A CLASS OF NON-PREEMPTIVE SCHEDULING ALGORITHMS FOR REAL-TIME SYSTEMS

Dan C. Marinescu

CSD-TR-850
January 1989

A CLASS OF NON-PREEMPTIVE SCHEDULING ALGORITHMS FOR REAL-TIME SYSTEMS

Dan C. Marinescu

Computer Sciences Department
Purdue University
West Lafayette, IN 47907

ABSTRACT

We introduce a class of scheduling algorithms, based upon the *estimated extinction time*, defined as the latest time a a task must begin its service in order to meet its deadlines in a real time system. At time t the algorithms use a scheduling window which covers the interval $[\,(t+g(t)),\ (t+g(t)+W_0)\,]$, and all tasks with estimated extinction time within this window are eligible for selection. The task with the earliest estimated extinction time is selected for execution. The algorithms use an estimate of the *gap at the time of completion of all tasks within the window*, computed from the current gap, g(t) minus the estimated execution time of tasks within the window to decide whether the system is in a safe state and rejects any new task which would lead to an unsafe state of the system.

1. OVERVIEW

Real-time computing has important applications ranging from industrial control of nuclear power plants to control of experiments in various research fields, from microprogramming to aircraft avionics, from traffic control to military applications. In all these case there are some computing resources used by tasks with real-time constraints. A scheduling algorithm has the objective to ensure that all tasks which compete for system resources meet their deadlines. Whenever the deadlines are not met the system experiences a failure.

In general a real-time system has to respond to external events and the occurence time as well as the processing time of these events are non-deterministic in nature, and as a result, the system will occasionally fail. If we accept the possibility of such failures then the objective of a scheduling strategy becomes to minimize the number of cases when tasks fail to meet their deadlines.

In this paper we discuss non-deterministic scheduling algorithms for real-time systems serving a mix of tasks, some with deadlines to the beginning of service and some with deadlines to the completion of service. Some of the tasks may have in addition to a deadline some form or another of priorities associated with their execution. We propose to treat all these cases uniformly by determining an *estimated extinction time* defined as the latest time a task with a deadline and priority must begin its execution, and then to schedule the task according to some policy based upon its estimated extinction time.

Scheduling policies for tasks with deadlines to the beginning of service have been investigated by Baccelli [1], Jackson [2], and more recently by Panwar et al [5]. Queues with customer deadlines to the beginning of service are generally used to model issuing policies for perishable inventory or for modeling communication systems with time delivery constrains as those in integrated voice, image, data networks. It has been shown that the *Shortest Time to Extinction*, STE, and *Shortest time to Extinction with inserted Idle time*, STEI, are optimal scheduling policies for such systems, [5].

Jackson has studied systems with tasks with given due dates [2]. Such queues are used to model robotic systems, manufacturing systems, etc. Jackson has shown that *Earliest Due Date* policy, EDD, is optimal in this case.

There are no known results concerning tasks with deadlines and priorities. We have studied a class of systems with a mix of tasks with deadlines and tasks without deadlines [3]. In such systems tasks are scheduled according to a priority assigned function of the deadline. We have investigated the impact of critical sections upon preemptive priority scheduling in such "semi-hard real time systems".

More recently we have proposed a random multiple access algorithm for communication with real-time delivery constraints, [4]. In this algorithm we use a modified extinction time to schedule transmission of data packets. The RTD algorithm schedules packet transmission in this distributed queuing environment according to a STE strategy. All stations are able to compute the position of a current window using the feedback received from the channel, collision, idle slot or success. All stations with packets with modified extinction time within the window are allowed to transmit in the next slot. If a collision occurs the window is split in half and the left sub-window becomes current window. The algorithm impose a minimum acceptable deadline, Δ and rejects any request with a deadline smaller than Δ.

Based upon the analysis of the RTD splitting algorithm we have understood that window scheduling algorithms provide an interesting alternative to explore and that the *gap* defined as the interval from the current time to the left margin of the window is a good synthetic measure of the system load, it gives indications when deadlines will not be met. The system starts missing deadlines when the gap becomes zero. We have also realized that in order to avoid missing deadlines it is necessary to keep this gap larger than a minimum value function of the actual parameters of the system. In other words it is necessary to make scheduling decisions well in advance of the actual deadline, to impose a safety margin. To control the system and keep it within a desired operational region we have to occasionally reject new load which would cause the gap to shrink beyond acceptable limits. The gap is used to establish whether the system is in a "safe" state or not.

These ideas form the basis of the scheduling algorithms proposed in this paper. We introduce successively ESTEI and ESTEW policies and we conjecture that they are optimal scheduling policies for the mix of tasks presented above. There we describe non-preemptive scheduling algorithms based upon these policies, and give a justification for the algorithms, We introduce the concept of a "safe" state of the system as a state when we can determine with a certain level of confidence that the system will be able to meet the deadlines of the tasks accepted by the system. We define a scheduling window positioned in the future and every time a scheduling decision is made, we use a shortest time to extinction within this window scheduling policy to determine which task is to be executed next. The relative position of this scheduling window is used to asses whether the system is in a safe state or not. When a task first arrives, a decision to reject the task is made if its deadline is earlier than a minimum acceptable deadline, or if the system is in an "unsafe" state.

2. NON-DETERMINISM AND SCHEDULING WITH TIMING CONSTRAINTS

In the following, we consider real-time systems which process tasks with deadlines associated with their execution. A task is characterized by a tuple $r_i = (t_i, d_i, s_i, q_i)$ with t_i the arrival time of the task r_i to the system, d_i its deadline, s_i the service time and q_i an indicator, $q_i = 0$ if the deadline is to the beginning of service and $q_i = 1$ if the deadline is to the completion of service, and $q_i = p$, $p > 1$ to specify a priority, p. Whenever the type of deadline is explicitly defined and no priority is involved, the task will be specified only as $r_i = (t_i, d_i, s_i)$.

In case of tasks with deadlines to the beginning of service, it is required that service starts by the *extinction time* defined as $e_i = t_i + d_i$. The arrival time, t_i as well as the deadline, d_i are known at the time a scheduling decision is made, while the service time is known only in the deterministic case. In general, the service time s_i is not known, only the distribution function of the service time is known. It follows that the completion time $c_i = e_i + s_i$ can only be estimated at the time when the scheduling decision is made.

When the task r_i has a deadline to the completion of service, this means that the service must be complete by the time c_i defined as $c_i = t_i + d_i$. Consequently, the service must start by the estimated extinction time $e_i' = c_i - s_i = t_i + d_i - s_i$.

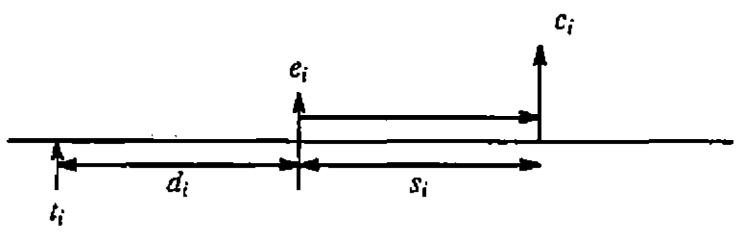

Figure 1a. A request $r_i = (t_i, d_i, s_i)$ with the deadline to the beginning of service. The extinction time is $e_i = t_i + d_i$.

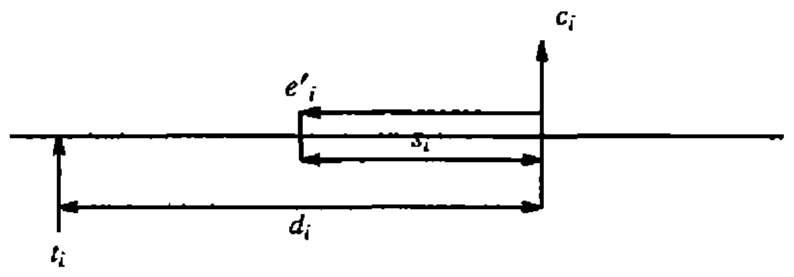

Figure 1b. A request r_i with deadline to the completion of service. Its estimated extinction time is $e'_i = t_i + (d_i - s_i)$.

When the processing of a task does not meet the required deadline, the task is considered lost. Due to the statistical nature of the arrival process $\mathcal{A}$, of the deadline process $\mathcal{D}$, and of the service process $\mathcal{S}$, tasks will occasionally be lost.

It is important to note the implication of the type of deadline for scheduling. Given a task r_i with a deadline to the beginning of service, it is guaranteed that the task will not be lost if it will start service at a time t^{start} which satisfies the condition $t^{start} \leq e'_i$. In case of a deadline to the completion of service even when the $t^{start} < e_i$, there is no guarantee that the deadline will be met, since in general, we can only estimate e'_i; the service time is not known at the time a scheduling decision is made.

A scheduling algorithm may attempt to optimize different performance measures e.g. response time, waiting time, turnaround time, which are measures of the quality of service, or throughput, server utilization, which are measures of the quantity of service. A characteristic curve of the algorithm defines the relationship between the quality and the quantity of service.

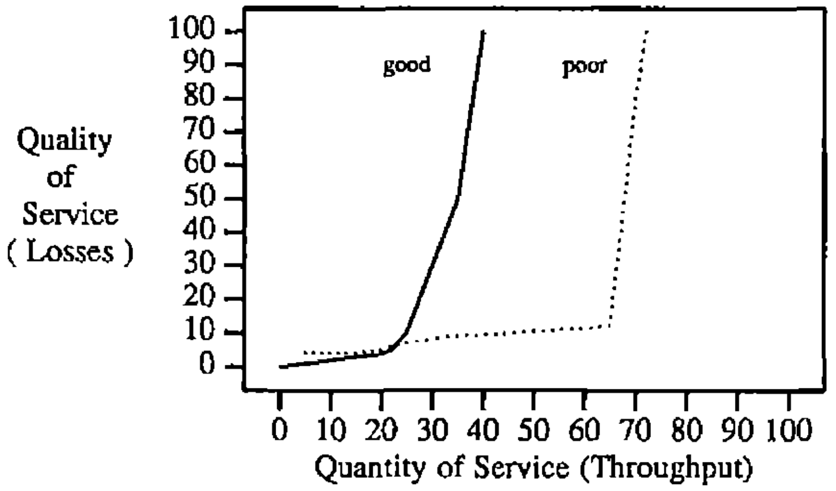

Figure 2a. The characteristic curves of two scheduling algorithms for real-time systems.

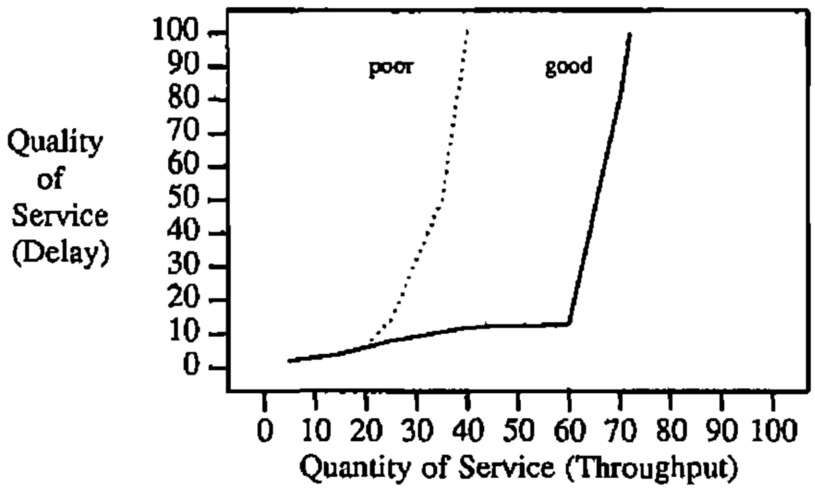

Figure 2b. The characteristic curves of two scheduling algorithms for non RT-systems.

In Figure 2b we show two such curves for scheduling algorithms for multiuser systems. The solid curve corresponds to a good scheduling algorithm which ensures an acceptable delay for a relatively high throughput, while the dotted curve corresponds to a poor algorithm. In the second case the delay becomes very large for a much lower throughput than in the first case. Informally we say that the system is stable if the delay is finite. An actual operational region of the system is a subset of the stability region determined from conditions that the quality of service satisfies preestablished standards. In Figure 2a we show the same curves for a real-time system. to point out two major differences from the previous case. First, in a real-time system the quality of service is characterized by the percentage of losses, and it is desired to keep this level as low as it can possibly be. As a result we expect a narrower operational region and lower throughput. Second, the concept of stability is no longer well defined.

3. OPTIMAL SCHEDULING POLICIES FOR A MIX OF TASKS, SOME WITH DEADLINES TO THE BEGINNING AND SOME WITH DEADLINES TO THE COMPLETION OF SERVICE

An early result concerning scheduling policies for systems accepting tasks with deadlines to the completion of service was obtained by Jackson [2], who has proposed the Earliest Due Date, EDD scheduling policy. Given n tasks $r = \{r_1, \ldots, r_n\}$ with the due dates $\{a_1, \ldots, a_n\}$, let $c_{i,\pi}$ be the finishing time of r_i under policy π. Define the *lateness or* r_i *under* π by $(c_{i,\pi} - a_i)$ and the *tardiness of r under* π by $\max\{0, c_{i,\pi} - a_i\}$. Jackson has shown that a policy π which schedules the tasks in the order of non-decreasing due dates minimizes the maximum lateness and the tardiness.

Shortest Time to Extinction, STE is a scheduling policy for tasks with deadlines to the beginning of service. It schedules the tasks in the order given by their extinction time, but it does not schedule tasks which have already exceeded their deadline.

Very recently [5], it has been shown that STE policies are optimal for a class of non-preemptive M|G|1 + G queues, which do not allow unforced idle times and that when enforced idle times are allowed, the optimal scheduling policies belong to the class of Shortest Time to Extinction with inserted Idle time, STEI for any G|G|1 + G queue. Such policy forces the server to be idle, even when the system is not empty, but the extinction time of the tasks present in the system are far away, based upon the idea that if such a task r_i is scheduled for service, it will force the system to miss the deadline of another task say r_j, arriving to the system after r_i begins service, and having a deadline before r_i completes its service.

We propose a class of scheduling policies for a mix of tasks with deadlines to the beginning and to the completion of service. The *Extended Shortest Time to Extinction with inserted Idle time*, **ESTEI**, policy is based upon the following concepts:

1. For each task $r_i = (t_i, d_i, s)$ with deadlines to the completion of service compute an estimate with confidence α, $s^{(\alpha)}$ of its execution time $e_i^{(\alpha)} = t_i + d_i - s^{(\alpha)}$. Treat r_i as a task with a deadline to the beginning of service with extinction time $e_i^{(\alpha)}$.
2. At time t, define a "scheduling horizon" $H_{(t)} > t$.
3. Consider only tasks r_i with extinction time e_i or modified extinction time $e_i^{(\alpha)}$ within the scheduling window $t \leq e_i \; or \; e_i^{(\alpha)} \leq H_{(t)}$. Schedule them using a STE, Shortest Time to Extinction policy.
4. Do not schedule a task if its extinction time or estimated extinction time has passed. If $e_i < t$ or if $e_i^{(\alpha)} < t$), consider r_i to be lost.

To estimate the service time $s^{(\alpha)}$, consider that the mean value μ_s and the standard deviation σ_s of the random variable s are known. Then the coefficient of variation of s, C_s is defined as $C_s = \frac{\sigma_s}{\mu_s}$. From Chebyshev inequality, it follows that for any $\alpha < 1$, $P\{s : s \geq s^{(\alpha)}\} \leq \alpha$ with

$$s^{(\alpha)} = \mu_s \left[1 + C_s \sqrt{\frac{1-\alpha}{\alpha}} \right]$$

For example, if we consider an exponential distribution of the service time, hence $c_s = 1$, then for $\alpha = 0.05$, $s^{(0.05)} = 5.36\mu_s$ and for $\alpha = 0.2$, $s^{(0.2)} = 3\mu_s$.

It is important to note that ESTEI policies can accommodate priority scheduling as well, using the following idea. If a task t_i with extinction time e_i has a priority p_i, then compute a modified extinction time $e_i^p = f_{(e_i, p_i)} < e_i$ and use e_i^p for scheduling based upon an ESTEI policy.

Note that better estimates of the execution time may be derived whenever the distribution function of the service time is known.

4. TASK ACCEPTANCE BASED UPON THE CONCEPT OF A "SAFE" STATE. STEW SCHEDULING POLICIES

We propose scheduling policies for RT-systems based upon the idea that tasks likely to miss their deadlines, should be rejected upon their arrival in the system, if the system is already in an "unsafe" state or if the acceptance of the task will lead to an "unsafe" state. Informally we define a state to be safe if we can determine with a desired level of confidence, that all the tasks accepted for processing will be processed before their deadline expire. For example, if $r_i = (t_i, e_i, d_i)$ and $r_j = (t_j, e_j, d_j)$ are the only tasks in the system and both have deadlines to the beginning of service, and if $e_i < e_j$ then the probability that r_j will be lost when service for r_i starts precisely at time e_i is at most 0.05 iff $e_j - e_i \geq s^{(0.05)}$. If service for r_i starts at time $t_i^{start} < e_i$, then the probability that r_j will be lost decreases even further.

To determine whether the system is in a safe state, we propose a class of window-based scheduling policies. At time t, define a scheduling window positioned in the future, as shown in Figure 3. Call $g_{(t)} = T_{(t)} - t$ the current "scheduling gap". Define the system to be in a safe state if $g_{(t)} > g_{\min}$ with $g_{\min} > 0$ a parameter of the algorithm to be determined from optimization conditions related to the system throughput and from the level of quality of service imposed upon the system.

Apply some scheduling policy, e.g. STE, to the tasks with extinction or estimated extinction time within the window, $T_{(t)} \leq e_i^{(\alpha)} \leq T_{(t)} + W_{(t)}$. Every time a scheduling decision is made select the task with the shortest time to extinction as the next runnable task and compute the "estimated gap at the time of completion of all tasks within the window", $g_{(t+s^{(\alpha)})}$, assuming that the task is accepted. If $g_{(t+s^{(\alpha)})} > g_{\min}$ declare the system to be in a safe state and accept the task. Note that $g_{\min}$, W and T are parameters of the algorithm to be determined by an optimization analysis. Note also that the actual gap at the time of completion of r_i, $g_{(t+s_i)}$ will become the scheduling gap for the next scheduling decision. This policy is called *Shortest Time to Extinction within the Window*, STEW. STEW is a window based scheduling policy using the STE strategy.

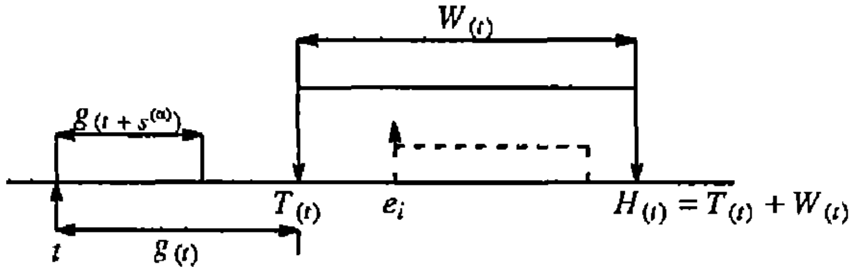

Figure 3.

It is intuitively clear that the sooner we are able to make a scheduling decision, the less likely is the possibility of a loss. The "scheduling gap" is a good measure of our flexibility in making scheduling decisions while the "estimated gap after completion" is a good synthetic measure of the system load and can be used to determine an optimal functioning point on the characteristic curves of the system. We point out that a system may move from a safe to an unsafe state, even in absence of task arrival, because the service time of the tasks already accepted in the system are larger than their estimated value. From an unsafe state, the system may either recover and move to a safe state, or may move to an undesired state when it misses deadlines.

5. SCHEDULING ALGORITHMS BASED UPON THE ESTEW POLICIES

An ESTEW scheduling policy combines the ESTEI policy which allows scheduling of a mix of tasks, some with deadlines to the beginning, some with deadlines to the completion of service, possibly taking into account task priorities with the STEW policies, which perform scheduling following an ESTEI policy for all tasks with extinction times within a scheduling window. The scheduling algorithm operates as follows.

1. The scheduling process has two phases, an "acceptance phase" and a "scheduling phase". The first one is triggered every time a task arrives to the system. Then a decision whether to accept or to reject the task is made. The second phase acts every time a task completes its service or whenever a new task is accepted into the system and the server is idle.
2. The "acceptance phase" makes a decision whether to accept or to reject an incoming task. As a first step the extinction time is estimated. This estimation is based upon knowledge of the distribution function of the service time, the level of confidence, α and possibly upon the priority of the task. Clearly the extinction time of requests with deadline to the beginning of service are known when the task arrives into the system. The acceptance is based upon two tests.

   2a. *The minimum deadline requirement test.* A minimum acceptable deadline, Δ is defined and for every task $r_i = (t_i, d_i, s_i, 0)$ it is checked that $d_i \geq \Delta$. For a task $r_i = (t_i, d_i, s_i, 1)$ the test is $d_i - s^{(\alpha)} \geq \Delta$. If the above conditions are satisfied, the task is marked as acceptable and a second test is made.

   2b. *The safe state test.* If the server is idle, the current scheduling gap $g_{(t)}$ is used to compute the estimated gap at time of completion r_i, $g_{(t+s^{(\alpha)})}$. If $g_{(t+s^{(\alpha)})} > g_{\min}$ the task t_i is accepted. If server is busy, the estimate of the gap at the time of completion of all tasks within the window, $g^{w}_{(t)}$ is used instead of $g_{(t)}$ for the test.
3. The scheduling phase implements an ESTEW policy. It considers a scheduling window (see Figure 3) with the initial position $T_{(t)} = \Delta$ and $W_{(t)} = W_0$. If r_i is the task with the earliest extinction time, it schedules r_i for execution. If two tasks r_i and r_j have the same extinction time, say e_i, then it defines an interval $(e_i - \varepsilon, e_i)$ and redistributes uniformly the extinction times over this interval. If c_i is the completion time of r_i, then at time $t + c_i$ the scheduler probes again the same scheduling window, which now is at a distance $g(t + c_i) = g_{(t)} - s_i$ and repeats the ESTEW decision. This process continues until at time $(t + \beta)$ the scheduler finds the window empty. At that time, a new window is defined such that $T_{(t-\beta)} = \min[(t + \beta + \Delta), (T_{(t)} + W_0)]$.

6. A JUSTIFICATION FOR REJECTION OF TASKS WHICH DO NOT SATISFY THE MINIMUM ACCEPTABLE DEADLINE REQUIREMENT

We conjecture that the ESTEI and ESTEW policies are optimal scheduling policies for G|G|1 + G systems. As shown earlier ESTEI are based upon STEI policies, which are optimal scheduling policies for tasks with deadlines to the beginning of service. For tasks with deadlines to the completion of service ESTEI implements in fact, an EDD policy which again is optimal.

It is more difficult to justify ESTEW policies which are based upon the idea of initial rejection of tasks. It seems counter intuitive to consider that a policy which rejects tasks when their deadlines are too close to their arrival in the system or when the system is in an unsafe state, could be optimal in a stochastic system which has the objective to minimize losses. The intuitive arguments in favor of initial rejection of tasks are the following.

(a) It is unwise to waste system resources by starting processing of a task r_i with deadlines to the completion of service when we know that the deadline will not be met. If we reject such a task upon its arrival in the system, we have extra capacity to process other tasks.

(b) From the system engineering point of view, the tasks rejected initially could be re-routed to a standby system, which might be able to process them observing their deadline. Considering the typical applications of real-time systems we expect such standby resources to be present in order to increase the reliability of the system. If we choose a proper operating region for the system, the task rejection rate will be fairly low.

We have reasons to believe that STEW scheduling policies, Shortest Time to Extinction within the Window policies are optimal at least for systems which can be modeled as load dependent service rate systems.

In [4] we describe a random multiple access splitting algorithm for real-time communication. In this algorithm all stations connected to a broadcast channel compute!! the current position of a scheduling window located in the future and stations with data packets with extinction time within the window are allowed to transmit in the current slot.

We have performed extensive simulation experiments upon the RTD algorithm. In Figure 4 we show the characteristic curve of the system. We see the effect of the minimum acceptable deadline, Δ. The larger is Δ the smaller is the number of losses due to packets which miss their deadline. Figure 5 shows the effect of the window size upon losses due to packets missing the deadlines. We observe an optimal window size of about twice the expected transmission time of a packet.

Finally in Figure 6, we see that total losses, namely losses due to initial rejection and those due to missing deadlines, indicate a minimum for an optimal value of the minimum acceptable deadline.

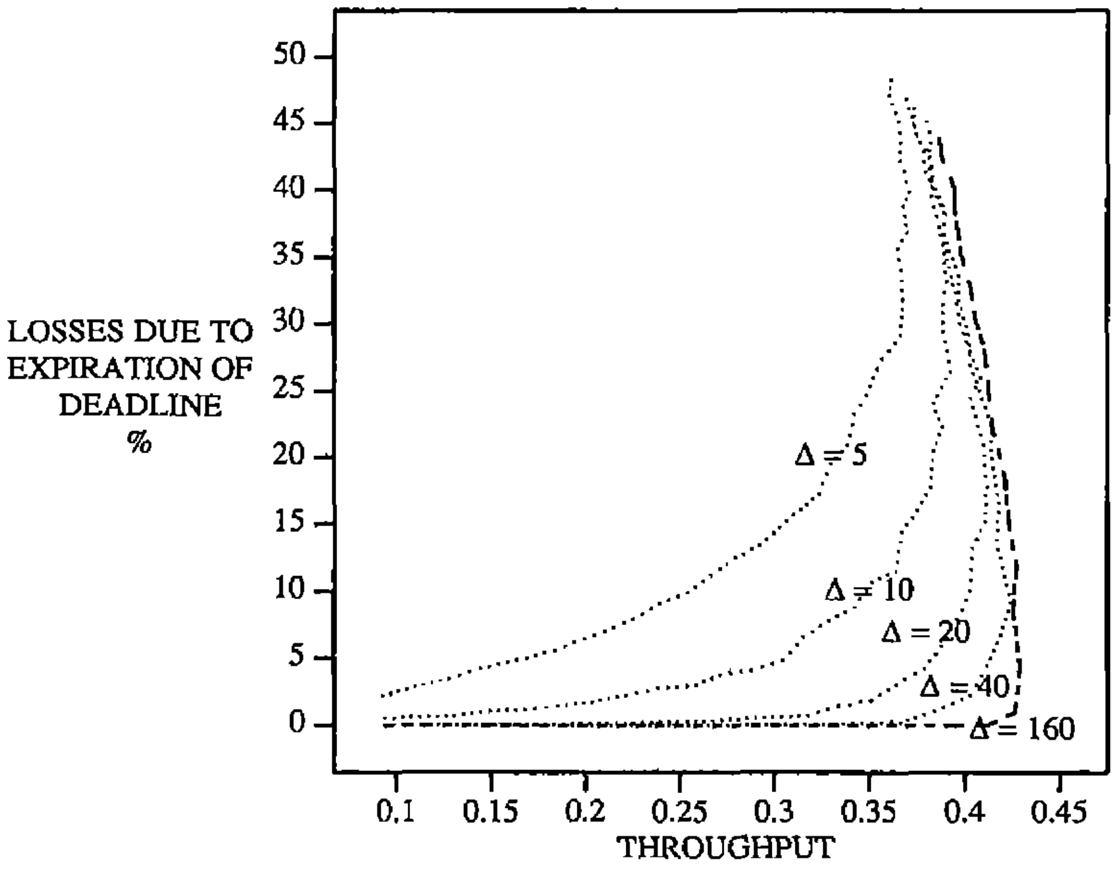

Figure 4. The percentage of losses due to expiration of deadline function of the throughput for different values of the minimum acceptable deadline, Δ. The mean value of the distribution of deadline is $\frac{\Delta}{\mu} = 16$. In this case the average value of losses of the first type, (losses because the requested deadline is shorter than Δ) is about 6.0 %. The window size is close to the optimal window size for maximum throughput, $W_0 = 2$. The arrival rates are in the range $0.1 \leq \lambda \leq 0.78$.

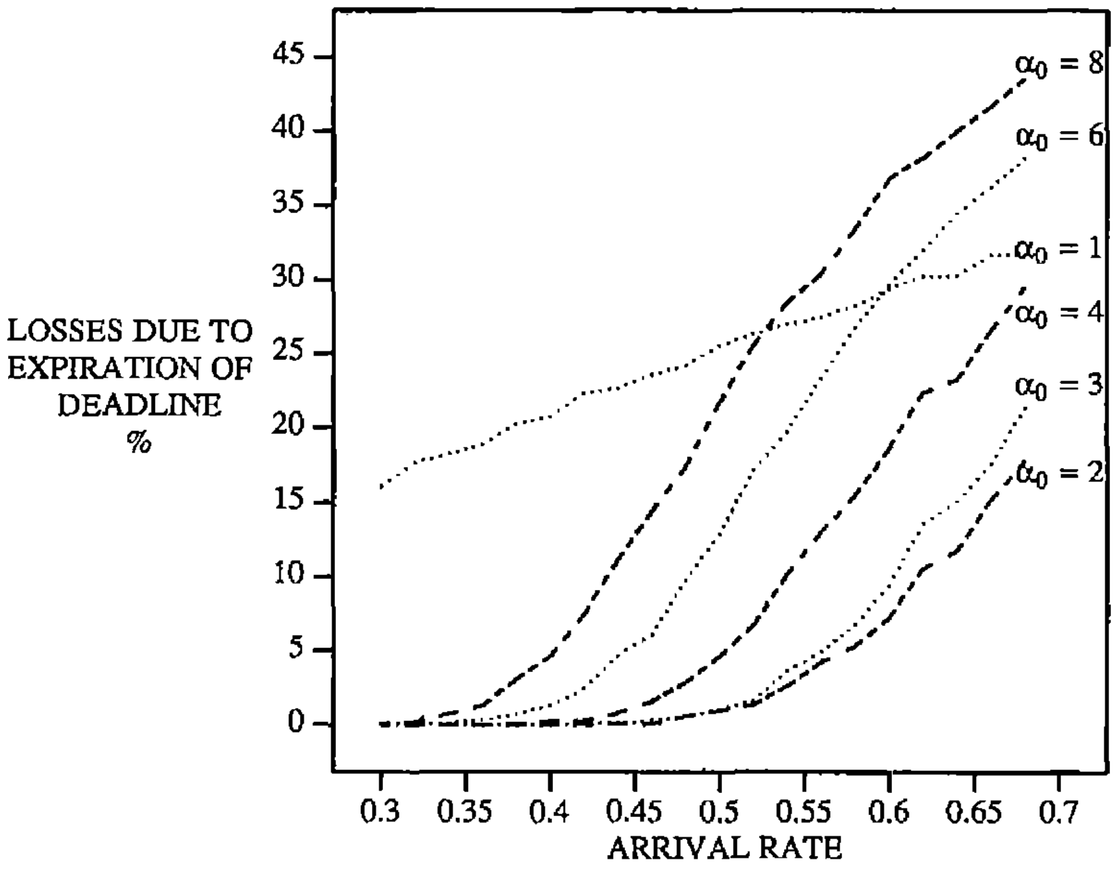

Figure 5. The effect of the maximum window size, α_0, upon losses of the second type for, W_0 in the range 1 to 8. Losses of the second type (percentage out of the total number of packets generated) function of the arrival rate for different values of the maximal window size. The ratio $\frac{\Delta}{\mu} = 4$, and $\Delta = 40$. The optimal window size is $W_0^{optimal} = 2$.

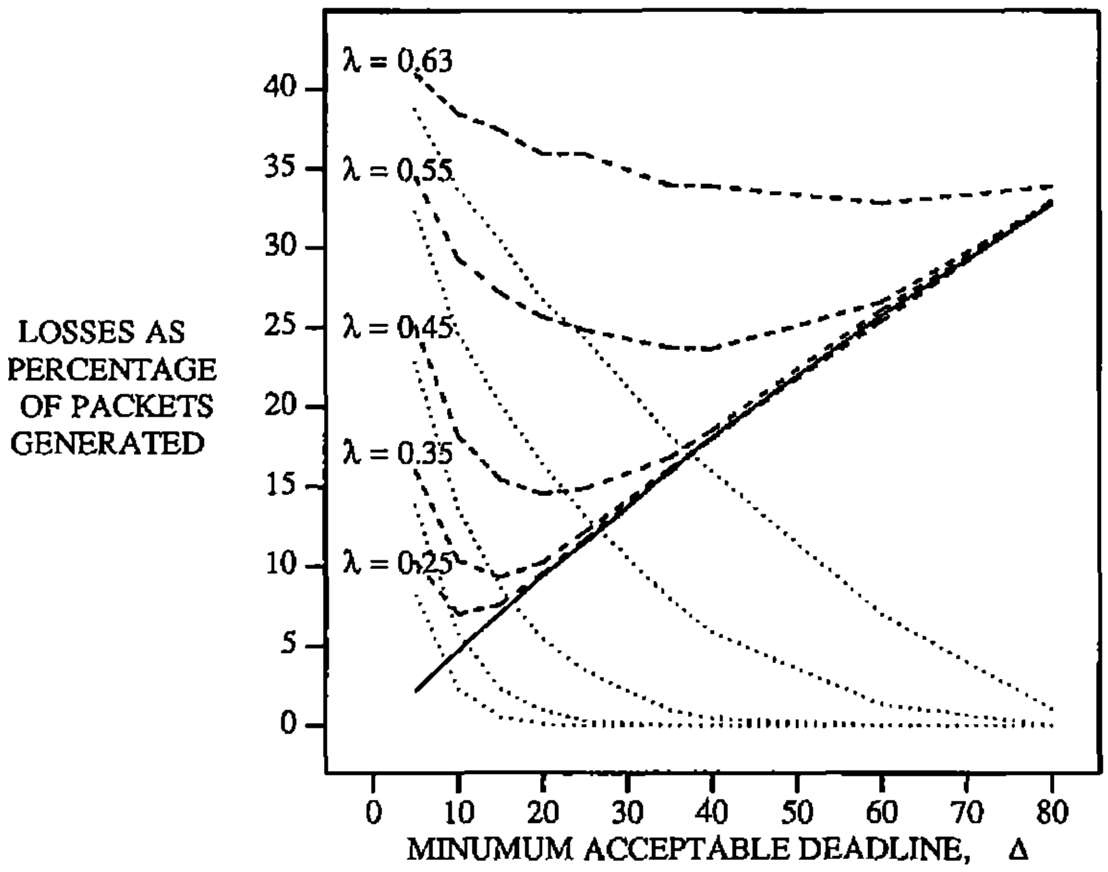

Figure 6. The effect of the minimum acceptable deadline, Δ upon the losses. The mean value of the deadline is fixed, $\mu = 200$. The ratio $\frac{\Delta}{\mu}$ varies in the range 0.4 to 16. Losses of the second type, due to the expiration of deadline, are represented by dotted curves for arrival rates $0.25 \le \lambda \le 0.63$. The losses of the first type, due to a too short deadline are represented by the solid curve. The losses of the first type are independent upon the arrival rate of packets. The cumulative losses are represented by dashed curves.

LITERATURE

[1] Baccelli, F., and Hebuterne, G., "On queues with impatient customers", in *Performance '81* (F. J. Klystra, ed.) North Holland, Amsterdam, pp. 159–179, 1981.

[2] Jackson, J.R. "Scheduling a production line to minimize tardiness", *Research Report 43, Management Sci.* Univ. of Calif., Los Angeles, 1955

[3] Marinescu, D. C., "Approximate analysis of distributed semi-hard real-time control systems", *IEEE Transactions on Automatic Control,* Vol AC-32, Number 12, pp. 1097-1100, December 1987

[4] Marinescu, D.C., "A splitting algorithm for communication with real-time delivery constraints", *Proceedings of Twenty-Six Annual Allerton Conference on Communication, Control and Computing,* pp. 955-964, September 1988

[5] Panwar, S.S., Towsley, D., and Wolf, J.K., "Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service" *Journal of ACM,* Vol. 35, No 4, pp. 832-844, October 1988