DISTRIBUTED ROUTING WITH ON-LINE MARGINAL DELAY ESTIMATION

C. Cassandras, M. Abidi and D. Towsley

COINS Technical Report 88-12

February 11, 1988

DISTRIBUTED ROUTING WITH ON-LINE MARGINAL DELAY ESTIMATION [1]

Christos G. Cassandras and M. Vasmi Abidi

Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
Tel. 413 - 545 1340

Don Towsley

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Most optimal routing algorithms for packet-switched networks require information about the sensitivity of the performance measure with respect to link flows. This information is generally difficult to obtain for real-time applications, due to the absence of closed form expressions for performance as a function of flows; in most interesting cases, standard assumptions (exponentially distributed packet lengths, Poisson arrival processes) do not hold. In this paper, we present a procedure for estimating *on-line* marginal packet delays through links with respect to link flows without making such assumptions, based on a technique known as Perturbation Analysis (PA). No knowledge of network parameters is required (arrival rates, link capacities). This is used in the context of a minimum delay distributed routing algorithm for real-time implementation. Experimental results are included investigating the effect of the algorithm step-size and observation period parameters, demonstrate the adaptivity of the approach and compare it to well-known analytical approximations.

July 1987

---

[1] This work was partly supported by NSF under grant ECS-8504675 and ONR under contract N00014-87-K-0304.

1. INTRODUCTION

The routing of packets from source node to destination node is an important issue in the design of packet-switched networks, since it affects several performance measures of interest. By routing we mean the set of decisions regarding the outgoing link to be used for transmitting messages at each network node. The objective of a routing algorithm is to optimize some performance measure, e.g. the mean packet delay or network throughput.

Routing may be done in a *centralized* or *distributed* manner. In the former case, a special node in the network, called the Routing Control Center (RCC), periodically receives information from all other network nodes and, based on this global information, it sets up and updates routing tables for all nodes. This method suffers from high communication overhead, and must deal with the problem of handling link and node failures. An even more serious problem is how to handle the failure of the RCC itself.

Distributed routing avoids some of these problems. In this case, each node makes its own routing decisions based on the (local) information it receives from its neighboring nodes. One potential problem here is that inconsistent routing paths can cause looping of packets and possibly deadlocks.

The problem of routing has been investigated using different approaches. Some are based on heuristics (e.g. shortest path algorithms, as in ARPANET), while others formulate the routing problem as one of optimal control and then attempt to solve it using standard optimization techniques ([3], [7], [14]). In general, the optimal routing problem reduces to minimizing (or maximizing) some objective function $D(f_{ik})$ with respect to the variables f_{ik}, where f_{ik} denotes the flow on link (i,k), subject to certain constraints such as flow conservation and non-negativity of flows. Optimal routing algorithms may differ in the precise formulation of the problem, and in the choice of the objective function. However, a common feature is the requirement for knowledge of the derivatives of the performance measure (e.g. the mean packet delay) with respect to the control

(e.g. the link flows f_{ik}). Therefore, it becomes important to be able to estimate these derivatives as efficiently and accurately as possible.

Gallager [8] has defined an interesting algorithm to solve the distributed routing problem in a quasistatic environment. A quasistatic environment is one where the offered traffic statistics for each origin-destination pair change slowly over time, and individual traffic functions do not show large and persistent deviations from the average. In such a case, it is valid to base routing decisions on the expected values of the flows. Routing changes are made periodically, or when required (say, due to a link failing). Thus, from the viewpoint of *adaptivity* to changing network conditions, "quasistatic" algorithms such as Gallager's are a sensible compromise between the purely static (routing decisions fixed in advance) and the fully dynamic (routing decisions based on continuously observed state information) routing algorithms.

Like most other optimal routing algorithms, Gallager's algorithm assumes knowledge of the delay gradient $D'_{ik}(f_{ik})$. The question is: how does one determine the derivatives of delays over the links? It can be done analytically if an appropriate formula giving the link delay as a function of the link flows is available. For example, Kleinrock [11] has shown that the average total delay/sec over a link is given by :

$$D_{ik}(f_{ik}) = \frac{f_{ik}}{C_{ik} - f_{ik}}$$

where f_{ik} denotes the amount of traffic on link (i,k), and C_{ik} is the capacity of link (i,k) (expressed in the same units as f_{ik}). This expression, however, is based on the assumptions of Poisson arrivals at nodes, exponentially distributed packet lengths, and the "independence assumption" of service times at successive nodes (i.e. when a packet arrives at a node, a new length is assigned to it from some common exponential distribution). Such assumptions do not hold in practice. Hence, instead of seeking closed form expressions for $D_{ik}(f_{ik})$, it is preferable to estimate the derivative of the delay directly, i.e. by using data available in a real network.

Few techniques have been proposed for on-line estimation of performance gradients of systems modeled as queueing networks. Segall [15] has proposed a "customer rejection" algorithm suitable

for estimating mean delay gradients in individual network links. Of greater generality are the Perturbation Analysis (PA) ([9], [10]) and Likelihood Ratio (LR) [13] techniques. In this paper, we show how PA can be used to estimate *on-line* the delay gradients $D'_{ik}(f_{ik})$, and be efficiently integrated in a minimum delay routing algorithm. As we shall see, PA posesses several attractive properties for our purposes, including lower variance estimates. In contrast to analytical models, links need not be characterized by exponentially distributed tranasmission times, and the processes for external arrivals at nodes need not be Poisson. Furthermore, the "independence assumption" is relaxed packets entering the network retain their identity (packet length) as they traverse links. An important feature of practical interest in our PA-based approach is that the parameters of the network (external arrival rates, link capacities) need not be known.

Using PA link delay gradient estimates, we have implemented Gallager's Minimum Delay routing algorithm on several simulated networks. We have identified two important parameters, the *observation period* and the *step size*, and have studied the performance of the algorithm with respect to these parameters. In some cases, we also discuss simple enhancements that can improve its performance.

In section 2 we give an overview of the routing algorithm considered here, which makes clear the need for estimation of the link delay gradients. Section 3 of the paper describes the Perturbation Analysis (PA) methodology, on the basis of which the algorithm used to estimate the delay gradients is derived. This section also compares the PA procedure to the estimation algorithm suggested by Segall [15] and to the Likelihood Ratio (LR) approach [13]. In section 4, we describe the models used for simulation, and investigate the performance of the algorithm for different cases - as the traffic pattern becomes more complex, as the size of the network increases, and as sudden changes occur in the network. A comparison with the routing algorithm using M/M/1 approximation models for links is also included. The paper concludes with section 5, where the practical significance of these results is discussed and some guidelines are presented for routing in practical cases.

2. MINIMUM DELAY ROUTING ALGORITHM

In this section we give a brief description of the main features of the algorithm suggested by Gallager [8]. Further details can be found in the original reference. The algorithm uses distributed computation to achieve optimal minimum delay routing, and is well-suited for packet-switched networks where the quasistatic assumptions hold. Each node constructs its own routing tables based on periodic updating information from neighboring nodes. Packets are sent over routes seeking to minimize the overall delay.

We first define some notation to be used. The ordered pair (i,k) denotes the directed link from node i to node k. In addition:

$r_i(j)$ = expected traffic entering the network at node i and destined for node j in packets/sec.

$t_i(j)$ = total expected traffic at node i destined for node j in packets/sec.

f_{ik} = expected traffic over link (i,k) in packets/sec; also called the *link flow*.

τ_{ik} = mean interarrival time of packets on link (i,k), i.e. $\tau_{ik} = 1/f_{ik}$.

$\phi_{ik}(j)$ = fraction of the traffic $t_i(j)$ that is routed over link (i,k); also referred to as the *routing variable* for link (i,k).

Network congestion is typically measured as some function of the flows f_{ik}. In this algorithm the measure considered is:

$$D_T = \sum_{(i,k)} D_{ik}(f_{ik})$$

where $D_{ik}(f_{ik})$ is the *average number of packets in queue or under transmission* at link (i,k). Note that, by Little's law, D_T is proportional to the mean delay of packets in the network. Hence, for a given (external) traffic input, minimizing D_T is equivalent to minimizing the mean packet delay.

We can now pose the optimal routing problem as follows: find a set of routing variables $\{\phi_{ik}(j)\}$ which, for a fixed and given set of inputs $\{r_i(j)\}$, minimizes the objective function D_T. Furthermore, in the context of distributed algorithms, we wish each node i to choose its own routing variables $\phi_{ik}(j)$ for each k,j.

It is shown in [8] that, in order to minimize D_T, each node i must incrementally decrease those routing variables $\phi_{ik}(j)$ for which the *marginal delay* defined as:

$$D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \qquad -(1)$$

is large, and increase those for which it is small.

Here, $D'_{ik} = \partial D_{ik}/\partial f_{ik}$ is the *sensitivity* of the mean packet delay/sec (i.e. the mean queue length) over link (i,k) with respect to the link flow.

The second term in the marginal delay expression (1) can be computed recursively as:

$$\frac{\partial D_T}{\partial r_i(j)} = \sum_k \phi_{ik}(j) \left\{ D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right\} \qquad -(2)$$

with

$$\frac{\partial D_T}{\partial r_j(j)} = 0$$

The first term in (1), i.e. the link delay sensitivity $D'_{ik}(f_{ik})$, will be estimated using the Perturbation Analysis technique, described in section 3.

The algorithm uses the following iterative scheme to update the routing variables at the (n+1)th iteration :

$$\phi_{ik}^{(n+1)}(j) = \begin{cases} \phi_{ik}^{(n)}(j) - \Delta_{ik}(j) & k \neq k_{min} \\ \phi_{ik}^{(n)}(j) + \sum_{k \neq k_{min}} \Delta_{ik}(j) & k = k_{min} \end{cases} \qquad -(3)$$

where:

$$\Delta_{ik}(j) = \min \left[\phi_{ik}^{(n)}(j), \eta a_{ik}(j)/t_i(j) \right]$$

$$a_{ik}(j) = \left[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right] - \min_n \left[D'_{in}(f_{in}) + \frac{\partial D_T}{\partial r_n(j)} \right]$$

and k_{min} is the neighboring node to node i such that the link (i,k_{min}) has the minimum marginal delay, and η is a scaling factor to be discussed below.

Hence, the algorithm reduces the fraction of traffic sent on non-optimal links and increases the fraction on the best link. The amount of reduction is proportional to the difference between the marginal delay of traffic to node j using link (i,k) and the marginal delay using the best link. It also depends on a scaling factor η , called the *step size* of the algorithm. As expected, η plays an important role in determining the convergence properties of the algorithm. In fact, the convergence of the algorithm has been proved for very small values of η. It will be seen, however, that η can be made considerably larger in practice.

It is worthwhile emphasizing the *distributed* nature of this algorithm: as is clear from (3), the routing variables at node i are adjusted based on information collected from neighboring nodes (indexed by k) only. Yet, the algorithm minimizes the global performance measure D_T.

A practical implementation of this algorithm requires the answer to two questions:

1. How can the delay sensitivity $D'_{ik}(f_{ik})$ be estimated accurately and in "real time" ?
2. What is the best value of the step size η ?

We shall address these questions in subsequent sections.

A practical implementation also requires that the algorithm be *loop free at every iteration*; otherwise, there would be a deadlock, i.e. a pair of neighboring nodes each waiting (forever) for information from the other. The protocol used by each node to prevent the formation of loops is discussed in [8].

3. ON-LINE ESTIMATION OF DELAY SENSITIVITY.

We have seen that the link delay sensitivity $D'_{ik}(f_{ik})$ is an important quantity in the Minimum Delay algorithm. Since analytical determination of $D'_{ik}(f_{ik})$ would involve restrictive, and possibly unreasonable assumptions (as discussed in section 1), we wish to estimate this quantity directly. In this section, we shall briefly discuss a technique known as Perturbation Analysis (PA), and demonstrate its use for our purpose.

The PA technique was originally developed to provide an efficient way to estimate performance sensitivities for complex discrete event dynamic systems ([9], [10]). A computer network with stochastic message flows is such a system. The PA method is based on the assumption that a sample path of the (real or simulated) system is observed. The information thus gathered is continuously used to predict performance of the system with perturbed parameter values, without having to observe any additional sample paths, and without actually implementing these perturbations. An important advantage of the PA approach is that it is based on real data, and is therefore independent of many restrictive assumptions required for stochastic modeling. Another is that it can provide *on-line estimation* of performance sensitivities merely by observation of the system and with little computational overhead. This feature is exploited here in order to implement optimal routing algorithms.

For a description of the general PA technique we point the reader to the literature in this area (e.g. [9], [10], [17]). Questions regarding the statistical validity and properties of the PA approach have been investigated in detail (e.g. [4], [5], [17]). For example, Suri and Zazanis [17] have shown that the so called *infinitesimal* PA mean response time estimates are unbiased and strongly consistent for the case of the M/G/1 queue, and give extensions for the case of the GI/G/1 queue as well [18].

3.1. Estimating Packet Response Time Perturbations.

For our purposes, we model a network link as a GI/G/1 queueing system (note that in reality the arrival process at links is not known) with a first-come-first-serve (FCFS) service discipline. The assumption we make is that a sample path (i.e. a stochastic realization) of the link in operation is directly observable. Hence, we can record the arrival time, a_k , and the departure time, d_k, of the kth packet on the link. Let S_k be the service time (i.e. the transmission time) of the kth packet. When the (k-1)th packet departs, two cases are possible :

(i) if there is at least one packet waiting for service, then: $d_k = d_{k-1} + S_k$

(ii) otherwise, there is an idle period of duration $I_k = (a_k - d_{k-1})$; therefore: $d_k = a_k + S_k$.

Combining these two cases, we can write :

$$d_k = \max \{ d_{k-1} , a_k \} + S_k , \qquad k = 1,2,3,... \qquad - (4)$$

with $d_0 = 0$.

Let $F_A(.;\tau)$ and $F_S(.;\sigma)$ be the distributions of the interarrival and service times on the link, respectively, where τ and σ are the corresponding mean values. This is called the *nominal* system. Equation (4) describes the departure time dynamics of the nominal system over a specific sample path. We may think of this sample path as being characterized by an underlying sequence $\omega = (\omega_1, \omega_2,...)$ of random numbers uniformly distributed in [0,1], from which the interarrival time sequence $A_1,A_2,...$ and the service time sequence $S_1,S_2,...$ are generated according to $F_A(.;\tau)$ and $F_S(.;\sigma)$ respectively (see [16] for details). A *perturbed* system is obtained if the parameter τ is perturbed by $\delta\tau$ (which may be positive or negative). The perturbed sample path is then created using the same ω, but by generating an interarrival time sequence $A_1',A_2',...$

In this framework, $A_k = F_A^{-1}(\omega_j,\tau)$ for some ω_j, and $A_k' = F_A^{-1}(\omega_j,\tau+\delta\tau)$ for the same ω_j. Interarrival time perturbations $\delta A_k = A_k' - A_k$ are therefore given by:

$$\delta A_k = F_A^{-1}(\omega_j,\tau+\delta\tau) - A_k$$

Thus, if F_A is known, one can always construct δA_k after observing A_k. Furthermore, if τ is a *scale* parameter of F_A (i.e. the distribution of A/τ is independent of τ), it is shown in [16] that:

$$\delta A_k = \left(\frac{\delta\tau}{\tau}\right) A_k$$

Note that in this case only the fraction $(\delta\tau/\tau)$ is required, while *the nominal value of τ may be unknown*. We shall assume in the sequel that whenever F_A is not available, its mean is a scale parameter. As we shall see, this provides good approximations for interarrival time perturbations at a network node, especially when traffic from multiple links merge at that node.

For the perturbed path created by $\delta\tau$, similar to (4), we get :

$$d_k' = \max \{ d_{k-1}', a_k' \} + S_k , \qquad k = 1,2,3,... \qquad - (5)$$

where a_k' and d_k' are the arrival and departure times, respectively, of the kth packet in the perturbed path. We define the arrival time perturbation (as a result of $\delta\tau$) of the kth packet as $\delta a_k = a_k' - a_k$. These arrival time perturbarions give rise to departure time perturbations, $\delta d_k = d_k' - d_k$.

From (4) and (5), after some algebra, the departure time perturbation dynamics can be obtained as :

$$\delta d_k = \begin{cases} \max\{\delta d_{k-1} - (a_k - d_{k-1}), \delta a_k\} & \text{if } a_k > d_{k-1} \\ \max\{\delta d_{k-1}, \delta a_k + (d_k - d_{k-1})\} & \text{if } a_k \le d_{k-1} \end{cases} \quad -(6)$$

This equation provides a recursive relationship for evaluating departure time perturbations by observing a single sample path of the nominal system. This enables us to determine the exact value of the departure time d_k' for that sample path. This algorithm may, however, require a substantial amount of state memory because if $a_k < d_{k-1}$ and several arrivals are observed in the interval (a_k, d_{k-1}), then all this arrival time information must be saved in order to evaluate δd_j, $j>k-1$ for future departures observed along the nominal path. In general the PA algorithm can be easily modified to trade off memory requirement with estimation accuracy [5]. For instance, parametric perturbations are termed *infinitesimal* if the resulting arrival and departure time perturbations are sufficiently small so as to satisfy the conditions :

if $a_k > d_{k-1}$, then: $a_k+\delta a_k > d_{k-1}+\delta d_{k-1}$, and

if $a_k \le d_{k-1}$, then: $a_k+\delta a_k \le d_{k-1}+\delta d_{k-1}$

for all k=1,2,...

This assumption is referred to as "deterministic similarity" between the nominal and the perturbed paths, and it implies that the perturbations are sufficiently small so as not to cause idle periods to be either created or eliminated in the perturbed path. In such a case, equation (6) may be simplified to obtain the *Infinitesimal Perturbation Analysis (IPA)* estimate :

$$(\delta d_k)_{IPA} = \begin{cases} \delta a_k & \text{if } a_k > d_{k-1} \\ (\delta d_{k-1})_{IPA} & \text{if } a_k \le d_{k-1} \end{cases}$$

It can be shown [5] that the IPA estimate provides at least a *lower bound* for departure time perturbations. More importantly, it can be shown [18] that, under certain conditions, the IPA estimate of the mean delay gradient for a GI/G/1 system is strongly consistent. In this paper, we have used the estimate of equation (6), which is more accurate in tracking the transient behavior of the perturbed path, as long as no constraints on the amount of state memory involved are violated (we have not found this to be a problem in practice).

Based on the PA framework described above, we shall now derive a procedure similar to (6) for estimating the mean response time of packets through a link. Let r_k denote the *response time* (queueing delay + transmission time) experienced by the kth packet on the link. Note that:

$$r_k = d_k - a_k$$

Thus, the departure time dynamics of equation (4) give:

$$\begin{aligned} r_k &= \max \{ d_{k-1}, a_k \} + S_k - a_k \\ &= \max \{ d_{k-1} - a_k, 0 \} + S_k \\ &= \max \{ r_{k-1} - A_{k-1}, 0 \} + S_k \end{aligned} \qquad - (7)$$

where $A_{k-1} = a_k - a_{k-1}$ is the interarrival time following the (k-1)th packet. If r_k' is the corresponding response time in the perturbed path created by $\delta\tau$, the response time perturbation $\delta r_k = r_k' - r_k$ can be recursively evaluated as:

$$\delta r_k = -\delta A_k + \begin{cases} \max \{\delta r_{k-1} - (A_k - r_{k-1}), \delta A_k\} & \text{if } A_k > r_{k-1} \\ \max \{\delta r_{k-1}, \delta A_k + (A_k - r_{k-1})\} & \text{if } A_k \le r_{k-1} \end{cases} \qquad - (8)$$

where $\delta A_k = A_k' - A_k$ is the kth interarrival time perturbation due to $\delta\tau$, evaluated as discussed above following the observation of A_k.

Then, for a sample path consisting of K observed packets, the *sample* mean packet delay is given by:

$$R_K = \frac{1}{K} \sum_{k=1}^{K} r_k$$

Assuming ergodicity, this expression provides an unbiased and consistent estimator of the mean delay over all sample paths. The corresponding sensitivity with respect to $\delta\tau$ is computed as:

$$\delta R_K = \frac{1}{K}\sum_{k=1}^{K} \delta r_k \qquad -(9)$$

where δr_k is evaluated by the PA algorithm in (8), or a simplified version of it (e.g. using the IPA departure time perturbation estimate). Subject to some technical assumptions (see [18]), the IPA estimate alone provides an unbiased and consistent estimator of the mean response time gradient as $\delta\tau \to 0$.

3.2. PA Estimate of D'_{ik} in the Routing Algorithm.

Returning to the notation of section 2, for a link (i,k) with mean interrrival time τ_{ik}, let R_{ik} be the mean response time of packets over that link. We can then use the PA procedure in (8)-(9) with some "small" $\delta\tau_{ik}$, in order to estimate the gradient $\frac{\partial R_{ik}}{\partial \tau_{ik}}$ from a single observed sample path with K packets through (i,k). Note that the gradient estimates *for all links* can be obtained along this one sample path.

Since D_{ik} in the routing algorithm represents the mean queue length over link (i,k), by Little's law:

$$D_{ik} = f_{ik} R_{ik}$$

and we get:

$$D'_{ik} = \frac{\partial D_{ik}}{\partial f_{ik}} = R_{ik} + f_{ik}\frac{\partial R_{ik}}{\partial f_{ik}}$$

However, since the PA gradient estimates are obtained in terms of mean interarrival time - rather than flow - perturbations, we have:

$$D'_{ik} = R_{ik} + \frac{1}{\tau_{ik}} \cdot \frac{\partial R_{ik}}{\partial \tau_{ik}} \cdot \frac{\partial \tau_{ik}}{\partial f_{ik}}$$

which yields:

$$D'_{ik} = R_{ik} - \tau_{ik}\left(\frac{\partial R_{ik}}{\partial \tau_{ik}}\right) \qquad -(10)$$

with $\frac{\partial R_{ik}}{\partial \tau_{ik}}$ estimated through equations (8), (9).

Furthermore, when τ_{ik} is treated as a scale parameter, as previously discussed, we need only fix the fraction $(\delta\tau_{ik}/\tau_{ik})$ in order to evaluate perturbations δA_k, with a corresponding δR_{ik} obtained through (9). Hence, from (10), our estimate of D'_{ik} is:

$$\hat{D}'_{ik} = R_{ik} - \left(\frac{\tau_{\iota\kappa}}{\delta\tau_{\iota\kappa}}\right)\delta R_{ik} \qquad -(11)$$

and the actual value of τ_{ik} need not be known or estimated.

Very few algorithms for such on-line gradient estimation have appeared in the literature. One is the "Customer Rejection" algorithm proposed by Segall [15]. In this approach, the delay sensitivity is estimated along an observed sample path by hypothesizing that packets arriving at a link are rejected (i.e. are denied service) with probability ε. The effective link flow is thus reduced on the average by:

$$\delta f = \frac{M\varepsilon}{T}$$

where M is the number of observed arrivals in the interval T. Note that in this algorithm the *number* of packets is (hypothetically) reduced, while in the PA approach the *interarrival* times of packets are individually perturbed. Thus, either technique can be used to model a given flow perturbation. One can see, however, that if ε is small in Segall's approach, then a long observation period is required to allow a sufficiently large number of packet "rejections". The advantage of the PA approach is in its relative simplicity and - more significantly - in that it inherently provides lower variance estimates. This is an advantage PA also maintains over the Likelihood Ratio (LR) approach [13]. In fact, unless the system is regenerative, the variance of LR estimates increases with longer observation periods (see [13]). The lower variance property ensures fast convergence of the PA algorithm. This is important for our purpose, since the convergence properties of the routing algorithm depend on the convergence properties of the estimates themselves.

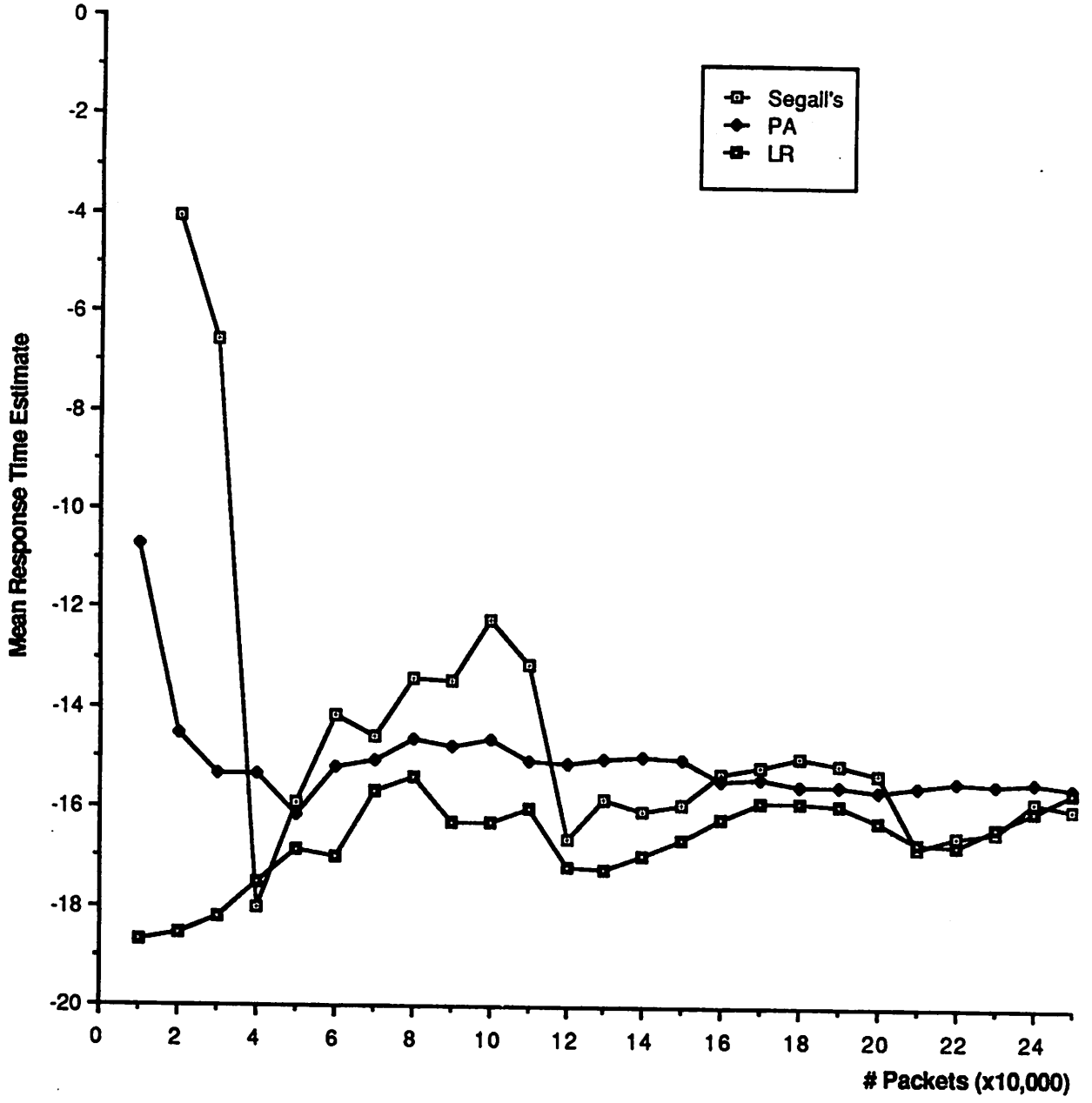

Fig. 1: *Comparing the Convergence of Segall's, PA and LR Estimates*

In order to compare the PA, LR and "Customer Rejection" estimation algorithms, all three were implemented to estimate the gradient of the mean delay (response time), R, for an M/M/1 link model. The mean interarrival time is $\tau = 10.0$ and the mean service time is $\sigma = 8.0$. The mean delay gradient $\partial R/\partial \tau$ was estimated by each of these methods over consecutive observation intervals, with 10,000 packets in each one. Analytically, it is easy to obtain $\partial R/\partial \tau = -16.0$. The results of the

algorithms for this example are shown graphically in Fig. 1, where the PA estimate is seen to converge rapidly and smoothly.

4. SIMULATION RESULTS

In this section, we present results from simulation to demonstrate how the PA estimation technique can be used in conjunction with a routing algorithm to achieve optimal routing in computer networks. The Minimum Delay algorithm of section 2 was implemented on several simulated networks, with the marginal delay estimation being done through PA, as described in section 3. In particular, D'_{ik} in the marginal delay expression (1) is evaluated through (11), with the mean response time perturbation estimated through (8) and (9). The value of $(\delta\tau_{ik}/\tau_{ik})$ used in subsequent results is 0.001. This is sufficiently small to maintain minimal arrival time storage requirements in implementing (8).

Packets enter the network at the source node and are disposed of at the destination node. The network is observed for a certain interval of time (the *observation period*), which defines one iteration of the routing algorithm. For each packet traversing the network, we record its delay and marginal delay on each link, and also its total sojourn time in the network. Then, at the end of the observation period, equations (2) and (3) are used to compute the marginal delay and to update the routing variables for all nodes. The initial routing in each case is chosen arbitrarily.

In the examples that follow, our first objective is to verify that the routing algorithm actually converges to the minimum delay value, given that marginal delay estimates, and not values based on some function $D_{ik}(f_{ik})$ which is not available, are being used (Examples 1 and 2). Obviously, the accuracy of the estimates used for each iteration depends on the time interval over which the network is observed. Thus, we investigate the performance of the algorithm as the observation period is varied. The speed of convergence of the algorithm depends on the value of the step size η. We shall, therefore, also investigate the behavior of the algorithm for varying step sizes, and

present guidelines for choosing an appropriate step size. In Example 3, we consider the same issues for more complicated traffic patterns.

Next, we investigate how the algorithm adapts to sudden changes in the network. Specifically, we shall consider the case where a link deteriorates, then completely fails, and finally improves again after some time (Example 4). Here, we consider a larger network with link capacities made as realistic as possible, based on values found in existing commercial networks. In Examples 5, we have included multiple source-destination pairs to test the algorithm in an environment of significant complexity.

Finally, we compare the routing algorithm performance with marginal delays estimated through PA and through an M/M/1 approximation for links (Example 6). As we shall see, the latter (which still requires estimation of link flows) yields significantly higher mean delays for the network and a typical traffic pattern considered.

In all examples that follow, packets entering the network at node O with destination node D are referred to as the O→D traffic.

• **Example 1 (Comparison of algorithm with analytically available results)**

We have applied the Minimum Delay routing algorithm, with PA marginal delay estimation, to the four-node network shown in Fig. 2. In this example, we consider only 1→4 traffic, i.e. packets enter the network at node 1 only, and are all destined for node 4. In order to compare our experimental results with analytical ones, we have assumed that interarrival and service times are exponentially distributed, and that the independence assumption holds. The mean interarrival time is 10.0 sec, and the mean service time of each link is 8.0 sec.

From the symmetry of the network, it is intuitively obvious that, for optimal routing, traffic at node 1 should be split equally on links (1,2) and (1,3), and that no traffic should be routed on link (2,3). Corresponding to this routing, we can obtain by standard queueing theoretic results (Jackson's theorem), the mean packet delay as $D_T = 26.7$ sec.

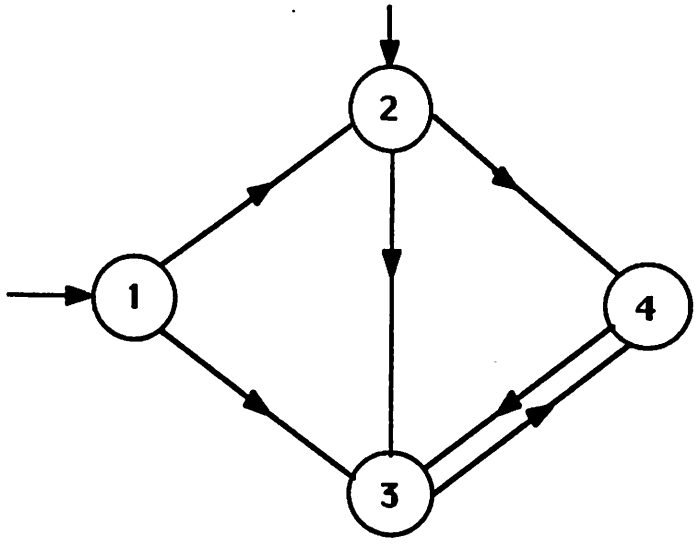

Fig. 2: *Network for Examples 1, 2 and 3*

In Fig. 3, we show the mean delay obtained as a function of simulated time for different sample path lengths per algorithm iteration (in terms of the number of packets observed). Note that convergence near the analytically determined optimal mean delay is obtained even for relatively short observation periods (500 packets). Thus, if the average link transmission time were 0.01 sec, convergence would be attained in less than 0.1 sec.

- **Example 2 (Algorithm performance when packet lengths are not exponentially distributed)**

We now relax the assumption of exponential service times (equivalently: exponentially distributed packet lengths). We also relax the independence assumption, so that all packets maintain their identity (packet length) at all nodes. In this example, packets entering the network fall in one of two classes, "long" and "short", with the former constituting 80% of the total traffic. The service time of each link is now specified as 9.0 sec if a packet is "long", and 0.9 sec if a packet is "short".

No analytical solution is available in this case, and the interarrival time distributions at links are difficult to obtain in closed form. However, from the symmetry argument, the optimal routing should remain the same as in the previous example.

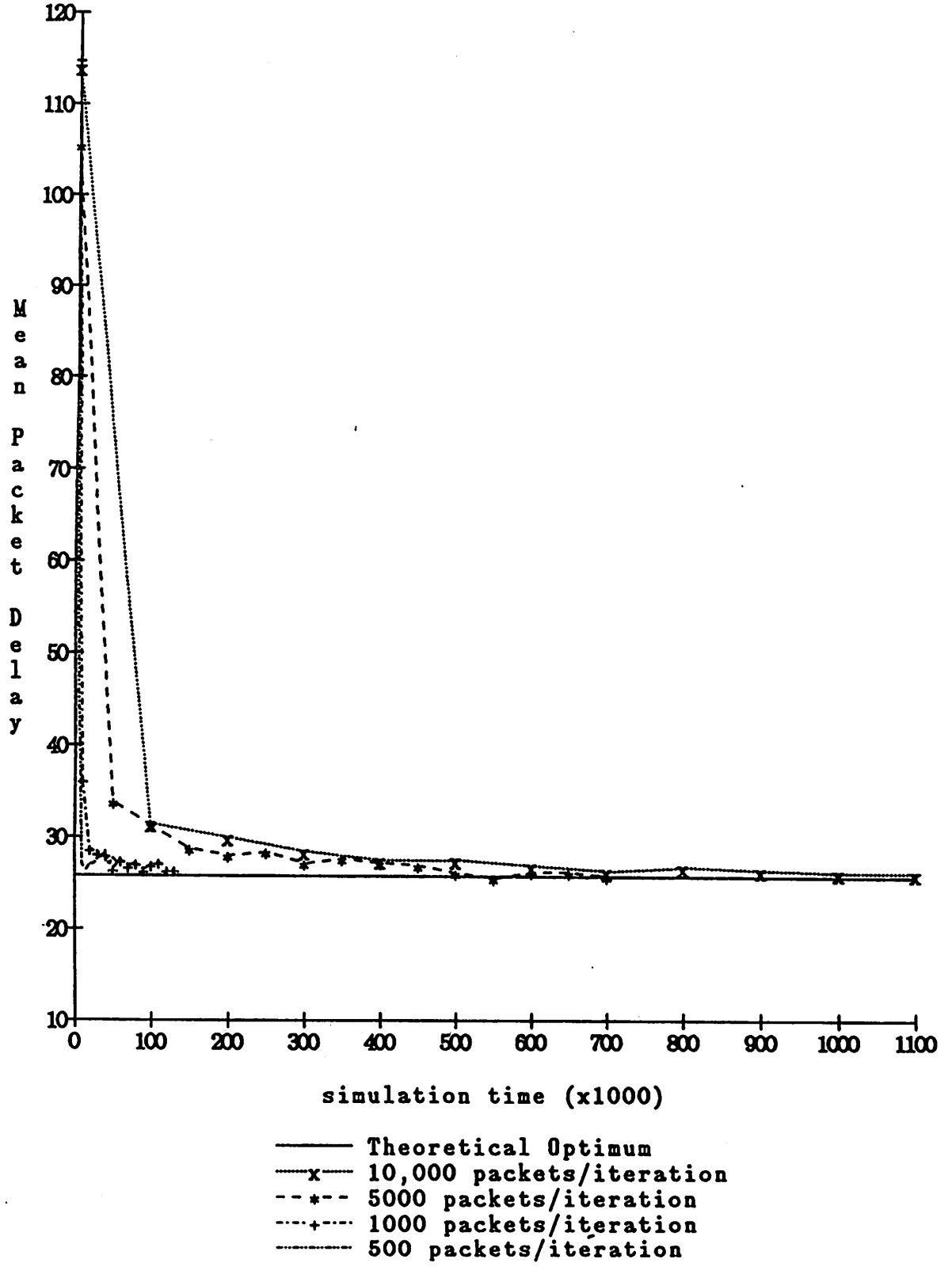

Fig. 3: *Effect of Observation Period Length on Routing Algorithm Convergence (Example 1)*

Figures 4a and 4b show the behavior of the algorithm for different values of the step size, with the observation period fixed to 5,000 packets/iteration. In Fig. 4a the step sizes are small enough to guarantee convergence, whereas Fig. 4b illustrates the instability resulting from larger step sizes.

Fig. 5a shows the behavior of the algorithm when the observation period is varied, for a fixed step size $\eta = 1.0\text{x}10^{-5}$. Fig. 5b is a magnified version of 5a to show convergence for short observation periods (as few as 50 packets).

In all cases (except when there is no convergence), the routing variables converge to the same final values, irrespective of the initial routing. These final values are, as expected:

$$\phi_{12}(4) = 0.5, \qquad \phi_{13}(4) = 0.5$$

$$\phi_{24}(4) = 1.0, \qquad \phi_{23}(4) = 0.0$$

After reviewing Figures 4 and 5, we can make the following observations :

(1) The mean delay decreases rapidly during the first few iterations; subsequently, changes become more gradual.

(2) A larger step size leads to faster convergence. However, if the step size is too large, the algorithm fails to converge, and the mean delay exhibits an oscillatory behavior, as illustrated in Fig. 4b with $\eta = 5.0\text{x}10^{-4}$. This is because a large step size results in an inordinately large amount of change in the routing variables at each iteration; as a result, the link flow is switched from from one path to another, and the optimal (bifurcated) routing is never achieved. Note that the oscillatory behavior of the mean delay may not manifest itself immediately (as in the case of $\eta = 1.0\text{x}10^{-4}$ in Fig. 4b), but only after link flows throughout the network have achieved steady state.

(3) Convergence occurs even with short observation intervals (i.e. sample paths for obtaining PA mean delay sensitivity estimates), as illustrated in Figures 5a, 5b for 50 packets per iteration. However, when observation intervals become too short, the mean delay exhibits "noisy" behavior near the optimal to which the algorithm converges. This is expected, since the PA estimates have larger variance when observation intervals are short.

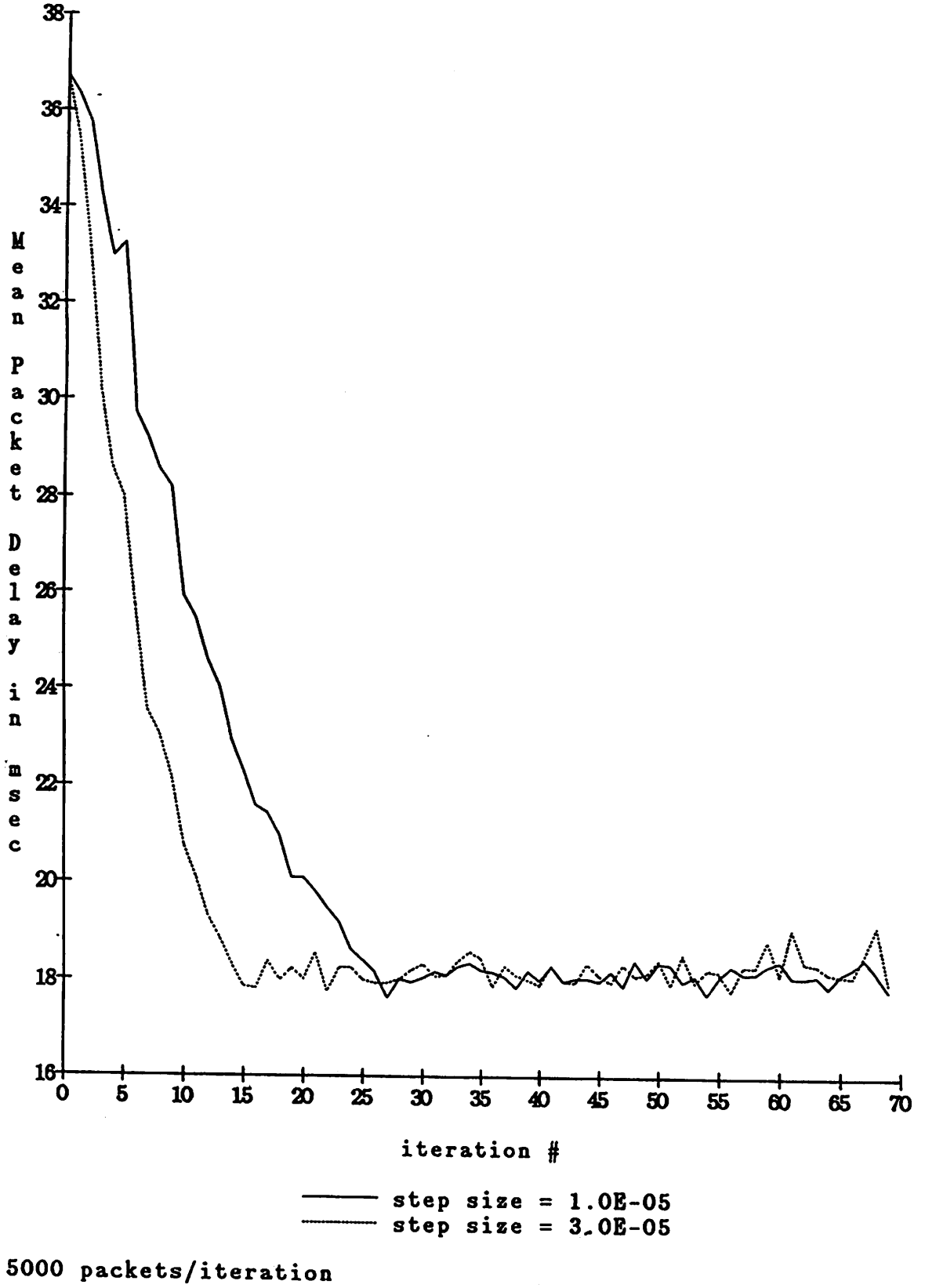

Fig. 4a: *Effect of Step Size on Routing Algorithm Convergence (Example 2) (Small Step Sizes)*

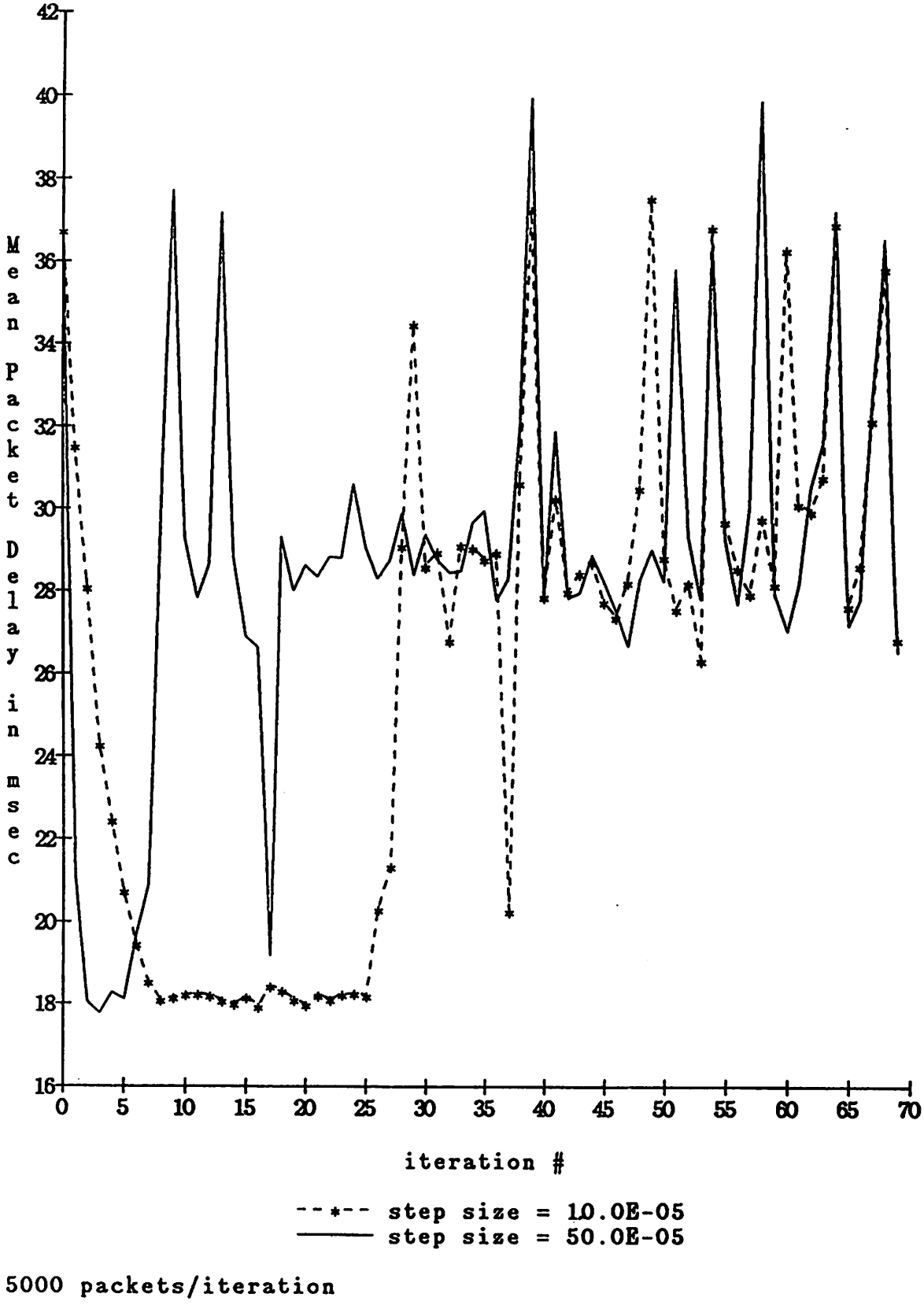

Fig. 4b: *Effect of Step Size on Routing Algorithm Convergence (Example 2). (Large Step Sizes)*

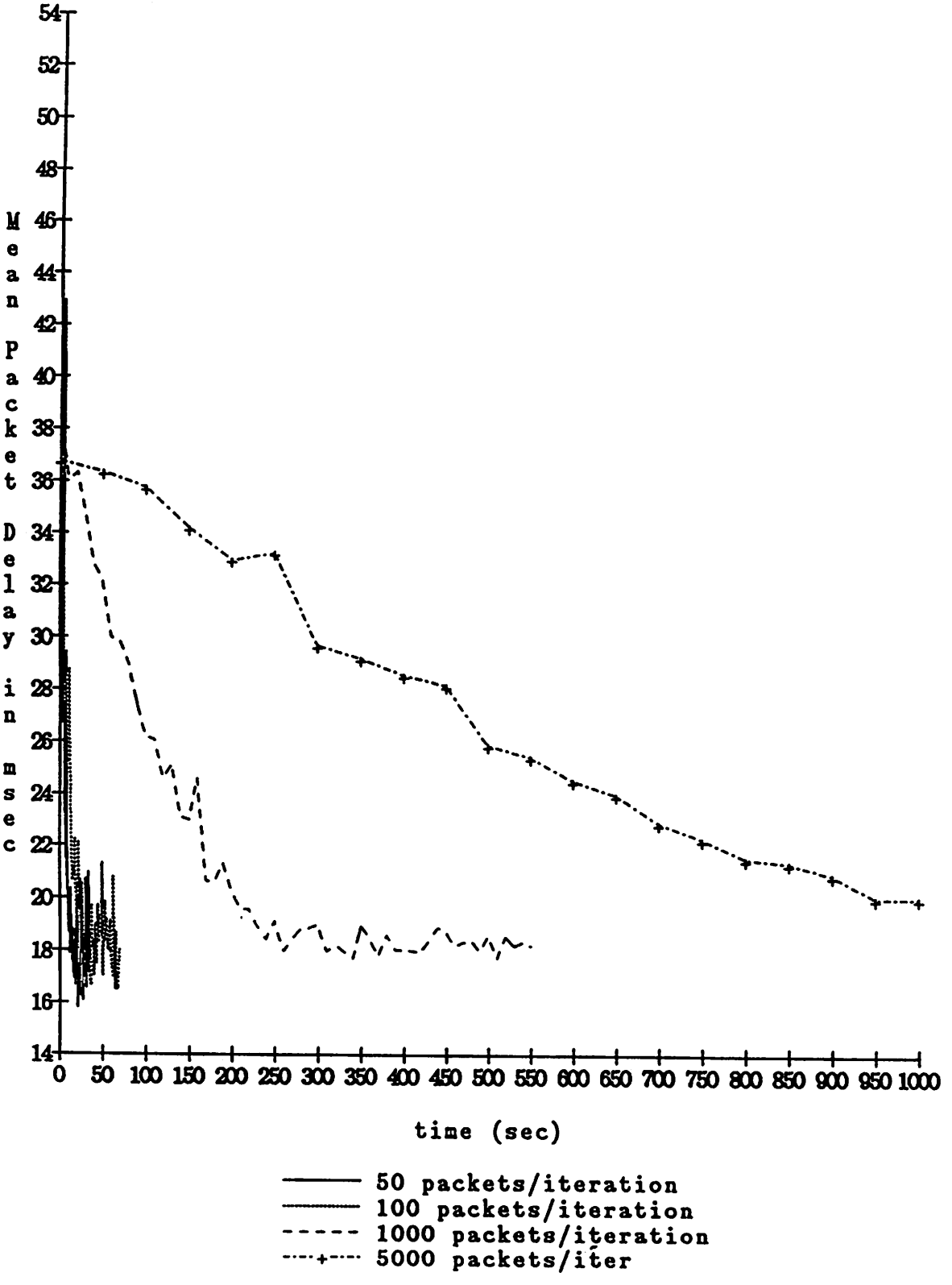

Fig. 5a: *Effect of Observation Period Length on Routing Algorithm Convergence (Example 2)*

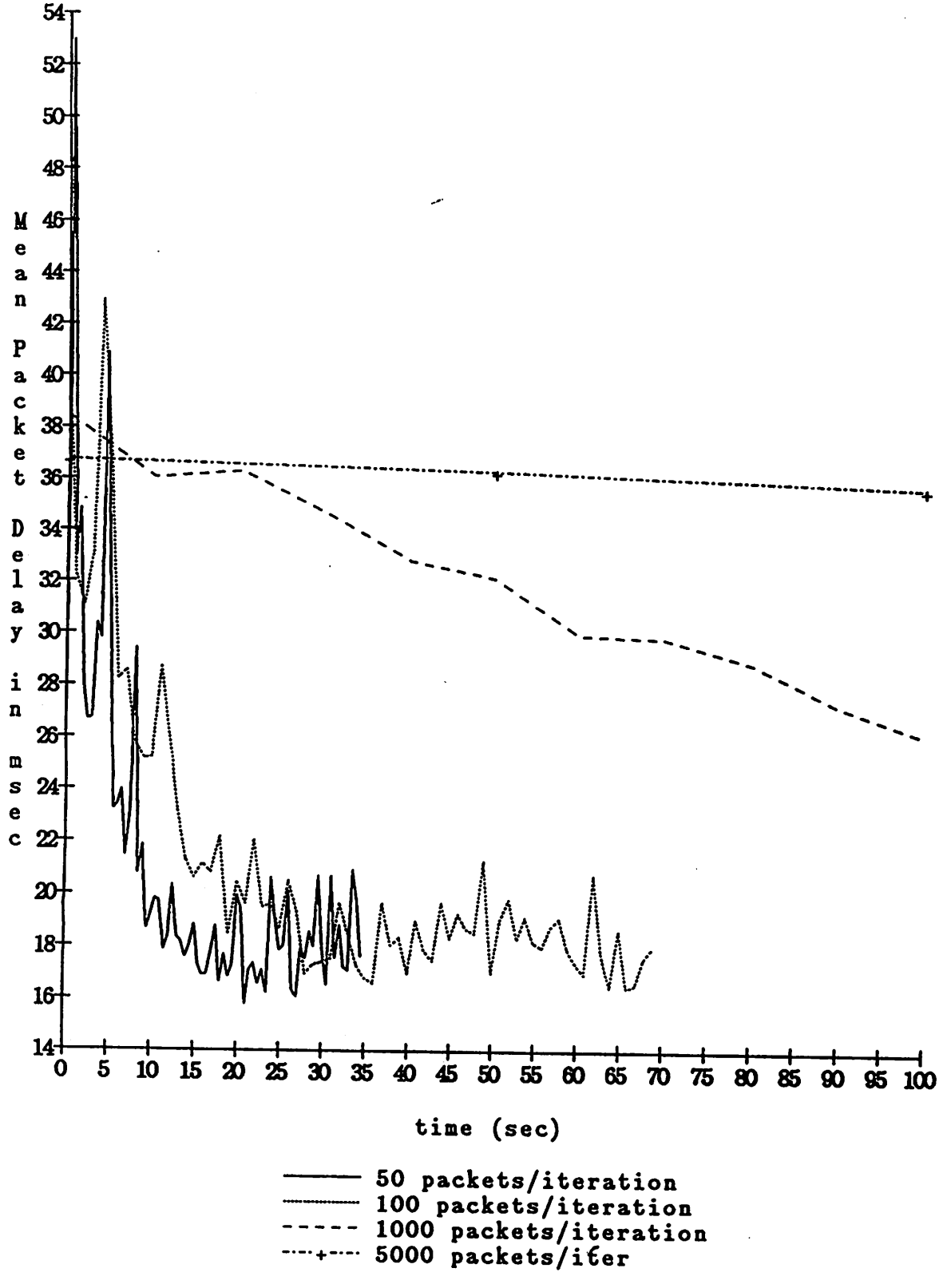

Fig. 5b: *Effect of Observation Period Length on Routing Algorithm Convergence (Example 2) (Short Observation Intervals)*

• Example 3 : (Algorithm performance for more complex traffic pattern)

Our objective here is to investigate the effect of interfering traffic at the nodes of a network, as several O→D pairs are included. Consider again the network of Fig. 2, but now with three O→D pairs: 1→4, 2→4 and 1→3.

The interarrival times of 1→4 and 2→4 traffic are both exponentially distributed with mean values of 15.0 sec, and the interarrival time of 1→3 traffic is uniformly distributed in the interval [12.0,18.0] sec.

The behavior of the algorithm in this case is presented in Figures 6 and 7. The observations made above for the single OD pair hold in this case as well. Note, however, that a smaller step size must be chosen (not much greater than $\eta = 1.0 \times 10^{-5}$). Interestingly, fast convergence is attained even for short observation periods (e.g. 100 packets/iteration in Fig. 7), at the expense of small oscillations near the optimal mean delay.

• Example 4 (Algorithm performance for larger networks and adaptivity to sudden changes)

We now investigate the behavior of the algorithm when the network size is increased, and also see how it adapts to a link failure. Consider the six-node network of Fig. 8. Packets enter the network at node 1 only with exponentially distributed interarrival times with a mean of 50.0 msec, and destined for node 5. We take 80% of the traffic to consist of 1024 bit "long" packets (e.g. data), and 20% of 128 bit "short" packets (e.g. control packets). Link capacities are 72 kbps, 56 kbps or 9.6 kbps, as shown in Fig. 8. Note that the maximum number of hops a packet must traverse in going from source node to destination node has increased compared to the network of Fig.2.

Fig. 9 shows that the algorithm converges in a manner similar to the previous cases for a step size $\eta = 0.3 \times 10^{-5}$ and observation intervals of 5,000 packets per iteration. The oscillatory behavior observed illustrates a limitation of the routing algorithm, causing some of the routing variables to switch between 0 and large positive values in successive iterations. Specifically, consider links

(2,6), (2,4) and (2,5) in Fig. 8. Because (2,3) is a much faster link, it is optimal for all (1→5) traffic at node 2 to be routed through that link. Thus, following some iteration, we have:

$$\phi_{23}(5) = 1,\ \phi_{24}(5) = 0,\ \phi_{26}(5) = 0$$

Now, in the next observation period, since no packets are routed through links (2,4) and (2,6), mean delay and marginal delay for these links cannot be estimated. For lack of information, the algorithm assumes these quantities to be 0. As a result, a large fraction of traffic is shifted to these links in the next iteration, the marginal delays will once again be found to be large, the previous (truely optimal) values for the routing variables are re-established and the process goes on. This explains why the mean delay in Fig. 9 periodically deviates from the optimal.

Clearly, if we are not concerned with the possibility of future changes, we may simply set the routing variables to their optimal values as soon as this behavior is detected. This gives rise to the smoother (solid) curve in Fig. 9. Alternatively, we may maintain memory for the marginal delays of unused links and allow only a small fraction of "test" traffic through them in such cases, which would prevent any significant mean delay deterioration. Yet another approach is to use an M/M/1 approximation for such links to estimate the marginal delay if measurements are not available in an iteration.

We now investigate the adaptivity of the algorithm when sudden changes occur in the state of the network (without attempting to avoid the oscillatory behavior discussed above). Consider the following scenario: first, link (1,3) suddenly deteriorates (i.e. its transmission time increases); after some time, it deteriorates further, and finally the link is repaired. Referring to Fig. 10, after the 83rd iteration, link (1,3) fails and is replaced by a slower 24 kbps "backup" link. As expected, the mean delay initially increases to a very large value, but, within a few iterations, the algorithm rapidly adapts to the changed network condition and finds a superior routing.

After the 120th iteration, link (1,3) is fcompletely removed. Again note the initial increase in delay, followed by rapid adaptation.

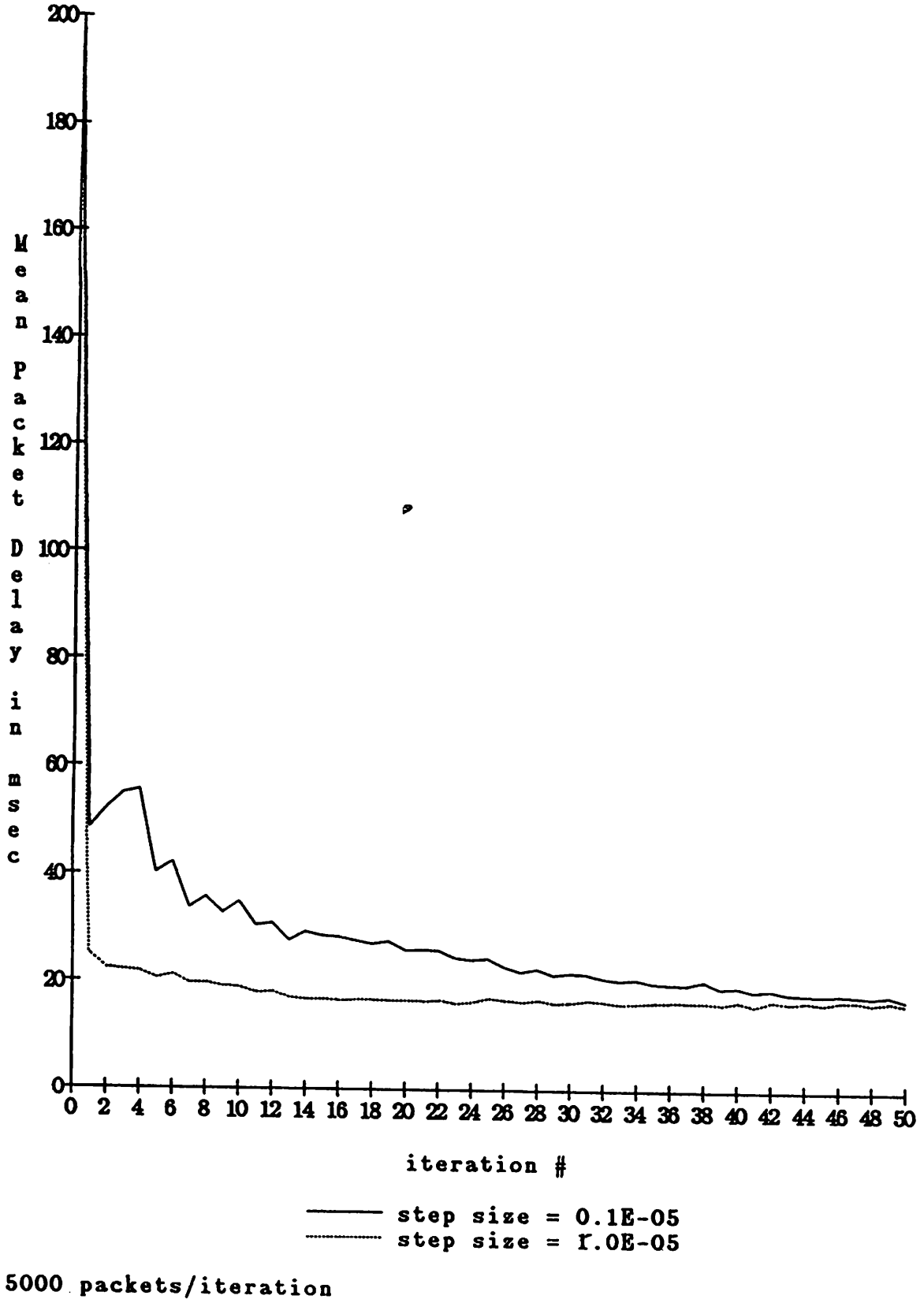

Fig. 6: *Effect of Step Size on Routing Algorithm Convergence (Example 3)*

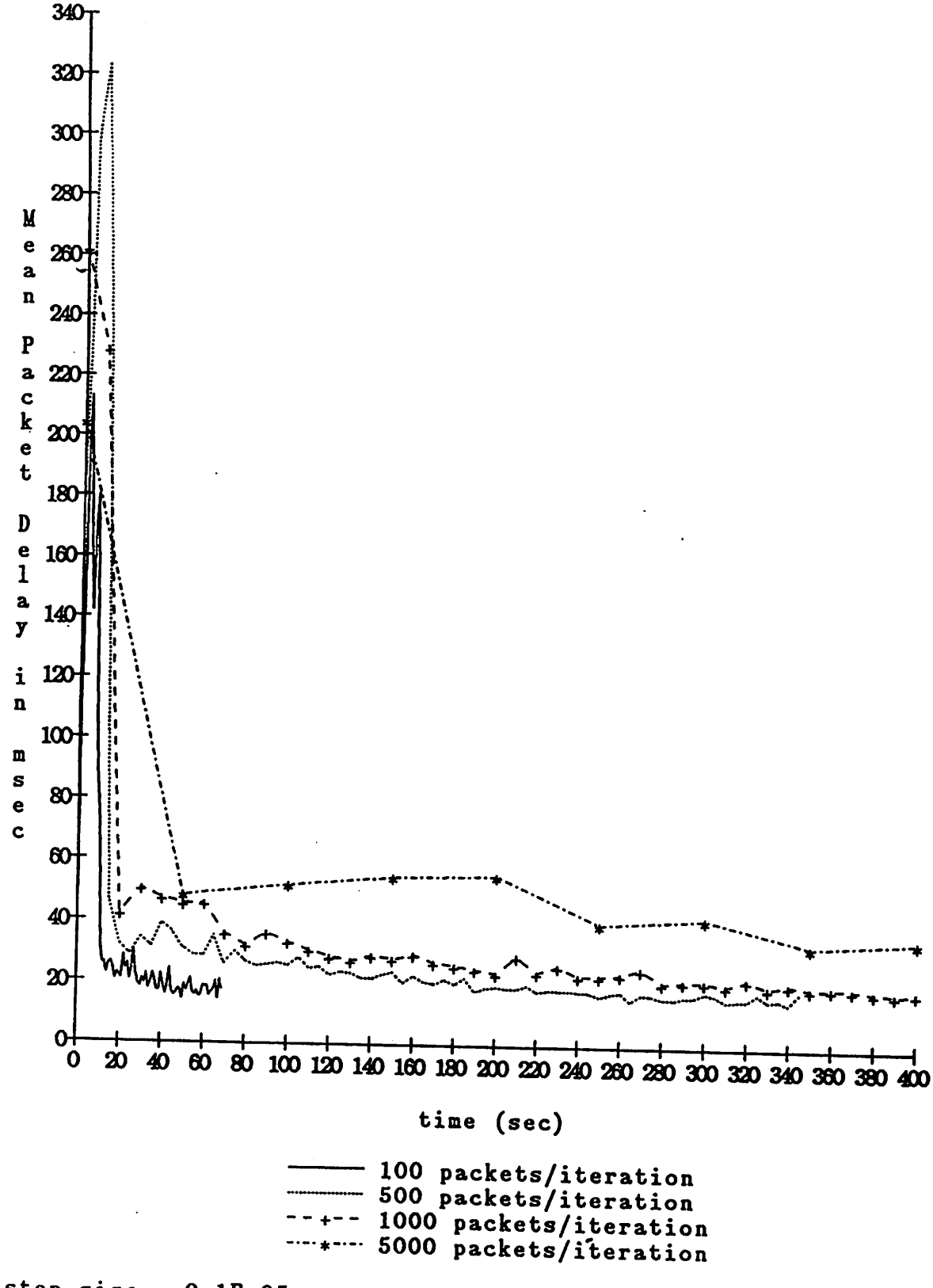

Fig. 7: *Effect of Observation Period Lenghth on Routing Algorithm Convergence (Example 3).*

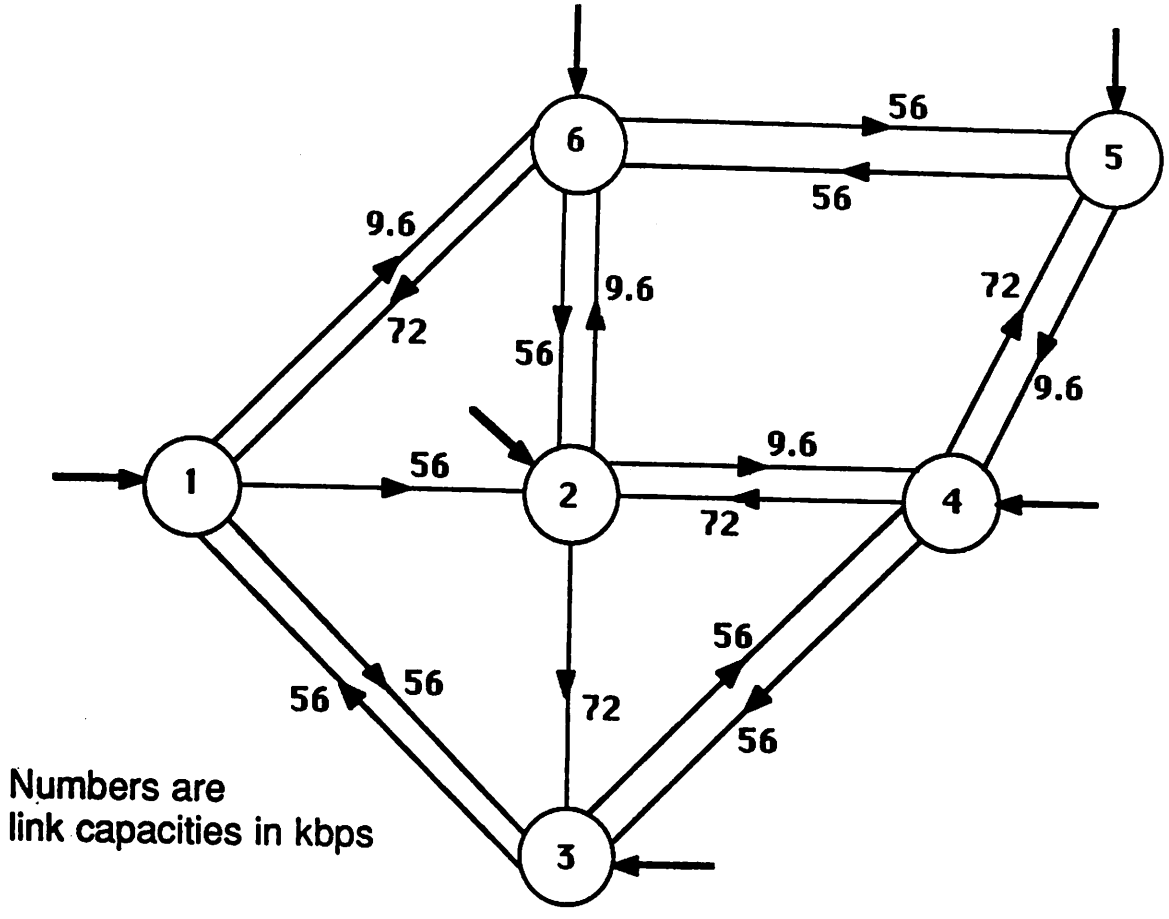

Fig. 8: *Network for Examples 4, 5 and 6*

Finally, after the 155th iteration, link (1,3) is repaired and regains its original 56 kbps capacity. The algorithm immediately responds by shifting flow to this link, and converges to the original optimal routing. Thus, even in the presence of large disturbances, the minimum delay routing algorithm has the ability to adapt rapidly.

- **Example 5 (Algorithm performance for larger networks and complex traffic pattern - 19 OD pairs)**

We now investigate the performance of the algorithm for the six-node network of Fig. 8 when the traffic pattern becomes more complex, with 19 OD pairs. Interarrival times are taken to be either exponentially or uniformly distributed, with means ranging from 60.0 msec for (1→3) traffic to 300.0 msec for (1→5), (1→6), (3→2), (3→5) and (4→3) traffic. Packets contain, as before, either 1024 (80% of the traffic) or 128 bits.

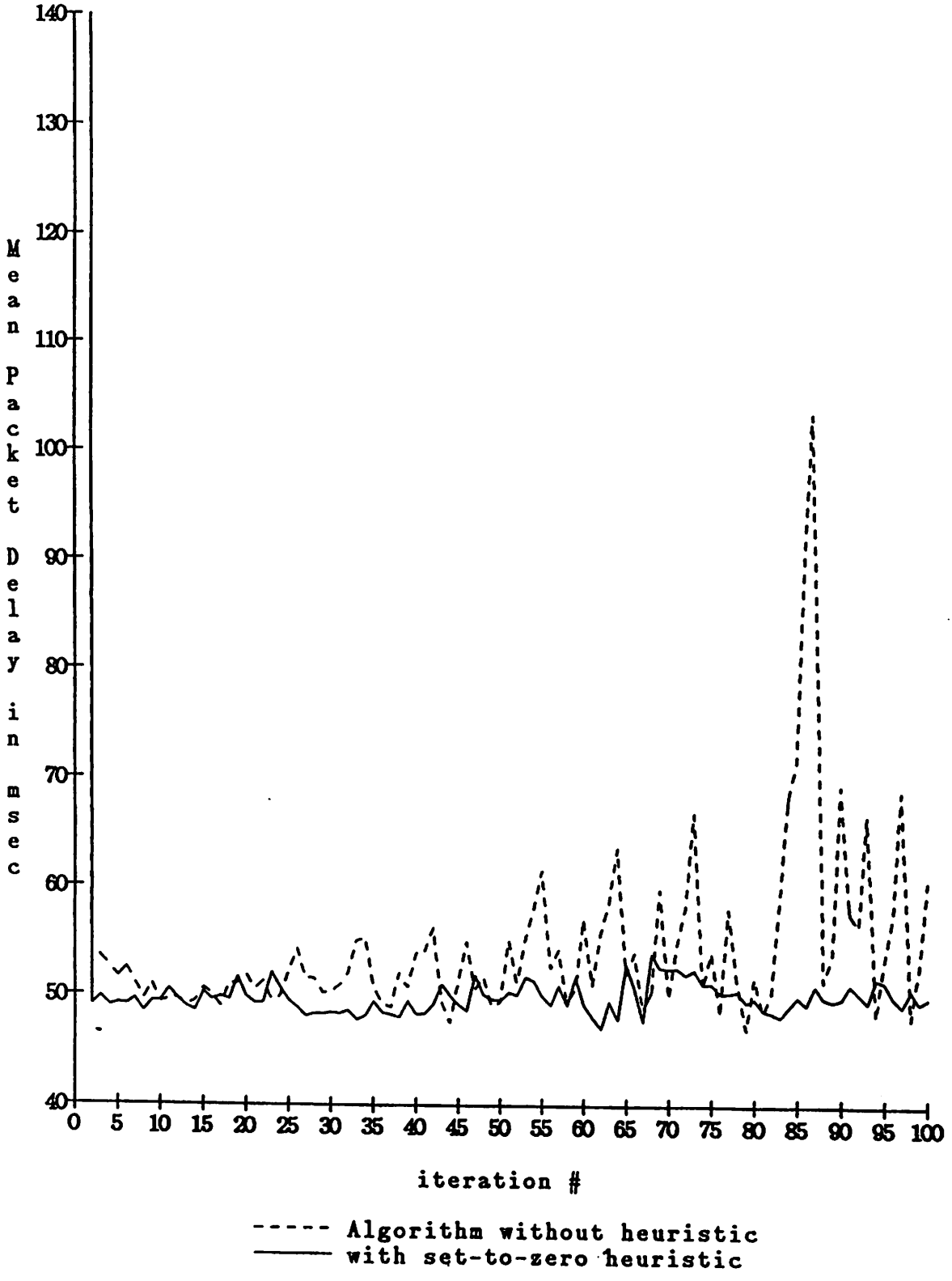

Fig. 9: *Algorithm Performance with Heuristic Modification (Example 4)*

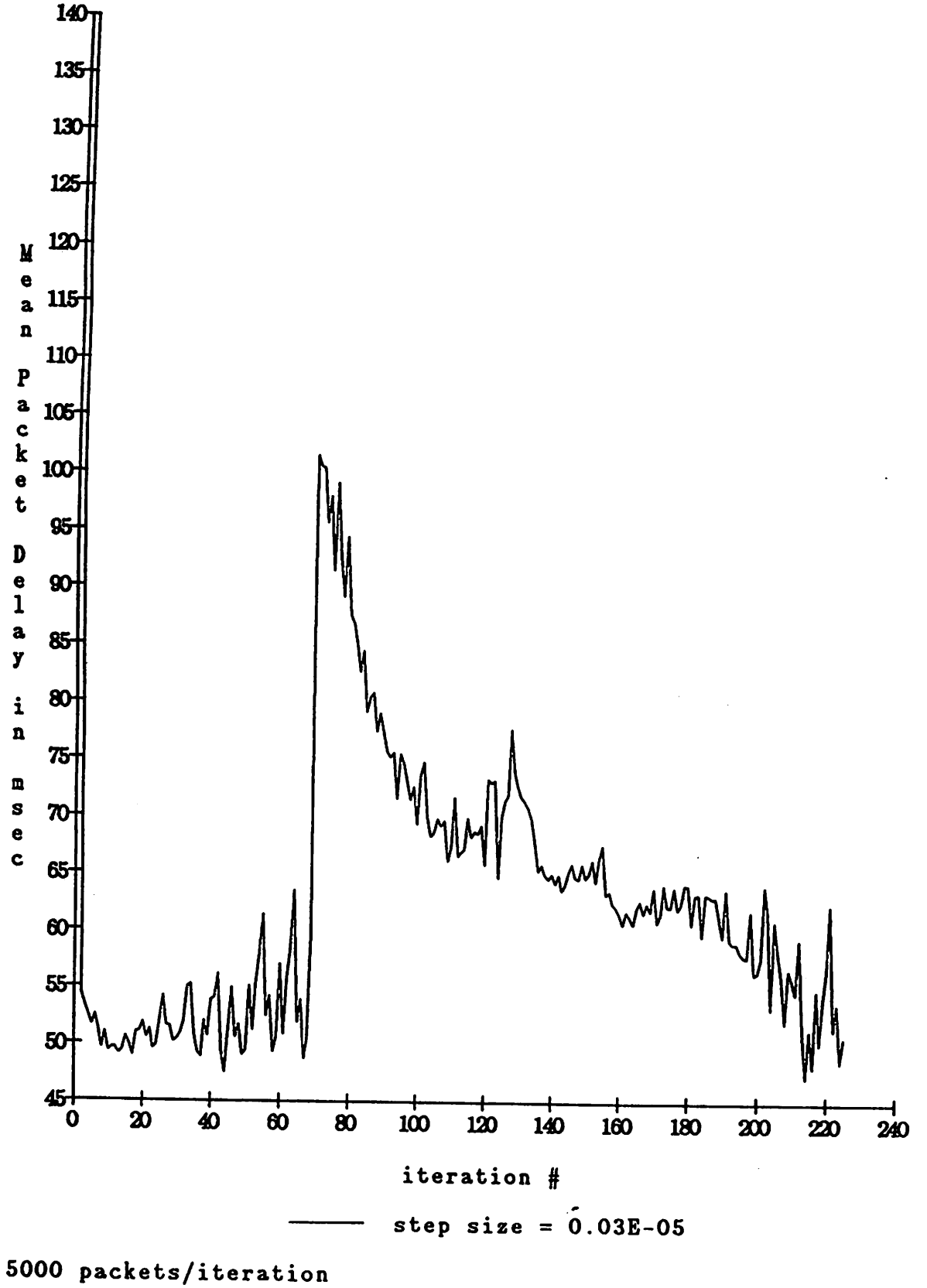

Fig. 10: *Adaptivity of Routing Algorithm (Example 4)*

The mean delay was found to converge to about 50 msec, as shown in Figures 11 and 12 (we omit listing the corresponding optimal routing variables). The main features of the algorithm are similar to the previous examples, with even smaller step sizes required to guarantee convergence (the algorithm became unstable for $\eta = 0.1 \times 10^{-5}$). In Fig. 12, note that convergence becomes more noisy as the observation periods are shortened; however, rapid and near-optimal performance is obtained for observation periods as short as 500 packets. For instance, with $\eta = 0.01 \times 10^{-5}$ and observation periods of 500 packets, the mean delay comes very close to optimal after 50 sec.

• Example 6 (Comparison between PA marginal delay estimation and M/M/1 link model approximation in algorithm performance)

As a first step, one may compare PA delay gradient estimates of a standalone GI/G/1 model with those obtained through an M/M/1 approximation. Next, since it is reasonable to approximate the interarrival time process at a network node with heavy traffic arriving from several links as having an exponential distribution, one can use an M/G/1 system instead. Examples showing that M/M/1 approximations can be highly inaccurate for such standalone models, compared to their PA counterparts, may be found in [1].

On the other hand, it is possible to argue that the M/M/1 approximations are sufficiently accurate in the context of the routing algorithm, since it is the *relative* sensitivities of links that are important in determining how to update routing variables. We have, therefore, repeated Example 5 with a step size $\eta = 0.01 \times 10^{-5}$ and observation periods with 5,000 packets using M/M/1 approximations for all network links, and hence analytically evaluating the marginal delays required in the algorithm. The results, comparing this approach to our implementation with PA estimation, are shown in Fig. 13. Note that the M/M/1 approximation yields higher mean delay routings with considerable variance for this particular case (long and short packets).

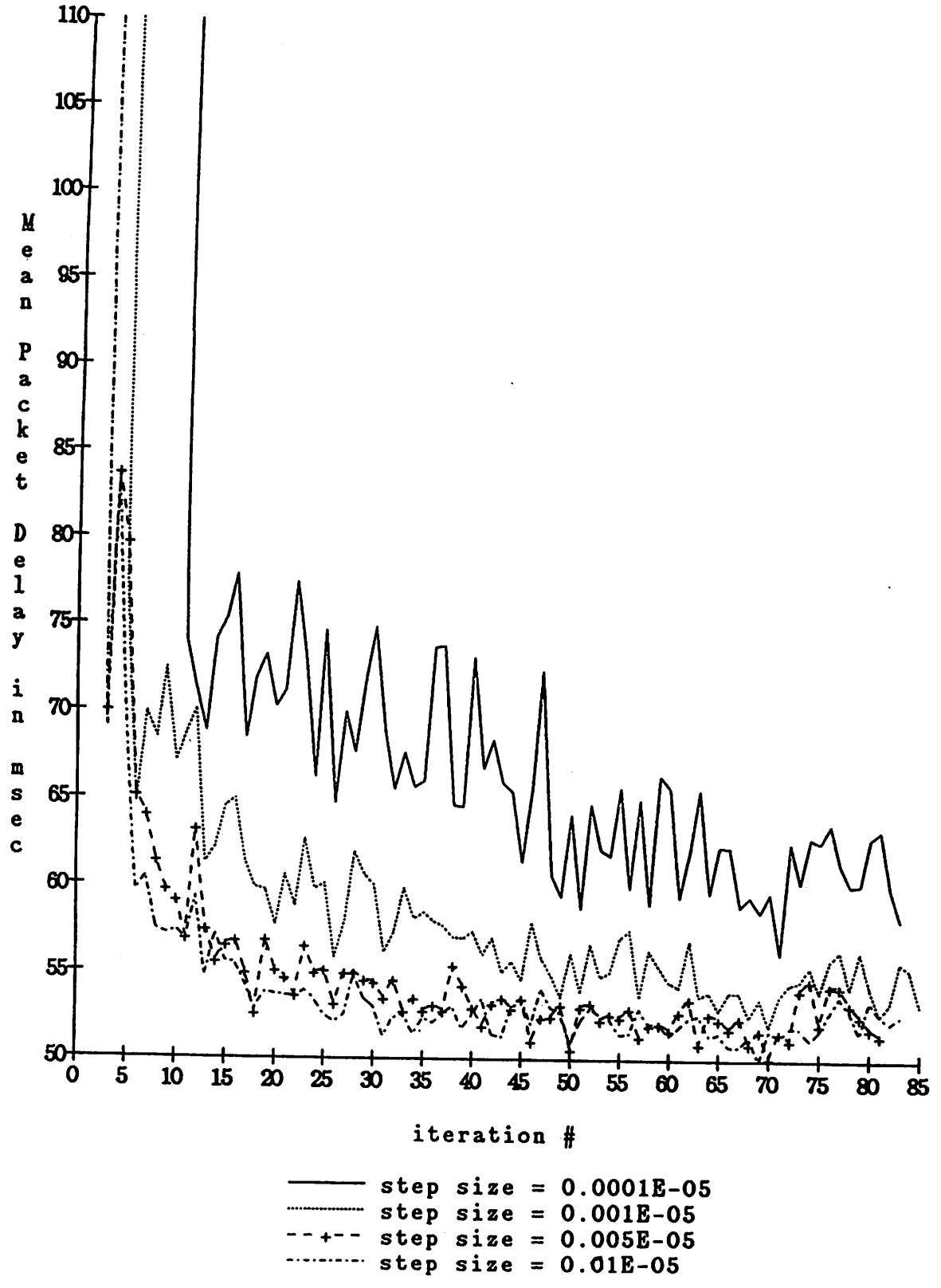

Fig. 11: *Effect of Step Size on Routing Algorithm Convergence (Example 5)*

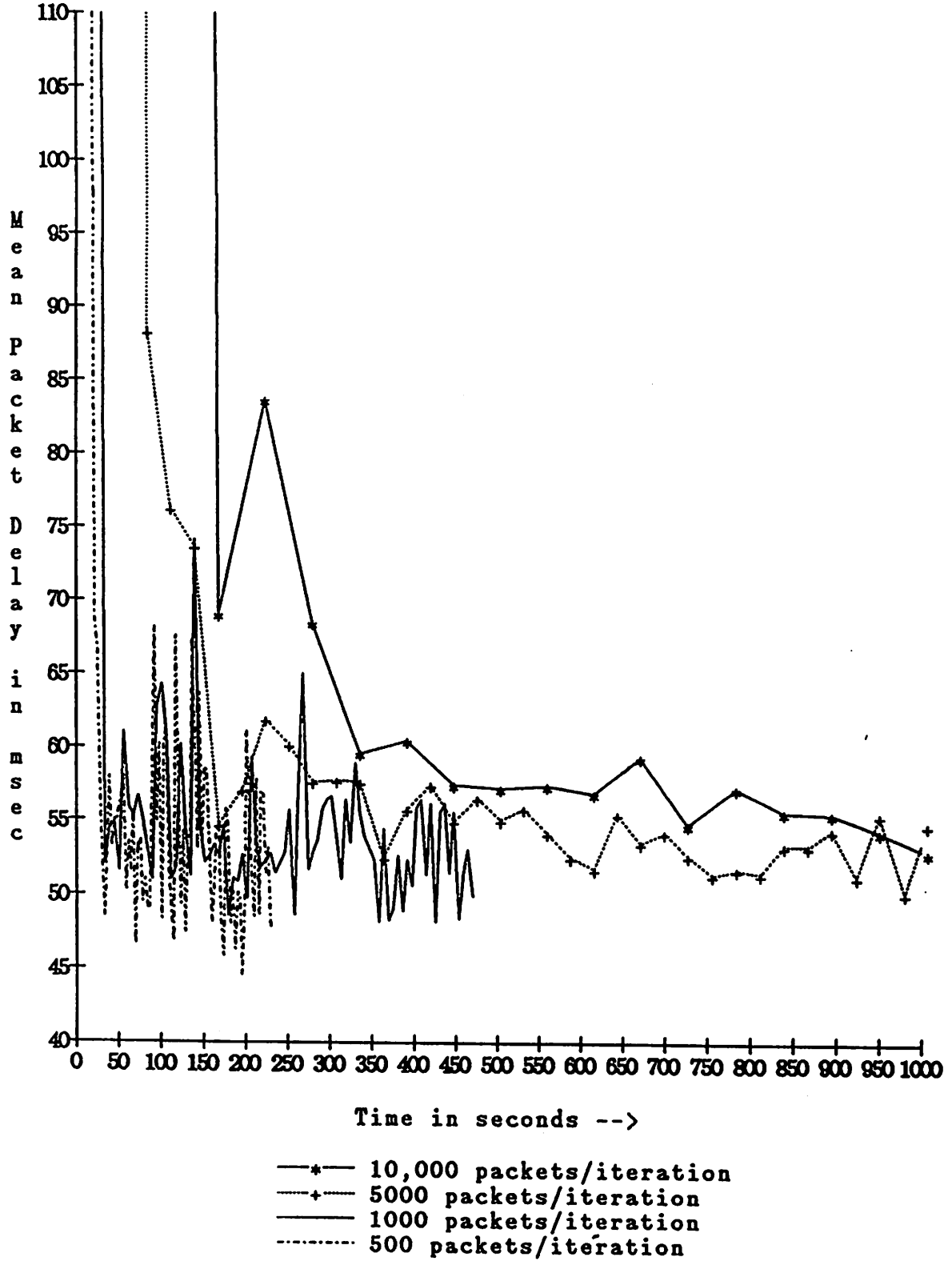

Fig. 12: *Effect of Observation Period Length on Routing Algorithm Convergence (Example 5).*

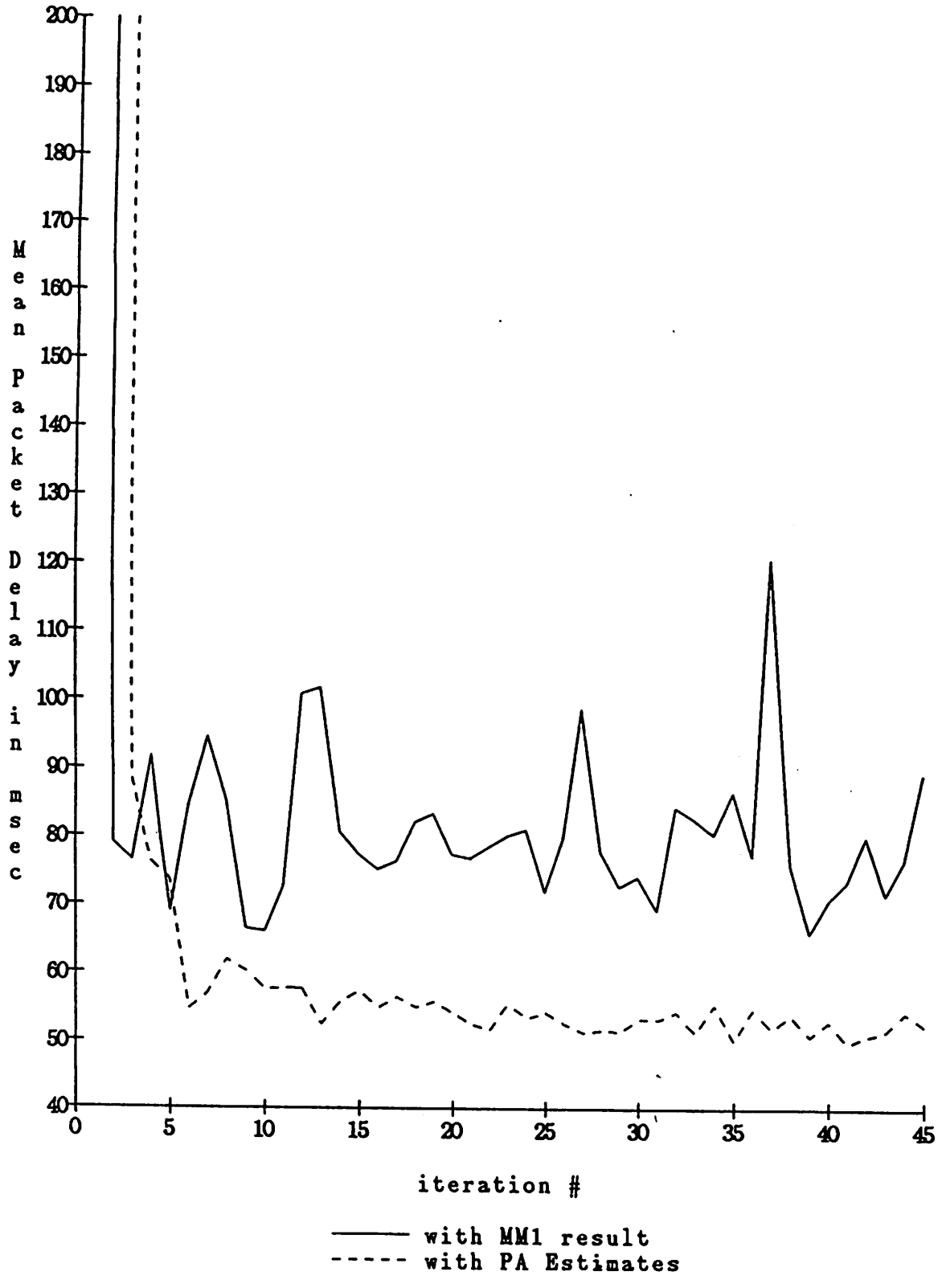

Fig. 13: *Comparison of Algorithm Performance with PA estimation vs. M/M/1 Approximation (Example 6)*

One final important point is that to use the M/M/1 approximation and corresponding analytically obtained marginal delays, one still has to estimate link flows throughout the network, in order to use them as arrival and service rate parameters in the M/M/1 link models. At the end of an iteration, one must then evaluate the response time sensitivity of all links, which is already available through PA estimation without explicit knowledge of link flows and at little extra cost (the implementation of equations (8) and (9)).

4. CONCLUSIONS

We have considered the problem of routing in computer networks in order to minimize the mean delay of packets. One reason why optimal routing algorithms have not yet been very popular in practical networks is that they are based on knowledge of delay gradients, and few methods exist to estimate such quantities simply, efficiently, and in real time, without having to make restrictive assumptions about the network traffic. In this paper, we have shown how the Perturbation Analysis (PA) technique can be used for this purpose: no assumption is made regarding the external arrival processes or the packet lengths. Furthermore, no explicit knowledge of the network parameters (arrival rates, link capacities) is required. We have used PA estimates to implement a distributed minimum delay routing algorithm on simulated networks. Through simulation experiments, we have verified that the algorithm is able to achieve nearly optimal routing (in a minimum delay sense) for several different cases, and that it is able to adapt to large changes in the network.

The performance of the routing algorithm depends on the step size and the observation period length used to apply PA in order to estimate delay gradients. We have studied the performance of the algorithm with respect to both these parameters. From the results of section 4, some general conclusions can be drawn.

First, most of the improvement in performance is obtained in the first few iterations. This initial improvement is particularly dramatic if the initial routing is poor. As the optimal value is

approached, the performance improvement between successive iterations becomes increasingly smaller. In practice, however, it is seldom necessary to attain the theoretically optimal routing. One would often be content to be within 5-10% of the (theoretical) minimum mean delay value, and, as seen from the examples in section 4, this can be achieved in relatively few iterations.

The convergence of the algorithm is critically dependent on the step size parameter, especially for more complex traffic patterns (i.e. several O→D pairs with interacting traffic). Too small a step size results in extremely slow convergence, while too large a value may preclude convergence altogether. A moderately large step size, on the other hand, leads to fast convergence, but the mean delay may exhibit some initial oscillations with very large values. In addition, the routing variables (and the mean delay) may oscillate near the optimal values. We have also observed that for large step sizes, a good initial routing results in faster convergence.

The algorithm converges even for short observation periods, which affects the accuracy of delay gradient estimates, at the expense of higher mean delay variance over the sample points defined by several iterations. In practice, it may be desirable to minimize the mean packet delay *in a given interval of time*. For instance, in a dynamically changing network environment, the specified time interval would be properly chosen so that the algorithm attempts to provide near optimum performance before a network change may occur. To achieve this goal, we have to choose a combination of large step size and short observation period, since this would give the fastest initial performance improvement.

One can expect that second derivative considerations (e.g. [2]) and certain heuristic modifications may lead to improved performance of the algorithm used here. One such modification for reducing oscillatory behavior in the routing variables was suggested and implemented in Example 4. Another possibility is to include a mechanism for adjusting the step size used at each iteration, based on the change in mean delay at successive iterations (see [6]) and to vary the step size depending on the link. Similarly, the observation period may be increased to obtain more accurate marginal delay estimates only after the mean delay is near the optimum.

Several challenging problems in routing remain to be studied. The algorithm we have used here is suitable for quasistatic environments, where periodic or infrequent routing changes are required. For networks experiencing more frequent changes, "dynamic" routing algorithms are preferable. Such algorithms make routing decisions on a packet-by-packet basis, using instantaneous state observations (e.g. queue lengths), preferably in a decentralized manner. One such approach, currently under study, uses *threshold dependent* routing. Finally, we have restricted ourselves to a single class of packets in the network, with FCFS service disciplines at all nodes. There is increasing evidence, however, that multiclass environments need to be analyzed (e.g. [12]), with each class characterized by different service requirements, including real-time constraints.

REFERENCES

[1] M.V. Abidi, "Use of Perturbation Analysis Techniques for Optimal Routing in Computer Networks", Dept. of Electrical and Computer Engineering Technical Report, University of Massachusetts/Amherst, 1987.

[2] D.P. Bertsekas, E.M. Gafni, and R.G. Gallager, "Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks", *IEEE Trans. Commun.*, COM-32, 8, pp. 911-919, 1984.

[3] D.G. Cantor, and M. Gerla, "Optimal Routing in a Packet-Switched Computer Network", *IEEE Trans. Comput.*, C-23, pp. 1062-1069, 1974.

[4] X. Cao, "Convergence of Parameter Sensitivity Estimates in a Stochastic Experiment", *IEEE Trans. Automatic Control*, AC-30, 9, pp. 690-700, 1986.

[5] C.G. Cassandras, "Error properties of Perturbation Analysis for Queueing Systems", subm. to *Operations Research*, 1987.

[6] F. Chang, and L. Wu, "An Optimal Adaptive Routing Algorithm", *IEEE Trans. Automatic Control*, AC-31, 8, pp. 845-853, 1985.

[7] L. Fratta, M. Gerla, and L. Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design", *Networks*, 3, pp. 97-133, 1973.

[8] R.G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation", *IEEE Trans. Commun.*, COM-25, 1, pp. 73-85, 1977.

[9] Y.C. Ho, and C.G. Cassandras, "A New Approach to the Analysis of Discrete Event Dynamic Systems", *Automatica*, 19, pp. 149-167, 1983.

[10] Y.C. Ho, X. Cao, and C.G. Cassandras, "Infinitesimal and Finite Perturbation Analysis for Queueing Networks", *Automatica*, 19, pp. 439-445, 1983.

[11] L. Kleinrock, "Communication Nets: Stochastic Message Flow and Delay", New York: McGraw-Hill, 1964.

[12] K.J. Lee, D.F. Towsley, and M. Choi, "Distributed Algorithms for Minimum Delay Routing with Constraints in Communication Networks", *Proc. INFOCOM '87*, pp. 188-201, 1987.

[13] M.I. Reiman, and A. Weiss, "Sensitivity Analysis for Simulations via Likelihood Ratios", manuscript, AT&T Laboratories, 1986.

[14] M. Schwartz, and C.K. Cheung, "The Gradient Projection Algorithm for Multiple Routing in Message-Switched Networks", *IEEE Trans. Commun.*, COM-24, pp. 449-456, 1976.

[15] A. Segall, "The Modeling of Adaptive Routing in Data Communication Networks", *IEEE Trans. Commun.*, COM-25, 1, pp. 85-95, 1977.

[16] R. Suri, "Implementation of Sensitivity Calculation on a Monte Carlo Experiment", *J. Optim. Theory and Applications*, Vol. 40, pp. 625-630, 1983.

[17] R. Suri, and M.A. Zazanis, "Perturbation Analysis gives Strongly Consistent Estimates for the M/G/1 Queue", subm. to *Management Science*, 1986.

[18] M.A. Zazanis, and R. Suri, "Estimating First and Second Derivatives of Response Time for G/G/1 Queues from a Single Sample Path", subm. to *Queueing Systems*, 1986.