Distributed Routing With On-Line Marginal Delay Estimation

Artin Spiridonoff

SE 733
Spring 2017

CONTENTS

ABSTRACT

In a data network, each packet enters the network from a source node and passes through different communication links and nodes to get to its destination node. In order for each packet to reach its destination, routing decisions should be made to choose its path. The routing of packets from source node to destination node is an important problem in the design of networks and it affects several performance measures of the network. The objective of a routing algorithm is to optimize some performance measure, e.g., the mean packet delay or network throughput.

In this report, we describe and study a distributed routing algorithm to minimize the mean packet delay using on-line marginal delay estimation. Then simulations of the algorithm are included to investigate the effects of different parameters on the performance of the algorithm.

1 INTRODUCTION

Routing may be done in centralized or distributed manner. In the centralized case, a special node in the network, called routing control center (RCC), collects information from all the nodes and based on them, calculates and broadcasts the routing variables to nodes. This method suffers from high communication overhead and needs to deal with the problem of handling link and node failures, and also needs to handle the failure of RCC itself.

Distributed routing avoids these problems. In this case, each node based on the information that it collects from neighboring nodes, makes its own routing decisions. It is safer in the sense that it doesn't rely on a single unit and also it solves the problem of scalability; since with increasing in the size of the network, the amount of information to be analyzed and the size of the problem that has to be solved by the RCC in the centralized routing, increases. However, in distributed routing, inconsistent routing may cause looping of the packets, which can be avoided.

The problem of routing has been approached in many ways in the literature. 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 require a knowledge of the derivatives of the objective function with respect to the control variables. Therefore estimating these derivatives accurately and efficiently, is very important.

Gallager [1] has defined an interesting algorithm to solve the distributed routing problem in a quasi-static environment. In a quasi-static routing problem, the 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. Like most other optimal routing algorithms, Gallager's algorithm assumes knowledge of the delay gradient $D'_{ik}(f_{ik})$. It can be calculated analytically if an appropriate formula giving the link delay as a function of the link flows is available. However, these formulas are based on many restrictive assumptions that do not hold in practice. Hence, instead of seeking closed form expression for $D_{ik}(f_{ik})$, it is preferred to estimate the derivative of the delay directly.

Few techniques have been proposed for on-line estimation of performance gradients of systems modeled as queuing networks. Here we use the method proposed by Cassandras et al. [2], which uses perturbation analysis (PA) to estimate link delay gradients. In [2], using PA link delay estimations, Cassandras et al. have implemented Gallager's minimum delay routing algorithm on several simulated networks. They have studied the effects of two important parameters, *observation period* and the *step size*, on the performance of the algorithm.

In this project, we simulate a data network and implement the algorithms proposed in [1] and [2]. The goal of this work, is to become familiar with the complexities involved in simulating a data network as a discrete event system (DES) and implementing the algorithm

in practice. In order to verify our results, we simulated the same network settings used in [2].

The rest of this report is organized as follows. In section 2, we describe the Minimum Delay Routing Algorithm. In section 3, we study use perturbation analysis to estimate a parameter which is needed in the routing algorithm. Simulation results are included in section 4, followed by Conclusions and further discussions in section 5.

2 MINIMUM DELAY ROUTING ALGORITHM

In this section we give a brief description of the main features of the algorithm suggested by Gallager [1]. Further details can be found in the original reference. The algorithm uses distributed computation to achieve optimal minimum delay routing. 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.

2.1 FORMULATION OF THE MODEL

We first introduce some notation that is used in the rest of this report. Let the nodes of an n-node network be represented by the integers 1, 2, ..., n and let a link from node i to node k be represented by (i,k). Let L be the set of links, L = {(i,k): a link goes from i to k}. In addition:

$r_i(j)$ = Expected traffic entering the network at node i and destined for node j in packets/s,

$t_i(j)$ = Total expected traffic at node i destined for node j in packets/s,

f_{ik} = Expected traffic over link (i,k) in packets/s; also called the link flow,

τ_{ik} = Mean inter-arrival time of packets on link (i,k), i.e., $\tau_{ik} = 1/f_{ik}$, and

$\phi_{ik}(j)$ = 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). We make the following assumptions for $\phi_{ik}(j)$:

$\phi_{ik}(j) = 0$ if $(i,k) \notin L$ or if $i = j$.

$\sum_k \phi_{ik}(j) = 0$ for all i.

For each i, j $(i \neq j)$ there is a routing path from i to j.

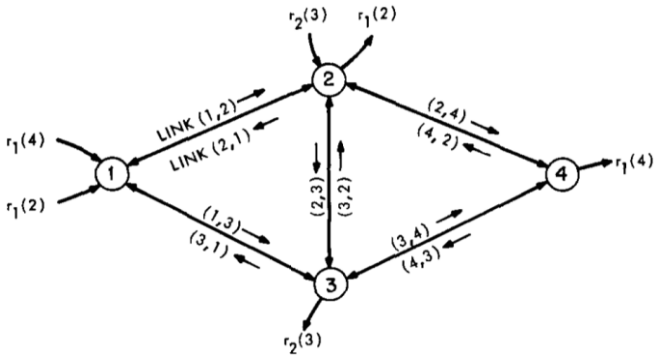

Figure 1. Nodes, links and inputs in a data network [1]

In this algorithm, the network congestion, in other words the objective function, is defined as follows:

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

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 traffic input, minimizing D_T is equivalent to minimizing the mean packet delay.

2.2 THE ALGORITHM

In the algorithm each node i must incrementally decrease those routing variables $\Phi_{ik}(j)$ for which the marginal delay $D'_{ik}(f_{ik}) + \partial D_T/\partial r_k(j)$ is large, and increase those for which it is small. The algorithm breaks into two parts: a protocol between nodes to calculate the marginal delays and an algorithm for modifying the routing variables; we discuss the protocol part first.

In order to see how node i can calculate $\partial D_T/\partial r_k(j)$ for a neighboring node k, define node m to be downstream from node i (with respect to destination j) if there is a routing path from i to j passing through m (i.e., a path with positive routing variables on each link). Similarly, we define i as upstream from m if m is downstream from i. A routing variable set Φ is loop free if for each destination j, there is no i, m ($i \neq m$) such that i is both upstream and downstream from m. If Φ is loop free, then for each destination j, the downstream (and the upstream) relation form a partial ordering on the set of nodes.

The protocol used for an update, now, is as follows: for each destination node j, each node i waits until it has received the value $\partial D_T/\partial r_k(j)$ from each of its downstream neighbors $k \neq j$ (i.e., nodes k with $\phi_{ik}(j) > 0$). The node i then calculates $\partial D_T/\partial r_i(j)$ from the following equation (using the convention that $\partial D_T/\partial r_j(j) = 0$),

$$\frac{\partial D_T}{\partial r_i(j)} = \sum_k \phi_{ik}(j)\left[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right] \tag{1}$$

where $D'_{ik}(f_{ik}) = \dfrac{\partial D_T}{\partial f_{ik}}$ is the sensitivity of the mean packet delay/s over link (i, k) with respect to the link flow and is estimated using PA. Then node i broadcasts this to all of its neighbors (except to the destination node j which has no need of the information). This procedure is free of deadlocks if and only if Φ is loop free.

In order to prevent the packets from cycling and for Φ to be loop free, it turns out to be necessary, for each destination j and each node i, to specify a set $B_i(j)$ of blocked nodes k for which $\Phi_{ik}(j) = 0$ and the algorithm is not permitted to increase $\Phi_{ik}(j)$ from 0.

The algorithm A, on each iteration, updates the current routing variable set $\Phi^{(n)}$ into a new set $\Phi^{(n+1)}$. Each iteration is defined by a sufficiently long observation period. The mapping is defined as follows.

$$A_i = \min_k \left[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right] \tag{2}$$

$$k_{\min} = \arg\min_k \left[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right] \tag{3}$$

$$a_{ik}(j) = \left[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right] - A_i \tag{4}$$

$$\Delta_{ik}(j) = \min \left[\phi_{ik}^{(n)}(j), \eta \frac{a_{ik}(j)}{t_i(j)} \right] \tag{5}$$

In (3), node i determines its "best" neighbor, $k_{\min}$, in the sense that link (i, $k_{\min}$) has the least marginal delay. In (4), node i computes the amount of flow with destination j to be transferred from link (i, k) to the "best" link (i, $k_{\min}$). This is proportional to a_{ik}, with the min(.) operation used to prevent routing variables becoming negative. The scaling factor η is an important parameter to be discussed later. Then

$$\phi_{ik}^{(n+1)}(j) = \begin{cases} \phi_{ik}^{(n)}(j) - \Delta_{ik}(j), & k \neq k_{\min} \\ \phi_{ik}^{(n)}(j) + \sum_{k \neq k_{\min}} \Delta_{ik}(j), & k = k_{\min} \end{cases} \tag{6}$$

The algorithm reduces the fraction of traffic sent on non-optimal links and increases the fraction on the best link. For η very small, convergence of the algorithm is guaranteed, but rather slow. As η increases, the speed of convergence increases but the danger of no convergence also increases.

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 $D'_{ik}(f_{ik})$ makes restrictive assumptions, we wish to estimate it directly. In this section we briefly describe the perturbation analysis (PA) method used to estimate this quantity.

The PA is an efficient way to estimate performance sensitivities in complex discrete event systems. The PA method assumes that the sample path of the system is observed. An important advantage of the PA is that it is based on real data and doesn't make many restrictive assumptions required for stochastic modeling. It also provides online estimation of the performance sensitivities.

3.1 ESTIMATING PACKET RESPONSE TIME PERTURBATIONS

For our purposes, we model a network link as a GI/G/1 queuing system. Let a_k, d_k and S_k be the arrival time, departure time and service time of the k-th packet on the link, respectively. We have:

$$d_k = \max\{d_{k-1}, a_k\} + s_k, \quad k = 1, 2, 3, \ldots \tag{7}$$

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. We refer to a system characterized by these distributions and the departure time dynamics in (7) as the nominal system. Clearly, a sample path of this system is characterized by an inter-arrival time sequence A_1, A_2, … and a service time sequence S_1, S_2, … generated according to $F_A(.;\tau)$ and $F_S(.;\sigma)$, respectively. Now suppose that there is an underlying sequence $\omega = (\omega_1, \omega_2, \ldots)$ of random numbers generated from a uniform distribution in [0, 1], and that A_k is obtained through the transformation $A_k = F_A^{-1}(\omega_k, \tau)$, k = 1, 2, … (and similarly, another sequence is used for S_k, k = 1, 2, …). In this framework, a perturbed system is obtained if the parameter τ is perturbed by $\delta\tau$ (which may be positive or negative), and the corresponding perturbed sample path is created by using the same, but by generating an interarrival time sequence $A'_1, A'_2, \ldots$ where $A'_k = F_A^{-1}(\omega_k, \tau + \delta\tau)$ for the same ω_k. Interarrival time perturbations $\delta A_k = A'_k - A_k$ are therefore given by

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

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 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. In a large class of distributions, including exponential distribution, the mean is a scale parameter. For the perturbed path created by $\delta\tau$, similar to (7), we get

$$d'_k = \max\{d'_{k-1}, a'_k\} + s_k, \quad k = 1, 2, 3, \ldots \tag{8}$$

where d'_k and a'_k are the departure and arrival times, respectively.
From (7) and (8) we get

$$\delta d_k = \begin{cases} \max\{\delta d_{k-1} - (a_k - d_{k-1}), \delta a_k\} & \text{if } a_k > d_{k-1} \\ \max\{\delta d_{k-1}, \delta a_k + (a_k - d_{k-1})\} & \text{if } a_k \le d_{k-1} \end{cases} \tag{9}$$

This equation provides a recursive relationship for evaluating departure time perturbations by observing a single sample path of the nominal system.
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 (7) give

$$r_k = \max\{r_{k-1} - A_{k-1}, 0\} + S_k \tag{10}$$

where $A_{k-1} = a_k - a_{k-1}$ is the interarrival time following the (k-1)th packet. The response time perturbation $\delta r_k = r'_k - r_k$ can be recursively evaluated as

$$\delta r_k = -\delta A_k \begin{cases} \max\{\delta r_{k-1} - (A_k - r_{k-1}), \delta A_k\} & \text{if } A_k > r_{k-1} \\ \max\{\delta r_{k-1}, \delta A_k + (A_k - A_{k-1})\} & \text{if } A_k \le r_{k-1} \end{cases} \tag{11}$$

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

$$\delta R_k = \frac{1}{K}\sum_{k=1}^{K} \delta r_k \tag{12}$$

Where δr_k is evaluated by the PA algorithm in (11).

3.2 PA Estimate of Marginal Delay in the Routing Algorithm

Let R_{ik} be the mean response time of the packets. By Little's law

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

and we get

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

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

which yields

$$D'_{ik} = R_{ik} - \tau_{ik}\left(\frac{\partial R_{ik}}{\partial \tau_{ik}}\right) \tag{13}$$

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 (12). Hence, from (13), our estimate of D'_{ik} is

$$\hat{D}'_{ik} = R_{ik} - \left(\frac{\tau_{ik}}{\delta\tau_{ik}}\right)\delta R_{ik} \tag{14}$$

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

4 SIMULATION RESULTS

In this section, we present the results of simulating a data network, in which we use the Gallager's routing algorithm, and PA to estimate the derivatives. The simulation has been done using Python language. In this code, the data network is simulated as a discrete event system with an event calendar, in which the set of all possible events is defined as packets entering network at different nodes with different destinations, and packets departing a link. The network is observed for a certain interval of time (the observation period), which defines one iteration of the routing algorithm. For each packet, we record its response time in each link, and also the total time it spend in the network. Then, at the end of the observation period, routing variables are updated using the estimated marginal delays. The initial routing in each case is chosen arbitrarily.

The value of $\left(\frac{\delta\tau_{ik}}{\tau_{ik}}\right) = 0.001$ is used in all the simulations. We have applied the minimum delay routing algorithm, with PA marginal delay estimation, to the four-node network shown in Figure 2.

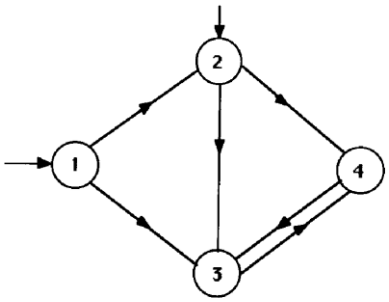

Figure 2. Network used in simulations [2]

4.1 EXAMPLE 1

This exact example has been simulated in [2]. In this report, we use the same simulation settings which are described in the following. This is to be able to compare and verify our results.

In this example, packets enter the network at node 1 only, and are all destined for node 4. In order to compare our experimental results to analytical ones, the inter-arrival and service times are assumed to be exponentially distributed, and that the independence assumption holds. The mean inter-arrival time is 10.0 ms, and the mean service time of each link is 8.0 ms.

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, using standard queuing theoretic results (Jackson's theorem), it can be shown that the mean packet delay is DT = 26.7 ms [2].

In Figure 3 we have plotted the Mean Packet Delay over time, for different observation period lengths, (T denotes the observation period length).

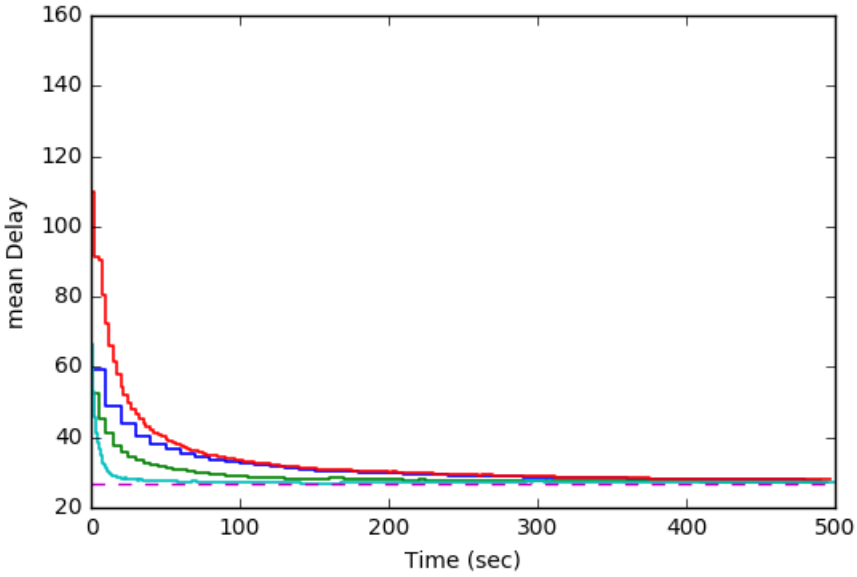

Figure 3. Mean packet delay (ms) over time (s),
Red: T=10 s, Blue: T=5 s,
Green: 2.5 s, Light Blue: T=1 s,
$\eta = 0.00015$

It can be seen that for shorter observation periods, the convergence speed is faster. Also we observe that in all of the simulations, the final value of the packet delay converges to the optimal analytical value (dashed line). Furthermore, the routing variables converged to

$$\phi_{12}(4) = 0.5, \quad \phi_{13}(4) = 0.5,$$
$$\phi_{23}(4) = 0.0, \quad \phi_{24}(4) = 1.0,$$

which is what we expected.

4.2 EXAMPLE 2

In this example, we make the network simulation more complex by increasing the packet origin-destination pairs to the following three: $1 \to 3$, $1 \to 4$ and $2 \to 4$ with inter-arrival times exponentially distributed with mean 0.1 ms for each pair, and service times of each link set to 0.5 ms.

There is no analytical solution for the optimal total packet delay for this example; however, we study the effect of two parameters, step size and observation period, on the algorithm performance.

In this example, once we fixed the step size and investigated the effect of observation period (Figure 4), and once we fixed the observation period length and simulated the network for different step sizes (Figure 5).

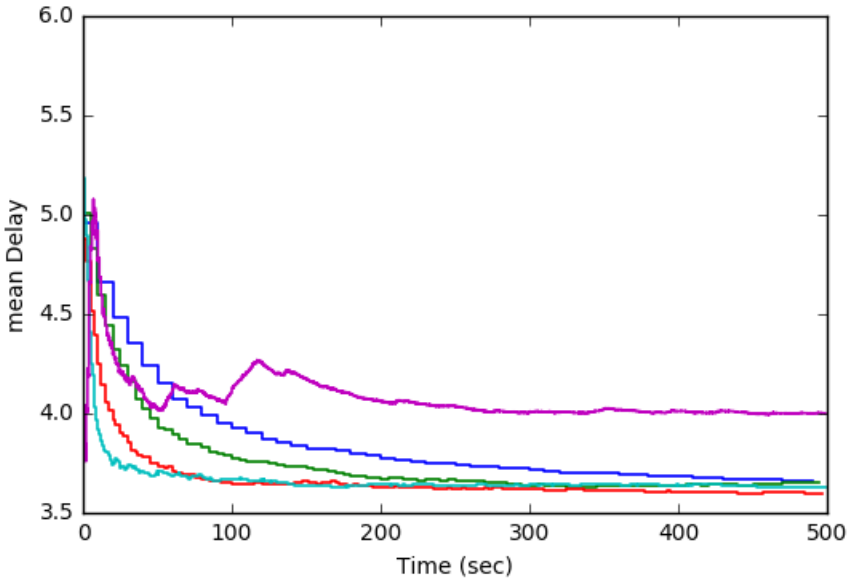

Figure 4. Mean packet delay (ms) over time (s),
Blue: T=10 s, Green: T=5 s,
Red: 2.5 s, Light Blue: T=1 s, Purple: T=0.05 s,
$\eta = 0.005$

From Figure 4, we can again see the effect of smaller observation period on faster convergence. However, in this case we added a very small observation period of 0.05 s in which the algorithm doesn't seem to be converging to the correct optimal value. The reason behind this is that, the algorithm doesn't have enough time to estimate the marginal delays correctly. The longer the observation period, the more precise is the PA estimate of marginal delays. On the other hand for longer observation periods, the algorithm has to wait a longer time to update its routing variables and hence it will converge slower.

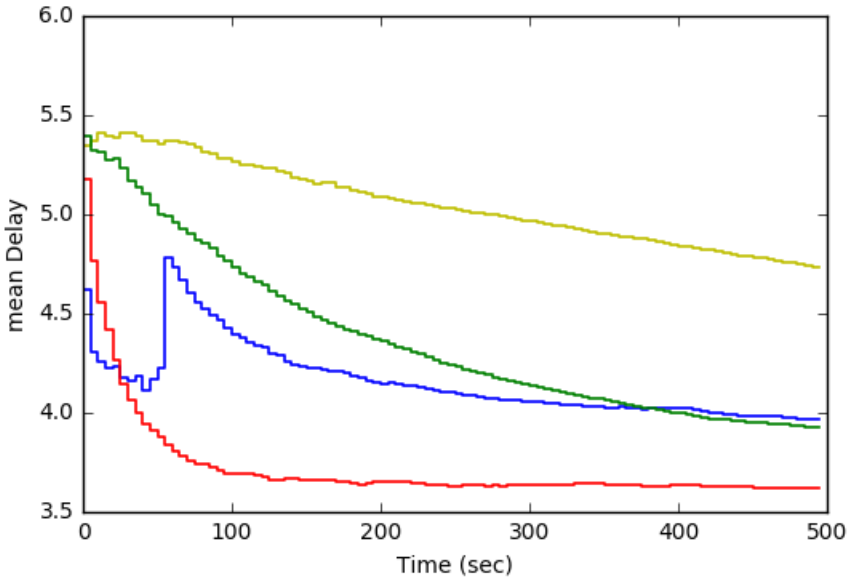

Figure 5. Mean packet delay (ms) over time (s),
Blue: $\eta = 0.05$, Red: $\eta = 0.005$,
Green: $\eta = 0.0005$, Yellow: $\eta = 0.0001$,
T=5 s

We can investigate the effect of step size in Figure 5. It is observed that the smaller the step-size, the slower the algorithm converges. However, increasing the step-size too much, the algorithm shows instability and might not converge to optimal value (blue curve).

5 CONCLUSION

In this report, we described and studied a minimum delay decentralized routing algorithm for data networks. This algorithm uses the links marginal delay estimates to update its routing variables. We used a PA technique to estimate link delay sensitivities online, using the observed sample path. We then implemented these algorithms together to simulate a data network with different settings to investigate the effect of two main parameters of the algorithm, *step size* and *observation period*. We compared our results with the analytical available solutions to verify the eligibility of the implementation and simulation.

The main contribution of this report was the simulation part, which deals with a lot of complexities that are involved with simulating a data network and tracking packets with different destinations. Moreover, the simulation was done using Python programming language, using only simple mathematical operations, without being reliant on any particular libraries suited for network simulations or discrete event systems.

The main advantage of the written code, is that it is structured so that it can be used for different network topologies and inputs, and not just to perform the simulations brought in this report.

In order to continue the research in this direction, one can add the feature of asynchronous routing updates to the network, so that each node can update its variables independently of the others, only when it detects it has collected enough information.

6 REFERENCES

[1] Robert G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation", IEEE Transactions on communications, vol. COM-25, NO. 1, pp. 73-85, January 1977

[2] Christos G. Cassandras, M. Vasmi Abidi, Don Towsley, "Distributed Routing with On-Line Marginal Delay Estimation", IEEE Transactions on communications, vol. 38, NO. 3, pp. 348-359, March 1990

7 APPENDIX

The written Python code is attached to this report in a separate file.