition is that, for large n (and, therefore, large $L(n)$) the estimates are close to the true gradient (by $A5$) and thus, the algorithm behaves deterministically; initially, if $D(X^{(0)}, \Theta)$ is far from optimal, small values of n will permit quick movement towards S^*, allowing refinement via more accurate estimates later. Theoretically, as discussed in [**?**], stepsize control requires the estimators to be unbiased whereas several proposed estimators in the literature are not. Sampling control, however, only requires strong consistency ($A5$), a property satisfied by some of these gradient estimators [**?**, **?**], including the one discussed in this paper. More practically, sampling control, because of increasing estimation periods, requires less communication between processors and sources and, as the experimental results indicate, performs at least as well as stepsize control.

As mentioned earlier, the approach taken in [**?**] involves using measurements of basic parameters such as the arrival rate in derivative formulas to obtain the gradient estimate F^m. For reasons of tractability, this typically requires the assumption that the underlying distributions be exponential. These assumptions were made in [**?**] in the context of centralized load balancing. On the other hand, correctness proofs for the gradient methods of our choice (verifying assumption $A5$) also require similar assumptions. For example, structure of the gradient estimators [**?**, **?**] depends on the arrival process, which is usually taken to be Poisson. One may then ask why the more complicated gradient estimators in [?, ?] should be used in place of simpler formula-based estimators such as an estimator based on $M/M/1$ queueing formulas. In the first place, the IPA estimator discussed earlier retains the same structure over a large family of distributions, the so-called *scale* parameter distributions [?, ?]. In addition, there is some evidence [?] to suggest that this structure of the IPA estimator is robust to changes in the arrival distribution. Furthermore, the Finite Perturbation Analysis (FPA) estimator [?], which is used in the simulation results of this paper, is relatively insensitive to the underlying distributions in the sense that, provided the perturbation is small enough, it should not matter whether the parameter of interest is of the scale or location [?] type. However, it estimates only a first difference approximation of the gradient and thus one might say that $A5$ is only approximately justified. In any case, even when the distributions are known a priori, simulation results [?] indicate that the FPA and IPA estimators appear to perform just as well as the formula-based estimators. We note that our convergence result will hold for formula-based estimators as well.

4 Experimental Results

In many cases, as will be shown, we find that both stepsize and sampling control result in approximately the same performance, with sampling control providing only slightly better performance. However, in certain special instances, such as when constraint estimation is present, sampling control can be seen to be significantly better than stepsize control. Our simulation results, while not comprehensive in the scope of parameters, at least document the feasibility of using the two techniques.

The simulated system is the one shown in Figure ?? with three sources and nine processors and is intended to be an extension of the single source case presented in [?] in two ways: there are now multiple interacting sources instead of one source and, each unit of work (call) uses several resources in sequence. In the general routing problem [?] one usually associates a source with each node in the network and therefore would require routing variables to be present at each of the nine queues. Our simulation example models the situation where only sources at the edge of a network exercise control. However, we believe that, apart from being applicable to such networks, this model captures the important aspects of the general routing problem. The key elements of our stochastic control problem, including the general routing problem, simply involve conflicts between sources operating asynchronously and the fact that each sensitivity estimate must be constructed from samples taken at a number of processors. Our simulation model, as does our theoretical formulation, contains both of these important elements.

The arrival and service distributions were all taken to be exponential, and thus each processor is an $M/M/1$ queue with the arrival rates at the sources chosen as $\lambda_1 = \lambda_2 = \lambda_3 = 1$ and the service rates at the queues $\mu_1 = 5, \mu_2 = 2, \mu_3 = 1.1, \mu_4 = ... = \mu_9 = 2$. The differing service rates were deliberately chosen so as to require load balancing. We use the FPA estimator with the same constant for each queue ($\Delta r_i = 0.001$); a comparison with IPA is made later. We plot the expected call setup delay with progress in time; the abscissa is clock time whereas the ordinate is taken to be the expected setup delay for a randomly selected call arriving in steady state. Each plotted point is an average from 20 simulation runs and the confidence intervals were found to be less than one-fourth of each ordinate unit and are not shown on the graph.

4.1 Results

In Figure **??** we plot two curves in order to compare stepsize and sampling control in the multiple source case. The initial routing policy was taken to be $X^{(0)}(1) = (0.5, 0.5, 0)$, $X^{(0)}(2) = (0.5, 0.5, 0)$ and $X^{(0)}(3) = (0, 0.2, 0.8)$. This initial routing policy is used in all our simulations, except where otherwise noted. The following curves are plotted:

- ST_1 : stepsize-controlled version with $a_n = \frac{11.1}{n}$ and $L(n) = L = 1$ for each source c. Here and in each subsequent experiment, the best possible stepsize version was selected, with stepsize parameter a being chosen from a brief search in the interval $[0.01, 20.00]$.
- SA_2 : Sampling-controlled version with $a_n = 0.6$ and $L(n+1) = L(n) + 1$, $L(1) = 1$ for each c.

We observe that, as in the single source case [**?**], there is slightly better performance from sampling-control.

In Figure **??** a different initial routing policy is taken, $X^{(0)}(1) = (0.33, 0.33, 0.34)$, $X^{(0)}(2) = (0.33, 0.33, 0.34)$ and $X^{(0)}(3) = (0.33, 0.33, 0.34)$, for the purpose of comparison. The above initial policy is typically what would be used when no information about the service rates is known.

- ST_2 : Stepsize-controlled version with $a_n = \frac{0.6}{n}$ and $L(n) = 1$.
- SA_2 : Sampling-controlled version with $a_n = 0.1$ and $L(n+1) = L(n) + 1, L(1) = 1$.

Once again, the graphs indicate a small difference in the performance with sampling control resulting in slightly better performance.

In the previous two figures, the sources were synchronized by virtue of requiring the same numbers of samples from each processor. We now focus on the case that the sources operate at different speeds and in an asynchronous manner. Our simulation of this case is only an approximation since we did not actually attempt to match conditions found in a network of operating systems. Our study, which constitutes a worst-case analysis, may be explained as follows. The source speeds were chosen as $1, \frac{1}{5}$ and $\frac{1}{10}$ with the first source being the fastest and the third, the slowest. Thus, under stepsize control for example, for every ten updates that source 1 makes, the second source makes two updates and the third makes one update. This remains the pattern with stepsize-control. In

sampling-control, however, once the number of samples becomes very large, each source will spend more time waiting for large-sample estimates and, thus the sources tend to get synchronized in their optimization iterations.

In Figure **??** we plot the following curves for the asynchronous case:

- ST_3 : Stepsize-controlled version with $a_n = \frac{0.6}{n}$ and $L(n) = 1$.
- SA_3 : Sampling-controlled version with $a_n = 0.6$ and $L(n+1) = L(n) + 1, L(1) = 1$.

The asynchronous operation of sources with different speeds exaggerates the difference between the two types of stochastic approximation. We conjecture that the reason for the good performance of sampling-controlled stochastic approximation lies in the eventual synchronization of the sources; the stepsize control version, on the other hand, results in the faster sources having to wait for the slower sources. We emphasize that the above scenario only constitutes a worst-case scenario and, in an implementation, stepsize control may perform much better than indicated above.

5 Unknown Constraints

We now consider the case where the *inequality* constraints in our setup problem are unknown and must themselves be estimated from observing the system. When b_i and c_i are unknown, estimators $\hat{b}_i$ and $\hat{c}_i$ may be constructed from observing the external arrival rates and service completions at the queues. Since $\hat{c}_i^{(n)}$ and $\hat{b}_i^{(n)}$ are noisy estimates, their use in our algorithms could result in the generation of infeasible vectors $X^{(n)}$, i.e., $X_i^{(n)} > b_i$ or $X_j^{(n)} < c_j$ for some i and j. For example, it is possible to underestimate the upper constraint and overload a particular queue.

In order to handle this problem, we will assume that $\hat{c}_i^{(n)}$ and $\hat{b}_i^{(n)}$ are strongly consistent. Loosely speaking, as the estimates $\hat{c}_i^{(n)}$ and $\hat{b}_i^{(n)}$ get better (i.e., $\hat{c}_i^{(n)} \approx c_i$ and $\hat{b}_i^{(n)} \approx b_i$), the probability of generating infeasible vectors is reduced. More precisely, we assume the existence of a real number ζ such that $0 < \zeta < \inf_{i,x\in S^*} \min\left(\frac{x_i - c_i}{2}, \frac{b_i - x_i}{2}\right)$ and define $b_i^{(n)} = \hat{b}_i^{(n)} - \zeta$ and $c_i^{(n)} = \hat{c}_i^{(n)} + \zeta$. By forcing $X_i^{(n)} \geq c_i^{(n)}$ and $X_i^{(n)} \leq b_i^{(n)}$, eventually for large enough n we will have $c_i \leq X_i^{(n)} \leq b_i$. Whenever the constraint $c_i^{(n)} \leq X_i^{(n)} \leq b_i^{(n)}$ is violated, a heuristically close value of $X^{(n)}$ is substituted to satisfy the constraint (a simple mechanism for achieving this is given in [**?**]).

We now return to the simulation system and, in figure **??**, plot a comparison between stepsize and sampling-controlled versions for the case in which the constraints are unknown at each queue. The constant ζ is taken to be 10^{-6} and estimates are maintained of each bound $\hat{b}_i^{(n)}$. The following curves are plotted:

- ST_5 : Stepsize-controlled version with $a_n = \frac{11.1}{n}$ and $L(n) = L = 1$ for each source c.
- SA_5 : Sampling-controlled version with $a_n = 0.6$ and $L(n+1) = L(n) + 1$, $L(1) = 1$ for each c.

For the same configuration as the first set of curves used earlier (figure **??**), we observe a significant difference between stepsize and sampling-control, the latter exhibiting better performance.

6 Different Estimation Schemes

In [**?**], some simulation results were presented for the case in which different (likelihood ratio [**?**]) estimators were used in place of Perturbation Analysis, which indicated that better performance resulted from using Perturbation Analysis. We now compare Finite and Infinitesimal Perturbation Analysis for the same system and configuration as described earlier by plotting curve SA_1 (which uses FPA) together with:

- SA_{IPA} : Sampling-controlled version using the IPA estimator [**?**] with $a_n = 0.2$ and $L(n+1) = L(n) + 1, L(1) = 1$ for each c.

The figure shows that the (heuristic) FPA estimator performs just as well as the theoretically correct (strongly consistent) IPA estimator.

As noted in [**?**], many gradient estimators are *biased* including the IPA and FPA estimators in this paper. It may be conjectured that this bias is the cause for poorer performance in the case of stepsize-control. We next explore this possibility through the removal of bias through using combinations of samples from different busy periods. The estimator $Y^n(\gamma)$ is biased due to the fact that (see equation (**??**) in Appendix B) $U^{(n)}(\gamma)$ is a nonlinear function and also that the random variables $A_j(i)$ and N_i are correlated for each j. The bias problem is further compounded by the inclusion of the rate estimator $\hat{\gamma}$. If we assume that the rates are known, the bias in equation (**??**) can be removed by using random variables from different regeneration periods; in this case, the numerator and denominator in (**??**) are taken from different busy periods. However, as Figure **??** shows the results for the multiple source system of the previous section, the added noise in using random variables from different busy periods is so high that convergence is slower:

- ST_6 : Stepsize control with $a_n = \frac{0.6}{n}$ and $L(n) = 1$.

We note that stepsize control does achieve as good estimates as sampling control in many cases, even with both methods using the FPA heuristic and thus, one may have to look elsewhere to evaluate the effects of bias. We note that the particular form of the algorithm in equation (**??**) suggests that estimator bias is significantly reduced through via subtraction of the average and hence may not play an important role for our problem of interest.

7 Conclusions

In this paper we formulated a problem addressing the efficiency of the call setup procedure in circuit- and packet-switched systems. We argued that with increasing demands placed on network operating systems, a load balancing mechanism installed among call processors could achieve improved setup times. One of the main goals was to achieve this load balancing on-line during the operation of the system and to do so using measurements taken from the system. In this case, traditional gradient-based iterative optimization algorithms must be modified to handle stochasticity in their input through stochastic approximation techniques. Our main contribution has been a formal theoretical treatment of a simplified version of this problem. We presented an algorithm that generalized earlier optimization algorithms for this type of problem and proved its convergence. Through simulation we have made comparisons between two stochastic approximation techniques and investigated several other interesting issues.

8 Appendix A: Derivative Estimation

We first describe the Infinitesimal Perturbation Analysis (IPA) estimator [**?**] of the derivative of the delay with respect to the arrival rate for a *single* queue and then show how estimators of this kind for single queues are then combined to obtain F^m.

For a single queue let N_i denote the i-th busy period, γ the arrival rate and $A_j(i)$ the interarrival time of the j-th unit in the i-th busy period. Thus, if units j and $j-1$ arrive at times $a_j(i)$ and $a_{j-1}(i)$ then $A_j(i) = a_j(i) - a_{j-1}(i)$. Define

$$U^{(n)}(\gamma) \;=\; \frac{\sum_{i=1}^{n}\sum_{k=2}^{N_i}\sum_{j=2}^{k} A_j(i)}{\gamma\sum_{i=1}^{n} N_i} \tag{3}$$

If we denote the steady state expected mean delay of the queue by w then for the $M/G/1$ queue it can be shown [**?**] that the estimator $U^{(n)}(\gamma)$ converges to the derivative of the mean delay with respect to the arrival rate: $\lim_{n\to\infty} U^{(n)}(\gamma) = \frac{dw}{d\gamma}$ $a.s.$ In this paper we are interested in the case in which γ itself is unknown and must be estimated on-line. The above estimator is easily modified to include estimation of γ. Let

$$\hat{\gamma}^{(n)} \;=\; \frac{\sum_{i=1}^{n} N_i}{\sum_{i=1}^{n}\sum_{j=1}^{N_i} A_j(i)}.$$

Then, since γ is one of two multiplicands in $U^{(n)}(\gamma)$, and assuming the interarrival times are an i.i.d. sequence,

$$\lim_{n\to\infty} Y^n(\gamma) = \lim_{n\to\infty} U^{(n)}(\hat{\gamma}^{(n)}) = \frac{dw}{d\gamma} \quad a.s.$$

Of course, we are interested in estimating the derivatives $f_i(x_i)$; we now construct such an estimator using the derivative estimators for each queue. Let $Y_i^n(\gamma_i)$ denote the IPA estimator for the ith queue and $\gamma_i(x)$ the arrival rate to the ith queue depending on the routing policy x. By summing the expected number in each queue and using Little's Law it can be shown that [**?**]:

$$\frac{\partial D(x,\Theta)}{\partial x_i} = \frac{1}{\sum_{i=1}^{C} \lambda_i} \sum_{i=1}^{R} \frac{d\gamma_i}{dx_i} \left(w_i + \gamma_i \frac{dw_i}{d\gamma_i} \right)$$

Hence, we can use the above equation to construct the estimator F_i^m for $\frac{\partial D(x,\Theta)}{\partial x_i}$ in terms of $\hat{\gamma}_i^m$ and Y_i^m such that

$$\lim_{m \to \infty} F_i^m = \frac{\partial D(x,\Theta)}{\partial x_i} \quad a.s.$$

The derivative estimates F_i^m are then used in the stochastic optimization algorithm at each source.

9 Appendix B

PROOF OF LEMMA 1 That the equality constraints are satisfied is easily checked by summing $\sum_{i \in A_c^{(n)}} X_i^{(n+1)}$ in equation (**??**). We focus now on proving that if $c_i \le X_i^{(n)} \le b_i$ then $c_i \le X_i^{(n+1)} \le b_i$, where the index i refers to a particular control variable $i \in K_c$. We do this by induction on the number of elements of the set K (or, equivalently, $A_c^{(n)}$), in the order in which K is constructed. Clearly, from equation (**??**) we have the constraints satisfied for $|K| = 1$ (here, $|K|$ denotes the cardinality of K). Let $|K| = m-1, m \ge 2$ and consider the addition of the m^{th} element into K. For the upper constraint, $X_i^{(n+1)} \le b_i$ we must have, using equation (**??**) $\forall i \in K \cup \{m\}$:

$$X_i^{(n)} + a_n \left(M_{K\cup\{m\}}^{(n)} - F_i^{L(n)} \right) \le b_i$$

i.e.,

$$M_{K\cup\{m\}}^{(n)} \le F_i^{L(n)} + \frac{b_i - X_i^{(n)}}{a_n}$$

Similarly, from the lower constraint $X_i^{(n+1)} \ge c_i$ we obtain the following condition:

$$M_{K\cup\{m\}}^{(n)} \ge F_i^{L(n)} + \frac{c_i - X_i^{(n)}}{a_n}$$

Note that we have removed the additional complication of using the sort indices in the proof here. The above two inequalities are exactly the inequalities checked inside the **for** loop in the computation of $A_c^{(n)}$ and thus, we get our result that $c_i \le X_i^{(n+1)} \le b_i$. $\square$.

We will use the results developed in [?] in our convergence proof. Note that we have assumed $A1$ in [?] (existence of moment generating functions) and $A2$ in [?] holds from the definition of the gradient estimator F^m. If assumption $A3$ in [?] could be shown, then we could use their Theorem 3 for our algorithm. However, due to the multiplicity of solutions in S^*, we prove a variation of their $A3$ and modify the proof to obtain our result. As in [?], we distinguish between *deterministic* and *stochastic* versions of the algorithm. The algorithm we have described so far is stochastic because *estimates* of the gradient (F^m) are used in place of the true gradient (f). We refer to the iteration as the deterministic version (for the purposes of this proof) when the true gradient is used in the algorithm:

$$x_i^{(n+1)} = x_i^{(n)} + a_n H_i\left(f(x^{(n)}), x^{(n)}\right) \tag{4}$$

Note that f replaces F^m in equation (**??**) and we use lowercase letters to emphasize determinism. The following lemma establishes $A3$ in [?].

LEMMA 2: . Let $f^{(n)}$ denote the deterministic gradient $f(x)$ evaluated at $z^{(n)}$. There exists α such that the following statement holds when $a_n = a < \alpha$. Given $\delta > 0$, $\exists N < \infty$ and $\epsilon > 0$ such that using the perturbed recursion

$$\bar{D}^{(n+1)} = D\left(z^{(n+1)}, \Theta\right)$$

where the recursion for z is defined as:

$$z_i^{(n+1)} = z_i^{(n)} + H_i\left(f^{(n)} + \epsilon^{(n)}, z^{(n)}\right)$$

with H_i defined as in equation (**??**), then, $\forall n > N$ and $\sup_n |\epsilon_n| < \epsilon$ it is true that

$$\left|\bar{D}^{(n+1)} - D^*\right| < \delta$$

for $N \leq n \leq 2N$.

PROOF: For convenience, we will drop the dependence on Θ and write $D(x) = D(x, \Theta)$. Let

$$\begin{aligned}
z(\gamma) &= (1-\gamma)z^{(n)} + \gamma z^{(n+1)} \\
\Delta\bar{D}(\epsilon^{(n)}) &= D(z^{(n+1)}) - D(z^{(n)}) \\
S_D(\delta) &= \{z : D(z) - D^* > \delta\} \\
\Delta z_i &= z_i^{(n+1)} - z_i^{(n)} \\
D(\gamma) &= D(z(\gamma))
\end{aligned}$$

In this lemma will use the proof technique in [?] for deterministic algorithms. By Taylor's theorem,

$$\Delta\bar{D}(\epsilon^{(n)}) = \left.\frac{dD(\gamma)}{d\gamma}\right|_{\gamma=0} + \frac{1}{2}\left.\frac{d^2D(\gamma)}{d\gamma^2}\right|_{\gamma=\check{\gamma}}$$

for some $0 \le \check{\gamma} \le 1$. We first focus on the contribution to $\Delta \bar{D}(\epsilon^{(n)})$ from a single source. Let

$$\Delta \bar{D}_c(\epsilon^{(n)}) = \sum_{i \in A_c^{(n)}} f^{(n)} \Delta z_i + \frac{1}{2} \sum_{i,j \in A_c^{(n)}} \left. \frac{\partial^2 D}{\partial z_i \partial z_j} \right|_{z=z(\check{\gamma})} \Delta z_i \Delta z_j \tag{5}$$

Then, $\Delta \bar{D}(\epsilon^{(n)}) = \sum_{c=1}^{C} \Delta \bar{D}_c(\epsilon^{(n)})$. We will demonstrate a stepsize a such that when $\delta > 0$ is given, there exists $\epsilon > 0$ for which $\Delta \bar{D}_c(\epsilon^{(n)}) < 0$ for $z^{(n)} \in S_D(\delta)$ whenever $\left|\epsilon^{(n)}\right| < \epsilon$. Thus, $\exists N : \left|\bar{D}(z^{(n)}) - D^*\right| < \delta$ for all $n > N$, in particular, for $N \le n \le 2N$.

Define

$$d(A_c^{(n)}) = \sup_{i,j \in A_c^{(n)}} \left| f_i^{(n)} - f_j^{(n)} \right|$$

Noting that $A_c^{(n)} \cap B_c \neq \emptyset$, we observe that $d(A_c^{(n)}) > 0$ for $z^{(n)} \in S_D(\delta)$, otherwise the Kuhn Tucker conditions will be satisfied at $z^{(n)}$. Let $\nu = \inf_{z \in S_D(\delta)} d(A_c^{(n)})$. Next, define

$$h_i(x) = x_i - \frac{1}{\left|A_c^{(n)}\right|} \sum_{j \in A_c^{(n)}} x_j, \qquad i \in A_c^{(n)}$$

Then, substituting from equation (**??**) for Δz_i in equation (**??**), we get

$$\begin{aligned}\Delta \bar{D}_c(\epsilon^{(n)}) = & -a \sum_{i \in A_c^{(n)}} f_i^{(n)} h_i\left(f^{(n)} + \epsilon^{(n)}\right) \\ & + \frac{1}{2} a^2 \sum_{i,j \in A_c^{(n)}} \left. \frac{\partial^2 D}{\partial z_i \partial z_j} \right|_{z=z(\check{\gamma})} h_i\left(f^{(n)} + \epsilon^{(n)}\right) h_j\left(f^{(n)} + \epsilon^{(n)}\right)\end{aligned}$$

Next consider what happens when $\epsilon^{(n)} = 0$, i.e., when no perturbations are used. This corresponds to the deterministic version of the algorithm. When $\epsilon^{(n)} = 0$, the first term above can be rearranged as in Theorem 1 of [**?**] and since the second derivatives are bounded (less than M, say) by assumption $A1$, we have

$$\Delta \bar{D}_c(0) \le -a \sum_{i \in A_c^{(n)}} h_i^2(f^{(n)}) + \frac{a^2 C(C-1) M}{2} \sum_{i \in A_c^{(n)}} h_i^2(f^{(n)})$$

Now, $h_i^2(f^{(n)}) \ge d^2(A_c^{(n)}) \ge \nu^2 > 0$. By choosing $a < \frac{2}{C(C-1)M}$, $\sup_{z \in S_D(\delta)} \Delta \bar{D}_c(0) < 0$.

Finally, consider the construction of $A_c^{(n)}$ when $\epsilon^{(n)} = 0$ and let $A_c^{(n)}(\epsilon^{(n)})$ be the set obtained in Step 1 of the algorithm when $f^{(n)} + \epsilon^{(n)}$ is used in place of $f^{(n)}$ (i.e., when using the perturbed version). We may choose ϵ' small enough so that $\left|\epsilon^{(n)}\right| < \epsilon'$ implies that exactly those elements in $A_c^{(n)}$ enter K in the **while**-loop of Step 1, i.e., $A_c^{(n)}(\epsilon) = A_c^{(n)}$. This means that h (which is linear in $\epsilon^{(n)}$) is continuous in $\epsilon^{(n)}$. Hence, $\exists\epsilon < \epsilon'$ such that $\left|\epsilon^{(n)}\right| < \epsilon$ implies $\sup_{z^{(n)}\in S_D(\delta)} \Delta D_c(\epsilon^{(n)}) < 0$, which was to be shown. We have incidentally also shown convergence for the deterministic version of the algorithm. We note that for some systems, the one-step strict decrease ($\Delta\bar{D}_c(0) < 0$) need not occur; however, for these systems it is possible to show that $\Delta\bar{D}_c(0) \leq 0$ and that strict decrease occurs after a fixed number of iterations, i.e., $\exists M_a : D(z^{(n+M_a)}) - D(z^{(n)}) < 0$. This case is discussed in detail in [**?**]. The above proof is easily modified to observe the algorithm every M_a iterations.

PROOF OF THEOREM: Let $D^{(n)} = D(X^{(n)})$. We will follow the proof technique in Theorem 3 of [**?**] and show that $P\left[\left|D^{(n)} - D^*\right| > \delta\ i.o\right] = 0$, i.e., that $D^{(n)} \to D^*\ a.s.$ Choosing N and ϵ from Lemma 2, we define

$$E_i = \left\{\left|D^{(n)} - D^*\right| > \delta \text{ for some } n,\ iN \leq n \leq iN + N\right\}$$

Then, following [**?**] one may show that ,

$$P\left[E_i \left| D^{(iN-N)} = d\right.\right] \leq \sum_{n=iN-N+1}^{iN+N} P\left[\left|F^{L(n)} - f\right| > \epsilon \left| D^{(iN-N)} = d\right.\right]$$

Hence we can follow the rest of the proof of Theorem 3 in [**?**] to conclude convergence.

Acknowledgements: The authors would like to thank Professor Paul Dupuis of the Department of Applied Mathematics, Brown University and Professor Don Towsley of the Computer Science Department, University of Massachusetts, Amherst for several helpful discussions. We would also like to thank an anonymous referee for substantial suggestions in improving the quality of the presentation.

References

[1] A.Becker, P.R.Kumar and C-Z.Wei, "Adaptive Control with the Stochastic Approximation Algorithm: Geometry and Convergence", *IEEE Transactions on Automatic Control*, Vol. AC-30, pp. 330-338, 1985.

[2] D.Bertsekas, E.Gafni and R.Gallager, "Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks," *IEEE Transactions on Communications*, Vol COM-32,pp. 911-919, 1984.

[3] D.G.Cantor and M.Gerla, "Optimal Routing in a Packet-switched Computer Network", *IEEE Transactions on Computers*, Vol. C-23, pp. 1062-1069, 1974.

[4] C.G.Cassandras, W.Gong and J.I.Lee, "On the Robustness of Perturbation Analysis Estimators for Queueing Systems with Unknown Distributions", *J. Opt. Theory and Appl.*, to appear.

[5] C.G.Cassandras, M.V.Abidi and D.Towsley,"Distributed Routing with on-Line Marginal Delay Estimation", *IEEE Transactions on Communications*, Vol. COM-38, No. 3, pp. 348-359, 1990.

[6] F.Chang and L.Wu, "An Optimal Adaptive Routing Algorithm", *IEEE Transactions on Automatic Control*, Vol. AC-31, pp.690-700, 1986.

[7] P.Chen, "Optimal File Allocation in Multilevel Storage Systems", *Proc. AFIPS*, VOl. 42, pp. 277-282, 1973.

[8] Y-C.Chow and W.H.Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System", *IEEE Transactions on Computers*, Vol. C-28, pp. 354-361, 1979.

[9] I.Cidon, I.Gopal, M.Kaplan and S.Kutten, "Distributed Control in PARIS', *Proc. 9th Annual Symposium on Principles of Distributed Computing*, Quebec, Canada, 1990.

[10] P.Dupuis and R.Simha, "On Sampling-controlled Stochastic Approximation", *IEEE Transactions on Automatic Control*, Vol. 26, No. 8, pp. 915-924, 1991.

[11] P.Dupuis and H.J.Kushner, "Stochastic Approximation and Large Deviations: Upper Bounds and w.p.1 Convergence", *SIAM Journal of Control and Optimization*, Vol. 27, pp.1108-1135, 1989.

[12] E.M.Gafni and D.P.Bertsekas, "Path Assignment for Virtual Circuit Routing", *ACM SIGCOMM*, pp. 21-25, 1983.

[13] R.G.Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Trans. on Communications*, Vol COM-25, pp. 73-85, 1977.

[14] A.Girard, Routing and Dimensioning in Circuit-switched Networks, *Addison-Wesley*, Reading, MA, 1990.

[15] B.Hajek, "Stochastic Approximation Methods for Decentralized Control of Multiaccess Communications", *IEEE Transactions on Information Theory*, Vol. 31, pp. 176-184, 1985.

[16] G.Heal, "Planning Without Prices," *Review of Economic Studies*, Vol. 36, pp. 346-362, 1969.

[17] P.Heidelberger, X-R.Cao, M.A.Zazanis and R.Suri, "Convergence Properties of Infinitesimal Perturbation Analysis Estimates", *Management Science*, Vol. 34, pp. 1281-1302, 1988.

[18] Y.C.Ho, L.Servi and R.Suri, "A Class of Center-Free Resource Allocation Algorithms," *Large Scale Systems*, Vol. 1, pp. 51-62, 1980.

[19] Y.C.Ho, X.R.Cao and C.Cassandras, "Finite and Infinitesimal Perturbation Analysis of Queueing Networks", *Automatica*, Vol. 19, pp. 439-445, 1984.

[20] I.Kogiku, T.Ohrui, T.Ohkubo and Y.Hamada, "A Real-time Portable Operating System Common to Switching and Information Processing Applications", *International Switching Symposium*, pp. B4.4.1-B4.4.5, 1987.

[21] A.Kumar and F.Bonomi, "Adaptive Load Balancing in a Multiprocessor System with a Central Job Scheduler", *Proc. 2nd Int. Workshop on Applied Mathematics and Performance/Reliability Models of Computer Communication Systems*, Univ. of Rome II, pp. 173-188, 1987.

[22] J.Kurose and R.Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems," *IEEE Transactions on Computers*, Vol. 38, pp. 705-717, 1989.

[23] H.J.Kushner and D.S.Clark, Stochastic Approximation Methods for Constrained and Unconstrained Systems, *Springer Verlag*, New York, 1978.

[24] D.A.Lawson, "A New Software Architecture for Switching Systems", *IEEE Transactions on Communications*, Vol. COM-30, pp. 1281-1289, 1982.

[25] D.G.Luenberger, Linear and Nonlinear Programming, *Addison-Wesley*, 1984.

[26] L.M.Ni and K.Hwang, "Optimal Load Balancing in a Multiple Processor System With Many Job Classes", *IEEE Transactions on Software Engineering*, Vol. SE-11, pp. 491-496, 1985.

[27] S.D.Personick and W.O.Fleckstein, "Communications Switching - From Operators to Photonics", *Proceedings of the IEEE*, Vol. 75, pp. 1380-1403, 1987.

[28] M.I.Reiman and A.Weiss, "Sensitivity Analysis for Simulations Via Likelihood Ratios", *Operations Research*, Vol. 37, pp. 830-844, 1989.

[29] D.Russell, Optimization Theory, *W.A.Benjamin Inc*, New York, 1970.

[30] M.Schwartz and T.E.Stern, "Routing Techniques Used in Computer Networks", *IEEE Transactions on Communications*, Vol. COM-28, pp. 539, 1980.

[31] M.Schwartz, Telecommunication Networks, Chapter 11, *Addison-Wesley*, Reading, Mass, 1987.

[32] R.Simha and J.F.Kurose, "Stochastic Approximation Schemes for a Load Balancing Problem", *Proc. 27th Allerton Conf. on Communication, Control and Computing*, Allerton, 1989.

[33] R.Simha and J.F.Kurose, "On-line Minimization of Call Setup Time via Load Balancing: A Stochastic Approximation Approach", Technical Report, Department of Computer Science, College of William & Mary, Williamsburg, Virginia.

[34] R.Suri, "Perturbation Analysis: The State of the Art and Research Issues Explained Via the GI/G/1 Queue", *Proceedings of the IEEE*, Vol. 77, 114-137, 1989

[35] R.Suri, "Implementation of Sensitivity Calculation on a Monte Carlo Experiment", *J.Optimization Theory and Applications*, Vol. 40, No.4, 1983.

[36] A.N.Tantawi and D.Towsley, "Optimal Static Load Balancing in Distributed Computer Systems", *Journal of the ACM*, Vol. 32, pp. 445-465, 1985.

[37] J.N.Tsitsiklis and D.P.Bertsekas, "Distributed Asynchronous Optimal Routing in Data Networks," *IEEE Transactions on Automatic Control*, Vol Ac-31, pp. 325-332, 1986.

[38] Y.Wardi and K.Lee, "Application of Descent Algorithms with Armijo Stepsizes to Simulation Based Optimization of Queueing Networks", *Proc. 30th Conf. on Decision and Control*, Brighton, England, December 1991.

[39] J.Yang and H.J.Kushner, "A Monte Carlo Method for Sensitivity Analysis and Parametric Optimization of Nonlinear Stochastic Systems", *SIAM J. Control & Optimization*, Vol. 29, No. 5, pp. 1216-1249, 1991.

[40] J.R.Yee, "Optimal Distributed Routing Algorithms for Datagram Communication Networks",*Proc. 27th Allerton Conf. on Communication, Control and Computing*, Allertion, 1989.

[41] T-K.Yum, "The Design and Analysis of a Semidynamic Deterministic Routing Rule", *IEEE Transactions on Communications*, Vol. COM-29, pp. 498-504, 1981.

[42] C.Ziegler, G.Weiss and E.Friedman, "Implementation Mechanisms for Packet Switched Voice Conferencing", *IEEE Journal Selected Areas in Communications*, Vol. 7, pp. 698-706, 1989.