A Dynamic Priority Assignment Technique for Streams with (*m*, *k*)-Firm Deadlines

Moncef Hamdaoui, *Member, IEEE*, and Parameswaran Ramanathan, *Member, IEEE*

***Abstract*—The problem of scheduling multiple streams of real-time customers. is addressed in this paper. The paper first introduces the notion of (*m*, *k*)-firm deadlines to better characterize the timing constraints of real-time streams. More specifically, a stream is said to have (*m*, *k*)-firm deadlines if at least *m* out of any *k* consecutive customers must meet their deadlines. A stream with (*m*, *k*)-firm deadlines experiences a dynamic failure if fewer than *m* out of any *k* consecutive customers meet their deadlines.**

The paper then proposes a priority-based policy for scheduling *N* such streams on a single server to reduce the probability of dynamic failure. The basic idea is to assign higher priorities to customers from streams that are closer to a dynamic failure so as to improve their chances of meeting their deadlines. The paper proposes a heuristic for assigning these priorities. The effectiveness of this approach is evaluated through simulation under various customer arrival and service patterns. The scheme is compared to a conventional scheme where all customers are serviced at the same priority level and to an imprecise computation model approach. The evaluation shows that substantial reductions in the probability of dynamic failure are achieved when the proposed policy is used.

***Index Terms*–Real-time systems, deadline-constrained scheduling, dynamic failure, priority queues, soft deadlines.**

I. Introduction

A real-time application is usually comprised of a set of cooperating tasks which interact with each other by exchanging messages. The tasks, and thus the corresponding messages, are often invoked/activated repeatedly. Each instance of a task has a deadline by which it is expected to receive complete service. Examples of real-time applications include process control, life-support systems, automated manufacturing systems, robotics, spacecraft, and multimedia. In some cases, the failure of a task or a message to meet its deadline can have catastrophic consequences, and therefore, it is very important for the task to complete its computation in a timely fashion. Such critical tasks and messages are said to have *hard* deadlines.

In contrast, there are many real-time applications in which it is not necessary for every instance of a repetitive task or message to meet its deadline. Such tasks and messages are said to have *soft* deadlines. Consider, for example, the real-time transmission of digitized full motion video. A source (e.g., a video camera) generates a stream of video frames at a rate of, say, 30 frames/sec. These frames are transmitted and are played back as they arrive at the destination. Each frame has a deadline before which it must reach the destination. A frame that misses its deadline is dropped and is considered lost. In this application, one can tolerate a few missed deadlines (i.e., dropped frames) without a significant degradation in the video quality.

The tolerance to deadline misses has traditionally been expressed as a maximum allowable loss percentage. For example, a video stream may be specified to tolerate a 10% loss rate. The problem with this specification is that it implicitly assumes that the lost frames are "adequately" spaced. That is, the video quality is not acceptable if too many consecutive frames are lost, even if the overall loss rate is below 10%. The requirements of such a stream can be more precisely expressed by specifying two constants *k* and *m* such that the quality of service is acceptable as long as at least *m* frames in any window of *k* consecutive frames meet their deadlines. We refer to such timing constraint (*m*, *k*)-*firm deadlines*.

A stream with (*m*, *k*)-firm deadlines experiences a *dynamic failure* if fewer than *m* customers meet their deadlines in a window of *k* consecutive customers. The rate at which a stream experiences dynamic failure is therefore a measure of how often the quality of service falls below the acceptable level. The problem addressed in this paper is to schedule a set of *N* real-time customer streams, each with its own deadline requirements, so as to reduce the probability of dynamic failure. This is achieved by carefully assigning priorities to customers. The basic idea of the proposed priority assignment scheme, called the distance-based priority (DBP) scheme, is to assign a priority to each customer based on the recent history of missed deadlines in the corresponding stream; if several recent instances from a stream have missed their deadlines, then the next customer from the stream is assigned a higher priority. The server uses this priority—in conjunction with other parameters like deadlines—to determine an order of service among the pending customers. The proposed DBP scheme is compared to a single priority (SP) scheme where all customers are serviced at the same priority level. Simulation results show that there is a substantial reduction in the probability of dynamic failure as a result of using the proposed scheme.

It is difficult to relate the proposed scheme to other work in the literature because of the new model. Many schemes have been proposed to meet a specified maximum allowable loss percentage, where the loss includes customers with missed deadlines [2], [3], [5], [8], [10]. As indicated earlier, however, the problem with this model is that the loss percentage requirement can be met even if a large number of consecutive customers miss their deadlines. Streams with (*m*, *k*)-firm dead-

Manuscript received Mar. 12, 1994.

M. Hamadoui is with Northern Telecom in Tunis, Tunisia.

P. Ramanathan is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706-1691; e-mail: parmesh@ece.wisc.edu.

To order reprints of this article, e-mail: transactions@computer.org, and reference IEEECS Log Number C95147.

lines can also be serviced using the imprecise computation model [4], [7], [6]. In this model, customers are statically classified as either mandatory or optional. Mandatory customers are always serviced. Optional customers are serviced on a best-effort basis. A stream with (m, k)-firm deadlines can be dealt with by statically classifying the customers in such a way that there are at least m mandatories in any window of k consecutive customers. Although this approach can result in a substantial reduction in the probability of dynamic failure, it does so at the expense of a higher overall probability of deadline miss.

The rest of this paper is organized as follows. The system model and the notion of (m, k)-firm deadlines are described in Section II. In Section III, we describe the proposed distance-based priority assignment technique. In Section IV, results of an empirical evaluation of the proposed scheme are presented. Section V discusses the issue of implementing the proposed scheme with a fewer number of priority levels. The paper concludes with Section VI.

II. System Model and Problem Formulation

We begin by presenting a new deadline model that generalizes the notion of hard and soft deadlines. Consider a stream of real-time customers. A customer may be either a task, a message, or any other schedulable entity in a real-time application. Each customer in the stream has a deadline by which it expects complete service from the system. If the customer is fully serviced before the deadline, the customer is said to have *met* its deadline. Otherwise, the customer is said to have *missed* the deadline. The stream is characterized by two parameters m and k such that at least m customers in any window of k consecutive customers (from the stream) must meet their deadlines. These parameters specify a desired quality of service for the stream. The stream is said to have (m, k)-firm deadlines. When fewer than m customers meet their deadlines in a window of k consecutive customers from the stream, we say that the stream experienced a *dynamic failure*. For a video stream, for instance, these parameters specify the desired video quality and a dynamic failure means that the video quality falls below the desired quality.

The notion of (m, k)-firm deadlines is fairly general. The traditional assumption that every customer from a stream must meet its deadline can be represented as (1, 1)-firm deadlines. A (1, 2)-firm specification corresponds to the constraint that no two consecutive customers from a stream should miss their deadlines. Likewise, in a stream with (4, 6)-firm deadlines, no three consecutive customers should miss their deadlines. Furthermore, at least four customers is any window of six customers must meet their deadlines. Also, note that an (m, k)-firm specification implies a $(k - m)/k$ maximum allowable loss rate, in addition to ensuring that the misses are adequately spaced. For example, a (4, 5)-firm specification implies a 20% maximum allowable loss rate. A (8, 10)-firm specification, which is less stringent than a (4, 5)-firm specification, also implies a 20% maximum allowable loss rate. The traditional soft deadline requirement can be represented as an (m, k)-firm deadline with k very large and $(k - m)/k$ equal to the maximum allowable loss rate.

We consider a system consisting of N streams of real-time customers, $R_1, R_2, \ldots, R_N$. Stream R_j has (m_j, k_j)-firm deadlines. The problem addressed in this paper is to schedule the customers on a single server so as to reduce the average probability of dynamic failure.

A straightforward approach for reducing the probability of dynamic failure is to reduce the probability of a customer missing its deadline. The rationale is that, by reducing the probability of a customer missing its deadline, the probability that at least m_j customers meet their deadlines in a window of k_j consecutive customers is increased. To accomplish this, several different scheduling policies have been proposed [1]. Most of these policies can be abstracted as follows. There is a separate first-In first-out queue for each stream (see Fig. 1). These queues ensure that customers from a given stream are serviced in arrival order. Only the customers at the head of these individual stream queues are candidates for service. The selection among these customers is based on the policy being used. For instance, in the First-In First-Out policy, the selection is based on the arrival times of the customers; a customer with the earliest arrival time is selected. This policy is not cognizant of the deadlines of the customers. A commonly used deadline-cognizant policy is the earliest-deadline-first policy in which the selection is based on the deadlines of the customers; a customer with the closest deadline is selected.

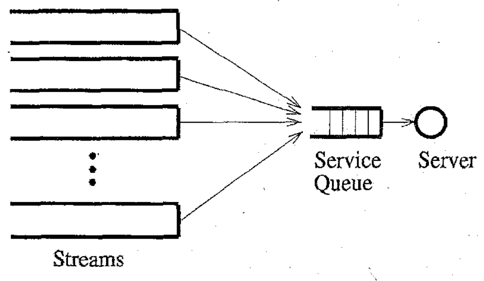


Fig. 1. System model.

The proposed approach can also be implemented using this generic service model. The customers at the head of the stream queues are assigned a priority as described below in Section III. Customers with a higher priority are given preference over customers with a lower priority. Among customers with the same priority, the selection can be based on any criteria used in the conventional approach. Since several studies have shown that the earliest-deadline-first criteria performs fairly well in the conventional approach, the rest of this paper will assume that this criteria is also used in the proposed approach. The key aspect of the proposed scheme is the way in which the priority is assigned to a customer at the head of the stream queue.

Prior to describing the priority assignment policy, there is one point about the service policy which needs further clarification. In some applications, the system can easily determine whether a waiting customer has already missed its deadline. In these applications, the system will not service this tardy customer and we refer to it as a *dropped* customer. However, if the application is such that one cannot determine if a waiting customer has missed its deadline, then the system must service all the customers. In the evaluation presented in Section IV, we consider both of these possibilities. For the next section, it does not matter which of these two possibilities is adopted.

III. Distance-Based Priority Assignment Technique

For each stream, the system maintains a state which captures the recent history of the deadlines met and missed. Let the customers from each stream be numbered 1, 2, 3, ..., in the order of their arrival. A customer is either serviced completely or dropped prior to service. A serviced customer may either *miss* or *meet* its deadline. All dropped customers are considered to have missed their deadlines. Let δ_i^j be the status of the ith customer from stream R_j, $i \geq 1$, where the status of a customer is a *miss* or a *meet*, depending on whether the customer missed or met its deadline.[1] The state of stream R_j at a given time is then the k_j-tuple $(\delta_{i-k_j+1}^j, \ldots, \delta_{i-1}^j, \delta_i^j)$, where i is the index of the most recent customer serviced or dropped from stream R_j. Since δ_i^j is meaningful only for $i \geq 1$, the above k_j-tuple is not well defined for $i \leq k_j$. However, for presentation, it is convenient if the k_j-tuple is defined for all integers i. Therefore, we arbitrarily define δ_i^j to be a miss for $i \leq 0$. Stream R_j can therefore be in one of 2^{k_j} possible states. If stream R_j is in a state with fewer than m_j meets, then it has encountered a dynamic failure. Therefore, states with fewer than m_j meets are considered *failing states* for stream R_j.

A stream gets "closer" to a failing state when its customer misses its deadline. The objective is to prevent streams from going to a failing state. The idea of the proposed approach is as follows. The closer a stream is to a failing state, the higher the priority assigned to its next customer so as to increase its chances of meeting the deadline and thus move the stream away from the failing state. In the proposed approach, the customer is assigned a priority value equal to the minimum number of consecutive misses required to take the stream from the current state to a failing state. The server gives higher priority to customers with lower priority values, i.e., customers with priority value 0 have higher priority than customers with value 1.

Fig. 2 shows the state transition diagram for a stream with (2, 3)-firm deadlines. The letters M and m are used to represent a meet and a miss, respectively. States are denoted by three-letter strings. For example, MMm denotes the state where the most recent customer missed its deadline and the two before that met their deadlines. The edges represent the possible state transitions. Starting from a state, the stream makes a transition to one of two states, depending on whether its next customer meets (denoted by M) or misses (denoted by m) its deadline. For example, if the stream is currently in state MMm and its next customer meets the deadline, then the stream transits to state MmM. The shaded states have less than two meets and are therefore failing states. For this stream, the proposed priority assignment technique works as follows. If the stream is in a failing state (i.e., one of the shaded states in Fig. 2) then its next customer is assigned a priority value of 0, i.e., the highest priority. If the stream is in state MMm or MmM, then the distance to the failing states is one. Therefore, the next customer from the stream gets a priority value of 1. Finally, if the stream is in state mMM or MMM, then the distance to the failing states is two, and the next customer is assigned a priority value of 2. Thus, if a stream is closer to a failing state, its next customer is assigned a higher priority to ensure that it fares well in the competition for service with customers from the other streams.

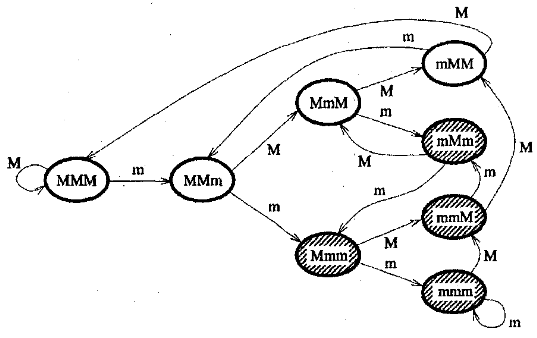


Fig. 2. State transition diagram example.

The proposed distance-based priority (DBP) assignment technique is a way to arbitrate between the streams in a system. The idea is to have streams give up their turn to more urgent streams. When a stream is close to a failing state, its customer is given a high priority so as to increase its chances of meeting the deadline. This is of course done at the expense of other streams that are not as close to a failing state (i.e., can afford a few misses.) However, streams that are in the same situation will not be adversely affected by this high priority customer since their customers will also have the same high priority. In particular, when the miss rates are extremely low (e.g., when the system is lightly loaded), most customers will be competing for service at the lowest priority. When the system is heavily loaded, streams will often be in or very close to a failing state, in which case most customers will be competing at the highest priority. In general, however, few streams will be close to a failing state at any instant. These streams will benefit from the higher priority and the other streams will not be severely affected. Thus, the proposed approach tends to reduce in the probability of dynamic failure.

The DBP scheme is especially beneficial when streams in the system have different deadline requirements. Consider, for example, two streams with (3, 5)-firm deadlines and (9, 10)-firm deadlines, respectively. Suppose that the two streams are otherwise identical.[2] The first stream can tolerate two misses in

1. Note that dropped customers are considered to have missed their deadlines.

2. We say that two streams are identical when they have the same customer service time distribution, the same customer interarrival distribution, and the same customer deadline distribution.

every five consecutive customers—a 40% loss rate. The second stream can only tolerate one miss in every ten consecutive customers—a 10% loss rate. The conventional single priority scheme is oblivious to the individual timing requirements of the streams, and, will therefore result in the same loss rate for both streams. Using the DBP scheme, the stream with the (9, 10)-firm deadlines will usually be given a higher priority since it will usually be closer to a failing state. In particular, note that, for the stream with the (9, 10)-firm deadlines, the distance from a miss-free state to a failing state is two. For the stream with the (3, 5)-firm deadlines, the distance from a miss-free state to a failing state is three. So, even if both streams are in miss-free states, the stream with the (9, 10)-firm deadlines is given a higher priority. As a result, the stream with the (9, 10)-firm deadlines experiences a lower deadline miss rate. Note that this is different from the static approach of giving the stream with the tighter requirements a higher priority. The stream with the (3, 5)-firm deadlines may be given a higher priority if it is judged to be beneficial. For example, when the stream with the (3, 5)-firm deadlines is one miss away from a failing state while the other stream is in a miss-free state, the stream with the (3, 5)-firm deadlines is given a higher priority.

The proposed scheme can be easily implemented in hardware and/or in software. The state of stream R_j can be kept in a k_j-bit shift register. Let 0 and 1 represent a deadline miss and a deadline meet, respectively. When the next customer is serviced, a 0 or a 1 is shifted in (from the right) depending on whether the customer missed or met its deadline. Let $l_j(n, s)$ denote the position (from the right) of the nth meet (or 1) in the state s of stream R_j. If there are less than n 1s in s, then $l_j(n, s) = k_j + 1$. For example, suppose stream R_1 has (1, 3)-firm deadlines. Then, $l_1(1, \text{MmM}) = 1$ and $l_1(2, \text{MmM}) = 3$. For $n > 2$, $l_1(n, \text{MmM}) = k_1 + 1 = 4$. Then, the priority assigned to customer $i + 1$ from stream R_j is given by

$$\text{priority}^j_{i+1} = k_j - l_j(m_j, s) + 1, \qquad (3.1)$$

where $s = (\delta^j_{i-k_j+1}, \ldots, \delta^j_{i-1}, \delta^j_i)$ is the current state of stream R_j. The priority of a customer can also be computed incrementally from the priority of the previous customer from the same stream. If the previous customer missed its deadline, then the next customer is given the next higher priority (i.e., the priority level is decreased by one) unless the previous customer was serviced at the highest priority. If the previous customer met its deadline, the priority of a customer remains the same or is decreased depending on the exact state of the stream. More specifically, in this case, the difference in the priorities of the previous and the current customers from R_j is $l_j(m_j + 1, s) - (l_j(m_j, s) + 1)$, where s is the state of stream R_j.

IV. Evaluation of the DBP Scheme

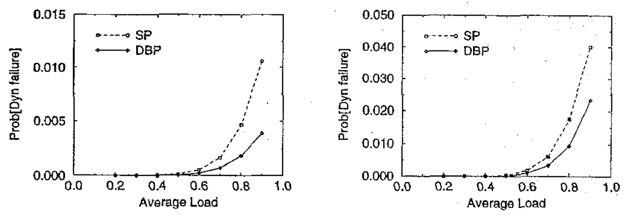
The performance of the proposed approach was evaluated through simulation. The number of streams in the system and the characteristics of each stream are specified to the simulator. The simulator computes the probability of dynamic failure for each stream under both the proposed distance-based priority (DBP) scheme and the conventional single priority (SP) scheme in which all customers are assigned the same priority. In both cases, customers within the same priority level are serviced in the earliest-deadline-first order.

In the results presented here, two customer generation patterns were considered: Poisson and bursty. In a Poisson stream, customer inter-arrival times are exponentially distributed. A bursty source alternates between ON and OFF states. When in the ON state, customers are generated periodically. No customers are generated when the source is in the OFF state. The durations of the ON and the OFF states are exponentially distributed with averages ON_{ave} and OFF_{ave}, respectively. Such a stream is often used to model a stream of voice samples in a conversation [9], [11]. The ON state corresponds to a talkspurt and the OFF state corresponds to a silence period.

We first consider the case where all streams in the system have the same timing requirements. We also assume that only the customers that meet their deadlines are serviced. Later in this section, we will compare the DBP and SP schemes in a system where all customers are serviced, regardless of whether or not they meet their deadlines. We will also consider heterogeneous systems where some streams have more stringent timing requirements than others. The DBP and SP schemes are then compared to an implementation of the imprecise model approach. Finally, we examine the effect of the number of streams in a system on the performance of the DBP scheme.

A. Poisson Streams

The plots in Figs. 3a and 3b show the probability of dynamic failure in two systems with (1, 2)-firm and (3, 4)-firm deadlines, respectively. Each system consists of five streams. All customers require a constant service time. Service deadlines are set equal to five times the customer service time. Customer interarrival times are exponentially distributed and the overall average load is varied from 0.2 to 0.9 by varying the customer arrival rate. The difference between the systems corresponding to Figs. 3a and 3b is in the tolerance to deadline misses. In Fig. 3a, a dynamic failure occurs if two consecutive customers miss their deadlines. The constraints are more stringent in Fig. 3b because three out of four consecutive customers must meet their deadlines. As a result, the probabilities of dynamic failure in Fig. 3b are higher than those in Fig. 3a. In both cases, however, the proposed DBP substantially reduces the probability of dynamic failure. Table I gives the percent reduction with respect to SP at each load. In Fig. 3a, DBP results in probabilities of dynamic failure that are more than 60% lower than those in SP. In Fig. 3b, the reductions are over 40%.



(a) (1, 2)-firm deadlines (b) (3, 4)-firm deadlines

Fig. 3. Probability of dynamic failure in SP and DBP for Poisson streams.

TABLE I
PERCENT REDUCTIONS IN THE PROBABILITY OF DYNAMIC FAILURE OF POISSON STREAMS

	(1, 2)-firm			(3, 4) -firm		
Avg Load	SP	DBP	% reduction	SP	DBP	% reduction
0.2	0.00000	0.00000	–	0.00000	0.00000	–
0.3	0.00000	0.00000	–	0.00000	0.00000	–
0.4	0.00002	0.00000	80.0	0.00007	0.00003	48.5
0.5	0.00014	0.00004	67.6	0.00046	0.00020	56.3
0.6	0.00054	0.00023	58.0	0.00188	0.00102	45.8
0.7	0.00165	0.00071	56.8	0.00608	0.00341	44.0
0.8	0.00466	0.00182	61.0	0.01747	0.00936	46.4
0.9	0.01058	0.00388	63.3	0.04006	0.02319	42.1

B. Bursty Streams

Fig. 4 shows plots of the probability of dynamic failure in a system with five bursty streams. The ON and OFF periods of each stream are exponentially distributed with ON_{ave} = 50 msec and OFF_{ave} = 100 msec. The offered peak load of a stream is therefore three times the average load. When in the ON state, a stream generates one customer every 5 msec. Customer deadlines are set to twice the generation period. The overall load is varied by changing the customer service times. Again, the probabilities of dynamic failure are higher in Fig. 4b than those in Fig. 4a because the tolerance to deadline misses is lower. The reductions in the probability of dynamic failure are more than 95% when the deadlines are (1, 2)-firm (see Table II). In Fig. 4b, the reductions are more modest since the miss tolerance is more stringent.

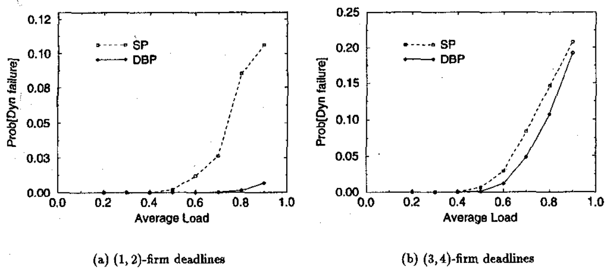


Fig. 4. Probability of dynamic failure in a system with bursty streams.

TABLE II
PERCENT REDUCTIONS IN THE PROBABILITY OF DYNAMIC FAILURE FOR BURSTY STREAMS

	(1, 2)-firm			(3, 4) -firm		
Avg Load	SP	DBP	% reduction	SP	DBP	% reduction
0.2	0.00000	0.00000	–	0.00000	0.00000	–
0.3	0.00000	0.00000	–	0.00000	0.00000	–
0.4	0.00002	0.00000	100.0	0.00011	0.00000	100.0
0.5	0.00005	0.00000	100.0	0.00627	0.00107	82.9
0.6	0.00196	0.00000	100.0	0.02920	0.01184	59.4
0.7	0.01200	0.00017	99.3	0.08376	0.04821	42.4
0.8	0.08507	0.00145	98.3	0.14595	0.10728	26.5
0.9	0.10631	0.00674	93.6	0.20822	0.19283	7.4

Comparing the results in Figs. 3 and 4 we observe that the probability of dynamic failure is higher when customer arrivals are bursty. This is because, at the higher loads, the peak load exceeds one in the bursty case and the system often gets overloaded.

C. No-Drop Policy

In the results presented above, it was assumed that customer service times are known in advance, and so it was possible to drop customers that miss their deadlines. When a customer is ready to be serviced, the system checks if the deadline will be met. If it is determined that the customer will miss its deadline, the customer is not serviced. Dropping such a customer is desirable. However, this is not always possible (e.g., the execution time of a task may not be known a priori). Fig. 5 shows a plot of the probability of dynamic failure in a system like the one examined in Fig. 3 except that all customers are serviced regardless of whether they meet or miss their deadlines. Because missed customers are not dropped, the effective load is higher than before, and as a result, the probabilities of dynamic failure are substantially higher than those in Fig. 3. Here, the reductions are in excess of 80% even at the higher loads.

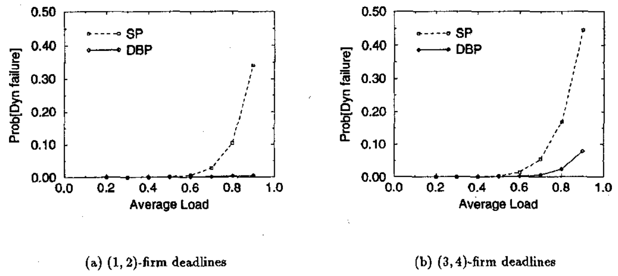
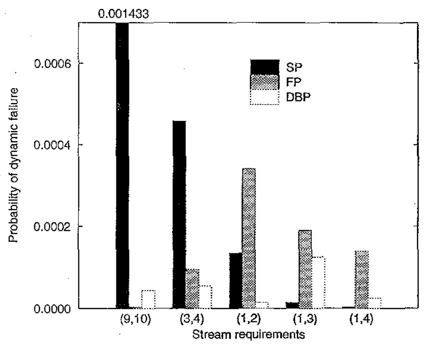


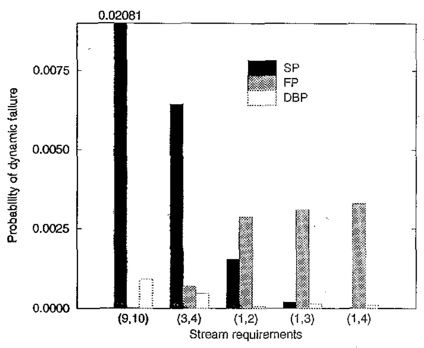
Fig. 5. Probability of dynamic failure in a system with a no-drop policy.

D. Heterogeneous Systems

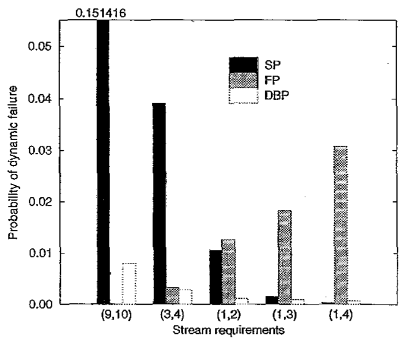
Each system considered so far consisted of streams with the same deadline requirements, i.e., $m_i = m_j$ and $k_i = k_j$ for all $i, j \in \{1, 2, ..., N\}$. Figs. 6a, 6b, and 6c show the results for a heterogeneous system in which streams have different deadline requirements. The system consists of five streams with (9, 10)-firm, (3, 4)-firm, (1, 2)-firm, (1, 3)-firm, and (1, 4)-firm, deadlines, respectively. The customer arrival, service, and deadline patterns in this system are like those for the streams examined in Fig. 3. The arrival rates of the streams are adjusted to get an average load of 0.5, 0.7, and 0.9 for Figs. 6a, 6b, and 6c, respectively. The figures compare the probability of dynamic failure experienced by the five streams in three different scheduling policies: single priority (SP), fixed priority (FP), and the DBP policies. In the SP policy, all customers are serviced in the earliest deadline first irrespective of their stream requirements, whereas in the FP policy, customers have different (fixed) priorities depending on the stringency of the deadline requirements of their streams. In particular, customers from stream with (9, 10)-firm deadlines are given the highest priority, followed by customers from streams with (1, 2)-firm,



(a) Avg. load = 0.5



(b) Avg. load = 0.7



(c) Avg. load = 0.9

Fig. 6. Probability of dynamic failure in a heterogeneous system.

(1, 3)-firm, and (1, 4)-firm deadlines, respectively. Within each priority, the customers are serviced in the earliest deadline first order.

Observe that, since all customers are serviced at the same priority level in the SP scheme, the probability of a customer missing its deadline does not depend on the deadline requirements of the stream. As a result, the stream with (9, 10)-firm deadlines has the highest probability of dynamic failure followed by streams with (3, 4)-firm, (1, 2)-firm, (1, 3)-firm, and (1, 4)-firm deadlines, respectively. In the FP scheme, on the other hand, higher priority is given to customers from streams with less tolerance to deadline misses. Since, the probability of a deadline miss is usually much smaller for streams with higher priority, this means that there are two conflicting factors which determine the probability of dynamic failure of a given stream. As a result, the resulting probability of dynamic failure depends on the balance between these two factors. In the DBP scheme, the priority of a customer depends on the state of its stream. Customers from streams closer to dynamic failure are given a higher priority. Consequently, the DBP scheme tends to have a lower probability of dynamic failure on the average. In particular, the DBP scheme tends to be substantially better than the SP scheme for streams with stringent deadline requirements and, it is slightly worse or comparable for streams which can tolerate a large number of deadline misses. On the other hand, the DBP scheme tends to be better than the FP scheme for streams with more tolerance to deadline misses and worse for streams with less tolerance to deadline misses. Also, the DBP scheme seems to balance in the probability of dynamic failure of streams with widely varying deadline requirements.

E. Comparison to the Imprecise Model Approach

As mentioned earlier in the Introduction, streams with (m, k)-firm deadlines can also be serviced using the imprecise computation model. Recall that, in this model, each customer is tagged as mandatory or optional. Mandatory customers are always serviced. On the other hand, optional customers may be dropped from the system without service. This typically happens when the system is congested, e.g., the queue is longer than a certain threshold. Customers from a stream with (m, k)-firm deadlines can be tagged in a way such that there are at least m mandatory customers in every k consecutive customers from the stream.

Reconsider the system examined earlier in Fig. 5b. An implementation using the imprecise computation model is compared to SP and the proposed DBP scheme in Fig. 7, where the probability of dynamic failure and the probability of a customer missing its deadline are plotted for three different loads. In this case, since the deadlines are (3, 4)-firm, every fourth

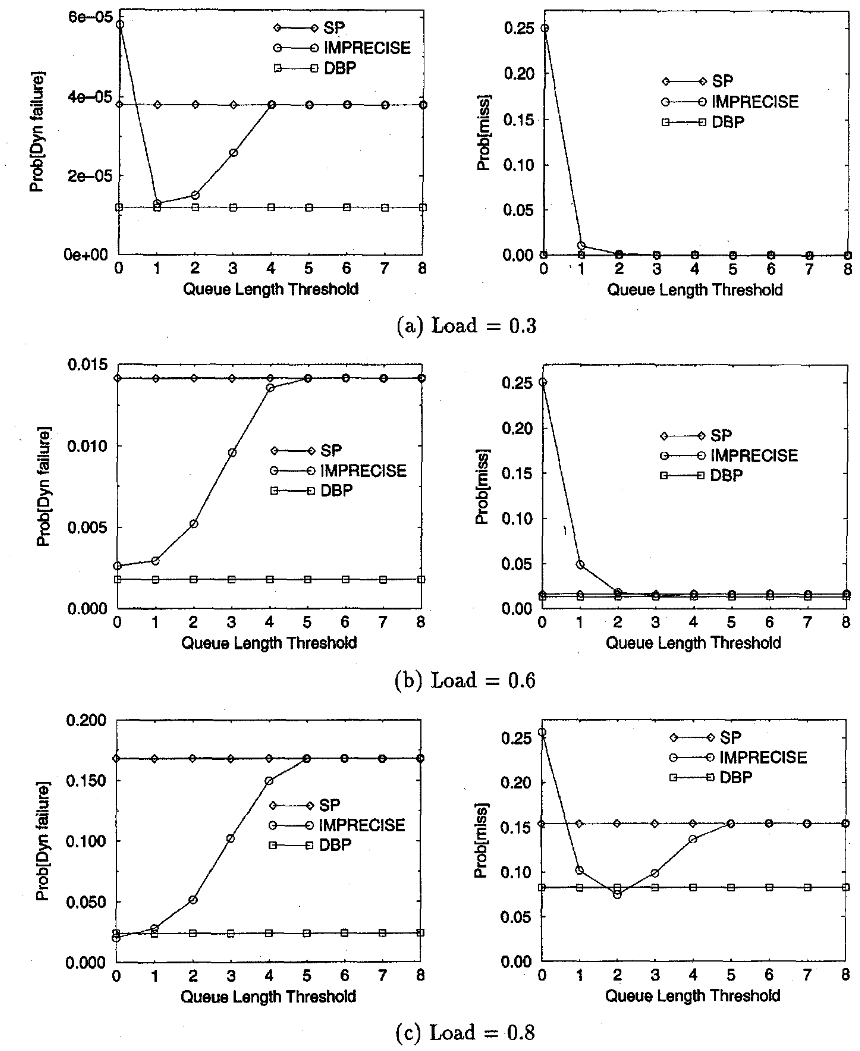


Fig. 7. SP vs. DBP vs. imprecise model approach at three different loads.

customer from a stream is tagged as optional. An optional customer is queued for service only if the total number of customers already in the queue is below a threshold *T*. Note that, when $T = 0$, all optional customers are dropped. At the other extreme, when $T = \infty$ (or large enough), all optional customers are serviced, and the scheme becomes equivalent to the conventional SP. In Fig. 7, the threshold *T* is varied from 0 to 8. Note that, the queue length threshold is a parameter of the imprecise approach only. The schemes SP and DBP are insensitive to this parameter; they are plotted along the same axis for ease of comparison only.

We observe that, with a proper queue length threshold, the imprecise model approach achieves a probability of dynamic failure that is comparable to that achieved by the proposed DBP scheme. For example, at higher loads, a $T = 0$ results in a very low probability of dynamic failure. However, this reduction is achieved at the expense of a dramatic increase in the overall probability of a customer missing its deadline. In this case, for example, dropping every customer implies a miss probability of at least 0.25. In contrast, the DBP scheme reduces the probability of dynamic failure without greatly affecting the overall deadline miss rate. Another problem with the imprecise model approach is that the optimal queue length threshold depends on the load and other parameters. In this example, $T = 0$ minimizes the probability of dynamic failure at higher loads. However, at a load of 0.3, $T = 0$ results in a probability of dynamic failure that is even higher than that of the SP scheme.

F. Effect of Number of Streams

The proposed approach reduces the probability of dynamic failure by prioritizing the streams (or their customers) based on the states of the streams. Therefore, the reductions depend on the number of streams in a system. In particular, if only one stream exists, then DBP and SP are equivalent. To study the effect of the number of streams on the performance of DBP, we varied the number of streams and adjusted the arrival rates to keep the overall average load constant. Plots of the probability of dynamic failure versus the number of streams for an overall average load of 0.7 are shown in Fig. 8. The customer interarrival times are exponentially distributed and all customers have identical service times. Also, the streams have (1, 2)-firm deadlines. In Fig. 8a, all customers are serviced regardless of whether they miss or meet their deadlines. In Fig. 8b, only the customers that meet their deadlines are serviced. With only one stream, SP and DBP result in the same probability of dynamic failure. The percent reduction in the probability of dynamic failure increases as the number of streams in the system increases. It should be noted however that a substantial reduction is achieved even with only three streams in the system. In particular, when all customers are serviced, the reduction is over 70% even with only three streams in the system.

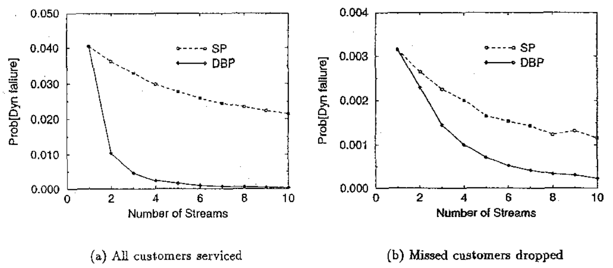


Fig. 8. Probability of dynamic failure vs. the number of streams.

V. LIMITED NUMBER OF PRIORITIES

The proposed DBP scheme assigns priorities to customers based on the state and the requirements of the corresponding stream. The number of distinct priorities that can be assigned to customers from a given stream depends on the tightness of the requirements of that stream. For example, customers from a stream with (2, 5)-firm deadlines can be assigned a priority 0, 1, 2, 3, or 4. Those from a stream with (4, 5)-firm deadlines can be assigned a priority 0, 1, or 2. In general, a customer from stream R_j with (m_j, k_j)-firm deadlines can be assigned any one of $k_j - m_j + 2$ priority levels. The system must therefore support

$$P = \max\{k_j - m_j + 2 : j = 1, 2, \ldots, N\} \qquad (5.1)$$

distinct priority levels, where N is the number of streams in the system. In practice, however, there is usually a limit on the number of priority levels a system can efficiently support. Let P_{max} be the maximum number of priority levels the system can support. (0 and $P_{max} - 1$ are the highest and the lowest priority levels, respectively.) In this section, we evaluate the performance of the proposed DBP scheme when $P_{max} < P$.

We adopt the approach of truncating the priorities at the lowest priority level $P_{max} - 1$. In other words, if the distance from the current state of a stream to a failing state is greater than $P_{max} - 1$, then its next customer is assigned the lowest priority, $P_{max} - 1$. More precisely, the priority value assigned to customer $i + 1$ from stream R_j is

$$\text{priority}_{i+1}^{j} = \min\{k_j - l_j(m_j, s) + 1, P_{max} - 1\} \qquad (5.2)$$

where $s = (\delta_{i-k_j+1}^{j}, \delta_{i-k_j+2}^{j}, \ldots, \delta_{i}^{j})$ is the current state of stream R_j and the function l_j is as defined at the end of Section III. Recall that the rationale behind the dynamic priority assignment scheme is to arbitrate between the streams in the system based on how close they are to a failing state. Giving a customer from a stream that is close to a failing state a high priority is an attempt to move the stream away from the failing state. The idea is then to use the few high priorities available to help the streams that need it the most—those closest to a failing state.

Consider a stream with (2, 5)-firm deadlines. In this case, the priority assigned to a customer is 0, 1, 2, 3, or 4. The system must therefore support five priority levels to fully implement the DBP scheme. Fig. 9 shows plots of the dynamic failure in a system consisting of five streams with (2, 5)-firm deadlines as a function of the maximum number priorities P_{max}. The customer interarrival times are exponentially distributed and all customers have identical service times. The interarrival rates are adjusted such that the average load is 0.8. The deadline of a customer is five times its service time. In Fig. 9a, a customer is serviced regardless of whether or not the customer meets its deadline. In Fig. 9b, a customer that is deemed to miss its deadline is dropped and not serviced. As expected, the probability of dynamic failure depends on the number of priorities available. When $P_{max} = 1$, all customers are serviced at the same priority level, i.e., using the conventional single priority scheme. At the other extreme, when $P_{max} = 5$, the DBP scheme is fully implementable. As P_{max} is increased from 1 to 5, the probability of dynamic failure is reduced. Note, however, that there is a substantial reduction even with only three priority levels. With $P_{max} = 3$, customers from streams already in a failing state are assigned the highest priority, 0. The second highest priority is assigned to customers from streams that are only one deadline miss away from a failing state. All other customers are serviced at the lowest priority level, 2.

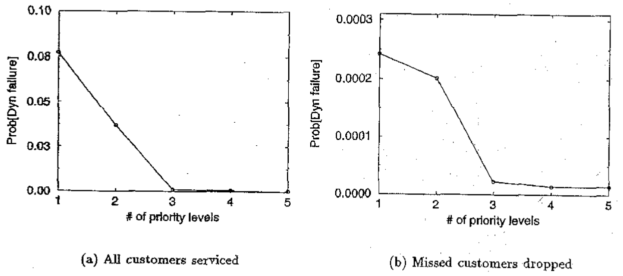
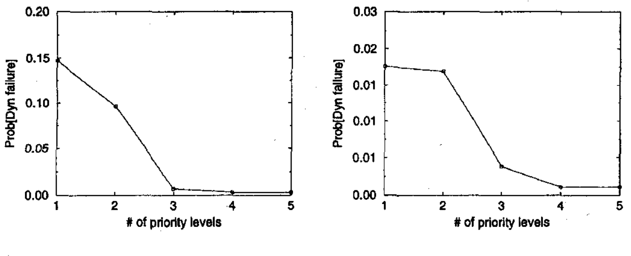


Fig. 9. Probability of dynamic failure vs. P_{max}.

Fig. 10 shows plots of the dynamic failure in a heterogeneous system as a function of P_{max}. The system consists of five streams with (9, 10)-firm, (3, 4)-firm, (1, 2)-firm, (1, 3)-firm, and (1, 4)-firm deadlines. The customer interarrival times are exponentially distributed and all customers have identical service times. The interarrival rates are adjusted such that the average load is 0.8. The deadline of a customer is five times its service time. Note that, in this case, the total number of priority levels, $k_j - m_j + 2$, is not the same for all the streams. For example, customers from the stream with (9, 10)-firm deadlines are assigned priorities 0, 1, or 2, whereas customers from the stream with (1, 4)-firm deadlines are assigned priorities 0, 1, 2, 3, or 4. As in the previous system, the proposed DBP scheme is fully implementable when $P_{max} = 5$. Here also, the reductions are substantial for $P_{max} = 3$.



(a) All customers serviced (b) Missed customers dropped

Fig. 10. Probability of dynamic failure vs. P_{max} in a heterogeneous system.

VI. Conclusion

Real-time customers have deadlines associated with them. Depending on the application, these deadlines are usually specified as hard (i.e., each customer must meet its deadline) or soft (i.e., misses can be tolerated as long as the miss rate is below some specified threshold). These models are inadequate to capture the requirements of many applications where hard deadlines would be too restrictive and soft deadlines too lax. This paper introduced a deadline model which generalizes the notion of hard and soft deadlines. In this model, a stream has two parameters m and k such that a dynamic failure occurs if less than m out of k consecutive customers meet their deadlines. The requirements of real-time applications can be more precisely expressed using the proposed deadline model.

The paper then proposed a service policy to reduce the probability of dynamic failure. The idea is to assign priorities to customers based on the recent history of the source stream. A customer from a stream that is close to a failing state (i.e., suffered too many recent misses) is assigned a higher priority so as to improve its chances of meeting the deadline. The approach was compared to a conventional approach where all customers are serviced at the same priority level. The approach was also compared to the imprecise computation model approach. Empirical results show that the proposed dynamic priority assignment technique results in substantial reductions in the probability of dynamic failure without greatly affecting the overall probability of miss.

Acknowledgment

The work reported here is supported in part by the National Science Foundation under Grant No. MIP-9213716. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation. A preliminary version of this paper appeared in the *Proceedings of the 24th Annual International Symposium on Fault-Tolerant Computing*, June 1994.

References

[1] C.M. Aras, J.F. Kurose, D.S. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *IEEE Proc.*, vol. 82, pp. 122–139, Jan. 1994.

[2] J.J. Bae and T. Suda, "Survey of traffic control schemes and protocols in ATM networks," *IEEE Proc.*, vol. 79, pp. 170–189, Feb. 1991.

[3] R. Chipalkatti, J.F. Kurose, and D. Towsley, "Scheduling policies for real-time and non-real-time traffic in a statistical multiplexer," *IEEE INFOCOM*, pp. 774–783, Apr. 1989.

[4] J.Y. Chung, J.W.S. Liu, and K.-J. Lin, "Scheduling periodic jobs that allow imprecise results," *IEEE Trans. Computers*, vol. 39, no. 9, pp. 1,156–1,174, Sept. 1990.

[5] K. Kubota, M. Murata, H. Miyahara, and Y. Oie, "Congestion control for bursty video traffic in ATM networks," *Electronics and Comm. in Japan*, vol. 175, pp. 13–19, Apr. 1992.

[6] J.W.S. Liu, K.-J. Lin, C.L. Liu, W.K. Shih, and J.Y. Chung, "Imprecise computations: A means to provide scheduling flexibility and enhance dependability," *Readings in Real-Time Systems*, Y.-H. Lee and C.M. Krishna, eds., ch. 2, pp. 81–97, IEEE CS Press, 1993.

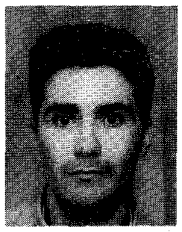
[7] J.W. S. Liu, K.-J. Lin, W.-K. Shih, A.C. Yu, J.-Y. Chung, and W. Zhao, "Algorithms for scheduling imprecise computations," *Computer*, vol. 24, no. 5, pp. 58–68, May 1991.

[8] D.W. Petr and V.S. Frost, "Optimal packet discarding: An ATM-oriented analysis model and initial results," *IEEE INFOCOM*, pp. 537–542, June 1990.

[9] M.A. Saleh, I.W. Habib, and T.N. Saadawi, "Simulation analysis of a communication link with statistically multiplexed bursty voice sources," *IEEE J. Selected Areas in Comm.*, vol. 11, no. 3, pp. 432–442, Apr. 1993.

[10] H. Schulzrinne, J.F. Kurose, and D. Towsley, "Congestion control for real-time traffic in high-speed networks," *IEEE INFOCOM*, pp. 543–550, June 1990.

[11] K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data," *IEEE J. Selected Areas in Comm.*, vol. 4, no. 6, Sept. 1986.



Moncef Hamdaoui (S'88) received the BS (with highest distinction), MS, and PhD degrees in electrical and computer engineering from the University of Wisconsin at Madison, in 1988, 1990, and 1994, respectively. He is currently working for Northern Telecom in Tunis, Tunisia. His research interests include real-time systems, communication networks, and fault-tolerant systems.

Parameswaran Ramanathan (M'89) received the BTech degree from the Indian Institute of Technology, Bombay, India, in 1984 and the MSE and PhD degrees from the University of Michigan, Ann Arbor, in 1986 and 1989, respectively. From 1984 to 1989 he was a research assistant in the Department of Electrical Engineering and Computer Science at the University of Michigan. At present, he is an associate professor in the Department of Electrical and Computer Engineering and in the Department of Computer Sciences at the University of Wisconsin, Madison. His research interests include the areas of real-time systems, fault tolerant computing, distributed systems, and parallel algorithms.