# Soft Real-Time Guarantee for Control Applications Using Both Measurement and Analytical Techniques

**Baek-Young Choi** choiby@umkc.edu
*University of Missouri,*
*Kansas City, MO, 64110, USA*

**Sejun Song** sjsong@tamu.edu
*Texas A&M University,*
*College Station, TX, 77843, USA*

**Abstract**

In this paper, we present a probabilistic admission control algorithm over switched Ethernet to support *soft real-time* control applications with *heterogeneous periodic* flows. Our approach is purely end host based, and it enables real-time *application-to-application* QoS management over switched Ethernet without sophisticated packet scheduling or resource reservation mechanisms in Ethernet switches or middleware on end hosts. In designing the probabilistic admission control algorithm, we employ both measurement and analytical techniques. In particular, we provide a new and efficient method to identify and estimate the queueing delay probability inside Ethernet switches for heterogeneous periodic flows with variable message sizes and periods. We implemented the probabilistic admission control algorithm on the Windows operating system, and validated its efficacy through extensive experiments.

**Keywords:** soft real-time, probabilistic admission control, periodic flows.

## 1. INTRODUCTION

A typical mission-critical real-time control system consists of sensors, actuators, controllers, data-intensive devices, and instrumentation devices. It works with the combination of periodic closed control loops. Controllers receive inputs from various sensors and data-intensive devices and perform control logic that determines how actuators and instrumentation devices should be operated. Each control loop has its Quality-of-Service (QoS) requirements, in particular, timing requirements including delay, jitter, and loss for a certain size message transmission. Traditionally, the mission-critical real-time control systems are designed to support the *hard* guarantee of their QoS requirements. However, for many practical real-time control systems found in industrial process controls, real-time signal processing, and telecommunications, a *hard* guarantee could be considered overly stringent by requiring excessive system resources. The *statistical* or *soft* guarantee accepts the performance as long as the violation probability of QoS requirements is below the pre-specified level. It is desirable to be designed and utilized for many real-time control applications to allow an efficient resource usage. In this paper, we focus on real-time control systems with soft QoS requirements, more specifically, soft delay guarantees.

Various real-time control networks have been developed with proprietary hardware and protocol solutions to provide deterministic controls for the specific applications. However, recent trends in the mission-critical control system industry replace proprietary networks with commercial-off-the-

shelf (COTS) or open networks so as to reduce product development cycle time and cost as well as to achieve system interoperability. Moreover, high-bandwidth sensors (e.g., infrared video, acoustic, and color sensors) are becoming increasingly common in control networks. Due to its ubiquity, simplicity and low cost, Ethernet has become a de facto choice for developing open mission-critical network strategies [2], [20], [13]. However, as the traditional Ethernet is a shared communication network using the CSMA/CD (Carrier Sense Media Access/Collision Detect) MAC protocol, packets to be transmitted may be held back arbitrarily long due to the random exponential back-off algorithm used to avoid collision. In other words, packet transmission delay over the traditional Ethernet is unpredictable, which makes it difficult to build a real-time control network over the traditional Ethernet. A better way to build real-time control systems over a local area network (LAN) is to use the switched Ethernet technology. Because of its switching capacity, an Ethernet switch not only enables fast packet transmission, but also reduces the chance of packet collision. Furthermore, with its internal buffer, a switched Ethernet can temporarily buffer packets that are competing for the same output port, further reducing the chance of an inside switch packet collision. However, a shared buffer inside of Ethernet switches introduces variable queueing delays of the packet transmission [16]. To build real-time control systems over switched Ethernet, it needs additional mechanisms to orchestrate competing resources according to the QoS requirements. However, QoS aware reservations or admission mechanisms are not available at Ethernet switches, unlike IP based solutions on more sophisticated IP routers.

In this paper, we propose a novel *probabilistic* admission control approach on the switched Ethernet environments to enable soft real-time guarantees for the mission-critical control applications. The proposed approach is designed for the typical control applications, which have *various periodic packet flows with different packet sizes*. It performs an admission control on the end control host, that starts a periodic real-time control flow, to determine whether a new connection (or a *flow*) between two control end hosts can be established. This admissibility check ensures that the *application-to-application* delay requirements of the new and existing flows can be guaranteed within a pre-specified probabilistic delay bound. The probabilistic admission control algorithm on the end control host works as follows. It first measures the *baseline* delay distribution when there are no competing flows in the switched Ethernet control network. This initial measurement of the baseline delay distribution captures the “fixed” delay components, including propagation delay, packet transmission time, and operating system overhead. According to the baseline delay distribution and the information regarding flow requests, it then estimates the probability of a queueing delay at the Ethernet switches when there are multiple competing flows for the same output port. It provides an efficient method to estimate the probability of a queueing delay for heterogeneous periodic flows in order to obtain a probabilistic delay bound.

The proposed approach is a pure end-host based solution that does not require any software or hardware modification to the Ethernet switches. For easy deployment, it is designed in the application layer that does not require any sophisticated middleware installation in the end host Operating System kernel. We implement the admission control algorithm on the COTS OS based end-hosts and conduct extensive experiments over the switched Ethernet environments. Through the experiments, we validate the effectiveness of the proposed probabilistic admission control approach.

The remainder of the paper is organized as follows. We first discuss the related work on real-time scheduling and admission control in a LAN control network environment in Section 2. In Section 3, we describe the problem setting, the queueing analysis and the proposed admission control algorithm in detail. The software design and implementation is presented in Section 4. In Section 5, we describe the experiment design and results. The paper is concluded with a summary of the work and future research directions in Section 6.

## 2. RELATED WORK

In the area of real-time research, a lot of efforts have been made to provide hard deadline guarantees over Ethernet with the expense of resource utilization and average performance. Most of the earlier work [21], [17], [7] modified the Ethernet MAC sub-layer to achieve a bounded channel access time. These proprietary approaches are quite costly compared to using the well-established and widely-used current Ethernet standard. Both [24], [6], [15] proposed a virtual non-collision token ring implementation over the collision-based Ethernet. Since the token management protocol is executed by the higher-layer (OS kernel of the hosts) rather than the MAC, the approach does not need to modify the network hardware architecture. The major shortcomings of this approach are the overhead of heavy token management including the token relay among the hosts and the restoration of the lost token and the performance limitation due to the overly conservative network usage. Recent studies try to achieve the hard real-time guarantee without modifying network hardware architecture. [12] proposed a traffic shaping software on the Ethernet switch to achieve hard guarantees with bounded delays and reserved bandwidths. The proposed solution in [8] designed on standard Ethernet switches with Layer 2 QoS/CoS protocol for traffic prioritization (IEEE 802.1p). It requires a separate queue for each priority class on the switch, but it cannot be smoothly deployed with currently widespread Ethernet switches that have a common buffer to share all priority classes. It may lead to the miss of QoS guarantees for high priority packets; for example, if the common buffer is already packed by the lower priority packets. [9] tried to resolve the problem of the shared buffer by using additional traffic shaping mechanisms available in other higher layer network elements such as routers.

While there is a lot of research on designing, validating, and facilitating traditional hard real-time systems, only few such techniques exist on soft real-time systems in spite of the recent proliferation of soft real-time applications. The existing soft real-time research mostly is focused on schedulability analysis techniques. Probabilistic Time Demand Analysis (PTDA) [23] and Statistical Rate Monotonic Scheduling (SRMS) [3] are the algorithms about the statistical behavior of periodic tasks to facilitate better design of soft real-time systems.

PTDA attempts to provide a lower bound of the missing deadline probability that is determined by the time supply that equals or exceeds the time demand at the deadline of the task. The time demand is computed by convolving the probability density functions of the execution times. It assumes that the relative deadline of all tasks are less than or equal to their period. SRMS attempts to schedule tasks with highly variable execution times in such a way that the portion of the processor time allocated to each task is met on the average. Variable execution times are smoothed by aggregating the executions of several jobs in a task and allocating an execution time budget for the aggregate. A job is released only if its task contains a sufficient budget to complete it in time and if higher priority jobs will not prevent its timely completion.

In a statistical real-time guarantee work [10], they analyzed the Ethernet MAC protocol using a semi-Markov process model and derived a network-wide input limit for achieving a target transmission success ratio. The network-wide input limit is kept by enforcing each component station to control its instantaneous traffic arrival rate under its station-wide input limit. To this end, they implemented a traffic smoothing middleware between the transport layer and the Ethernet data link layer at each station. The middleware keeps the traffic arrival rate under the station-wide input limit by smoothing a busty packet stream. An enhancement on the traffic smoother is made by [5]. They used the overall throughput in tandem with the number of collisions as network load indicators to feed into their fuzzy traffic smoother to give the flexibility on the sporadic traffic process. Unlike our approach based on switched Ethernet, these studies focused on designing a traffic smoother using an ordinary shared Ethernet hub.

Queueing system studies on multiplexing periodic flows have been limited to flows with the same packet size [14] ($N * D/D/1$, $\sum D_i /D/1$ queue) in the past. A relevant work in the switch can be found in Raha et al.'s work on real-time ATM [19], [18]. Their research focuses on the worst case queuing analysis. They devised Gamma functions to represent the flows' worst-case bandwidth requirements across different time scales. These gamma functions can be used to compute the worst-case queueing delay in an ATM switch buffer. Since ATM cells are of fixed size, the switch's processing rate is constant. Therefore, the worst-case delay on each node can be computed. Since the exact gamma functions are too complex to compute, they designed three approximation methods. However, their analysis based on gamma functions only works for fixed packet size such as ATM cells. It is therefore not applicable for switched Ethernet with variable packet size.

## 3. PROBABILISTIC ADMISSION CONTROL APPROACH

### 3.1 Problem Setting

Although the proposed probabilistic admission control approach can be implemented in either a distributed or a centralized fashion, figure 1 illustrates a distributed approach based control network environment. The control network environment has several control application hosts connected by a typical Ethernet switch with simple FIFO port buffers. The probabilistic admission control software is distributed to each control host running on the application layer. Each control host also maintains the control flow information database that is synchronized over the entire control network. *Flow* is used to refer to a connection between two control applications to transmit periodic messages. When a flow is requested on a control host, the host performs an *admissibility test* to check whether its delay requirement can be satisfied within the pre-specified probabilistic delay bound. If the flow request is admitted, a broadcast message of flow addition is sent to other control hosts in order to update their flow information database. When a flow is terminated, a broadcast message of flow deletion is sent to remove the flow from the distributed hosts' flow information database. In general, since a distributed approach has multiple admission control points (on each control host) over the control system network, the concurrent flow requests on the distributed control hosts can potentially lead to unexpected QoS violations. To deal with this issue, there is a proposal [11] to utilize a safety margin in order to absorb the potential impact of concurrency.

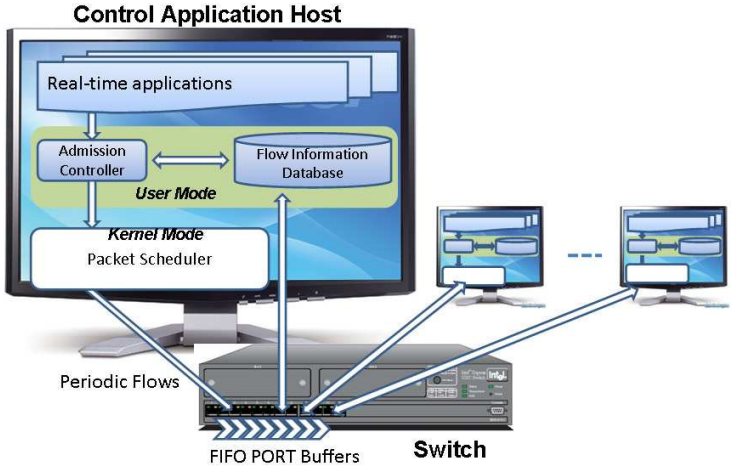


**FIGURE 1:** Network System Architecture.

The admission control algorithm is designed to perform an efficient on-line admissibility test by simplifying the calculation formula of the delay probability estimation. We consider the periodic flow characteristics with various periods and different message sizes. As summarized in Table I, a flow with its QoS requirements is defined by $S_i$ ($L_i$, $P_i$, $D_i$, $DP_i$) where $L_i$ is the (maximum) message size (bytes), $P_i$ is the period of the flow (ms), $D_i$ is the maximum acceptable delay (μs), and $DP_i$ is the bound on the probability that the actual message delay $d$ exceeds $D_i$ (i.e., $P\,r(d \geq D_i) \leq DP_i$). We assume that all flows are independent of each other.

| Notation | Explanation |
|---|---|
| $S_i$ ($L_i$, $P_i$, $D_i$, $DP_i$) | a flow with: |
| $L_i$ | message size (bytes) |
| $P_i$ | period (ms) |
| $D_i$ | maximum acceptable delay (μs) |
| $DP_i$ | minimum acceptable probability that the actual delay is less than $D_i$ ( i.e., $Pr(d \leq D_i) \geq DP_i$ ) |

**TABLE 1:** Flow Specification.

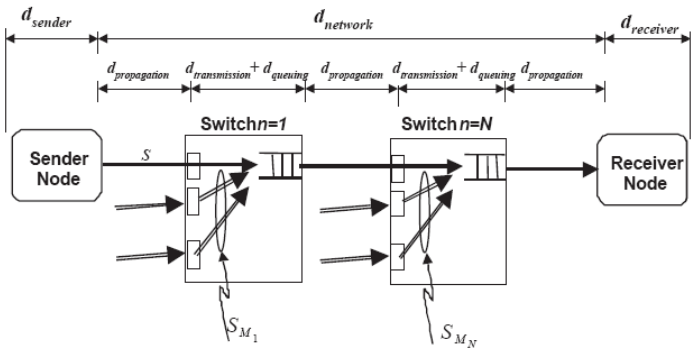


**FIGURE 2:** Network Delay Model.

As illustrated in figure 2, the performance metric (i.e., the message delay $d$) is the *application-to-application* delay that consists of the processing delays on both hosts and a network transmission delay as shown below:

$$d = d_{sender} + d_{network} + d_{receiver}. \quad (1)$$

The delay in network $d_{network}$ can be further decomposed into three components:

$$d_{network} = d_{\text{propagation}} + d_{\text{transmission}} + d_{\text{queueing}}, \quad (2)$$

where $d_{\text{propagation}}$ is the propagation delay of a message across the network, $d_{\text{transmission}}$ is the transmission time of a message, and $d_{\text{queueing}}$ is the queueing delay experienced by a message at Ethernet switches. The main variability in application-to-application delay is contributed by $d_{\text{queueing}}$ that varies with the network load and the number of competing flows in the network. The other delay components are independent of any competing flows in the network system. Hence, we group these delay components together and denote it as the baseline delay, $d_{\text{baseline}}$, i.e.,

$$d_{\text{baseline}} = d_{sender} + d_{receiver} + d_{\text{propagation}} + d_{\text{transmission}}. \quad (3)$$

This baseline delay of a flow can be estimated by directly measuring application-to-application delay with no competing flows in switches along the transmission path. In general, the measured baseline delay of a flow demonstrates certain distribution patterns, rather than a single delay point, due to the inherent delay variations introduced by the host OS and possible measurement

errors. In figure 4, we illustrate a hypothetical probability density function, *f(x)*, of the baseline delay $d_{baseline}$.

Given the measured baseline delay distribution, in the remainder of this section we devise an efficient method to estimate the queueing delay probability when there are competing flows at a switch. In section 3.2, we consider the simple case of two competing flows. The general case with multiple competing flows is then analyzed in section 3.3. Based on the analysis, we devise an efficient probabilistic admission control algorithm that is presented in section 3.4.

### 3.2 Simple Case: Two Competing Flows

In this section we consider the simple case where there are only two flows competing for the same output port. Note that even when the network load is relatively light-loaded, packets may still be queued at a switch, due to the coincidental arrival of packets forwarded to the same output port. In the following, we analyze the probability that packets will be queued at a switch under the assumption that there are only two competing two flows.

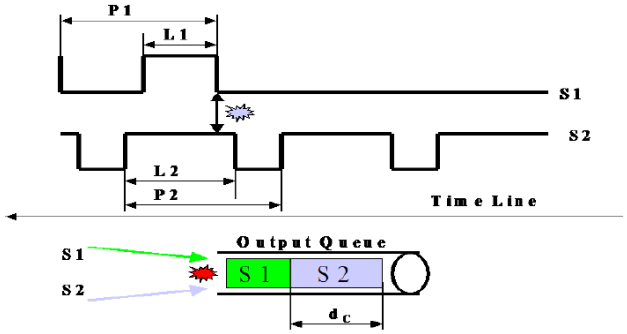


**FIGURE 3:** An Example of Competing Flows.

Suppose two flows, flows $S_1$ ($L_1$, $P_1$, $D_1$, $DP_1$) and $S_2$ ($L_2$, $P_2$, $D_2$, $DP_2$) send periodic transmissions through the same switch output port and the queue is initially empty. As illustrated in figure 3, if a packet of $S_1$ arrives before an $S_2$ packet is transmitted, $S_1$'s packet would be delayed by $S_2$'s packet. We assume that only one $S_2$ packet is in the queue when an $S_1$ packet arrives. This assumption is reasonable in a practical system environment and we will discuss in detail later. The worst case queueing delay experienced by an $S_1$ packet will be the time it takes the switch to transmit a whole $S_2$ packet. Since it depends on the message size and the period of competing flow $S_2$, we can calculate the queueing probability p of the flow $S_1$ as follows:

$$p = \frac{T_2}{P_2} \qquad where\ T2 = \frac{L_2}{Switch_{capacity}} \tag{4}$$

We denote the corresponding queueing delay $T_2$ that is the transmission time of $L_2$, as $d_C$ for consistency with the next section.

A packet of $S_1$ will be delayed by $d_C$ with a probability of $p$. There will be no queueing delay for the packet with a probability of $1 - p$. Hence, the estimated delay for flow $S_1$ with the presence of competing flow $S_2$ is

$$d = (d_{baseline} + d_{queueing}) * p + d_{baseline} * (1 - p) \tag{5}$$

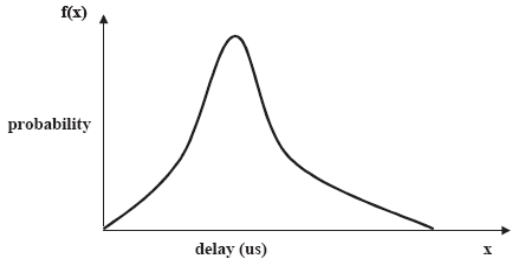


**FIGURE 4:** Probability Density Function for Baseline Delay.

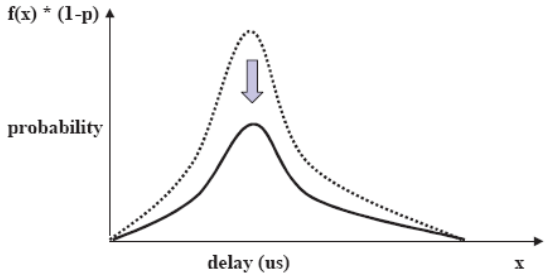


**FIGURE 5:** Partial Probability Density Function for No-Queuing Case.

As we will see in section 5, $d_{baseline}$, the measured delay of a certain flow without interference from other flows, demonstrates certain distribution patterns rather than a single delay point, due to the variation of the delay factors. In figure 4, we illustrate a hypothetical probability density function, $f(x)$, of the baseline distribution, $d_{baseline}$. The partial probability density function of the no-queuing case $f(x) * (1 - p)$ is illustrated in figure 5. The partial probability density function of the queuing case, $f(x - d_C) * p$ is equivalent to shifting $f(x)$ by $d_C$ and multiplying by $p$ as shown in figure 6. As illustrated in figure 7, the estimated delay distribution density function $f'(x)$ in the presence of the competing flow is derived by adding up partial delay distributions.

$$f'(x) = f(x - d_C) * p + f(x) * (1 - p) \quad (6)$$

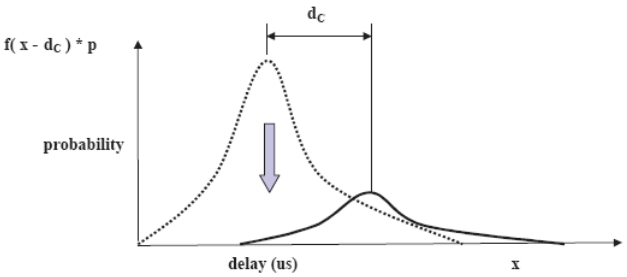


**FIGURE 6:** Partial Probability Density Function for Queuing Case.

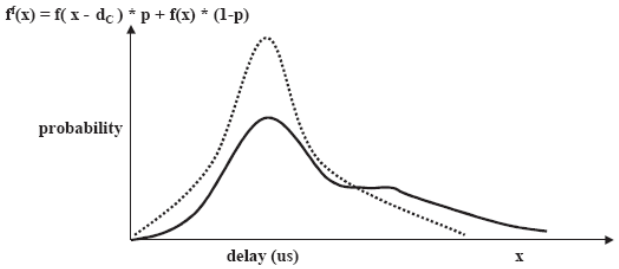


**FIGURE 7:** Probability Density Function for Output Delay.

In short, if we know the baseline distribution of a new flow and the characteristics (message size and period) of the existing competing flows in the switch, we can estimate the output delay distribution. Finally, the calculated output delay distribution is converted to the cumulative density function $F^f(x)$ for the admission decision. A flow can be admitted, if $F^f(D)$ is greater than DP. For example, in figure 8, the flow $S_1$ ($L_1$, $P_1$, $D_1$, 0.9) is rejected for the delay requirement $D_1$ but the flow $S_1$ ($L_1$, $P_1$, $D_1$, 0.8) is admitted.

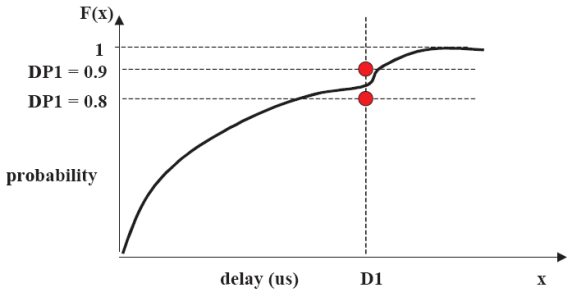


**FIGURE 8:** Cumulative Density Function for Admission Control.

### 3.3 The General Case: Multiple Competing Flows

In this section, we consider the generalized case where there are more than two competing flows at a switch. We first introduce a general method called the Benes˘ approach to express the queueing delay of a *G/G/1* queue. We then derive the formula for system multiplexing periodic flows with variable message sizes and variable periods using the Benes˘ approach.

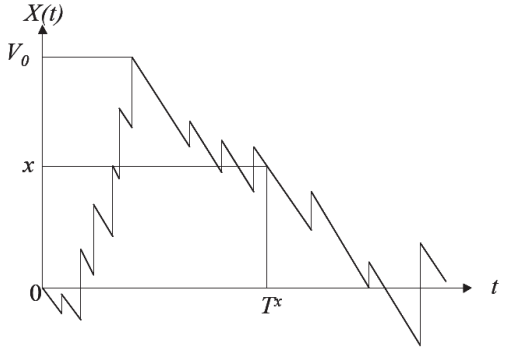


**FIGURE 9:** A realization of the process *X(t)* with the last exit time $T^x$.

*1) Virtual Waiting Time*: The virtual waiting time, also referred to as queueing delay in a system can be obtained by a theorem due to Benes˘ for *G/G/1* queue [4] that has general independent inter-arrival times and general service times. We review the approach here only to the extent that it is essential for our work. Consider a system where a constant rate server with unlimited buffer capacity is submitted work according to a random process. The server capacity is assumed to be 1 unit of work per unit of time and the system is assumed stationary, so that 0 represents an arbitrary time instant. Let *A(t), t ≥ 0*, denote the amount of work arriving to the system in the

interval [–t, 0), and let $V_t$ be the amount of work still in the system at time –*t*. Define $X(t) = A(t) – t$ to be the excess work arriving [–t, 0). Then $V_t$ is given by Reich's formula

$$V_t = \sup_{w \geq t} \{X(w) - X(t)\} \tag{7}$$

In particular,

$$V_0 = \sup_{t \geq 0} \{X(t)\} \tag{8}$$

Let $T^x$ denote the largest value of t such that $X(t) = x$ (see figure 9), then the following equivalence can be deducted:

$$\{V_0 \geq x\} \quad \Leftrightarrow \quad \{\exists \; unique \; T^x \; such \; that \; X(T^x) = x \; and \; X(w) < x \; for \; w > T^x\} \tag{9}$$

The complementary distribution function of $V_0$ can therefore be expressed by the generic Benes˘ principle

$$Pr\{V_0 > x\} = Pr\{T^x \in [0, \infty)\} \tag{10}$$

Employing the definition of $T^x$ from (9) we have

$$Pr\{V_0 > x\} = \int_{u=0}^{\infty} Pr\{u \leq T^x < u + du\}$$

$$= \int_{u=0}^{\infty} Pr\{X(u + du) < x \leq X(u) \tag{11}$$

$$and \; X(w) < x \; for \; w > u\} \tag{12}$$

Now, applying relation (7) at the point $t = u$ yields

$$\{V_0 = 0\} = \{X(w) \leq X(u) \; for \; w > u\} \tag{13}$$

Applying this to (11), it leads to the Benes˘ formula:

$$Pr\{V_0 > x\} = \int_{u>0} Pr\{X(u) \geq x > X(u + du) \; and \; V_u = 0\} \tag{14}$$

2) *Multiplexing periodic flows with variable message sizes and variable periods*: We address the problem of periodic flows with variable message sizes and variable periods. We refer to this system as $\sum D_i / \sum D_i / 1$ queue. We derive bounds for the queue length distribution of $\sum D_i / \sum D_i / 1$ queue by Benes˘ approach. Suppose *M* flows are sharing a switch port. Then there are $2^M$ numbers of flow combinations. We denote the set of all flows as $C_2^M$ and a combination of the flows that is a subset flow of $C_2^M$, as $C_m$. In a system fed by periodic flows, work arrives discontinuously. Then the probability in the right-hand side of (14) is concentrated on the values of *u* such that $x + u$ is an integer number of packet processing time. Let $d_{Cm}$ denote the sum of the delays caused by the packets of the flow set $C_m$.

$$d_{C_m} = \sum_{i \in C_m} T_i \tag{15}$$

We can thus replace the integral in (14) by a summation and give the virtual waiting time formula for $\sum D_i / \sum D_i / 1$ queue:

$$Pr\{V_0 > x\} = \sum_{d_{C_m} > 0} Pr\{A(d_{C_m} - x) = d_{C_m} \; and \; V_{d_{C_m} - x} = 0\} \tag{16}$$

$$= \sum_{d_{C_m} > 0} Pr\{A(d_{C_m} - x) = d_{C_m}\} \tag{17}$$

$$\cdot Pr\{V_{d_{C_m} - x} = 0 | A(d_{C_m} - x) = d_{C_m}\}$$

The first part of equation (17) is computed as follows:

$$\begin{aligned} Pr_{C_m} &= Pr\{A(d_{C_m} - x) = d_{C_m}\} \\ &= \prod_{i \in C_m} \frac{T_i}{P_i} \times \prod_{i \in (C_{2^M} - C_m)} (1 - \frac{T_i}{P_i}) \end{aligned} \quad (18)$$

$Pr_{Cm}$ is the queueing probability where a combination of packets from flow set $C_m$ are queued among the all active flows in the switch and the packets from the rest of the flows ($C_2^M - C_m$) are not queued. The actual corresponding queueing time is less than or equal to $d_{Cm}$, since the first packet in the queue may be already being served at the time of its arrival. The order of the packets does not affect its queueing time. The second part of equation (17) is replaced by $(1 - \rho_c)^+$ where $\rho_c$ is the conditional arrival intensity at time $d_{Cm} - x$ :

$$\rho_c = \sum_{i \in (C_{2^M} - C_m)} \frac{T_i}{P_i(1 - \frac{T_i}{P_i})} = \sum_{i \in (C_{2^M} - C_m)} \frac{T_i}{P_i - T_i} \quad (19)$$

In practice, this quantity is a very close one, since for a high capacity link that multiplexes a large number of flows, the system behaves like a multi-server system for which the empty queue probability is very much closer to 1 than $1 - \rho$. Omission of the condition $\{V_0 = 0\}$ in (14) yields an upper bound for $Pr\{ V_0 > x\}$. The approximation has been shown to be reasonably accurate [22]. This approximation is assumed at the two competing flow cases studied in the earlier section. The notations are summarized in Table 2.

| Notation | Explanation |
|---|---|
| $M$ | the number of existing flows sharing the switch port |
| $C_{2^M}$ | a set of all $M$ flows |
| $C_m$ | a subset flow of $C_{2^M}$ |
| $T_i$ | packet transmission time of flow $i$ (i.e., $L_i/Switch\_Capacity$) |
| $Pr_{C_m}$ | probability of queueing by a flow set $C_m$ |
| $d_{C_m}$ | delay caused by $C_m$ |

**TABLE 2:** Notations for Analysis.

The estimated distribution gives an approximation of a worst case delay distribution, since it counts the whole packet processing time of all the packets in the queue, even though the first packet in the queue is already being served. Queueing system studies on multiplexing periodic flows have been limited to flows with the same packet size [14] ($N * D/D/1$, $\sum D_i /D/1$ queue) in the past. Equation (17) together with Equations (18) and (19) provides a new method for estimating the queueing delay probability for periodic flows with variable message sizes and variable periods, and for obtaining a probabilistic delay bound.

### 3.4 The Admission Control Algorithm

The admission control algorithm running on hosts makes a decision if QoS requirements of flows on the host would be violated or not. In this section, we present the admission control algorithm using the queueing analysis described in the previous section. Since measured delay $d_{baseline}$ shows a distribution rather than a point, we cannot directly apply the complementary distribution $Pr\{V_0 > x\}$ to the admissibility test. Instead, the partial probability density function $f'(x)$ is obtained in equation (20), using its queueing probability $Pr_{Cm}$ and the corresponding delay $d_{Cm}$ for each queueing flow set $C_m$.

$$f'(x) = Pr_{Cm}\, f(x - d_{Cm}) \quad (20)$$

The partial probability density functions are summed up to be the output delay distribution $f^f(x)$. The output delay distribution is converted to the cumulative density function $F^f(D)$ to see if the deadline is satisfied with the required probability. The probability that the delay is less than the deadline $D$ is evaluated from the final cumulative density function $F^f$ as below (equation 22):

$$F^f(x) = \int_0^x \sum_{m \in C_{2^M}} Pr_{C_m} f(t - d_{C_m}) dt \tag{21}$$

$$F^f(D) > DP \tag{22}$$

The algorithm is particularly complex to evaluate, since the number of possible queueing combinations grows exponentially with the number of competing flows. For each possible competing flow combination, the algorithm needs computation process and memory usage to modify the partial probability density function (to be shifted and to be multiplied) and to convert it to the cumulative density function.

**Algorithm 1** Admissibility Test Algorithm

$F^f(x) \leftarrow$ baseline distribution of flow $s$
$F^f_{req} \leftarrow 0$ // initial probability for admission decision
**foreach** $m \in C_{2^M}$ // for each subset flows
    // Calculate the probability by a set $C_m$
    $Pr_{C_m} = \prod_{i \in C_m} \frac{T_i}{P_i} \times \prod_{i \in (C_{2^M} - C_m)} (1 - \frac{T_i}{P_i})$
    **if** $(Pr_{C_m} < \epsilon)$
      break // Speed up the decision
    $d_{C_m} = \sum_{i \in C_m} T_i$ // Calculate the caused by a set $C_m$
    $F^f_{req} += Pr_{C_m} \times F(D - d_{C_m})$
    **if** $(F^f_{req} > DP)$
      goto Decision //condision satisfied already
Decision:
    **if** (no reject signal received and $F^f_{req} > DP$)
      s is admitted
    **else**
      **if** $s$ is on this host
        $s$ is rejected
      **else**
        send reject control message

To make this approach practical, we made a couple of improvements to the algorithm. First, we observed that the packet queuing probability, due to a large number of other competing flows is very small when the stable network condition is satisfied. In most of the experiment settings, the packet queuing probability due to more than five competing flows was less than $10^{-10}$. Therefore, the computation of those combinations can be waived. More importantly, we found a derivation that enables us to make the admission decision without handling partial probability density function modifications and the cumulative density function conversion. In the following derivation (equation (23)), the admission decision is made by using the cumulative density function of the baseline distribution. Only one point (D – $d_C$) of the baseline distribution needs to be evaluated for each modification of the competing flow combinations.

$$F^f(D) = \int_0^D \sum_{m \in C_{2M}} Pr_{C_m} f(t - d_{C_m}) dt$$

$$= \sum_{m \in C_{2M}} Pr_{C_m} \int_0^D f(t - d_{C_m}) dt$$

$$= \sum_{m \in C_{2M}} Pr_{C_m} F(D - d_{C_m}) \quad (23)$$

For a requested flow to be admitted, the same admission decision process should also be made by all hosts that have existing flows competing for the same switch output port.

## 4. SOFTWARE DESIGN AND IMPLEMENTATION

We have implemented the probabilistic admission control software on the Windows operating system. As shown in figure 10, the proposed admission control software is implemented in the application layer and the packet classifier and the token bucket regulation-based packet controller in the Windows operating system kernel are used to maintain the flow period.

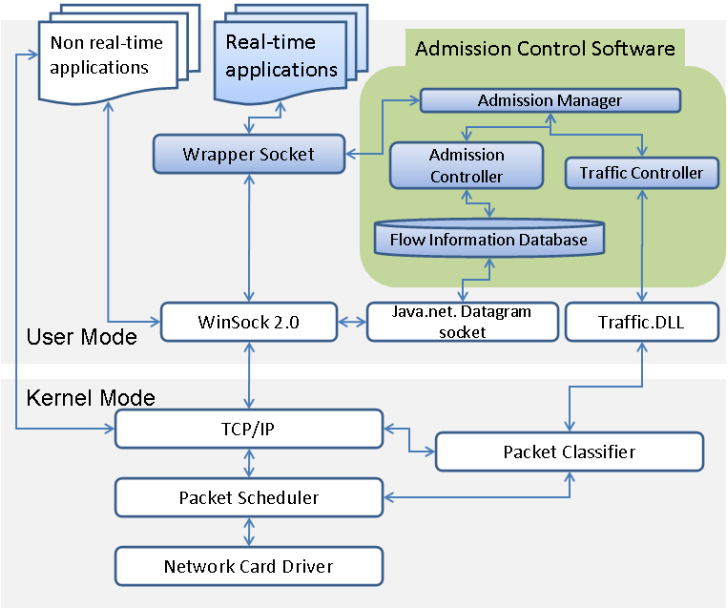


**FIGURE 10:** Windows Based soft real-time system Architecture.

The admission control software comprises the admission manager, the admission controller, the traffic controller, and the flow information database. The admission manager provides a registration and deregistration method to the traffic controller as well as an interface to the admission controller. If a new flow request is admitted, the admission manager registers the flow to the traffic controller. The traffic controller is responsible to relay flows to the packet scheduler in the kernel through the traffic.DLL calls. The flow informs the packet control method to be used for the packet scheduler. It also creates a filter as specified by the specification to instruct the packet classifier to filter the list of packets. The flows are controlled according to the flow specification described by the admission manager. The admission controller performs admissibility test for a new flow request based upon the flow specification and the existing flow information from the flow information database. If the flow request is admissible, the admission controller creates a flow and adds the new flow to the flow information database. The flow information database maintains a consistent image of the network topology and the existing flow information of the entire network control system. It also interfaces to the broadcast socket to send a broadcast message of flow

addition to other control hosts in order to update their flow information database. The admission control software is used by the real-time applications via Wrapper Socket API calls that are the Java-based wrapper APIs implemented above Winsock 2. The Wrapper Sockets pass the flow specification to the admission manager along with the source and destination IP addresses, destination port, and QoS enable indication. It relays the application flows to the Winsock 2 according to the admission decision.

## 5. EXPERIM ENTS AND RESULTS

To validate the efficiency of the proposed probabilistic admission control approach, we have conducted extensive experiments on the real networks. In this section, we present the experimental settings design and results. As illustrated in figure 11, 9 PCs are connected to a 12 port Intel express 520T fast Ethernet switch. A test flow generator generates a test message flow to a test flow receiver. Other 7 PCs are used to inject competing traffic flows. Competing flows are generated by seven other hosts and ten flows are generated per host. Competing flows are sent to the same receiving host as the test flow, ensuring that all competing flows use the same output port of a switch. A combined time delay of the hosts, network, and switch buffer is measured using time-stamps derived from the TrueTime™ [1] clock synchronization tool that provides one microsecond time resolution. To capture variability of flows both in terms of message size and period, we utilize an extensive set of experimental parameters. The experimental parameters, such as system environment and flow specifications, are summarized in Table 3. Although the algorithm can be applied to the flows with various periods with different message sizes, we have conducted the experiments with the same message size and period to illustrate the behavior of the delay as a function of message size and period. For the experimental parameters, we choose the message size less than the maximum packet fragmentation size (1500 bytes) in order not to introduce additional unnecessary complexity. We also keep the period greater than 10 milliseconds due to host delay stability issues (the flow control resolution of Windows). In a practical control system, most control message sizes are less than the maximum packet fragmentation size and the periodicity constraints are greater than tens of milliseconds [2].

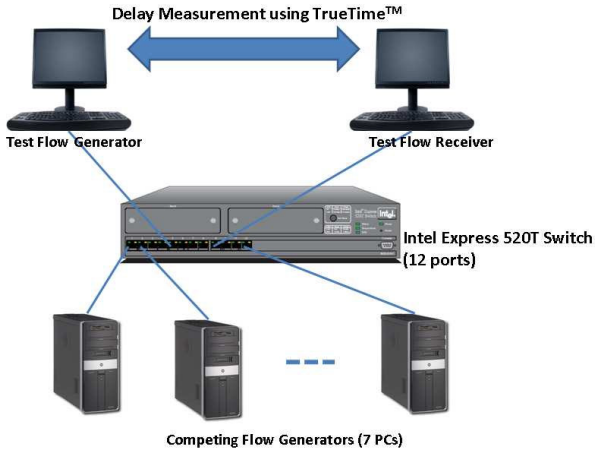


**FIGURE 11:** Testbed Configuration.

| Parameters | Values |
|---|---|
| System Environment | |
| Number of Switches between hosts | 1 |
| Switch Capacity (Mbps) | 100 |
| Baseline Flow | |
| Message size (bytes) | 50, 500, 1400 |
| Message period (ms) | 50, 200, 500 |
| Competing Flows | |
| Number of competing hosts | 7 |
| Number of flows per competing host | 10 |
| Message size (bytes) | 50, 500, 1400 |
| Message period (ms) | 50, 200, 500 |
| Test Flow QoS Required | |
| Acceptable Delay ($\mu s$) | 700, 800, 900 |
| Min Prob Actual Delay $<$ Acceptable Delay | 0.90 $\sim$ 0.99 |
| Windows Flow Control Mechanism | |
| Max Packet Size (bytes) | 1526 (for Ethernet) |
| Token Rate Token | Based on period |
| Bucket Size Peak | Based on message size |
| Bandwidth | Based on period and message size |

**TABLE 3:** Experiment Parameters.

### 5.1 Baseline and Competing Flow Experiments

We first generate the baseline delay probability distributions that characterize a single sender/receiver host pair without any other competing flows. This baseline distribution is used to estimate the theoretical distribution with competing flows. We then conduct experiments to measure delay distributions with competing flows to validate the theoretical distribution estimations. Each experiment is performed for 30 minutes with 5 minute warm-up interval. The delay distributions captured on the receiver are presented as PDF (Probability Density Function) and CDF (Cumulative Density Function).

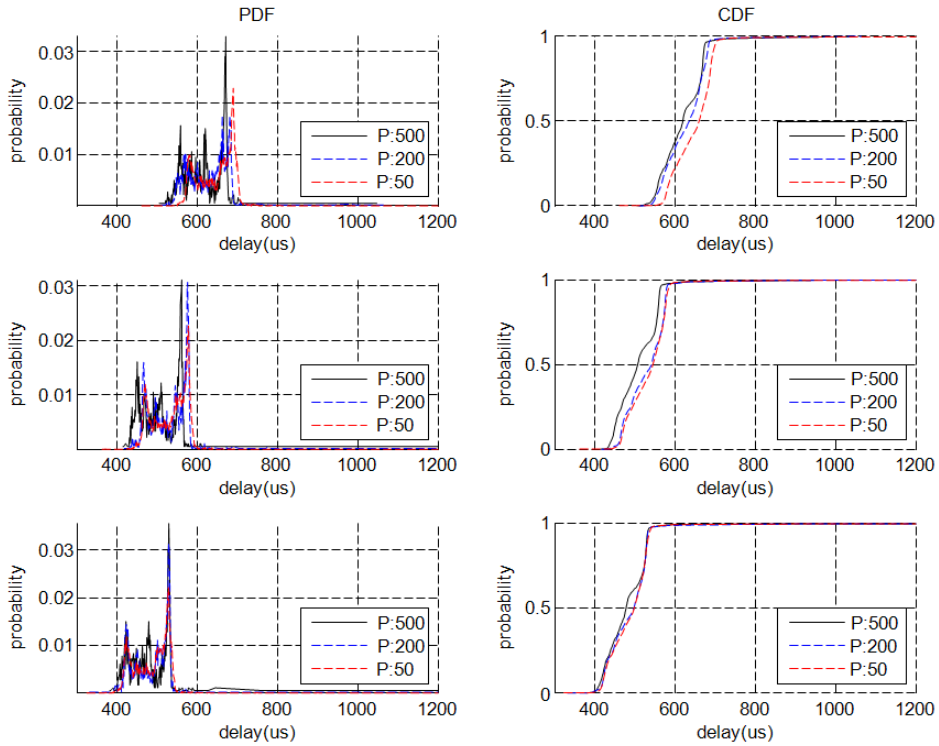


**FIGURE 12:** Baseline Distributions: PDFs and CDFs: Fixed Message Size (1400/500/50B from the top row), Variable Periods.

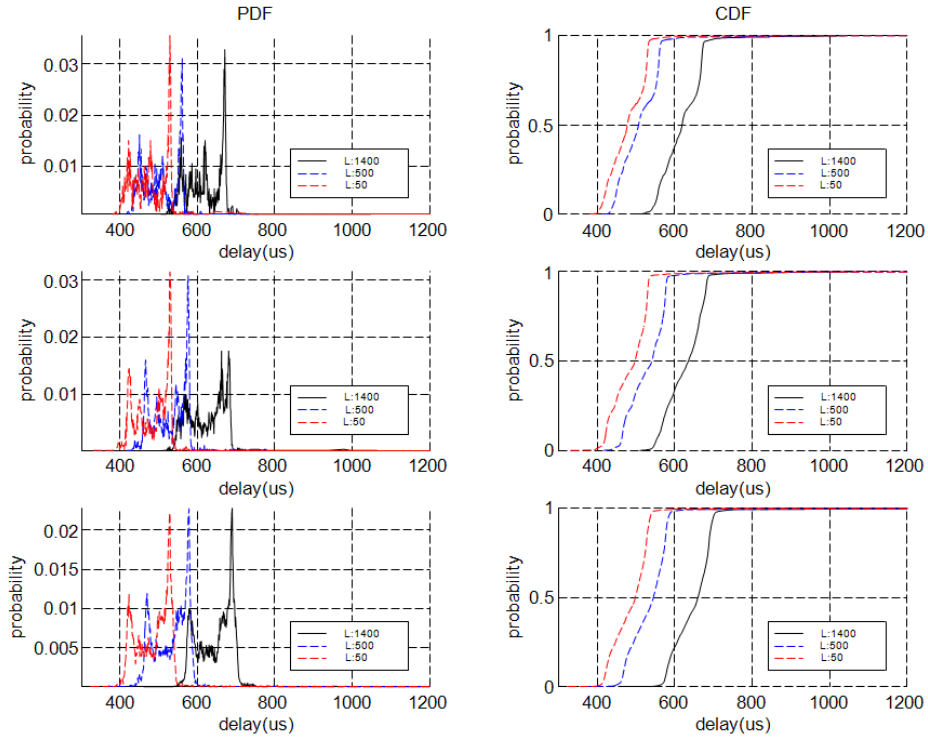


**FIGURE 13:** Baseline Distributions: PDFs and CDFs: Fixed Period (500/200/50 ms from the top row), Variable Message Sizes.

Figures 12 and 13 present the distribution of baseline measurements. In Figure 12, PDFs and CDFs are plotted for message sizes of 1400, 500, and 50 bytes, respectively from the first row. Each plot is compared with variable periods of 500, 200, and 50 ms. The PDF results illustrate that there are multiple modes on the distribution. These distributions look different from any well-known distributions that can be analytically tractable. Therefore, it seems infeasible to model this distribution parametrically. The distributions are, however, quite consistent with the entire experiment and are stable enough to use them as the basis of the estimated distributions with computing flows. Figure 13 shows the PDF and CDF results of various packet sizes (1400, 500, and 50 bytes) with a fixed message period per plot. Each plot illustrates message periods of 500, 200, and 50 ms. The CDF results show that the larger message size has more processing delay on the host and switch due to the longer transmission time that causes larger application-to-application delays. An observation from the experiment confirms that the CDF distribution shapes are consistent with the same period across the message sizes. Hence, if we have one baseline distribution for a flow, we can use the distribution for the same message size with different periods and we may further estimate the baseline distribution of the different message sizes without measurements. Using the same parameters as the baseline configuration for the test flow, we ran simultaneous competing flows to obtain a delay distribution that accounts for message queueing in the switch. The measured distribution is used to compare with the estimated distribution to assess its accuracy.

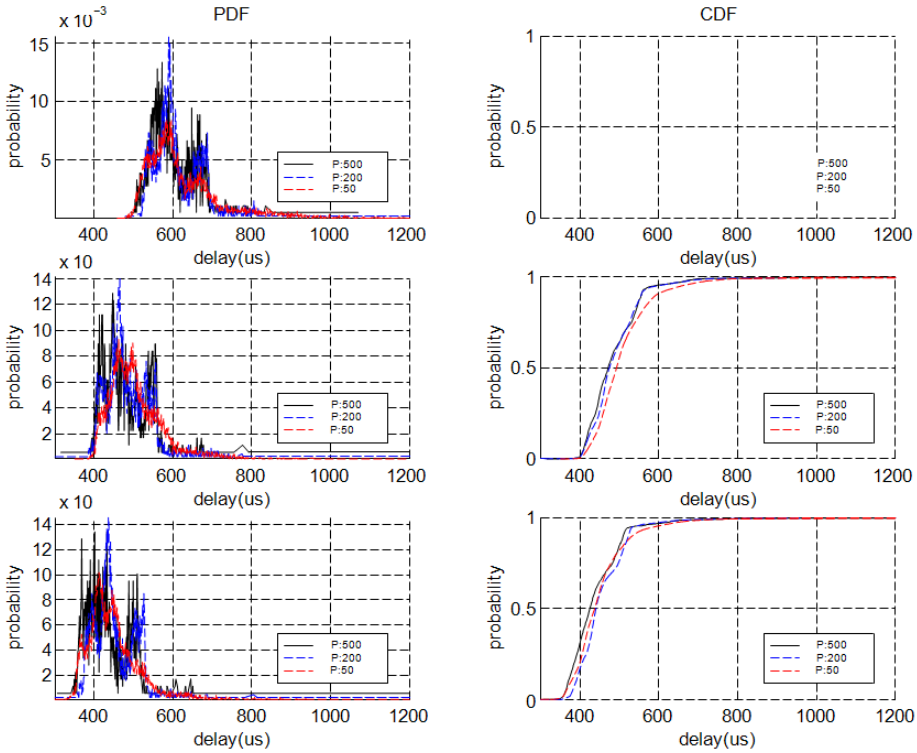


**FIGURE 14:** Competing Flows: PDFs and CDFs-Fixed Message Size (1400/800/50B from the top row), Variable Period.

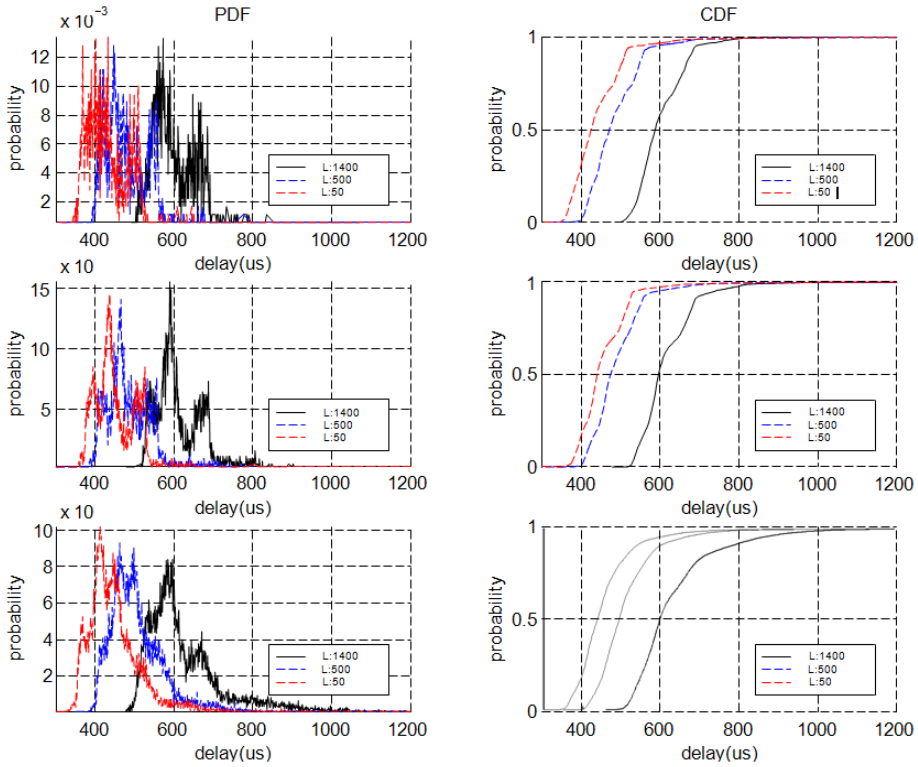


**FIGURE 15:** Competing Flows: PDFs and CDFs-Fixed Period (500/200/50ms from the top row), Variable Message Size.

The delay distributions with competing flows are shown in figures 14 and 15. In figure 14, PDFs and CDFs are plotted for message sizes of 1400, 500, and 50 bytes, respectively, from the first row. Each plot is compared for variable periods of 500, 200, and 50 ms. figure 15 presents the results of various packet sizes (1400, 500, and 50 bytes) with a fixed message period per plot. Each plot illustrates with message periods of 500, 200, and 50 ms. The experiment uses 7 competing flow generators and each host has 10 competing flows that makes 71 flows in total including the test flow. To ensure the flow independence and to reduce the effect of phase synchronization on the periodic flows, competing flows are generated with different start times that are made to be much bigger than the maximum period (if the start time difference is less than the maximum period, it may end up phase synchronization with the repeating experiments). To have solid statistical results, the experiments are performed repeatedly (more than 200 times). The experiments with competing flows clearly show that flows with longer messages and shorter periods experience longer delays with higher probability due to queueing events in the switch. In figure 14, shorter periods tend to result in a high probability of extreme delay for the fixed message sizes. i.e., PDFs are more widely distributed (compared to baselines) and in CDFs, it reaches to one slowly especially in high probability regions. In figure 15, the distributions of larger messages sit on the right-hand side of the smaller messages and the gaps are bigger than the case for baselines since that includes not only transmission delay of its message but also queueing delays.

### 5.2 Algorithm Validation

The measurements of various baseline distributions are used to estimate delay distributions that account for queueing in the switch for the same number and parameters of flows as in an earlier section. The estimated distributions are compared with the actual competing flow experiment distributions jointly with baselines in figure 16. The figures are shown for the high probability region (> 0.9) that aligns with the actual real-time requirements. The results show that the estimated distributions with competing flows approach to the experimental distributions especially in the higher probability regions. It is also observed that the estimated distributions are more conservative than the experiment distribution because it considers the worst case distribution. These experimental results in the real implementation validate the proposed approach.

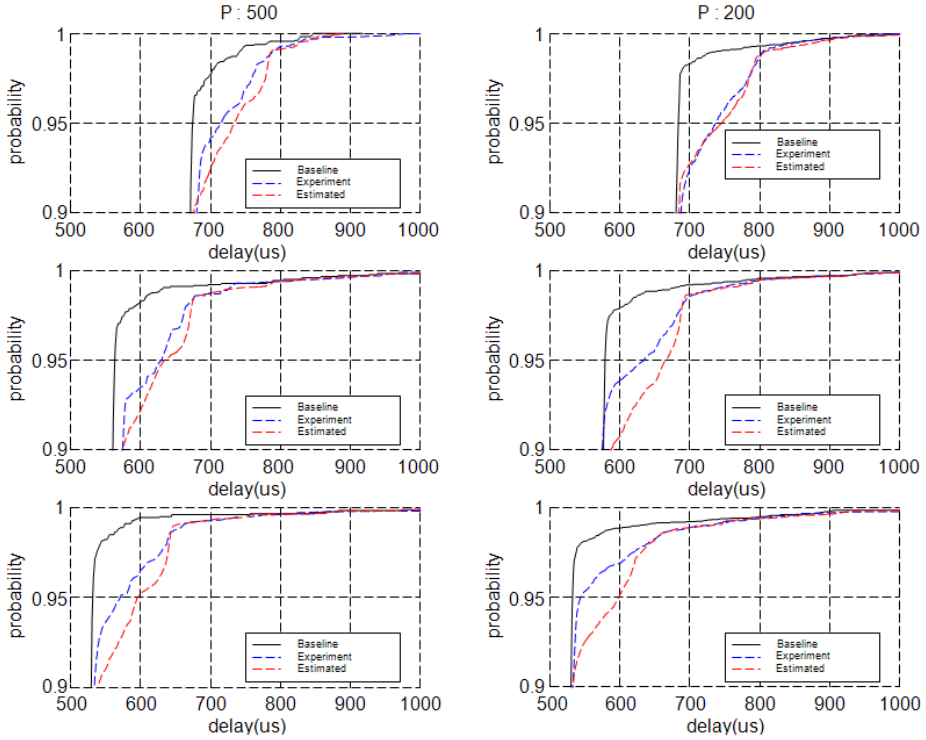


**FIGURE 16:** Delay CDFs from the Proposed Algorithm with Variable Periods.

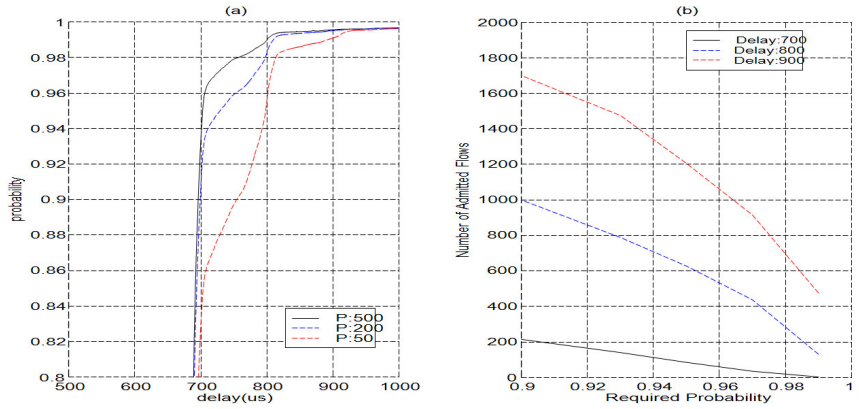


**FIGURE 17:** Estimated distributions (a) Estimated output distribution for variable period, (b) Number of admitted flows.

The trend of the estimated distributions for different parameters is shown in figure 17 (a). Figure 17 (b) also shows the trend in the number of admitted flows in various QoS requirements by successively admitting flows to a system of existing flows. The experiment was performed with a message size of 1400 bytes and a period of 200 *ms*. Since the algorithm admits flows as long as the bandwidth sum is less than the switch capacity and the baseline gives enough probability for the longer delay requirements, we test for shorter delay requirements (700, 800, 900 µs) to show the accuracy. The result shows that the flow admissibility increases for the less requested probability or the longer delay requirements. This indicates that the utilization can be increased by relaxing QoS requirements. The utilization gain is linear rather than exponential since the flows are periodic.

## 6. CONSLUSIONS

We have presented a novel and efficient probabilistic admission control approach to support soft real-time control applications over switched Ethernet. Our approach enables real-time application-to-application QoS management over switched Ethernet without sophisticated packet scheduling or resource reservation mechanisms in Ethernet switches or middleware on end hosts. Application-to-application delay is estimated based on the delay distribution of baseline measurements and queueing analysis with competing flow information. As part of our contributions, we have provided a new and efficient method to identify and estimate the queueing delay probability inside Ethernet switches for heterogeneous periodic control system applications with variable message size and period. This queueing analysis is interesting in itself. We have implemented the probabilistic admission control algorithm on the Windows operating system, and validated its efficiency through extensive experiments.

## 7. REFERENCES

[1] TrueTime, Inc. http://www.truetime.com/.
[2] Worldwide Plant Automation Systems Outlook-Market Analysis and Forecast Through 2010. Automation Research Corporation, 2005.
[3] A. K. Atlas and A. Bestavros. Statistical Rate Monotonic Scheduling. In 19th IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.
[4] V. E. Benes˘. General Stochastic Processes in the Theory of Queues. Addison-Wesley, 1963.
[5] R. Caponetto, L. L. Bello, and O. Mirabella. Fuzzy Traffic Smoothing: Another Step towards Real-Time Communication over Ethernet Networks. In 1st RTLIA, Vienna, Austria, 2002. IEEE Computer Society.
[6] T. Chiueh and C. Venkatramani. Supporting real-time traffic on Ethernet. In Proceedings of IEEE real-time Systems Symposium,1994.
[7] R. Court. Real-Time Ethernet. ACM Computer Communications, 15(3):198–201, Apr. 1992.

[8] Xing Fan and Magnus Jonsson. Guaranteed Real-Time Services over Standard Switched Ethernet. In Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN,05), Sydney, Australia, 2005. IEEE Computer Society.
[9] Robert Janowski, Piotr Krawiec, and Wojciech Burakowski. On assuring QoS in Ethernet access network. In Proceedings of the Third International Conference on Networking and Services ICNS'07, Athens, Greece, 2007. IEEE Computer Society.
[10] S.-K. Kweon and K. Shin. Real-Time Communication over Ethernet with Adaptive Traffic Smoothing. In RTAS '00: Proceedings of the Sixth IEEE Real-Time Technology and Applications Symposium, Washington, DC, USA, 2000. IEEE Computer Society.
[11] S. Lima, P. Carvalho, and V. Freitas. Distributed Admission Control in Multiservice IP Networks: Concurrency issues. Journal of Communications, 1(3):1–9, June 2006.
[12] J. Loeser and H. Haertig. Low-latency Hard Real-Time Communication over Switched Ethernet. In Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS 04), Catania, Italy, 2004. IEEE Computer Society.
[13] D. Loy and R. Schmalek. Thoughts About Redundancy in Fieldbus Systems Anchored in OSI Layer-4 and Applied to the Lontalk Protocol on Neuron-Based Network Nodes. In IEEE International Workshop on Factory Communication Systems, October 1995.
[14] I. Norros, J. W. Roberts, A. Simonian, and J. T. Virtamo:. The superposition of variable bit rate sources in an ATM multipler. IEEE Journal of Selected Areas on Communication, 9(3):378–387, 1991.
[15] P. Pedreiras, L. Almeida, and P. Gai. The ftt-ethernet protocol: Merging flexibility, timeliness and efficiency. In Euromicro ECRTS'02, Vienna, Austria, 2002. IEEE Computer Society.
[16] P. Pedreiras, R. Leite, and L. Almeida. Characterizing the Real-Time Behavior of Prioritized Switched-Ethernet. In 2nd RTLIA, Porto, Portugal, 2003. IEEE Computer Society.
[17] D. W. Pritty, J. R. Malone, S. K. Banerjee, and N. L. Lawrie. A real-time upgrade for Ethernet based factory networking. In Annual Conference of the IEEE Industrial Electronics Society (IECON), 1995.
[18] A. Raha, S. Kamat, and W. Zhao. Guaranteeing End-to-End Deadlines in ATM networks. In 15th International Conference on Distributed Computing Systems, 1995.
[19] A. Raha, S. Kamat, and W. Zhao. Admission Control for Hard Real-Time Connections in ATM LANs. In IEEE INFOCOM, San Francisco, CA, March 1996.
[20] Automation Strategies Report. Ethernet-Based Control Network Strategies. Automation Research Corporation, Oct. 1997.
[21] Y. Shimokawa and Y. Shiobara. Real-Time Ethernet for industrial applications. In Annual Conference of the IEEE Industrial Electronics Society (IECON), 1985.
[22] V. Sivaraman and F. Chiussi. Providing End-to-End Statistical Delay Guarantees with Earliest Deadline First Scheduling and Per-Hop Traffic Shaping. In IEEE INFOCOM, 2000.
[23] T. S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. C. Wu, and J. W. S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In IEEE Real-Time Technology and Applications Symposium, Chicago, IL, May 1995.
[24] C. Venkatramani and T. Chiueh. Design, Implementation, and Evaluation of a Software-based Real-Time Ethernet Protocol. In ACM SIGCOMM, Cambridge, MA, August 1995.