Design Trade-Offs for Networks with Soft End-to-end Timing Constraints
*

HaiFeng Zhu[1] John P. Lehoczky[2] Jeffery P. Hansen[3] Ragunathan Rajkumar[1]
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

As broadband capabilities on wired, wireless and mobile phone networks proliferate, real-time multimedia traffic is expected to consume higher portions of the available bandwidth. Usage models in such networks will range from low-cost VoIP to high-cost, high-quality, high-resolution video streams along with many intermediate data streams. Video flows, however, will have stochastic processing requirements, and therefore traditional real-time techniques may be too pessimistic for providing timing assurances. In this paper, we illustrate a set of analysis techniques for reasoning about lateness in such stochastic flows, and apply it to wired multi-hop networks. We make use of 4 dominant parameters to characterize the design space: end-to-end deadline, acceptable lateness, the number of hops and the network workload. Given any of these three parameters, the remaining parameter can be computed. For instance, if a stochastic flow passes through 10 nodes, its end-to-end deadline is 150ms, and no more than 0.1% of the packets can be late, then we can compute the Maximum Allowable System Workload that would satisfy the lateness requirements. This maximum allowable workload can then be used to develop an admission control policy to ensure that the specified timeliness requirements are met. Simulation results validate the results predicted by our model. Our techniques are applicable to even heterogeneous network cases where some nodes on the flow path have different service time characteristics or cross traffic from other flows.

*This work is supported in part by ONR under contract N00014-92-J-1524, DARPA under contracts F33615-00-C-1729 and F33615-00-C-1701, and ONR/DARPA MURI under contract N00014-01-1-0576.

1. Department of Electrical and Computer Engineering, *{haifengz,raj}@ece.cmu.edu.*
2. Department of Statistics *jpl@stat.cmu.edu.*,
3. Institute for Complex Engineered Systems *hansen@cmu.edu.*

1 Introduction

1.1 End-to-end Timing Guarantee

As multimedia traffic consumes larger portions of available bandwidth, developing network strategies to meet the needs of such traffic is becoming more critical[1, 4, 11]. Multimedia traffic is generally characterized as having both large bandwidth requirements and end-to-end timeliness requirements. While the end-to-end timeliness requirements are critical to achieving acceptable response times for applications such as video conferencing, in most cases a small portion of the traffic can miss deadlines without having a significantly adverse effect on application quality. While much is known about real-time scheduling of periodic tasks with hard deadlines, new techniques are needed to deal effectively with the combination of soft real-time with stochastic workload.

As a first step toward the goal of an integrated framework for stochastic real-time networking, we develop a set of design tools and apply them to a generalized pipeline network. In our example, we assume Earliest Deadline First (EDF) scheduling in the nodes to maximize real-time performance. Our approach is based on Real-Time Queueing Theory (RTQT), and we include a set of theory-based and experiments-based analysis techniques. In order to facilitate network decomposition, we have also developed techniques to analyze bottleneck nodes (nodes with longer times compared to other nodes) and nodes that include cross traffic from other flows.

There is a series of issues that arise in engineering a network. As a simple example, suppose the network shown in Figure 1 has two flows. Flow 1 goes from node A to node B passing through a slow node R1, two fast nodes R2 and R3, and then through a slow node, R4. Flow 2 goes from C to D cutting through R4. Assume that R2 and R3 are very high speed with small delay. Either or both of the nodes R1 and R4 will be bottleneck nodes. Given the network capacity and the deadlines of the flows, we want to determine the fraction of packets from each flow that will be late, the "Deadline Miss Rate" of each flow. Alternatively, we might specify the maximum fraction of packets that can be late and the end-to-end deadlines. From this specification, we want to compute the Maximum Allowable Workload

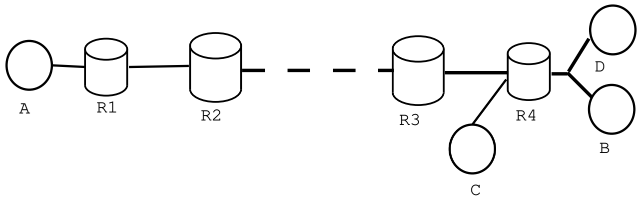

Figure 1. A real-time networked system

that would satisfy the lateness requirements. The Maximum Allowable Workload can then be used to develop an admission control policy to ensure that the specified real-time requirements are met. Finally, we can improve performance by upgrading some of the links. Our analysis can be used to determine which links should be upgraded and what increase in performance will result from the upgrade.

We use Earliest Deadline First (EDF) scheduling in the nodes to maximize real-time performance. In this setting, we distribute the end-to-end deadline to each node evenly, to obtain a per-hop deadline that is used for scheduling at that node. We call this the *EDF Static discipline (ES)*. Our approach is based on Real-Time Queueing Theory (RTQT), and we describe a set of analysis techniques in the paper. These techniques are based on approximations, but simulation results show that they are very accurate. Our primary example is the pipeline network case but we introduce techniques to handle networks with bottlenecks and cross-traffic that hold the promise of decomposing more complicated networks into simpler ones.

We define four fundamental design parameters for stochastic real-time network flows: end-to-end deadline, lateness allowance network workload, and number of hops. The end-to-end deadline D is the time after which the data in a packet is no longer usable. For example, in the case of a video stream, after a frame is displayed packets with data for that frame are no longer usable. For most applications the end-to-end deadline for a flow is constant, but the framework we present can be generalized to stochastic deadlines. The lateness allowance ϕ is the fraction of packets which are allowed to miss their deadline, sometimes called the miss rate. Typically ϕ is 1% or less, but its selection can have a tremendous impact in performance. Increasing ϕ from 0% (hard real-time) to 1% can increase system throughput by a factor of 20 or more when tasks have stochastic characteristics[3]. The workload at a node is characterized by a parameter θ that depends not only on the mean traffic intensity, but on the amount of variability in the service and arrival times. Finally the number of hops, N, is simply the number of queues through which a flow must pass. This can be either the actual number of hops, or as we will show later the number of bottleneck nodes.

Using Real-Time Queueing Theory analysis we develop a principle that we call the Diminishing Miss Rate Effect (DMRE). According to DMRE, in an N node pipeline network with a fixed stochastic traffic source having an end-to-end deadline of $N\tilde{D}$ (where $\tilde{D}$ is the per-hop deadline) and homogeneous per node stochastic service times, the fraction of packets missing their deadline, ϕ, will go to 0 as N goes to infinity. That is, the deadline needed to maintain a specific miss rate scales sub-linearly with the number of hops. Corollaries to this basic principle can be used to estimate any of the four fundamental design parameters when given the other three including the Minimum Feasible Deadline, the Maximum Allowable Workload and the Maximum Hop Count. Additional decomposition principles include the Position Independence Effect (PIE), the Bottleneck Effect (BE), and the Cross Traffic Effect(CTE). These principles expand the applicability of the techniques to a wider class of networks.

1.2 Brief Background to our Analytical Methodology

The analysis underlying the results presented in this paper is based on *Real-Time Queueing Theory (RTQT)*. In this subsection, we present an intuitive, non-technical introduction to RTQT. An outline of the specific mathematical analysis underlying the approximations used is given in the appendix.

RTQT is a real-time scheduling methodology introduced to analyze systems that exhibit significant stochastic behavior with respect to task arrivals, service requirements, or deadlines. This methodology assumes a soft-real-time formulation in which the fraction of late tasks is required to be less than some specified value as opposed to the hard-real-time formulation in which all tasks must always meet their deadlines.

The hard deadline formulation requires that a schedulability analysis must be carried out assuming worst case conditions, thus tasks are assumed to be periodic with period equal to the minimum possible interarrival time and computation time are equal to the maximum possible such time. The resulting worst case task utilization will be larger than the average utilization for that task. As a consequence, the task set being scheduled can have an average utilization much lower than the worst case utilization and the system can be, on average, highly underutilized.

The RTQT methodology uses the actual probability distributions associated with the task interarrival times, their computation requirements and their deadline requirements rather than their worst case values. Approximations based on heavy-traffic queueing theory are then developed to compute the fraction of tasks that will be late. These approximations have been shown to be highly accurate. Furthermore, the methodology results in an *average case schedulability analysis* from which one can find the largest load that can be permitted while still ensuring that the fraction of late tasks is constrained below some specified bound. That bound can be chosen to be a very small number, e.g. 10^{-7}. With such a specification the soft-real-time formulation is nearly equivalent to the hard-real-time formulation for all systems except for truly safety-critical systems in which every task deadline must be met. In spite of the near equivalent formulation, the worst case schedulability bound can be much lower than the average case schedulability bound. Consequently, using the RTQT methodology we often can achieve near hard-real-time behavior at much higher utilization levels than would be permitted by a traditional deadline monotonic or EDF analysis.

In this paper, we consider a generalized pipeline network structure having N nodes with possible cross-traffic at one or more nodes. There are K independent flows that are processed by this pipeline, each of which is associated with a sequence of jobs requiring processing at each stage of the pipeline. The jobs of a particular flow arrive to the network randomly with independent interarrival times governed by a probability distribution with c.d.f F_{ak}, mean α_k, and variance σ_{ak}^2, $1 \leq k \leq K$. The arrival rate of jobs from flow k to the network is $\lambda_k = 1/\alpha_k$. Thus each flow is a renewal process of jobs, and the different flows are assumed to have independent processing requirements at each node governed by the c.d.f. F_{skn} with mean τ_{kn} and variance σ_{skn}^2, $1 \leq k \leq K$ and $1 \leq n \leq N$. Finally, each job from flow k has an end-to-end deadline given by the c.d.f. G_k with mean $\bar{D}_k$, $1 \leq k \leq K$. Typical for many real-time applications that , jobs from a flow are periodic and the end-to-end deadlines are constant. This will, of course, simplify the analysis somewhat.

Real-time performance depends upon the load at each node and the variability of that load. Load is commonly measured by traffic intensity. The traffic intensity from flow k at node n is given by $\rho_{kn} = \lambda_k \tau_{kn}$. The traffic intensity at node n is given by $\sum_{k=1}^{K} \rho_{kn}$. RTQT is useful when the traffic intensities are near 1, i.e. when utilization of the nodes is near 100%. It turns out that for each node, one can compute a single parameter value that characterizes the delay behavior at that node. We begin by computing a total variability parameter for node n defined by $\sigma_{Tn}^2 = \sum_{k=1}^{K} \lambda_k \tau_k^2 (a_k^2 + b_{kn}^2)$ where $a_k^2 = \frac{\sigma_{ak}^2}{\alpha_k^2}$, the squared coefficient of variation of the interarrival distribution of jobs from flow k and $b_{kn}^2 = \frac{\sigma_{skn}^2}{\tau_{kn}^2}$, the squared coefficient of variation of the computation requirements of jobs from flow k at hop n.

All the parameters at node n can be consolidated into a single parameter $\theta_n = \frac{2(1-\rho_n)}{\sigma_{Tn}^2}$, $1 \leq n \leq N$. Under heavy loading conditions, the workload (the sum of execution time of tasks) at node n will, in equilibrium, be approximated most of the time as an exponential distribution with mean $\frac{1}{\theta_n}$ [9]. Small values of θ_n correspond to high levels of work at node n on average. The major result of RTQT is that under these conditions, one can accurately compute, given the workload, the time until the end-to-end deadlines of the tasks at that node elapses (we refer to this as the *lead-time profile*). This prediction will depend upon the scheduling policy that is used. The relationship between workload and the time until a task's deadline elapses is given in the Appendix.

We are interested in the end-to-end performance of tasks, not the performance at a single node. Using heavy traffic queueing theory, one can show that the joint workload at nodes in heavy traffic can be modeled as a multidimensional Brownian motion process. In many interesting cases (including the ones studied in this paper), the equilibrium distribution of the joint workloads will be the product of the marginal distributions, each of which is exponential(θ_n). Using this, one can determine the set of workload values at each node for which the end-to-end deadlines of some flow will be missed. We then need to integrate the joint equilibrium distribution over that set of workload values to determine the overall fraction of tasks that will be late. This fraction depends upon the number of node, the flow deadlines, and the θ_n parameters. By manipulating these, one can alter the fraction of tasks missing their deadlines.

This paper is organized as follows. Section 2 introduces the basic pipeline architecture, the associated tradeoff equation, and introduces the DMRE. Sections 3 and 4 introduce two generalizations, the case in which one or more nodes form a bottleneck and when there is cross-traffic at some of the nodes. The results provide a method for decomposing more complicated networks to the pipeline case. This is illustrated in Section 5 for the example presented in Figure 1. Section 6 presents our conclusions.

2 Basic Pipeline

The basic pipeline network consists of N nodes lined up as a pipeline or tandem structure. Each node has the same mean workload.

2.1 Experimental Illustration

We begin with an experiment showing the DMRE (Diminishing Miss Rate Effect). A task flow passes through a number of hops, and the end-to-end miss rate is measured at the last hop. The interarrival time of tasks is exponentially distributed with mean of 1 and the service time is exponentially distributed with mean of 0.95, thus the traffic intensity at each node is the same, i.e. 0.95. According to heavy traffic theory (for example, [9]) the mean workload of each node can be computed as $\frac{1}{\theta} \approx 18$. We set the deadline to be constant. In Figure 2, the per-hop deadline LD is set to $LD = \tilde{D} = 50 > \frac{1}{\theta} = 18$ and $LD = \tilde{D} = 10 < \frac{1}{\theta} = 18$. The x-axis gives the number of hops on the pipeline, and the y-axis gives the deadline miss rate. We note that the miss rate goes down near exponentially fast as the number of hops grows, when the per-hop deadline is larger than mean workload. However, when the per-hop deadline is smaller than the mean workload, the deadline miss rate is found to shoot up quickly to nearly 100% as the number of hops grows. This phenomenon is not unique to the constant deadline

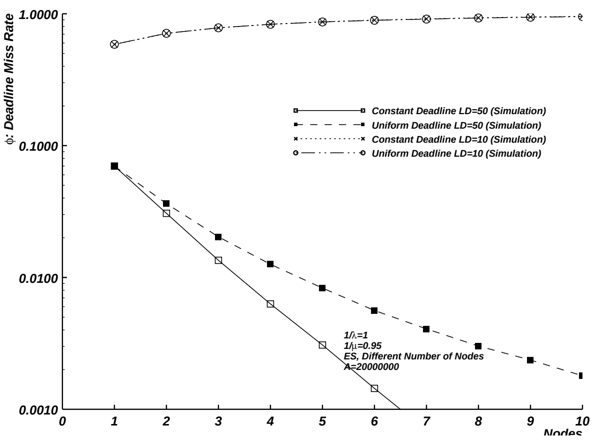

Figure 2. Miss Rate in a Pipeline Network (Diminishing Miss Rate Effect)

case. When the deadlines have a uniform distribution at mean per-hop deadlines given as $\tilde{D} = 50$ and $\tilde{D} = 10$. Figure 2 shows that the DMRE also occurs. These experiments show that in many cases, when the mean per-hop deadline is larger than the mean workload, the miss rate diminishes exponentially fast as the number of hops grows, while the miss rate rapidly increases when the it is smaller than the mean workload. This illustrates the DMRE phenomenon.

2.2 Analysis

For a given workload level, in heavy traffic the set of task lead-times (i.e. the lead-time profile) converges to a deterministic form that depends upon the initial task deadlines and the scheduling policy. Negative values of the lead-time profile correspond to late tasks. For the EDF scheduling policy, the smallest lead-time of the profile corresponds to the lead-time of the task that is exiting the system. If that lead-time is negative, then the exiting task is late. Consequently, for a given scheduling policy and distribution of initial task deadlines, we are interested in the workload levels at which the smallest lead-time values are negative. To determine the overall level of lateness, we need to integrate the equilibrium distribution of the workload levels over the set of workload values that correspond to task lateness. Using heavy traffic queueing theory, this equilibrium distribution can be determined, perhaps numerically. In the pipeline case we consider in this paper, for constant deadline flows, the equilibrium distribution for the N node case will be of product-form, with each node having an exponential distribution with parameter θ_n, $1 \leq n \leq N$. Harrison [9] also showed that for many cases in which the equilibrium distribution is not exactly product form, it is provides a good approximation. Consequently, in this paper we use the product of expontials, the nth node having parameter θ_n, for the equilibrium distribution of the workload.

In this section, we seek to determine the deadline miss rate, ϕ_{ES} and then to prove the Diminishing Miss Rate Effect (DMRE). We use the static deadline scheduling policy in which a task having end-to-end deadline D passing through N nodes will be assigned a deadline $\frac{D}{N}$ at each node independent the time it took to finish work at the previous nodes it visited. We consider two cases: 1) constant deadlines and 2) general deadline distributions.

For the constant deadline case, if the workload is W_n at node n and tasks arrive at each node with deadline $\frac{D}{N}$, then the lead-time profile will be uniform on the interval $[\frac{D}{N} - W_n, \frac{D}{N}]$. Consequently, task lateness occurs when the smallest value, $\frac{D}{N} - W_n$, is negative, or when $W_n > \frac{D}{N}$. This is lateness only at node n with respect to the local deadline. We are only interested in end-to-end lateness, i.e. when the sum of all the local lateness values is negative. Consequently, end-to-end lateness is associated with the situation in which $\sum_{n=1}^{N}(\frac{D}{N} - W_n) < 0$ or when $\sum_{n=1}^{N} W_n > D$, the end-to-end deadline. The random variables W_n, $1 \leq n \leq N$ are approximated to have independent exponential θ_n distributions. Thus the sum $\sum_{n=1}^{N} W_n$ has a generalized gamma distribution and the probability of the end-to-end lateness event can be calculated from this probability distribution, $P(\sum_{n=1}^{N} W_n > D)$.

The special case in which $\theta_n = \theta$, $1 \leq n \leq N$ allows us to develop a closed form expression [13]. Specifically,

$$\phi_{\mathrm{ES}} = P(\sum_{n=1}^{N} W_n > D) = \exp(-\theta D)(1+\theta D+(\theta D)^2/2! + \cdots + (\theta D)^{(N-1)}/(N-1)!). \tag{1}$$

Formulas can be developed for the case in which the the θ parameters are not equal. For example, the formula for two nodes is given in equation (2) as a part of our analysis of the network in Figure 1. Figure 3 shows a close match between Equation (1) and the experimental results. In this figure, three experiments were run for a flow passing through two nodes with different service times (constant, uniform and exponential). The dotted curves give the experimental results and lines give the theoretical predictions.

If we fix the per-hop deadline to be $\tilde{D}$ and let the end-to-end deadline $D = N\tilde{D}$, then the deadline miss rate, ϕ_{ES} converges to 0, provided the $\tilde{D} > \frac{1}{\theta}$, the mean workload. We refer to this phenomenon as the Diminishing Miss Rate Effect (DMRE). More general proofs are available in [13].

2.3 Engineering Design Consideration: Minimum Feasible Deadline Principle

Equation (1) above gives us the ability to gain insight into the behavior of the pipeline network and to develop

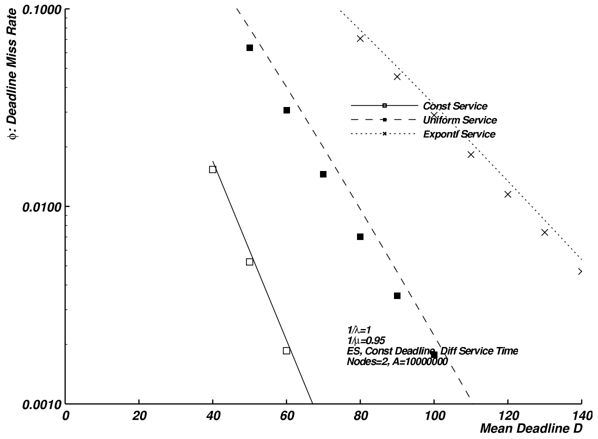

Figure 3. Two Node Pipeline: Constant Service Constant Deadline

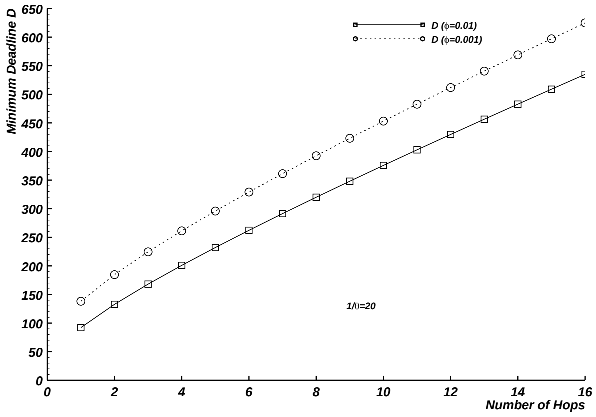

Figure 4. Minimum Deadline Needed for Specified Number of Hops

various design principles. In particular, we would like to use this equation to understand the relationships among various system parameters including the number of hops, the task deadlines, the variability of the flows and the resulting percentage of late tasks. We begin by illustrating one such design principle: the Minimum Feasible Deadline principle (MFD).

First, we use the static EDF scheduling policy at all nodes. Second, we designate the per-hop deadline of the tasks to be larger than the mean workload at each node (i.e. $\frac{D}{N} > \frac{1}{\theta}$); otherwise, as shown in Figure 2, on average all tasks will take longer to complete than their per-hop deadline and almost all will be late. In this case, the Diminishing Miss Rate Effect indicates that the fraction of task lateness will decrease exponentially with the number of hops for fixed per-hop deadline. This observation means that as the number of hops increases, we can reduce the per-hop deadline and still keep the miss rate constant. Using Equation (1), we can fix ϕ_{ES} and θ and solve for D as a function of N.

For example, Figure 4 shows the minimum end-to-end deadline needed to guarantee that the miss rate ϕ_{ES} is at or below 0.01 or 0.001. The two curves show the minimum per-hop deadline needed for such a guarantee as a function of the number of hops. We can see the minimum end-to-end deadline to guarantee $\phi_{\mathrm{ES}} \leq 0.01$ for $N = 1$ is 92. A naive design to achieve the .01 miss rate for $N = 10$ hops would be to use an end-to-end deadline of $92 \cdot 10 = 920$. Figure 4 shows that it only needs to be 376, which is approximately only 40% of the deadline required by the naive design.

We can also use Equation (1) to determine the maximum allowable workload. In Figure 5, if each flow takes 1% of the link capacity, for a fixed end-to-end deadline of 50 and a miss rate constraint of 0.01 or 0.001, this figures shows the maximum flows that can be admitted as a function of the number of hops. Note that as θ increases, the mean workload decreases. As the number

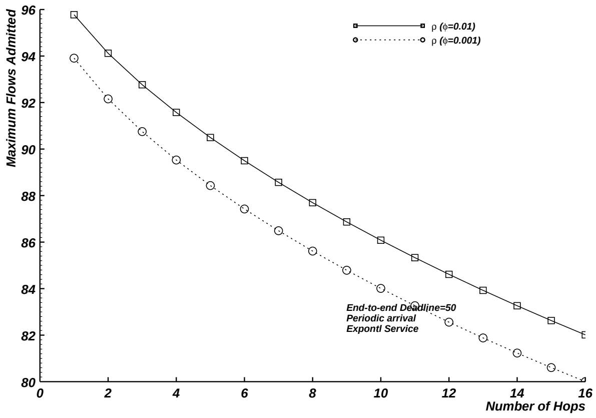

Figure 5. Maximum Allowable Workload for Specified Number of Hops

of hops grows while the deadline miss rate is held constant, the maximum allowable workload at each node decreases, therefore the number of flows admitted decreases. We can also fix the task lateness percentage and investigate the tradeoff between the end-to-end deadline and the number of hops permitted. Figure 6 considers two miss rates, .01 and .001 and $\theta \leq 0.1$ (which standards for the minimum workload must be kept greater or equal to 10 seconds at each node). One can see the sharp increase in the maximum number of hops permitted as the end-to-end deadline increases.

3 Bottlenecked Pipeline

In Section 2 we dealt with the homogeneous pipeline case in which the loads on each node were identical ($\theta_n = \theta$, $1 \leq n \leq N$). In this and next section we will consider the cases in which the nodes are not homogeneous either

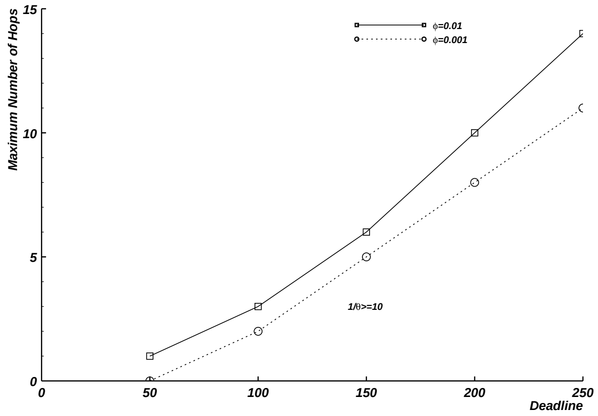

Figure 6. Max Number of Hops Allowed for Specified Deadline and Lateness Guarantee

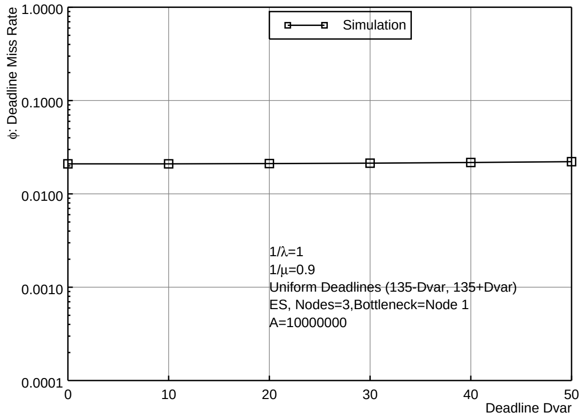

Figure 7. EDF Pipeline Network with Node 1 as Bottleneck

because of different processing capabilities at different nodes (which will cause certain nodes to be bottlenecks) or the loads at different nodes differ because of cross-traffic.

3.1 Bottleneck Effect and Position Independence Effect

A bottlenecked pipeline is a tandem of N nodes where some of nodes are significantly more congested than the others. We considered the following case. A flow with uniform deadlines passes 3 nodes in tandem where each node's scheduling policy is EDF static. We chose one of the nodes to be more heavily loaded than the other nodes, by multiplying the service times of the tasks arriving at that node by a constant factor that is larger than 1. The first observation is that the end-to-end deadline miss rate is the same, no matter which of the three nodes becomes the bottleneck. We call this phenomenon the *Position Independence Effect* (PIE). One of these three miss rate graphs is shown in Figure 7.

PIE gives us the ability to extend a simple pipeline where all nodes have the same traffic intensity to more complex pipeline cases. Furthermore, when the variance of the deadline increases, while keeping the mean deadline fixed, we see slight differences in the miss rate. This is useful for miss rate approximations for non-constant deadline cases. Similar simulation results were found when feeding in multiple flows to this pipeline.

Most importantly, we observe that if there is one or more node (say, N^* nodes, where $N^* \leq N$) that have identical mean workloads significantly larger than the mean workload of the other nodes, then the end-to-end miss rate is close to the miss rate assuming that there are only the N^* tandem nodes present. That is, the bottleneck nodes dominate the miss rate and makes other nodes' effects negligible. We call this the *Bottleneck Effect* (BE). This means that we can approximate this network by a pipeline network with only the N^* nodes.

3.2 Defining Bottleneck nodes

As we discussed in Section 2, a miss rate formula generalizing Equation (1) for the heterogeneous node case can be written using the generalized gamma distribution; however, this expression is extremely complicated. It is, therefore, useful to develop simpler approximations. To accomplish this, we must first define what is meant by a bottleneck node. Ordinarily, the load or traffic intensity, ρ, is taken as the primary indicator for each node. In fact, traffic intensity by itself is not a sufficient measure, because it does not take into account the variability of the flows at the node. Increases in variability result in increases in delay, hence increases in task lateness.

Consider the following two stage system in which the incoming task interarrival time is exponentially distributed with a mean of 0.9 seconds. We now let the first stage processing time be deterministic for any task with the mean of 0.97 seconds, while the second stage processing time is exponentially distributed with mean of 1 second. Then the traffic intensities at each node satisfy $\rho_1 = 0.93 > \rho_2 = 0.9$. This would lead to the conclusion that node 1 is the bottleneck. In fact, $\theta_1 = 0.15 > \theta_2 = 0.11$, which means the second node is actually the bottleneck. This can serve as a principle for identifing bottleneck for upgrades, which we called LUS (Link Upgrade Selection) principle.

We follow the direction specified by RTQT and use θ_n the reciprocal of the mean workload at node n to be the indicator of load. Here, small values of θ_n correspond to large values of load. We define $\theta_{\min} = \min_{1 \leq n \leq N} \theta_n$. This gives the heaviest load in the network. Now, we introduce a threshold, T_b and define the set of bottleneck nodes to be $S_b = \{j | \frac{\theta_j}{\theta_{\min}} \leq T_b\}$. The ratio is 1 for the most heavily loaded node(s). T_b is a parameter that allows us to tune the accuracy of our approxima-

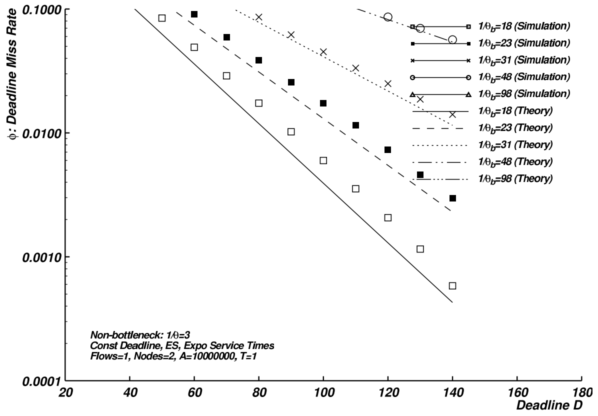

Figure 8. Two Node Pipeline with Bottleneck at Node 1, Experimental Miss Rate and Theoretical Approximation

tion. The larger the value of T_b the larger will be the set of bottleneck nodes. On the other hand, as we reduce T_b toward 1, the number of bottleneck nodes decreases. In the latter case, the analysis becomes much simpler; however, the approximation may not be as accurate becomes some nodes with significant congestion may be ignored. A detailed analysis of the approximations and the errors that result is given in [13].

3.3 Experimental Illustration

Figure 8 shows results from an experiment with two nodes in a pipeline in which one of the nodes has a slower service rate and is a bottleneck with mean workload given by $1/\theta_b$. The other node has mean workload given by $1/\theta = 3$. The bottleneck mean workload increases from 18 to 98. The theory curve is plotted assuming there is only one node in the pipeline (the bottleneck node) with no alteration to the end-to-end deadline. The experiment shows the higher the workload of the bottleneck, the more accurate the approximation.

4 Cross Traffic

In this section we generalize the pipeline network case to allow nodes to have cross-traffic flows. For this case, we show how we can use the ideas in the section 3 on bottleneck traffic to reduce the network back to the pipeline case with bottleneck nodes. A position independence effect is also presented with facilitates this decomposition.

4.1 Position Independence Effect

A flow passing through the N nodes can be viewed as a flow on a pipeline with some of the nodes having *cross-traffic*, some of which may continue along the same path for more than a single hop. For example, Figure 9 shows

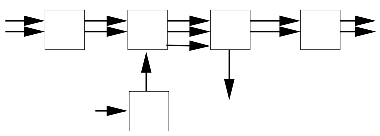

Figure 9. One Cross-Traffic Node

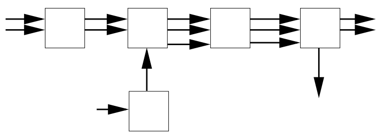

Figure 10. Two Cross-Traffic Nodes

a flow with one cross-traffic node, and Figure 10 shows a network with two cross-traffic nodes.

To study this more general network topology, we take a two-step approach. First, we study an N node pipeline network within which N^* ($N^* \leq N$) nodes have their θ parameters increased by slightly increasing their average service times. Second, we then compare the resulting network to a network with cross-traffic nodes that causes similar increases in their θ parameters.

Since the location of the N^* nodes can be arbitary, we experimented with different locations. Those experiments showed that the Position Independence Effect (PIE) also holds for this cross-traffic case. Figure 11 presents a representative graph. This PIE finding indicates the possibility of analytically reducing the case of K flows passing N tandem nodes with N^* cross-traffic nodes to the case of an N^* node pipeline. This is described next.

4.2 Cross-Traffic Effect: Approximating Bottleneck Pipeline

Suppose we have a primary flow (flow 1) entering node 1 and continuing to node N, then exiting the system and an interfereing flow (flow 2) entering and exiting node N_c. Suppose further that each node is experiencing heavy traffic conditions, and at node N_c, flow 1 takes a proportion α of that node's traffic, while flow 2 takes the rest $1-\alpha$ portion . We want to find the deadline miss rates in this case. Denote the local per-hop deadlines for flow 1 and flow 2 by $\tilde{D}_1$ and $\tilde{D}_2$ respectively. We further assume the workload at each node W_n is independently and exponentially distributed with mean $1/\theta_n$, $n = 1, \ldots, N$. Using the analysis in [13] we found the upper bound for miss rate of flow 1: $\phi_1 \leq \phi_1^{up} = P\{\sum_{n=1}^{N} W_n > (N-1)\tilde{D}_1 + \tilde{D}_1\rho_{N_c,1} + \tilde{D}_2\rho_{N_c,2}\}$

The ϕ_1^{up} equation is easier to calculate than the exact ϕ_1. After plugging in the workload, the θ's of each node will appear and play important roles in determining the miss rate. Thus we can perform the same analysis as bottlenecked pipeline in the previous section. Let us assume $\rho_n = \rho$ for $n = 1, \ldots, N$ are all the same and

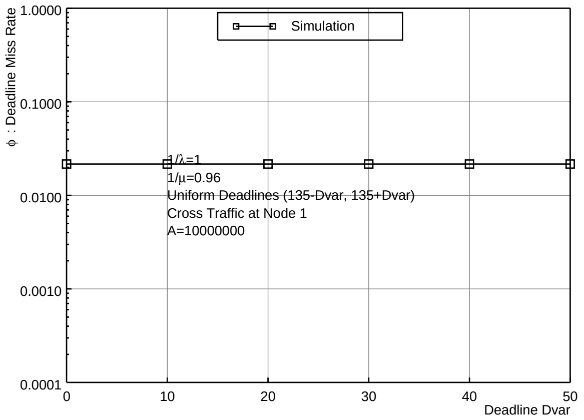

Figure 11. EDF Pipeline Network with Cross-Traffic at node 1

assign $\lambda_k = \lambda$ for $k = 1, 2$. If α is large (i.e. close to 1), $\theta_n \approx \theta_{N_c}$ ($n \neq N_c$), and this is equivalent to the "Basic Pipeline" case where all the nodes are essentially same, and the effect of flow 2 is negligible. If α is small (i.e. flow 2 matters), $\frac{\theta_n}{\theta_{N_c}}$ depends on $\frac{C_2^2}{C_1^2}$, the ratio of the sum of squared coefficient of variations of the interarrival and service times of flow 2 to that of the flow 1. If this value (sum of squared coefficient of variations of the interarrival and service times) of the flow 2 (the interfering flow) is larger than that of flow 1 (the primary flow), this means the interfering flow is more variable in its traffic characteristics. In this case, θ_{N_c} is smaller, therefore we can see if node N_c can be considered to be a bottleneck node. To do so, we can simply apply the "bottlenecked pipeline" analysis described in previous section. If $\theta_n > \theta_{N_c} T_b$ ($n = 1, \ldots, N$ and $n \neq N_c$), node N_c can be viewed as a bottleneck node. Otherwise, the other $N - 1$ nodes (except node N_c) can be viewed as nodes with heavier traffic. For these $N - 1$ nodes, if $\theta_n T_b < \theta_{N_c}$ is true for their θ_n, they can be viewed as bottleneck nodes. Thus, we reduce the cross traffic pipeline case to a simple bottleneck pipeline case.

Some applications are able to generate traffic with different compression and sending schemes that produce different traffic characteristics, such as periodic or aperiodic. By selecting different schemes, we can change $\frac{C_2^2}{C_1^2}$ through which we can reduce the congestion of the cross-traffic at node N_c thus preventing it from becoming a bottleneck node. We call this technique as "Sending Traffic Selection" principle, which is useful in choosing compression and encoding algorithms that generate different types of traffic.

5 Engineering a real-time network

In this section we return to the example in Figure 1 and illustrate the use of our design principles to engineer this network. The figure indicates that nodes R2 and R3 will not be bottlenecks, so using the principles described in Section 3 and 4, we can reduce our network to the two node case, R1 and R4. Now we must address the cross-traffic effect at Node 4. Suppose there are two compression algorithms that can be used for Flow 2, one of which creates a constant bit-rate flow with fixed packet sizes, while the other creates a variable bit-rate flow with variable packet sizes. The compression algorithm giving the constant bit-rate flow will reduce the ratio $\frac{C_2^2}{C_1^2}$. This will increase θ_4 and make node R_4 less congested. If the congestion is sufficiently reduced, then R1 would be the sole bottleneck and we could treat the network as though it has a single node. The miss rate would then be given by $e^{\theta_1 D_1}$.

Suppose instead that the compression algorithm and the cross-traffic serve to make R4 similarly congested as R1, hence the resulting network must consider both nodes R1 and R4. Equation (1) can be used if the flows result in $\theta_1 = \theta_4 = \theta$, and the miss rate would be given by the formula $e^{(-\theta D_1)}(1+\theta D_1)$. If θ_1 and θ_4 are not equal but not far enough apart for one of the two nodes to be a bottleneck, then we would need to treat the network as having two nodes with unequal θ values. As mentioned earlier, this can be easily done by using the generalized gamma probability distribution rather than the standard gamma distribution. In this more general case, the miss rate is given by the formula:

$$\phi_{\mathrm{ES}} = \frac{\theta_1 e^{-\theta_4 D_1} - \theta_4 e^{-\theta_1 D_1}}{\theta_1 - \theta_4} \tag{2}$$

where we assume $\theta_1 \neq \theta_4$. One can use this more general equation to determine tradeoffs among the lateness rate, the workload levels at the two nodes and the task deadlines just as Equation (1) was used earlier.

To complete the numerical example, we assume for simplicity that the $\theta_1 = \theta_4 = .03$, an average workload of 33 at both nodes with θ_2 and θ_3 being significantly larger that they can be removed from the network in analysis. We hope to seeking for the minimum feasible deadline D from R1 to R4. Consequently, we approximate the task deadline miss rate by the formula $e^{-0.03D}(1 + 0.03D)$. Suppose we want the miss rate to be no larger than 0.1. We can numerically solve the equation $.1 = e^{(-0.03D)}(1+0.03D)$ to find the minimum feasible end-to-end deadline to be $D = 129.7$.

Figure 12 presents the theoretical miss rates using the above analysis as a function of D, and we also see simulation results for a set of different deadline values. Notice the excellent agreement between the simulation and theoretical approximation. Consequently, this simple example shows how we can use the network decomposition principles to analyze and design networks of this type to meet real-time specifications.

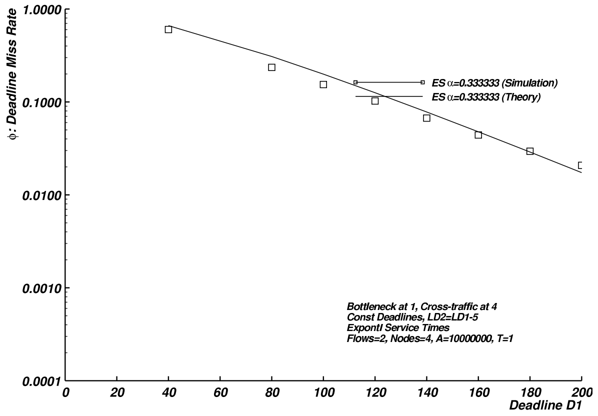

Figure 12. Experimental and Theoretical Comparison in Example Network

6 Conclusion

Earliest Deadline First is a widely studied real-time scheduling algorithm, although it's real-time behavior has been analyzed almost exclusively under worst-case deterministic assumptions. This paper studies a particular EDF network scheduling algorithm in a stochastic setting in which the tasks' interarrival time, service (execution) time, and deadlines are allowed to be random. We decomposed the network into three types: *basic pipeline* without bottleneck, where every node has the same mean workload, *bottlenecked pipeline* in which different nodes can have different mean workloads, and *cross-traffic* where a number of flows cut through the pipeline at various nodes. These different node conditions can be caused by different traffic intensities (or link speeds), or different network topologies. We consider the *static deadline* EDF discipline where the end-to-end deadline is distributed evenly to each node on a path and is treated as a local deadline at each node. We focused on the end-to-end deadline miss rate, and we found that it exhibited an interesting phenomenon that can serve as a useful engineering guideline for designing a real-time EDF network. That phenomenon is called the Diminishing Miss Rate Effect (DMRE) and it states that in a pipeline network architecture, the task end-to-end miss rate decreases exponentially as the number of hops increases assuming that the per-hop deadline is held constant. The Bottleneck Effect (BE) and the Position Independent Effect (PE) were defined, and together they indicate that a larger network can be reduced to one consisting of only the bottleneck nodes, independent of the location of those nodes. Finally the Cross-Traffic Effect(CTE) was defined and it allowed certain nodes to be treated a bottleneck nodes.

We illustrated these phenomena with simulation experiments and theoretical analyses based on Real-Time Queueing Theory (RTQT), and the two showed good agreement. Section 5 presented the methods on a 4 node network with two flows.

Finally, we developed a set of design principles: the Minimum Feasible Deadline (MFD) principle, the Link Upgrade Selection (LUS) principle, and the Sending Traffic Selection (STS) principle. They can be used in system design to approximate the end-to-end miss rate using fairly simple computations, and they can be used to significantly reduce deadline miss rates by tuning the traffic, adjusting deadline parameters, and making intelligent network topology choices.

Appendix: Brief Introduction to Real-Time Queueing Theory

This section provides a brief technical background on RTQT. The ideas were first introduced by Lehoczky [7]. The single queue case was put on a firm mathematical foundation in the paper by Doytchinov, Lehoczky and Shreve (DLS) [2]. The theory has also been extended to the case of open acyclic queueing networks with multiple independent traffic flows [6].

The analysis of real-time queues scheduled using EDF requires that deadline information be kept for every task. In particular, for each task we track its "lead-time" (task lead-time = time remaining until the task's deadline elapses). The addition of these lead-time variables results in a high dimensional state space. As a result, the analysis of the general problem is analytically intractable. Fortunately, excellent approximations are available in the heavy traffic case ($\rho \to 1$). Moreover, the heavy traffic case is the most important for real-time systems, since if one can accurately approximate the behavior of the system in this case, it will provide a bound for any lighter traffic case. Heavy traffic queueing theory is a well developed theory (e.g. see [9]). RTQT adds the analysis of real-time tasks to this theory.

Traditional network queueing theory studies the network workload process, $\{W(t) = (W_1(t), \ldots, W_N(t)), t \geq 0\}$, and the network queue length process, $\{Q(t) = (Q_1(t), \ldots, Q_N(t)), t \geq 0\}$. Here $W_i(t)$ equals the sum of the remaining computational requirements of all tasks present at node i of the system at time t. If $W_i(t) > 0$, then the workload process at any node decreases linearly at unit rate until the node empties or until a new task arrives at which time it jumps by the computational requirements of the arriving task. For simplicity we focus on the single node case, although the ideas generalize to the network case. Heavy traffic queueing theory considers a sequence of queueing systems indexed by some parameter $n \to \infty$. At each node, as n increases, the traffic intensity of the corresponding queueing system is assumed to increase to 1. Specifically, we define $\rho^{(n)} = \lambda^{(n)}/\mu^{(n)} = 1 - \gamma/\sqrt{n}$ for some constant $\gamma > 0$. Appropriately scaled versions of the workload and the queue length processes converge to drifted reflected Brownian motion processes. Specifically, if $W^{(n)}(t)$ and $Q^{(n)}(t)$ represent the workload and queue length of the

n^{th} system at time t, then $\hat{W}^{(n)}(t) = \frac{W^{(n)}(nt)}{\sqrt{n}} \Rightarrow W^*$, and $\hat{Q}^{(n)}(t) = \frac{Q^{(n)}(nt)}{\sqrt{n}} \Rightarrow \mu W^*$, where W^* is a reflected Brownian motion with drift $-\gamma$ and some scale parameter.

Much is known about this limiting process, so many features of the system performance can be approximated using facts about Brownian motion. While the above is written for the single node case, the ideas extend to networks with the Brownian motion process being replaced by a multi-dimensional Brownian motion restricted to an orthant.

Real-time queueing theory builds on heavy traffic queueing and includes task lead-times. At time t, let $\mathcal{Q}^{(n)}(t)(I)$ represent the number of tasks at time t in the nth system having lead-times taking values in some interval I. For example, I might be $[0,\infty)$, in which case, we would be focusing on the number of tasks with positive lead-times, i.e. those tasks that are not yet late. Similarly, we can define $\mathcal{W}^{(n)}(t)(I)$, the remaining work requirements of all tasks in the system at time t having lead-times at time t contained in I. DLS [2] proved that suitably scaled versions of these two processes converge to deterministic limits. Those limiting results form the basis of the real-time queueing approximations used in this paper. Specifically, we define the limiting scaled frontier process, $F^*(t) = H^{-1}(W^*(t))$ where $H(y) = \int_y^\infty (1-G(u))du$. The frontier is the longest lead-time of any customer that has received any service. All tasks with lead-times longer than the frontier have never received any service. We associate lateness with the frontier reaching or lying to the left of 0, consequently we seek to compute the probability that the frontier is negative.

Now the DLS prove that the scaled workload measure process can approximated by

$$\hat{\mathcal{W}}^*(t)(B) = \int_{B\cap[F^*(t),\infty)} (1-G(u))du,$$

for any Borel set B. This can be used to characterize the lead-time profile of all the tasks in the system. This profile is given by

$f(x) = \frac{(1-G(x))}{W(t)}, F(t) \le x < \infty$

where G is the c.d.f. of the task deadline distribution, $W(t)$ is the current queue length, and $F(t)$ is the frontier, where $W(t) = \int_{F(t)}^\infty (1-G(x))dx$. For given values of $W(t)$, this equation can be used to determine the frontier $F(t)$.

Lateness is associated with $F(t) \le 0$, or equivalently

$$W(t) \ge \int_0^\infty (1-G(x))dx = E(D).$$

Consequently, for the single node single flow case

$$P(\text{Lateness}) = P(W(t) \ge E(D)) = e^{-\theta E(D)},$$

or

$$\log(P(\text{Lateness})) = -\theta E(D).$$

The single flow single node case has been extended to the single-node multiple flow case by Kruk, Lehoczky, Shreve, and Yeung (KLSY1) [5]. This was, in turn, extended to the feed-forward case with multiple flow by Yeung and Lehoczky [10]. Finally, the work was extended to the case of acyclic networks by Kruk et.al. (KLSY2) [6].

References

[1] Chandra,P., Fisher, A., Kosak, C., Ng,T.S.E., Steenkiste,P., Takahashi, E., Zhang,H., "Darwin: Resource Management for Value-Added Customizable Network Service", From *6th IEEE International Conference on Network Protocols (ICNP'98)*, October 1998

[2] Doytchinov, B., Lehoczky, J.P. and Shreve, S.,"Real-time queues in heavy traffic with earliest deadline first queue discipline," *Ann. of Appl. Prob.*, **11**, 2001, 332-378.

[3] Hansen, J.P., Zhu, HF., Rajkumar,R., "Probabilistic Bandwidth Reservation by Resource Priority Multiplexing", *IEEE RTAS 2001*, Taiwan,2001.

[4] Hoover,C.L., Hansen, J.P.,Koopman,P., Tamboli,S., "The Amaranth Framework: Policy-Based Quality of Service Management for High-Assurance Computing", *International Journal of Reliability, Quality, and Safety Engineering*, 2001.

[5] Kruk, L., Lehoczky, J., Shreve, S., and Shu-Ngai Yeung, "multiple-input heavy-traffic real-time queues," *Annals of Applied Probability*, **13(1)**, 2003, 54-99.

[6] Kruk, L., Lehoczky, J., Shreve, S., and Shu-Ngai Yeung, "Earliest-deadline-first service in heavy-traffic acyclic networks," to appear *Annals of Applied Probability*, 2004.

[7] Lehoczky,J.P.,"Real-Time Queueing Theory", In *Proceedings Real-Time Systems Symposium*, pg. 186-195, December 1996

[8] Lehoczky, J.P., "Scheduling communication networks carrying real-time traffic", In *Proceedings Real-Time Systems Symposium*, December 1998, 470-479.

[9] Peterson, W.P., "A heavy traffic limit theorem for networks of queues with multiple customer types," *Math. of Operations Research*, **16**, 1991, 90-118.

[10] Yeung,S.N., Lehoczky,J., "End-to-end Delay Analysis for Real-Time Distributed Networks" , An early version submitted to *IEEE Real-Time System Symposium* , 2001

[11] Zhang,L., Deering,S., Estrin,D.,Shenker,S., Zappala,D., "RSVP: A New Resource ReSerVation Protocol", From *IEEE Network*, vol. 7, no. 5, p. 8-18, September 1993

[12] Zhu,HF., Hansen,J.P., Lehoczky,J.P., Rajkumar,R., "Optimal Partitioning for Quantized EDF Scheduling", *Proceedings of IEEE Real-Time Systems Symposium*, Austin, Texas, 2002.

[13] Zhu, HF.,Lehoczky,J.P.,Hansen,J.P.,Rajkumar,R.," Design Trade-Offs for Networks with Soft End-to-end Timing Constraints", *CMU ECE Technical Report, Pittsburgh, Pennsylvania,2004.*