Passive Estimation Algorithms for Queueing Delays in LANs and Other Polling Systems*

D. Manjunath

Dept of Electrical Engineering
Indian Institute of Technology
Kanpur — 208 016 India
dmanju@iitk.ernet.in

Mart L. Molle

Computer Science Dept
University of California, Riverside
Riverside, CA 92521
mart@cs.ucr.edu

Abstract

Queue inferencing algorithms are used to derive estimates of queue lengths and/or customer waiting times from a priori information about the customer arrival process and the observed sequence of times at which each customer enters and leaves service. In this paper, we extend these techniques by decoupling the arrival time constraints from the customer departure times, which allows us to handle additional features like server vacations. We then show how these techniques can be used to monitor a single station in a polling system, or in a shared medium Local Area Network such as Ethernet, Token Ring and FDDI. Using these results, passive, non-intrusive network monitoring tools could be developed to estimate waiting times and queue lengths for any host on the network by observing only the packet departure times from the nodes.

1 Introduction

Recent results by Larson [1], Bertsimas and Servi [2], and Daley and Servi [3, 4] provide algorithms for estimating queue lengths during a busy period for an M/G/1 queueing system, given transactional data that is visible to a passive external monitor. The visible data consist of only the service initiation and service termination events for each customer, or equivalently the customer departure instants together with the beginning and the end of each busy period. Larson introduced this queue inference problem and proposed an $\mathcal{O}(n^5)$ algorithm to estimate the queue lengths over an $n-$customer busy period. Bertsimas and Servi later developed an $\mathcal{O}(n^3)$ algorithm, while Daley and Servi describe a reasonably accurate approximate $\mathcal{O}(n^2 \log(n))$ algorithm. In all of these algorithms, the estimates are computed *off-line*, at the end of each busy period, and the algorithms are independent of both the customer arrival rate (as long as it is a stationary Poisson process) and the service time distribution. However, it is necessary that the service discipline be FCFS.

Some estimation algorithms have also been derived for more general queueing problems. For example, Bertsimas and Servi [2] also considered the case where the arrival process is a time dependent Poisson process or a renewal process with a known distribution, while Daley and Servi [4] also considered M/G/1 queues with balking, reneging and finite buffers, and the M/G/m queueing systems. In addition, Bertsimas and Servi provide an *on-line* $\mathcal{O}(n)$ estimation algorithm for the current queue length at the nth departure during an ongoing busy period, which is general enough to handle time dependent Poisson arrival processes.

The rest of the paper is organized as follows. In section 2 we develop a new *off-line* estimation algorithm for the waiting times of departing customers in an M/G/1 queue with FCFS service. In section 3 we introduce the problems of monitoring one of the stations in a polling a system, state our assumptions and introduce some additional notation. In section 4 we provide a framework for applying our waiting time estimation algorithm, and the queue length estimation algorithms of Bertsimas and Servi, to the monitoring of polling systems with exhaustive, k-limited and time-limited service. We then discuss applications to the monitoring of LAN systems in sections 5 and 6.

*This work was done while the authors were with the Dept. of Computer Science, University of Toronto, Toronto, Canada M5S 1A4, and was supported by the Natural Sciences and Engineering Research Council of Canada under grant A5517.

2 Estimation Algorithms

In this section we derive our algorithm for estimating the waiting times for departing customers in the busy period of an M/G/1 queue.

2.1 Arrival Time Estimation

Consider an interval $\mathcal{I} = (0, t_n]$ that includes n customer arrivals from a Poisson source, subject to the constraints that the m^{th} arrival occurred no later than time t_m, $m = 1, \cdots, n$. In this section, we will derive the *minimum mean square error estimate* for τ_m, the actual arrival time for the m^{th} customer. (To simplify the expressions we obtain, we find it convenient measure time from the left-hand boundary of $\mathcal{I}$ and to define $\tau_0 \equiv 0$.) This problem formulation is a generalization of the usual situation in queue inferencing algorithms, where the interval $\mathcal{I}$ represents a busy period for a single server queue, and the constraint t_m corresponds to the departure time y_{m-1} for the $(m-1)^{st}$ customer. We give the more general form here to permit easy application of our results to polling systems, Local Area Networks, etc., in later sections. We assume that the interval $\mathcal{I}$ and the constraints t_m are known to our monitoring algorithm, with which we must estimate the unknown τ_m's. Define the vectors $\underline{t} = [t_1, t_2, \cdots, t_n]$ and $\underline{\tau} = [\tau_1, \tau_2, \cdots, \tau_n]$.

If all we knew about the arrivals was that n of them from a Poisson source fell within the interval $\mathcal{I}$, then the joint arrival time density, $f'_n(\underline{\tau})$, would be [1]

$$f'_n(\underline{\tau}) = \frac{n!}{(t_n)^n} \qquad (1)$$

However, we do know more about the arrivals, in particular that the m^{th} arrival occurred before t_m. Thus, for any *feasible* arrival vector $\underline{\tau}$, the corresponding joint density $f_n(\underline{\tau})$ is obtained from $f'_n(\underline{\tau})$ by multiplying by the normalization constant $1/p_n$, where p_n is the probability that a randomly chosen arrival vector would satisfy these constraints and is given by

$$p_n = \int_{\tau_1=0}^{t_1} \int_{\tau_2=\tau_1}^{t_2} \cdots \int_{\tau_n=\tau_{n-1}}^{t_n} \frac{n!}{(t_n)^n} d\tau_n \cdots d\tau_2 \, d\tau_1$$

Therefore, the joint arrival time density, given all the information we have available, is given by

$$f_n(\underline{\tau}) = \frac{f'_m(\underline{\tau})}{p_n} \qquad (2)$$

as long as $\underline{\tau}$ satisfies the given constraints, i.e., $\tau_{i-1} < \tau_i < t_i$ for all $i = 1, 2, \cdots, n$; otherwise $f_n(\underline{\tau}) = 0$.

[1]Note the difference from Eqn. 1 of [2]: in our model we allow the first customer arrival to occur *within* the interval $\mathcal{I}$, rather than *defining* its left-hand boundary.

Lemma 1 *For $p \geq 0$, let $\phi_{p,p}(\tau_p) = 1$ and for $q > p$ let*

$$\phi_{p,q}(\tau_p) = \int_{\tau_{p+1}=\tau_p}^{t_{p+1}} \int_{\tau_{p+2}=\tau_{p+1}}^{t_{p+2}} \cdots \int_{\tau_q=\tau_{q-1}}^{t_q} d\tau_q \cdots d\tau_{p+1}$$

Then $\phi_{p,q}(\tau_p)$ is a polynomial of degree $q-p$ in τ_p and can be represented by

$$\phi_{p,q}(\tau_p) = \sum_{j=0}^{q-p} c_{p,q}(j)\tau_p^j \qquad (3)$$

where the coefficient $c_{p,q}(j)$ of the j^{th} power of τ_p is

$$c_{q,q}(0) = 1 \qquad (4)$$

For $r = q-1, q-2, \cdots, p$

$$c_{r,q}(j) = \begin{cases} \sum_{k=0}^{q-r-1} c_{r+1,q}(j)\frac{t_{r+1}^{(k+1)}}{k+1} & j = 0 \\ -\frac{c_{r+1,q}(j-1)}{j} & 0 < j \leq q-r \end{cases} \qquad (5)$$

Proof: For $q = p$, the lemma is true with $c_{p,p}(0) = 1$. Assuming the lemma holds for some $q \geq p$, we evaluate $\phi_{p,q+1}(\tau_p)$ as follows

$$\begin{aligned} \phi_{p,q+1}(\tau_p) &= \int_{\tau_{p+1}=\tau_p}^{t_{p+1}} \sum_{j=0}^{q-p} c_{p+1,q+1}(j)\tau_{p+1}^j d\tau_{p+1} \\ &= \sum_{j=0}^{q-p} c_{p+1,q+1}(j) \left[\frac{t_{p+1}^{(j+1)}}{j+1} - \frac{\tau_p^{(j+1)}}{j+1} \right] \\ &= \sum_{j=0}^{q+1-p} c_{p,q+1}\tau_p^j \end{aligned}$$

where $c_{p,q+1}(j)$ is defined by Eqn. 5. Hence the lemma.

Lemma 2 *For $m = 1, 2, \cdots$*

$$\begin{aligned} &\int_{\tau_1=0}^{t_1} \int_{\tau_2=\tau_1}^{t_2} \cdots \int_{\tau_m=\tau_{m-1}}^{t_m} \tau_m^j d\tau_m \cdots d\tau_2 \, d\tau_1 \quad (6) \\ &= \sum_{i=1}^{m} (-1)^{i+1} \, t_{m+1-i}^{j+i} \, \phi_{0,m-i}(0) \frac{j!}{(j+i)!} \quad (7) \end{aligned}$$

Proof: Proof is by direct evaluation.

Using the above lemmas, it is interesting to note that p_n and $f_n(\underline{\tau})$ for a feasible arrival vector can be rewritten as

$$\begin{aligned} p_n &= \frac{n!}{(t_n)^n} \phi_{0,n}(0) \\ f_n(\tau_1, \tau_2, \cdots \tau_n) &= \frac{1}{\phi_{0,n}(0)} \end{aligned}$$

Theorem 1 *The minimum mean square error estimate, $E\tau_m$, of the arrival time for the mth customer out of n in the interval $\mathcal{I}$, given the arrival process is Poisson and the constraints $\underline{t}$, is given by*

$$E\tau_m = \sum_{j=1}^{n-m+1} \frac{c_{m,n}(j-1)}{c_{0,n}(0)} \sum_{i=1}^{m} \frac{j!(-1)^{i+1} t_{m+1-i}^{j+i} c_{0,m-i}(0)}{(j+i)!} \tag{8}$$

Proof: Proof is by direct evaluation of $E\tau_m$.

2.2 Waiting Time Estimation

The results from the previous section provide all the information we need to create an *off-line* queue inferencing algorithm for estimating waiting times. The inputs to our algorithm are the starting and ending times for a busy period in a single server FCFS queueing system with Poisson arrivals, together with the departure times $y_1, \cdots, y_n$ for each of the n customers served in that busy period. The output is the *minimum mean square estimate*, Ew_m, for the mth customer's waiting time. Note that our algorithm also permits the first customer of the busy period to receive extraordinary service, i.e., it must wait for some time before its actual service begins. Examples of such systems include some types of M/G/1 queues with server vacations and some types of priority queues. We will use y_0 to represent the end of this server vacation, when it exists.

Theorem 2 *The minimum mean square error estimate, Ew_m, of the waiting time for the mth customer in an n-customer busy period of an M/G/1 queueing system with extraordinary service for the first customer is given by*

$$Ew_m = y_{m-1} - \sum_{j=1}^{n-m+1} \frac{c_{m,n}(j-1)}{c_{0,n}(0)} \sum_{i=1}^{m} \frac{j!(-1)^{i+1} y_{m-i}^{j+i} c_{0,m-i}(0)}{(j+i)!} \tag{9}$$

Proof: The waiting time of the mth customer in an n-customer busy period is

$$w_m = y_{m-1} - \tau_m \tag{10}$$

where y_{m-1} is an input to our monitoring algorithm, and τ_m is an unknown that requires estimation. Taking expectations, we have

$$Ew_m = y_{m-1} - E\tau_m \tag{11}$$

and it remains to evaluate $E\tau_m$. But recall that for each customer served within the same busy period for a FCFS queueing system, its arrival time must have occurred before the departure of the previous customer (if any). Thus, since $E\tau_m$ can be computed using theorem 1 when we let $t_m = y_{m-1}$ for all $m = 1, \cdots, n$, the theorem follows immediately from Eqn. 11.

3 Extending the Applications

Unlike most of the literature on queue inferencing, we derived our waiting time estimation algorithm in terms of an arbitrary interval $\mathcal{I}$ that is known to include the customer arrivals, together with some additional constraints $\underline{t}$ on the arrival times of individual customers. This generalization makes it obvious how to apply our algorithm to other situations where bounds on the arrival times can be determined. The queue length estimation algorithms of Bertsimas and Servi can be treated similarly.

In this section, we show that these algorithms can be applied to polling systems. Priority queueing models and to vacation server models can be handled similarly. In applying these algorithms, we assume that the monitor can determine the associated transactional data, such as the beginning and end of each busy period and the bounds on the arrival times for each customer.

3.1 Monitoring a Polling System

Our goal in this section is to estimate the average waiting times and queue lengths at the individual stations in a polling system. To this end, we define the problem and state our assumptions.

We assume that the monitor can observe the times of arrival, $A_i(j)$, and departure, $D_i(j)$ of server on its jth visit to station i, along with the departure times $y_{1,i}(j), y_{2,i}(j), \cdots$ for each of the $n_i(j)$ station i customers who were served during the jth visit.

We assume that the monitor is not told explicitly when the queue at station i becomes empty, but must instead deduce this for itself from the above transactional information based on *a priori* knowledge about the service discipline that the server is using, and its parameters. Thus, for example, under the k-limited service discipline, we assume that the monitor knows $K_i(j)$, the maximum number of customers that can be served at the jth of the server to station i. Similarly, for the time-limited service discipline we assume that the monitor knows $\theta_i(j)$, the maximum time that the server can spend at station i during its jth visit, beyond which it must not begin serving any new customers.

Given this context, we can now define a busy period for station i *as seen by the monitor.*

Definition 1 *A* **monitored busy period** *at station i consists of the interval between $V_i + 1$ consecutive departures from station i by the server, such that:*

1. *The first $V_i - 1$ of the included visits by the server ended when it was time for the server to leave station i according to the service discipline.*
2. *The V_i^{th} included visit by the server ended before it was required to leave station i.*
3. *At least one customer was served at station i.*

Observe that a monitored busy period for station i always begins and ends with some server departure from that station, i.e., it includes an integral number of service "cycles." However, the set of all monitored busy periods does not cover the entire transactional log: a "cycle" represents a server vacation in the station i model if no station i customers were served in that cycle and the server was not required to leave station i according to the service discipline on the previous "cycle." Thus, we define $V_i(r)$ to be the total number of server visits included in the r^{th} busy period at station i, where $F_i(r)$ is the visit number of first of these visits and $L_i(r)$ is the visit number of the last.[2] Thus

$$V_i(r) = L_i(r) - F_i(r) + 1 \qquad (12)$$

Let $M_i(r)$ be the total number customers served in the r^{th} busy period for station i, where

$$M_i(r) = \sum_{j=L_i(r)}^{F_i(r)} n_i(j). \qquad (13)$$

In order to apply the previously described queue inferencing algorithms to the r^{th} busy period for station i in a polling system, we adopt $D_i(F_i(r) - 1)$ as our time origin. We will also be using the notion of *service completion time* of a customer defined as follows.

Definition 2 *The* **service completion time** *of the m^{th} customer at station i, $\tilde{y}_{m,i}$, is the time at which the server is ready to accept the $(m+1)^{st}$ customer at station i (if any) into service.*

This definition is similar to the one used by Gaver [8] for priority queueing systems. Thus, using a similar approach to our treatment of priorities and vacations, we let $\tilde{y}_{0,i} = A_i(F_i(r))$ be the end of the server vacation that initiates the r^{th} monitored busy period at station i, and thereafter for $m = 1, \cdots, M_i(r)$, we define $\tilde{y}_{m,i}$ for the m^{th} customer served in that busy period to be its departure time, if the given customer was not the last one served in a particular visit to the station, or the server's next time of arrival at station i, otherwise. As before, we assume that the customers at a given station are served on a FCFS basis and that the arrival and service processes are continuous in time. In our subsequent discussion, we will be dealing with only a specific busy period. For notational convenience, when referring to a busy period we will drop the reference to r, the sequence number of the busy period.

[2]Note that according to the above definition, no station i customers need be served at the $L_i(r)^{th}$ visit, if $L_i(r) > F_i(r)$.

4 Application to Polling Systems

4.1 Exhaustive Systems

In the exhaustive service discipline, when the server departs from station i at time $D_i(F_i - 1)$ the queue at station i is empty by definition. Thus, each busy period includes only one visit to the given station by the server. We can, therefore, state the following properties for exhaustive service polling systems by inspection.

Property 1 *In the exhaustive service polling scheme, the time between two consecutive departures of the server from station i is either a monitored busy period for station i, if at least one station i customer is served during the visit, or a vacation, otherwise. For any monitored busy period at station i we have*

$$\begin{aligned} V_i &= 1 \\ F_i &= D_i \\ M_i &= n_i(F_i) \end{aligned}$$

Using the beginning of the busy period, $D_i(F_i - 1)$, as a time origin, the elements of the service completion time vector are given by

$$\begin{aligned} \tilde{y}_{0,i} &= A_i(F_i) - D_i(F_i - 1) \\ \tilde{y}_{m,i} &= y_{m,i} - D_i(F_i - 1) \qquad 1 \le m < M_i \\ \tilde{y}_{M_i,i} &= D_i(F_i) - D_i(F_i - 1) \end{aligned}$$

and the interval $\mathcal{I}$ containing the arrivals runs from 0 to $\tilde{y}_{M_i-1,i}$, subject to the constraints that the m^{th} customer arrived no later than $t_m \equiv \tilde{y}_{m-1,i}$.

4.2 Limited Service systems

There are two kinds of limited service systems, the k-limited system and the time limited system. In the k-limited systems, the server limits the number of customers it will serve during one visit to a station, $K_i(j)$ for station i during visit j. In most polling models $K_i(j)$ is a constant for all j, although there are some exceptions such as the Bernoulli schedule where it is a geometrically distributed random variable. If the queue at station i is empty before serving $K_i(j)$ customers, the server moves on to the next station.

In time limited service polling systems, the server limits the amount of time it will spend at the station. A new service will begin at the station if the queue is non-empty and if the time spent at the station by the server during the current visit has not exceeded the limit for the visit, $\theta_i(j)$. The FDDI protocol belongs to this class with $\theta_i(j)$ being determined by the server at the beginning of the visit using a token rotation timer.

It is easy to see that, under limited service, the monitored busy period at a station does not necessarily end when the server leaves the station. If the server leaves station i early i.e., if the server leaves after serving less than $K_i(j)$ customers during the visit for k-limited systems and if it leaves before $\theta_i(j)$ for time limited systems, it is an indication that the queue at station i is empty and hence the monitored busy period at station i has ended with this visit of the server. On the other hand, if the server leaves station i on time, i.e., after serving exactly $K_i(j)$ customers or after staying there for a duration of at least $\theta_i(j)$, the monitor is not sure if the busy period ended. In particular, the monitor cannot distinguish between the case where the server left some customers behind in queue i and the case where the server's maximum visit was just enough to empty queue i, but the next arrival occurred before the server returned. Thus, the customers (if any) that are served during the server's next visit are included in the same monitored busy period, even if queue was empty when the server departed.

From the above discussion we can state the following properties for limited service polling systems

Property 2 *In limited service polling schemes, a monitored busy period for station i is an interval that begins and ends at server departure instants from station i, and includes V_i visits by the server to station i. During the first $V_i - 1$ of those visits, the server departed on time, i.e., following conditions are satisfied:*

$$\begin{array}{rcll} n_i(j) & = & K_i(j) & \textit{for } K\textit{-limited systems} \\ D_i(j) - A_i(j) & \geq & \theta_i(j) & \textit{for time limited systems} \end{array} \quad (14)$$

For any monitored busy period at station i we have

$$\begin{aligned} V_i &\geq 1 \\ M_i &= \sum_{j=F_i}^{L_i} n_i(j) \end{aligned}$$

Using the beginning of the busy period, $D_i(F_i - 1)$, as a time origin, the elements of the service completion time vector are given by

$$\begin{aligned} \tilde{y}_{0,i} &= A_i(F_i) - D_i(F_i - 1) \\ \tilde{y}_{m,i} &= y_{m,i} - D_i(F_i - 1) \qquad 1 \leq m < n_i(F_i) \\ \tilde{y}_{n_i(F_i),i} &= A_i(F_i + 1) - D_i(F_i - 1) \\ \tilde{y}_{n_i(F_i)+m,i} &= y_{n_i(F_i)+m,i} - D_i(F_i - 1) \\ & \qquad\qquad 1 \leq m < n_i(F_i + 1) \\ &\vdots \\ \tilde{y}_{M_i,i} &= D_i(L_i) - D_i(F_i - 1) \end{aligned}$$

and the interval $\mathcal{I}$ containing the arrivals runs from 0 to $\tilde{y}_{M_i-1,i}$, subject to the constraints that the m^{th} customer arrived no later than $t_m \equiv \tilde{y}_{m-1,i}$.

5 Application to LAN Monitoring

In this section we show how the monitoring techniques described above can be used to estimate queue lengths and/or waiting times at some device(s) which are connected to a shared medium LAN. Given the requirements of the algorithms, the approach is simple: we need only connect a passive receiver to the medium from which we can record a "transaction log" describing the activity of the target device(s). That is, we need a connection to the medium from which we can determine the starting and ending times for each packet transmission by the target device, together with the necessary information for deciding whether or not the device used each opportunity to transmit.

In general, this information can be obtained by connecting the monitor to the medium and configuring its network interface for promiscuous receive. Note that, in general, the event times in the "transaction log" collected by the monitor will be offset from true event times because of the propagation delays across the network as well as clock drift. However, this offset is of no consequence for our results since it is applied equally to every event. In the following sections, we will discuss the monitoring of both Ethernet and token rings, such as FDDI, in more detail.

5.1 Monitoring Ethernet

In Ethernet (or IEEE 802.3) networks, the monitor must be connected to the same collision domain as the target device(s). Clearly, the monitor can determine the starting and ending times for each packet transmitted successfully by the target device by filtering the traffic by source address. However, the difficulty with Ethernet arises when we try to decide if the target device is using all of its transmission opportunities, and hence that is in the midst of a busy period. In general, all activity by the target device will not be visible to a monitor with a promiscuous receiver because collision fragments are often removed by the runt filter in the network interface, and even if they are not the identities of the colliding transmitters cannot be determined reliably from the garbled packet fragments. Moreover, because CSMA/CD is a random-access protocol, there aren't even any specific assigned transmission times where we can look for activity by the target host. Indeed, because of the binary exponential backoff algorithm, an active device that has experienced sufficiently many collisions with its current packet may not transmit for a long time *even if the network is completely idle.*

Thus, we will assume that an Ethernet monitor has a direct physical connection to the target device with which it can detect *all* transmission attempts — even those that fail as collisions. In the case of a coaxial cable network (i.e., 10Base5 and 10Base2), this can be done by inserting a hardware monitor into the connection between the target device and the cable — for example, see [5]. Fortunately for us, however, the most widely used version of Ethernet is 10BaseT, where every host is directly connected to an active hub. In this case, the required information can easily be collected on demand via some simple additions to the management software in the hub.

Given this information about transmission attempts by the target host, it is now a simple matter for the monitor to identify its busy periods. For each busy period for that host, we define t_m as the beginning of the first transmission attempt for its m^{th} packet. Because of deference to ongoing transmissions under 1-persistent CSMA, we define the start of the busy period to be t_1, if that first transmission attempt took place while the channel was quiet, or at the previous start-of-carrier event if it follows immediately after some other transmission. After the i^{th} packet of the busy period has been transmitted successfully, the target host must wait for an interframe gap (i.e., 96 bit times) and then can begin transmitting its next packet.[3] Thus, if the target host does not make another transmission attempt at this time, the monitor can conclude that the busy period is over.

5.2 Monitoring Token Ring Networks

In token ring networks, the token arrival at host i and its release correspond to the server arrivals and departures to the i^{th} station in our earlier discussion of polling systems. For our queue inferencing algorithms to be applicable, the monitor must be able to determine the times of the j^{th} token arrival and release events at host i, as well as the start and end of transmission events for each packet transmitted by host i. In addition, the monitor must be able to determine whether or not the target host released the token before its turn had expired.

[3]There is no *requirement* for the host to continue with its next packet this quickly, so many host interfaces introduce longer delays. These minimum interpacket delays can be determined by direct observation of the traffic generated by the target host.

Let γ_i denote the propagation delay along the ring from host i to the monitor. In general, γ_i will be unknown to the monitor, but we now show that it is not required either. Clearly, since a token ring is not store-and-forward, the monitor will see the beginning and end of each packet from host i exactly γ_i after the target host began and ended its transmission.

Also, let the hosts between the monitor and host i be called "upstream" and the hosts between host i and the monitor be called "downstream" hosts. Clearly, the idle token visits the monitor exactly once during the interval $[D_i(j-1),\ A_i(j))$. Furthermore, the idle token visits each of the "upstream" hosts exactly once between its visit to the monitor and $A_i(j)$, and *every packet transmitted by any of those hosts, together with the preceding busy token, must pass by the monitor before it is removed by the sending host.* Thus, the network monitor can determine $A_i(j)+\gamma_i$ as the next time it receives the end of an idle token, or the end of a busy token followed by a packet whose sender is either i or a "downstream" host. Similarly, the network monitor can determine $D_i(j)+\gamma_i$ as the next time it receives the end of an idle token, or the end of a busy token followed by a packet whose sender is a "downstream" host. Notice that all of these events (token arrivals/departures and packet transmission start/end) are delayed by exactly the same amount, i.e., γ_i, which is no different than connecting the monitor directly to the output of host i — but with its clock shifted by γ_i.

Using the results from Section 4, it is clear that the only missing element for applying our queue inferencing algorithms to token ring networks is determining whether the token left the target host "on time." Since this determination is trivial for k-limited service, we will now consider the FDDI protocol. In this case, the monitor must be able to calculate the value of the Token Holding Timer, $THT_i(j)$ at node i for the j^{th} visit of the token. It is easy to see that if the monitor knows the value of the Target Token Rotation Time (TTRT) in the network (which can be found by monitoring the control traffic on the ring), then

$$THT_i(j) = TTRT - [A_i(j-1) + A_i(j)] \qquad (15)$$

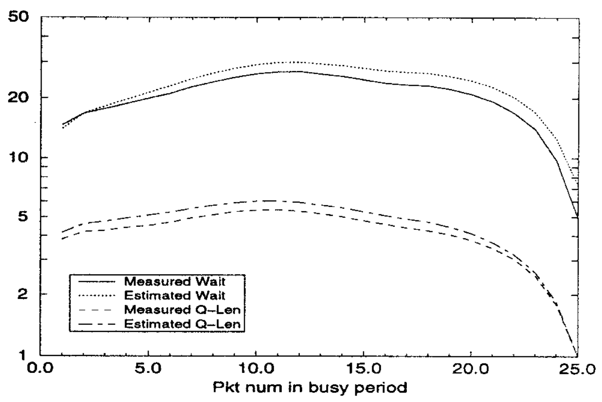


Figure 1: Application of Estimation Algorithms to 25-customer busy periods in an FDDI Ring with Erlang-2 distributed interarrival times

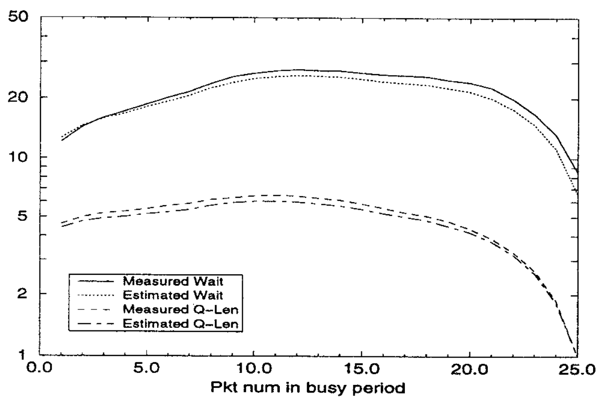


Figure 2: Application of Estimation Algorithms to 25-customer busy periods in an FDDI Ring with Hyper-exponentially distributed interarrival times

6 Results and Discussion

We simulated an FDDI network with 5 nodes. Time is normalized so that the average packet transmission time is unity. We also assumed a walk time around the ring of unity, and chose a target token rotation time of 10. Packet lengths were exponentially distributed, and the average arrival rate was fixed at 0.75.

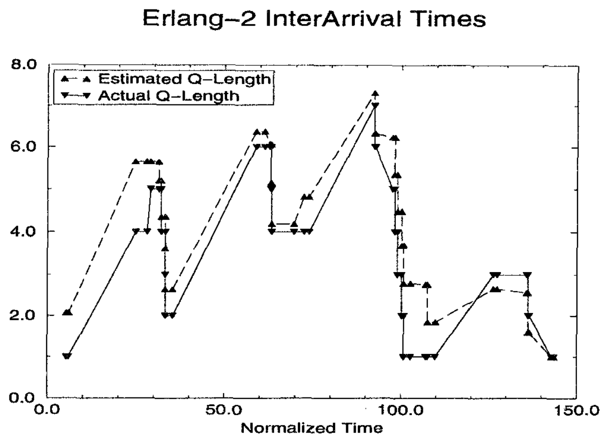
In Figs. 1 and 2, we show a comparison between the measured and estimated values of the waiting times for each packet and the queue lengths at each service completion time for 25-packet busy periods under Erlang-2, and hyperexponential interarrival time distributions with respective squared coefficients of variation of 0.5, and 2.3. Experiments were also run with an exponential interarrival time distribution and/or 15-packet busy periods, but are not shown here to conserve space. The results shown are averages obtained using the following methodology.

For each interarrival time distribution, the FDDI simulator was run for a very long time until 2000 busy periods of each target length were identified. For each of these busy periods, the measured queue length and waiting time values at each service completion time were recorded, and the associated monitor data was fed into the queue inferencing algorithms to produce the corresponding estimated values. The 2000 samples of each measured or estimated value were then averaged to produce point estimates, which we plotted in the Figures.

The measured and estimated values were in complete agreement in the case where the simulator used an exponential interarrival time distribution, just like the analysis. This accuracy is not surprising, since these are averages over many busy periods rather than the data obtained from applying the algorithm to a single busy period. The agreement in Figs. 1 and 2 is almost as good, but there is some systematic bias in our estimations because of the approximation error in replacing Erlang-2 and hyperexponential arrival processes, respectively, in the simulation by exponential arrivals in the estimation procedure. As expected, the estimates in Fig. 1 are slightly pessimistic, because our analysis was derived for an arrival process with higher variance. Conversely, the estimates in Fig. 2 show a slight optimistic bias, because our analysis assumed an arrival process with lower variance. Nevertheless, the relative error in these estimates was remarkably small in all the cases we tested.

Figure 3 compares the sequence of estimated and actual queue lengths at each service initiation event during a single monitored busy period in which the target host transmitted 25 packets and the interarrival time distribution is either Erlang-2 or hyperexponential. The beginning and end of each packet transmission is marked by a closely-spaced pair of triangle symbols (recall that the average transmission time is unity), separated by longer gaps where the token was elsewhere around the ring. It is evident that the transmission opportunities by the target host were quite erratic, including several large gaps followed by burst of many consecutive packet transmissions.

In both cases, the estimation procedure is doing a remarkable job of tracking the transient fluctuations in the queue length. Under the Erlang-2 arrival pro-



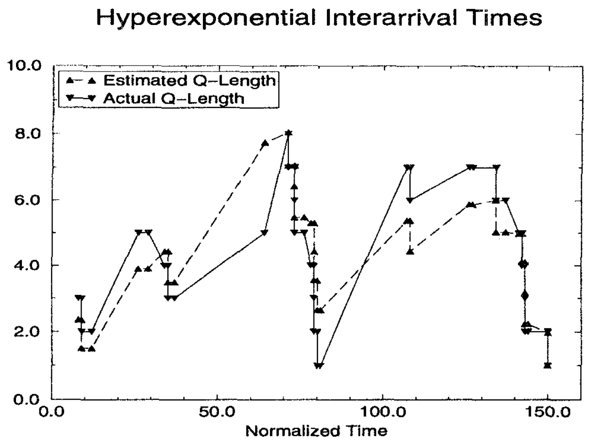


Figure 3: Transient Response of the Queue Length Estimation Algorithm over a Single Busy Period

cess, we can again see a slight systematic bias towards overestimation, although the error in the estimate is generally quite small (less than one customer). Larger estimation errors occur under the hyperexponential arrival process, because it is harder to predict the arrival times from a more variable process. Because of these larger estimation errors, it is more difficult to note the presence of a systematic bias in the results, although it is obvious that the estimate is a smoother function than the actual queue length.

7 Conclusion

We believe that these queue inferencing algorithms can be used to construct a new and more powerful class of network monitoring tools. Such tools would allow the network administrator to identify transient performance bottlenecks as well as hardware failures, protocol errors and long term traffic patterns. That is, if a specific user is experiencing low quality of service from the network, then using these monitoring tools will allow the exact location of the problem to be identified.

The accuracy of the estimates is quite remarkable, considering how little information they have available about the activity that is internal to a monitored device. Because the algorithms work directly with the measured packet length and interpacket vacation sequences, they do not require any statistical assumptions about the distribution of packet sizes, including independence, stationarity, etc. However, the sensitivity of the estimates to the variance of the arrival process suggests that further refinement of these algorithms may be needed to deal with the bursty traffic patterns that are characteristic of real networks.

References

[1] R. C. Larson, "The Queue Inference Engine; Deducing queue statistics from transactional data", *Management Science*, vol 36, 1990, pp 586-601.

[2] D. Bertsimas and L. D. Servi, "Deducing Queueing From Transactional Data : The Queue Inference Engine Revisited", *Operations Research*, vol 40 Supp No. 2, May-June 1992, pp S217-S228.

[3] D. A. Daley and L. D. Servi, "Exploiting Markov Chains to Infer Queue Length From Transactional Data", *Journal of Applied Probability*, vol 29, 1992, pp 713-732.

[4] D. A. Daley and L. D. Servi, "A two point Markov Chain Boundary Value Problem", *Advances in Applied Probability*, vol 25, 1993, pp 607-630.

[5] M.L. Molle, "A New Binary Logarithmic Arbitration Method for Ethernet", Technical Report CSRI-298, University of Toronto, 1994.

[6] H. Takagi, "Analysis of Polling Systems", MIT press, 1986.

[7] H. Takagi, "Application of Polling Models to Computer Networks", *Computer Networks and ISDN Systems*, 1991, pp 193-211.

[8] D.P. Gaver, "A Waiting Line with Interrupted Service including Priorities", *Journal of the Royal Statistical Society B*, vol B24, 1962, pp 73-90.