Survey of Weakly-Hard Real Time Schedule Theory and Its Application

Zhi Wang, Ye-Qiong Song, Enrico-Maria Poggi, Youxian Sun

Survey of Weakly-Hard Real Time Schedule Theory and Its Application

Zhi WANG[1*] Ye_qiong SONG[2] Enrico-Maria POGGI[2] Youxian Sun[1]
(1 National Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China)
(2 LORIA–ENSEM, 2, av. de la Forêt de Haye, 54516 Vandoeuvre-lès-Nancy, France)

ABSTRACT

Normally, tasks are classified into real time and non real time according to temporal constraints for the processing and transmitting of these tasks, consequently the worst-case response time and average performance should be focused on them. However, in practical engineering context, partly violated temporal constraints can be tolerated if the violation meets certain distribution. Nevertheless, the loss-rate (within real time region, an instance of a task is regarded as loss if it violates its temporal constraint) under stable state or statistical real time can solve the problem in some extent, it can't include the permitted distribution of violation. For completely solving the problem, weakly-hard real time schedule theory or window-constraint real time schedule theory, which is used to investigate the problem related to allowing violation of instances over a finite range, consecutive instances or a time window, is proposed. In order to effectively utilize the fact that a practical application can tolerate some violations of temporal constraint under certain distribution, the fundamental research must be done from the aspects of specification of temporal constraint, schedule and schedulibility, and implementation, which are explained in detail in this paper.

Keyword: Weakly-Hard Real Time, (*m*, *k*)-firm, Real Time, Loss-rate, Quality of Service, Differentiated Service

1 INTRODUCTION[1]

Real time schedule theory is mainly applied to a kind of system, wherein the temporal aspects of their behavior are parts of their requirement. The correctness of the result of a task is not only related to its logic correctness, but also to when the result occurs. Normally, such system refers to real time system. Traditionally, within real time schedule theory, real time systems are classified into two types, HRT (hard real time) system and SRT (soft real time) system.

For applications with HRT requirements, non deadline missed is tolerated. It means that the time period from receiving a task to completing the task, refers to response time, must meet a temporal constraint enforced on this task, refers to deadline, otherwise the task comes to a failure. In practical engineering, many systems may be regarded as HRT system. For instance, within process control or manufacture, a controller has to collect message of controlled application from sensor. After processing the message, the controller transfers its command to an actuator, which directly controls application according to the command.

The traditional perspective of real time schedule theory has served the temporal safety-critical system community well. Most temporal safety-critical applications are regarded as HRT system on the temporal aspect of processing a task. To guarantee each instance of a task meet its deadline, a HRT system is designed and tested under the worst-case situation, where consider a task with its maximum executing time, and minimum arrival interval and minimum deadline. The analysis of such systems is performed with worst-case analysis to estimate an upper bound for application response time using service curve approaches [Cruz91] or classical worst-case response time analysis [Lehoczky90].

However there are still some pessimistic factors in them. Since even a safety-critical system, a computer automatic control system as an instance, not every task must be guaranteed its deadline because of the deadline being over looked. Within this kind of system, tasks periodically sample input signals, perform some computation and then send command to some actuators. Normally, the sample rate, which plays an important role in schedulibility, should be a multiple of the bandwidth frequency, the intrinsic physical characteristic of an application, therefore some deadline lost can be neglected if the lost do not occur consecutively over a long period. Besides, some signal interpolation techniques can compensate the deadline missed.

Normally, most no temporal safety-critical applications are regarded as SRT system. For SRT, it is permitted to miss some deadline occasionally. Examples of real time but non safety-critical application are multimedia applications, such as video-on-demand or streamed audio. It is important that information is received and processed at an almost constant rate, such as 30 frames per second for video information. However, some packets comprising of video frame can be lost, resulting in little or no noticeable degradation in the quality of service at the receiver. Similarly, a data source can lose certain fraction of information during its transfer across a network as long as the receiver can process the received data to compensate for the lost information. However，the term occasionally is not precise, but for SRT systems we should specify a probability to meet the deadline requirements. In general, the analysis of such systems is made using stochastic approaches and queuing theory [Takagi 90, King90, Rom90]

Therefore, there are several reasons why such a restricted perspective is proving increasingly inadequate for many emerging applications, which are not safety-critical, but nevertheless have significant real time performance requirement:

- A deadline missed in such system is usually not catastrophic; instead, such a failure typically leads to a gradual quality of service degradation.
- In practical engineering, the occasional loss of some deadlines usually can be tolerated due to over pessimistic assumption about temporal property of a task, such as its execution time, deadline arrival interval, and the robustness of related control algorithm to an application.
- Run-time scheduling algorithms are typically designed to be correct only on a feasible system (system in which is indeed possible to always meet all deadlines); on an infeasible system, such algorithm may perform unacceptably poorly. A good example of this is EDF algorithm, which is optimal under non-overload condition, but has been observed to perform miserably upon overload.
- Overload management should carefully discard instances of tasks in order to improve effective utilization of resource and to minimize the amount of degradation caused by discarded instances.

[1] This paper supported by NSFC-60203030, 60084001and PRA S101-04.
Author of correspondence: zhiwang_iipc@yahoo.com or wangzhi@iipc.zju.edu.cn.

Further, on the aspect of performance metrics, the worst-case performance is focused on in HRT system, but average performance is in SRT system. In fact, for a system, its behaviors are great difference between its worst-case situation and average performance. Design or select a system based on the worst-case performance is surely poor resource utilization. Not having to meet every deadline allows the capacity of real time system resource to be smaller than it would be to meet all of them. This permits the creation of simple and more cost-effective system that make better use of the available resources while guarantee, in the worst-case, a minimum level of services.

From the above analysis we know, most real time application can tolerate certain deadline lost. The situation that practical engineering requires for better characterizing the temporal constraints of real time tasks and effectively managing these tasks with these temporal constraints, make a new challenge to real time schedule theory. However, these two classes, HRT and SRT, might be insufficient to appropriately describe a real-time system. Traditional real time schedule theory only deals with how to guarantee deadline of each instance instead of allowing partly deadline missed. From this sense, traditional real time schedule theory greatly lags the requirement of practical engineering. Actually, for SRT systems, stochastic analysis gives only probability of deadline missed, and can not guarantee that these deadlines are missed in right manner to hold the good behaviour of real-time system.

Exploiting the emerging real time applications, which tolerate certain deadline missed provided that the deadline missed occurs in a clear, predictable and bounded way, has the following advantages:

- Alleviating the pessimism in parameter of system and worst-case scenarios.
- Providing a mechanism for fair degradation of quality of service
- Obtaining a fair mechanism for deciding which task need to be skipped during transient overload

To implement the above object, we need:

- Realize the behavior of a task in case some of its instances miss deadline
- Provide clear and intuitive constraints for specifying the number of deadlines missed and met over a period.

From the aspects of specification of temporal constraint, schedule and schedulibility (schedule analysis), and implementation, this paper gives introduction in detail.

2 SPECIFICATION OF WEAKLY HARD REAL TIME SCHEDULE THEORY

The above section suggests that most real time applications can tolerate certain deadline missed, but the corresponding real time schedule theory only investigates how to guarantee every deadline of a task under worst-case situation. Naturally, the lost rate, the percentage of deadlines to be met or missed (although it is common practice to do so), is the direct performance metrics for these real time applications, and based on which statistical real time channel with P[response time < deadline] > p is proposed[20]. However, this concept take deadline missed to be evenly distributed for granted. In fact it can't guarantee even distribution of deadline missed at all. For example, the requirement of a task like "less than 10% of deadlines can be missed" only represents average information over a large period of time. It may mean that one deadline is missed every 10 instances of the task or that 100 deadlines may be missed followed by 900 deadlines met. Clearly, the two cases are not the same. It appears that the tolerance to deadlines missed cannot be adequately specified by a single parameter. Up to present, only a little work is in the region of WHRT schedule theory. Although lots of works have been done to deal with real time problem, most work is focused on how to meet deadline of each instance in HRT context.

2.1 Weakly-Hard Real Time Schedule Theory

To deal with the practical issues, a new real time schedule theory is necessary. We refer this type of theory to weakly-hard-real time schedule theory. Formally,

Definition 1 WHRT (Weakly-Hard-Real Time) Schedule Theory: A weakly-hard-real time schedule theory is a conceptual framework which investigates the characteristics of a system that can tolerate certain deadline missed under a precise distribution over a finite time window. Hence, the tolerance to deadline missed is established within a window of consecutive invocations of the tasks.

Correspondingly, the temporal constraint under the context of WHRT schedule theory refers to weakly-hard-real time constraint (WHRTC).

2.2 Specification of WHRT Constraints[3,4,5,6,7][15,16]

To capture the situation that deadlines missed with a permitted distribution over a finite range can be tolerated, a second parameter describing the window of time within which the number of deadlines must hold should be specified. To address the problem, server QoS criteria have been proposed. First, Nagarajan proposes two criteria called interval QoS and Block QoS. Unlike a state-state QoS measure, the quality of service is measured over finite intervals of time. Nagarajan points state-state analysis is inadequate in minimizing the occurrence of high loss periods or in maximizing the occurrence of no loss period. However, Nagarajan has not provided any (m, k)-like WHRTC. In fact, the statistical real time channel proposed by King, essentially is a kind of state-state QoS although which provides a probability real time requirement over a point-to-point channel.

In this sense of WHRTC, the work is first done by Koren and Shasha, who propose an approach of description of deadline missed with deterministic distribution, skip factor. A task which has a skip factor of s will have one instance skipped out of its s consecutive instances. It is apparent that a skip factor at least deterministically guarantees at most one deadline missed occurring over a finite time, s consecutive instances. Further, Hamdauoi and Ramanthan expand the notion of the skip factor with (m, k)-firm, to specify a task that is desired to meet deadline of m instances among its consecutive k instances. Similarly, Richard and Christian propose windowed lost rate, that specifies a task can tolerate x deadline missed over a finite range or window, among consecutive y instances. Recently, Bernat and Burns summarize temporal properties of specifications available of WHRTC, point out the relations between various specifications. Further, they point out that a specification should be considered from two aspects: a task maybe is sensitive to the consecutiveness of deadline met while another is only sensitive to the number of deadline missed; a task maybe is sensitive to the consecutiveness of deadline missed while another is only sensitive to the number of deadline missed. Concretely, they provide four types of basic specifications of WHRTC, these are:

- A task τ "meets any m in k deadlines", denoted with (m,k), if in any window of k consecutive instances of the task, there are at least m instances that meet the deadline.
- A task τ "meets consecutive m in k deadlines", denoted with $\langle m,k \rangle$, if in any window of k consecutive instances of the task, there are at least m consecutive instances that

meet the deadline.

- A task τ " not misses any m in k deadlines", denoted with $(\overline{m}, \overline{k})$ if in any window of k consecutive instances of the task, there are no more than m instances are missed
- A task τ "not misses consecutive m in k deadlines", denoted with $\langle \overline{m}, \overline{k} \rangle$ if in any window of k consecutive instances of the task, it is never the case that m consecutive instances miss their deadline.

2.3 Relation between Different Specifications of WHRTC

For the present, although there are various specifications of WHRTC, fortunately, almost all of them have intrinsic relationships with the four types of basic specifications.

- Hard Real-time with response time<deadline is equivalent to (k, k).
- Skip factor with skip one instance of s instances is equivalent to $(s-1, s)$ or $(\overline{1}, \overline{s})$.
- Statistical real-time channel with P [response time < deadline] > p is equivalent to $p = \lim_{m,k \to \infty} \frac{m}{k}$.
- Windowed lost rate with tolerating x deadline miss over y consecutive instances is equivalent to $(y-x, y)$ or $(\overline{x}, \overline{y})$.
- No real time just corresponds to (m, k) in the case of $m=0$.

Therefore, the four types of specifications are basis on the aspect of they can describe most specifications available.

Further, there are a certain relationship between these basic specifications, these are:

- $(m,k) = (\overline{k} - \overline{m}, \overline{k})$
- $\langle \overline{m}, \overline{k} \rangle = \langle \overline{m} \rangle$

2.4 Model of a Task with WHRTC

We only consider periodic tasks in the following, although tasks can be periodic or aperiodic (i.e. instances are randomly generated). In fact, in real-time community it is common to also consider sporadic traffic as periodic by taking the minimum inter-arrival time of instances as period. In practice, for most of transmission systems this minimum inter-arrival time does exist (e.g. 64-bytes packet + 96-bits IFS in Ethernet, leaky bucket smoothed input traffic).

We characterize a system with a set of n independent periodic tasks, $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n, \}$. Task τ_i can be described as the following model:

$$\tau_i = (O_i, D_i, T_i, C_i, \beta_i) \quad (1)$$

where O_i denotes release time of the instance job of τ_i, referred to initial time; D_i denotes maximum time allowed from the release time to the completion time of τ_i's instance, referred to deadline; T_i denotes interval between release times of two consecutive instances of τ_i, referred to period; C_i denotes maximum time needed to complete τ_i without any interruption, referred to execution time; β_i denotes WHRTC enforced to task τ_i. The j[th] instance of task τ_i is denoted as τ_{ij}.

Further, we can give definition of failure state and success state of task τ_i according to whether it meets its WHRTC.

Definition 2 Failure state and success state: a task τ_i is in success state if its last consecutive instances meet WHRTC, otherwise is in failure state.

From the above description, we can see a task may experience different states. Take (2, 3) of WHRTC as example. The state transition diagram of task model with WHRTC of (2, 3) is indicated in Fig.1.

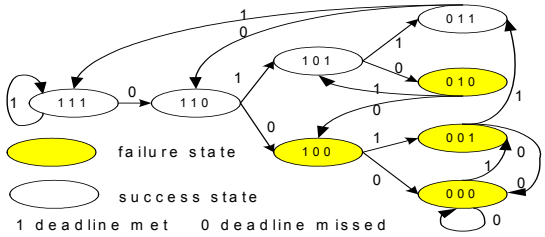


Fig.1 State transition diagram of task with WHRTC of (2, 3)

2.5 μ-pattern and WHRTC

The only information that a scheduler uses for tasks with WHRTC is the pattern of zeros and ones that represent missed and met deadlines on the history of the task (and possibly of the possible future of the tasks). These patterns refer to μ pattern[3][4].

Definition 3 μ-pattern: A μ-pattern of a task is a sequence of symbols of $\sum = \{0,1\}$ that characterizes the execution of the task. $|\mu| = p$ is the length of the pattern, and $\mu(k) \in \sum (1 \le k \le p)$. 1 means that a task has met its deadline, and 0 means that the task has missed its deadline. $\mu(1)$ is the oldest invocation and $\mu(p)$ is the most recent invocation. An example with (4, 5)-firm constraint is given in Fig.2.

The μ-pattern is a word of k bits ordered from the most recent to the oldest invocation in which each bit keeps memory of whether the deadline is missed (bit = 0) or met (bit =1). In this paper, the leftmost bit represents the oldest. Each new invocation causes a shift of all the bits towards left, the leftmost exits from the word and is no longer considered, while the rightmost will be a 1 if the task has met its deadline (i.e. it has been served within) or a 0 otherwise.

In essential, all of these algorithms deal with a problem of partition of instances of a task to meet its WHRTC. Therefore, we can generalize the problem of scheduling tasks with WHRTC, further evaluate a schedule algorithm and investigate optimal schedule algorithm.

From definition 3 we can see that schedule for a task with a WHRTC is just to select a proper μ-pattern that meets the WHRTC. Therefore, whether there is an optimal μ-pattern among different μ-patterns that meets WHRTC is important.

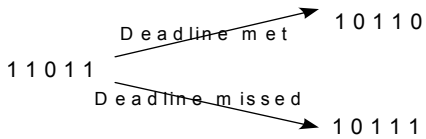


Fig.2 Possible evolution of the μ-pattern

Definition 4 Optimal μ-pattern : Given a task τ with a WHRTC, let the mandatory instances defined by a set of μ-pattern be assigned fixed priorities and the optional instances be assigned the lowest priority. The optimal μ-pattern is a μ-

pattern that meets there are no other μ pattern can satisfy the WHRTC if the optimal μ-pattern cannot.

3. BASIC SCHEDULING ALGORITHMS OF WEAKLY HARD REAL TIME SCHEDULE THEORY

Although there are still lack general conceptual frame for weakly-hard real time schedule theory, in order to fairly distribute resource to meet temporal requirement of tasks with their respective WHRTC, there are parts of scheduling algorithms being proposed. The goals of these scheduling algorithms are various, part for providing deterministic guarantee of temporal requirement, part for improving flexibility by just providing best-effort service, part for implementing total performance when real time and no real time tasks co-exist. In this section, we just introduce some typical scheduling algorithms from different aspects under the context of weakly-hard real time schedule theory.

3.1 DBP (Distance Based Priority) Schedule[10]

For each τ_j , in order to guarantee its WHRTC, its priority is assigned based on the number of deadline misses that task can still stand before violating its (m, k) requirement. This allowing number of deadline misses is referred to as distance, i.e. the distance to a failure state from current state. When a task is violating is (m, k) requirement, that task is said to be in failure state. The evaluation of this distance can be done exactly considering the recent history of τ_j .

Fig.1 suggests that the closer of a task to its failure state, the more easily the task suffers failure. That activates the idea of DBP schedule. As for an instance of a task with WHRTC of (m, k), DBP designs priority to the instance according to the information of its last k consecutive historical instances. Further, Fig.1 also suggests that a task with WHRTC of (m, k), the trend of its state to failure state is relevant to the position of m^{th} 1 (position of m^{th} deadline meet occurs) from the last k instances.

The priority assigned by DBP to a job at a given instant is equal to the distance of the current μ-pattern to a failure state. This distance can be easily evaluated, by adding in the right side 0s until failure state and the number of added 0s is the priority. If a stream is already in failure state (i.e., less than m 1s in the μ-pattern), the highest priority 0 is assigned. This is also given by equation (1). For example, considering a task with (3,5)-firm constraint, the current job j_{i+1} is set the priority of 2 if its previous 5 consecutive jobs construct the state of (11011), and is set the priority of 3 if its previous 5 consecutive jobs construct the state of (10111). Formally,

Let $s_j = \left(\delta_{i-k_j+1}^j, \cdots, \delta_{i-1}^j, \delta_i^j\right)$ denote the state of the previous k consecutive instances of task τ_j, $l_i(n,s)$ denote the position of n^{th} meet in the state of s_i, then the priority of current instances of task τ_i is

$$priority_j^{i+1} = \begin{cases} k_j - l_j(k_j, s_j) + 1 & \left(l_j(k_j, s_j) \le m_j\right) \\ 0 & \left(l_j(k_j, s_j) > m_j\right) \end{cases} \quad (2)$$

For example, considering a task with (3, 5) of WHRTC, the current instance of the task is set the priority of 2 if its previous 5 consecutive instances construct the state of (11011), and is set the priority of 3 if its previous 5 consecutive instances construct the state of (10111).

One of the problems faced with DBP, is that it assigns priorities only considering one τ_j without comparing it to the others sharing the same server. This self-reference behaviour may lead to a situation where more than one stream get the same priority at the same time, in this case an algorithm to choose among them should be defined. It is also important to underline that DBP chooses priority based on the history of the stream's μ-pattern, and doesn't take into account any specific information on the actual attributes of the stream like its length c_j, its minimum inter-arrival time T_j , and its deadline D_j.

The simplest and common way to overcome these problems is to assign DBP-based priority to the jobs and, in case of priority equality, use another scheduling algorithm among the already known ones. In their paper, Hamdaoui and Ramanathan [Hamdaoui95] combined DBP with Earlier Deadline First (EDF). However this solution gives to Deadline less importance than that given to the μ-pattern, since EDF would be used only when μ-pattern is not sufficient, i.e. when two streams get the same DBP-priority.

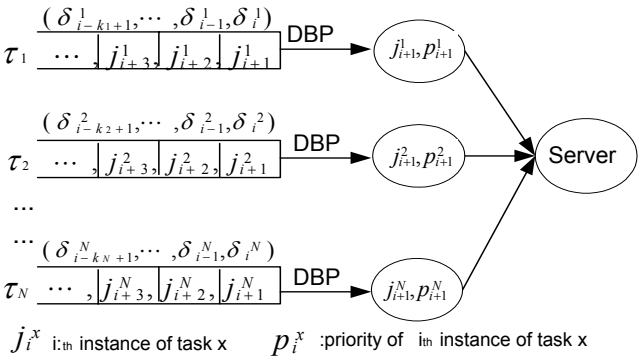


Fig.3 The Implementation of DBP Schedule Algorithm

To overcome the limits of DBP, the properties of periodic tasks, and the relationships among these properties should be considered, and an improved DBP, which integrated the above neglected factor should be researched in detail.

3.2 DWC (Dynamic Window Constrained) Schedule[16]

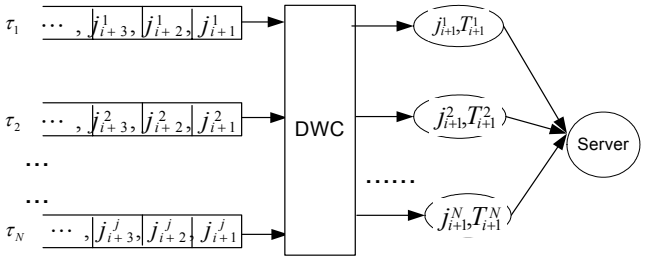
DWC schedule maintains information of each task just like DBP schedule does, but how to utilize the information is significantly different from DBP schedule. Whereas DBP schedule processes the current instance a task using its the state transition and its k last historical instances to capture the relative priority of a task, DWC schedule using the notion of dynamic window in which m and k are allowed to change. In DWC schedule, each time an instance of a task τ_i is transmitted or dropped, the information of (m_i, k_i) is adjusted accordingly.

Table.1 Main rules of DWC for Assigning Priority to an Instance

Lowest loss-tolerance first
Same non-zero loss-tolerance, order EDF first
Same non-zero loss-tolerance & deadlines, order lowest loss-numerator first
Zero loss-tolerance & denominators, order EDF first
Zero loss-tolerance, order highest loss-denominators
All other cases: first-come-first-serve

The main rules for DWC schedule are as follows: DWC schedule algorithm processes instances of tasks based on the current values of their loss-tolerance and deadlines, and gives precedence to the instance with the lowest loss-tolerance. Instances of the same task all have the same original and current

loss-tolerance, and are scheduled in their order of arrival. The loss-tolerance of an instance (and hence the corresponding task) changes over time depending on whether or not another earlier instance from the same task has been scheduled for transmission by its deadline. Whenever an instance misses its deadline, the loss-tolerance for all ongoing instances in the same task is adjusted to reflect the increased importance of transmitting these instances. This approach avoids starving the service granted to a given task and attempts to increase the importance of serving any instance in the task which is likely to violate its original loss constraints. Conversely, any instance serviced before its deadline causes the loss-tolerance of other instances (yet to be serviced) in the same task to be decreased, thereby reducing their priority.



j_i^x:Ith instance of task x T_{i+1}^x :loss_tolerance of i th instance of task x

Fig.4 The Implementation of DWC Schedule Algorithm

3.3 ERM (Enhanced Rate Monotonic) Schedule[11]

DBP and DWC schedules essentially are dynamic priority based schedule and belong to best-effort, that means they can't provide any deadline guarantee at all. At least, there is still not an effective approach to check whether they guarantee a task meet its WHRTC of (m, k). In fact, the goal of any schedule is to effectively manage tasks and distribute proper resource for these tasks. Because most static priority based schedule algorithms can provide deterministic guarantee of a task temporal constraint, and because the essential of WHRTC of (m, k) is to meet m of k deadlines, it is possible to construct a static priority promotion approach that can select m instances from any consecutive k instances. If we can guarantee the selected m instances of a task from any its k consecutive instances, we can at least meet WHRTC of (m, k) of this task. From this sense, the idea of Imprecise Computation (IC), that divides instances of a task into a mandatory and an optional part and the latter is rejected when system overload, is the same as WHRTC of (m, k). Further, Rate Monotonic (RM) schedule, a well-known static priority schedule algorithm for periodic tasks, effectively schedule periodic tasks based on the tasks' period. Combining the ideas from the IC and RM schedule, other schedule, ERM schedule, is proposed. That is the instances of a task is divided into a mandatory and an optional part, and the instances of mandatory part are scheduled according to Rate Monotonic policy, and instances of the optional part are assign the lowest priority and scheduled according to First Come First Service (FCFS).

The implementation of ERM schedule is very simple and easy; the key problem is how to select mandatory part of instances for a task. An approach for classification of instances of task τ_i as mandatory or optional is given, that is based on the value of m_i and k_i.

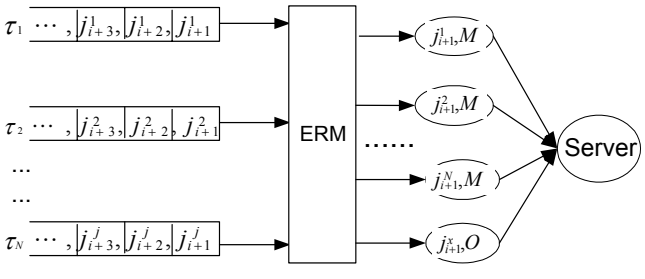
The instances of task τ_i activated at $(a = 0,1,\cdots,)$ are classified as mandatory if a meets

$$a = \left\lceil \frac{a \cdot m_i}{k_i} \right\rceil \cdot \frac{k_i}{m_i} \qquad (3)$$

Take the following two tasks as example:

$\tau_1 : C_1 = 1,\ T_1 = 2,\ m_1 = 2,\ k_1 = 3,$
$\tau_2 : C_2 = 3,\ T_2 = 4,\ m_2 = 2,\ k_2 = 3$

For task τ_1, one out of its every three instances is classified as optional, starting with the instance activated at time 8. For task τ_2, the instances with activation times 24, 48, 84, 108, … , are classified as optional.



j_i^x:Ith instance of task x M :mandatory task O :optional task

Fig.5 The Implementation of ERM Schedule Algorithm

Take ERM schedule as example, it has the following property from the aspect of selecting a proper μ-pattern,

- ERM implicitly selects mandatory instances of a task evenly among its m consecutive instances, and solely depends on the ratio of m over k of the task.
- The first instance of every task is always designated as mandatory or given the highest priority.

Apparently, judicious selection of mandatory v.s. optional instances and promotion of priority for instances play a critical role in scheduling tasks with WHRTC. However, ERM has following intrinsic disadvantages:

- Regardless of period and execution time, the mandatory instances of two tasks, having the same ratio of m over k, are always distributed in the same way among the m consecutive instances.
- Such even distribution lacks flexibility and may not be advantageous in certain situations.

4. EXPEBDED SCHEULING ALGORITHMS OF WEAKLY HARD REAL TIME THEORY

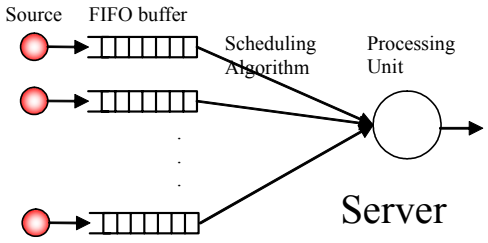


Fig.6 MIQSS model

In fact, all the above scheduling algorithms are basic scheduling algorithm in the sense that they are applied in MIQSS (multiple input queues single server) and under the condition where that only real time tasks with WHRTC exist. MIQSS model can be used to study a large category of computer and telecommunication systems such as multiple tasks execution in a CPU, transmission of messages issued from multiple message sources sharing a same transmission medium or network interconnection equipment. The proposed model is made up of *N* sources generating *N* streams of jobs τ_i (*i =1, 2, ... N*) attempting to be served by a single server. Each stream is formed by a source and a waiting queue, where a job (can represent a task or

a message) issued from the source waits until chosen by the server. The server chooses jobs at the head of queues according to its scheduling policy.

However, in most actual applications, tasks or messages are diversity on the aspect of hops of end-to-end connection over which tasks have to transfer, the number of tasks and types of tasks a server processes.

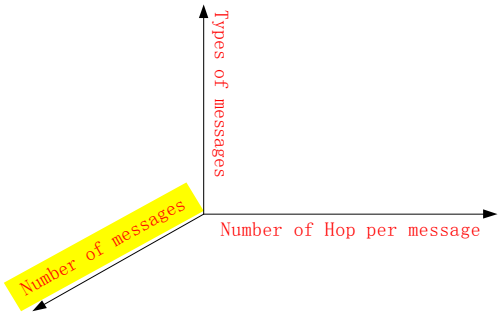


Fig.7 Complexity of schedule algorithms within WHRT

- **Multi-hop end-to-end connection**

Within a distributed real time system, a task needs data packet crossing multiple networks, multi-hop connection, and accordingly requires QoS in end-to-end. However, on the aspect of multi-hop, most schedule algorithms, such as DBP, DWC and DCQ, are not applicable any more. It is because the deadline in DBP like schedule algorithms is only a local deadline in multi-hop. Correspondingly, the rejected packet due to exceed its deadline is actually exceeding its local deadline in multi-hop, and maybe still have a chance to meet its end-to-end deadline.

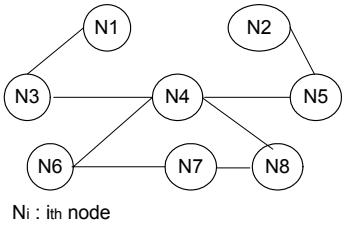


Fig.8 Network Topology of a Multi-hop Network

Parameter of Task

Task	Route	Deadline	(m,k)
T1	N1-->N3-->N4-->N8	30ms	(3,4)
T2	N2-->N5-->N4-->N6	25ms	(5,7)
T3	N3-->N4-->N5	15ms	(5,6)
T4	N6-->N7-->N8	30ms	(3,4)
T5	N7-->N8-->N4-->N3	24ms	(6,7)
T6	N8-->N4-->N5-->N2	35ms	(1,1)

Fig.9 Parameter in a Multi-hop Network

For example, suppose a network of multi-hop in Fig.8,9, which has 8 nodes $n_i\,(1 \le i \le 8)$, and 6 tasks $\tau_i\,(1 \le i \le 6)$. These nodes are connected by point-to-point, data packets of tasks are transmitted across multi-hop connected by nodes. For task τ_2, which must travel three hops of $n_2 \to n_5 \to n_4 \to n_6$ and has an end-to-end deadline of $D_i = 25ms$ and WHRTC of $(5,7)$. That means an data packet of τ_2 generated from its source node of n_2 at time t must reach its destination node of n_6 at time $t + D_i$, and its at least 5 data packets meet end-to-end deadline among its every 7 consecutive data packets. For the present, in the context of multi-hop end-to-end connection, only EDP-M (Modified-DBP) and EDBP (Enhanced EDP-M) are proposed and implemented.

- **Number of Tasks**

All of the schedule algorithms of DBP, ERM and DWC are common in this aspect, the schedulers utilize information of each task. Correspondingly, the above schedule algorithms are tasks-aware. However, the overheads of maintaining the information will rapidly increase with the increasing number of tasks. Example of application is real time media server, which will be responsible for lots of client, with a wide range of QoS requirement.

To overcome the limitation, the schedule algorithms must be scalable, where scalable means the overheads of schedule algorithms is independent to the number tasks in these schedule algorithms. Scalability has get attention in IETF, which focuses on providing architecture for real time service through Internet. Two systematic approaches have been proposed by IETF, one is InteS (Integrated Service), another is DiffS (Differentiated Service). InteS provides real time service by processing information of each task, and accordingly does scale well. Contradictorily, DiffS provides real time service by class-based schedule policy, wherein each task is partitioned into correspondingly class. In fact, the information of each class is the result of aggregation of tasks belonged to this class.

On the aspect of scalability, per class based schedule algorithms scale well than per task based, but there are fundamental problems to be solved on task with WHRTC. One of them is partition of tasks with WHRTC into a class, which is concerned with relationship of various WHRTC, such as whether a WHRTC being hard than another on the aspect of temporal constraints, how to determine a general metrics to various WHRTC. Up to present, only DCQ (Dynamic Class-based Queue) is proposed.

- **Types of Tasks**

Because of diversity of tasks on the aspect temporal requirement in an application or a system, not all tasks are real time, and part tasks maybe no real time. Therefore, effectively scheduling SRT task while guarantee the behavior of HRT task is becoming problem, and many techniques have been proposed to solve the problem. In fact, in many real time systems, hard real time task and soft real time task co-exist. Within this situation, a number of approaches have been proposed to deal with this mixed task set, such as DS (Deferrable Server), PE (Priority Exchange), SS (Slack Stealing), and DP (dual priority). Among these techniques, dual priority schedule is an intuitively simple method and lower overhead.

Although DP has some advantages in its simplicity and low overhead compared to other approaches, nevertheless B. Ganja has proved that DP is not always better than background scheduling on the aspect of improving responding time of no real time task[25]. For the present, to my best knowledge, EDP (Enhanced Dual Priority) is the only schedule algorithm being investigated under the co-existence of tasks with WHRTC and no real time tasks, let alone comparison among these schedules.

4.1 BDP-M and EDBP: Schedule Algorithm under Multi-hop End-to-end Connection

It is obvious that the DBP like schedule algorithms can't be directly used because all of them are applied under the assumption that sever can determine whether a task misses or meets its deadline. However, when a task must be relayed through multiple nodes, then all intermediate nodes can't locally determine whether the task meet its end-to-end deadline. Apparently, a direct approach is distributing end-to-end deadline

of a task into local deadline of hops where the task has to cross.

The simplest approach is evenly distributing end-to-end deadline according to the number of hops. In fact, BDP-M (Modified-DBP) and EDBP (Enhanced EDP-M) belong to the simplest approach on the aspect of distributing end-to-end deadline. The main idea of these two approaches is first check whether an instance (data packet) of a task violate its end-to-end deadline; then the selected instance is scheduled according EDP schedule.

- **Adaptability of Schedule Algorithms under Multi-hop End-to-end Connection**

On the aspect of multi-hop, adaptability of schedule algorithms is concerned with how distributing WHRTC of end-to-end (referred to global WHRTC here) into local WHRTC, and adjusting local WHRTC online to adapt actual situation. Solving the above problem consists of two challenges.

The global policy for dealing with data packet transmission across multi-hop:

- How to design end-to-end deadline of a task along the hops over which the task transfers
- How to design WHRTC of (m, k) of a task along the hops over which the task transfers

The local policy for dealing with data packet transmission in local hop:

- How to processes a task to adapt the actual requirement in its global WHRTC. A local deadline has two types, actual delay and setting deadline (given by global policy), and local policy processes instance of a task according to its local WHRTC, including local deadline and local (m, k). Presently, EDP-M and EDBP only locally process its instances of tasks separately, without considering the actual global situation of these tasks, although the both schedule algorithm do check of whether an instance exceeding its end-to-end deadline first. In fact, if the actual delay of a task in a local hop is near to end-to-end deadline, the task will get more chance to be processed if its local deadline decreased correspondingly. The reason is simple, the shorter local deadline, the more probability of deadline violation, and accordingly higher priority. Intuitively, it is a complicated problem, but we can get some general guidance through simulation.

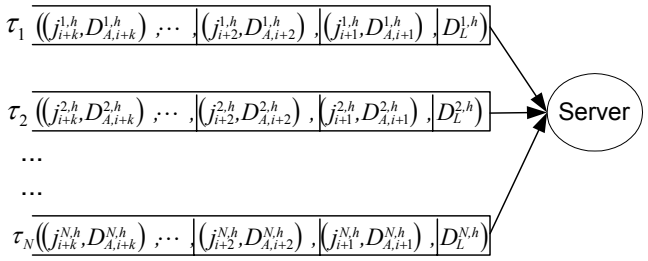


Fig.10 The Implementation of EDBP Schedule Algorithm

4.2 DCQ: Schedule Algorithm Being Scalable

As for QoS of Internet and scalable, DCQ schedule algorithms is first proposed to deal with tasks with WHRTC. The key issue to DCQ schedule has to solve is group membership, that consists of mapping a task with WHRTC into a class, and automatically adapting membership of each class when a task joining or leaving. DCQ schedule implements its goal through two level schedules, the first one is dealing with group membership, the other is scheduling each class according a given schedule algorithms, such as DBP schedule and DWC schedule.

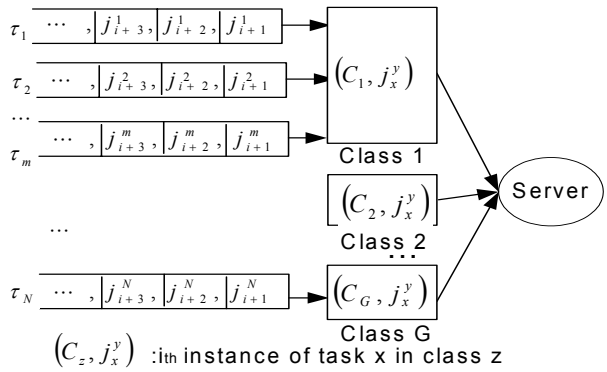


Fig.11 The Implementation of DCQ Schedule Algorithm

4.3 EDP: Schedule Algorithm for Co-existence of Tasks with WHRTC and SRT Tasks

DP schedule consists of three priority bands, there are lower, middle and upper. A HRT task is assigned two priorities, lower and upper, and the SRT task is only assigned middle priority. Upon invocation of an instance of a HRT task, the instance is assigned a low priority and it is promoted to a high priority to guarantee the deadline of the HRT task be met.

The key problem for dual priority schedule is to select proper promotion time Y_j for an instance of task τ_j after the instance is released.

$$Y_j = D_j - R_j \tag{4}$$

where, R_j is response time of task τ_j.

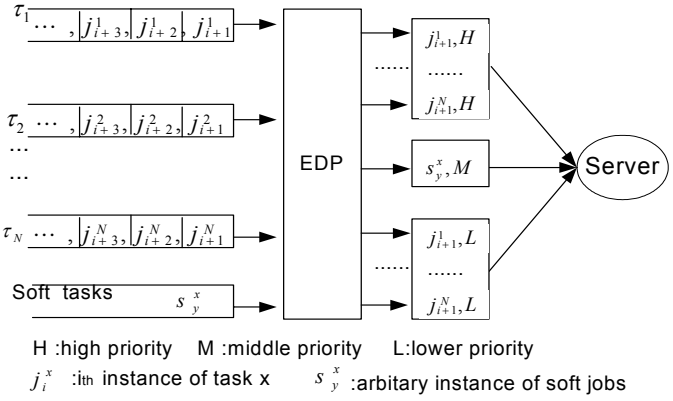


Fig.12 The Implementation of EDP Schedule Algorithm

The later a task has its priority promoted, the more slack time is available for SRT tasks and the better it is distributed. Consequently response time of SRT task is improved. Unfortunately, the value of Y_j may be quite close to the invocation if there is tight deadline for task τ_j. If we exploit the fact that tasks with WHRTC can tolerate some of their deadline missed, we can improve the responsiveness of SRT tasks. The improved DP refers to EDP Schedule.

This can be done by the following two approaches:

- Optimizing promotion time Y_j, so that tasks meet their WHRTC of (m, k).
- Optimizing selection strategies, thus offering more computation time to middle band for soft tasks whilst still guaranteeing real time tasks with WHRTC of (m, k).

5. PROBLEM YET TO BE SOLVED

Although many works have been done in this region, there are still some basic issues to be solved.

5.1 Specifications of WHRTC

Although Bernat and Burns propose four types of basic WHRTC, which are still lack enough capability for describing temporal constraint of a task. Take $\langle \overline{m}, \overline{k} \rangle$ as example, a μ-pattern of $(0,0,0,1,0,0,0,1,0,0,0,1,\cdots,)$ seems un-convincible despite of it meeting the requirement of $\langle \overline{3}, \overline{4} \rangle$. For solving this problem, $\langle \overline{m}, k \rangle$ like WHRTC maybe more convincible, where $\langle \overline{m}, k \rangle$ denote a task τ "not misses deadlines of m consecutive instances and at least meets deadlines of k consecutive instances". Similarly, there are other different WHRTC. It is note that the WHRTC should be easy implemented and analyzed.

5.2 Dynamic and Static Schedule Algorithms

DBP and DWC are dynamic schedule, on the aspect that priority of each instance of a task is automatically adjusted according the information of the previous instances of the task. For the present, almost all are dynamic schedule algorithms are DBP or DWC like, there is lack new idea on how to adjust priority of each instance dynamically.

ERM is static schedule, on the aspect that priority of each instance of a task is fixed, which only depended on the ratio of m to k in its WHRTC, instead the previous instances of the task. For the present, only one static schedule, ERM, is proposed and investigated.

Further, all schedule algorithms available focus only on WHRTC of (m, k), no matter static schedule or dynamic schedule, there are still lack all schedule algorithms on other WHRTC, such as $\langle m, k \rangle$, $(\overline{m}, \overline{k})$ and $\langle \overline{m}, \overline{k} \rangle$.

DBP and DWC are dynamic schedule, on the aspect that priority of each instance of a task is automatically adjusted according the information of the previous instances of the task. For the present, almost all are dynamic schedule algorithms are DBP or DWC like, there is lack new idea on how to adjust priority of each instance dynamically.

ERM is static schedule, on the aspect that priority of each instance of a task is fixed, which only depended on the ratio of m to k in its WHRTC, instead the previous instances of the task. For the present, only one static schedule, ERM, is proposed and investigated.

Further, all schedule algorithms available focus only on WHRTC of (m, k), no matter static schedule or dynamic schedule, there are still lack all schedule algorithms on other WHRTC, such as $\langle m, k \rangle$, $(\overline{m}, \overline{k})$ and $\langle \overline{m}, \overline{k} \rangle$.

5.3 Schedulability and Optimization of Schedule Algorithms

Providing online or offline schedulability checks for determining whether tasks with WHRTC being satisfied under given load and schedule algorithm is critical.

Schedulability checks is discovered as checks μ-patterns, nevertheless solving the problem consists of two challenges:

- How to determine if one set of μ-patterns is better or easier to be scheduled than another under a given task set with a WHRTC?
- How to predict if the corresponding mandatory jobs are all schedulable under a given set of μ-patterns?

Unfortunately,

- Searching optimal μ-pattern for each task is a NP-hard problem.
- Determine the schedulability of arbitrary μ-pattern of a task with a WHRTC is NP-hard problem.

It is well known that there are two main approaches for schedulability of a schedule algorithm. However, it is fronted great change in the calculation of utilization of resource and worst-case responding time (WSRT). Take the second as explanation; the key of calculation of WSRT of a task is in calculation of its busy period. However, the busy period of a task with WHRTC fronts the following challenges:

- Critical instant, the instant that all tasks arriving with their maximum is not critical instant in the sense of meeting WHRTC of a task, which is related to the information of previous instance. It is obvious the critical instant in traditional analysis is not the instant that a task is nearest to its failure state.
- Feasible load at time t, in traditional analysis and load at time t is very easy, but the situation is serious because for a task with WHRTC, wherein the distribution of discarded instance must be considered.

Intuitively, determine the WSRT of a task with WHRTC is NP-hard problem.

The schedulability of a task with a WHRTC and its sub-optimal μ-pattern can be further improved if one can tolerate spending more time on finding better mandatory/ optional partitions off-line. In this regards, a probabilistic optimization algorithm can be very effective[8][9].

- Genetic algorithms
- Simulated annealing

One challenge in applying such schedule algorithms is to formulate an appropriate objective function.

5.4 Sensitiveness of Tasks with WHRTC

For a system, we are interesting to schedulable or un-schedulable of the system, but also interesting to the margin by which the system un-schedulable or schedulable. On one aspect, how much reduction of capability of the system, such as rate of CPU and bandwidth of network, is allowed, or how many tasks are allowed to join if a system is schedulable. On another aspect, how much addition of capability of the system is needed, or how much WHRTC of tasks is needed if a system is un-schedulable. This type of analysis refers to sensitiveness analysis (SA). In fact, for task with WHRTC also needs SA. SA is important to admission control of network, especially in the negotiation between user and provider of network service. Traffic Model of Task with WHRTC

5.5 Traffic Model of Task with WHRTC

For the present, most researches focus on periodic task, except for a few on aperiodic tasks with Poisson arrival. Actually, within Internet most tasks is described by a traffic model (σ, ρ) proposed by Cruz, instead of by either periodic task or by aperiodic task. Within (σ, ρ), σ denotes arrival average rate of task, and ρ denotes burst size. It is feasible that we research tasks with WHRTC under the frame that the tasks have temporal characteristic of (σ, ρ) arrival.

6. CONCLUSION

With the emergence of lots of real time applications that can tolerate a certain deadline missed, the understanding real time must be improved instead of just guaranteeing deadline of each instance of a real time task. Accordingly, the real time schedule theory need be expanded to investigate the new phenomena. This paper summarizes the-state-of-art of weakly real time schedule theory on the aspect of specification, schedule algorithm and schedulability, and its applications. Further, the basic issues to yet be solved are pointed out in this paper.

REFERENCES

[1.] G. Bernat and A. Burns. Combining (n, m)-hard deadlines and dual priority scheduling. *Proceedings of Real-Time Systems Symposium*, pages 46–57, Dec 1997.

[2.] G. Bernat and A. Burns. Weakly_Hard real-time systems, *IEEE Transactions on Computer*s, 50(4),pp.308-321,2001.

[3.] G. Bernat and R.Cayssials, Guaranteed On-line Weakly_Hard real-time systems, *Proceedings of IEEE conference on real-time system*s, 2001.12

[4.] G. Quan and X. Hu. Enhanced Fixed-priority Scheduling with (m, k)-firm Guarantee, *12th IEEE Europmicro Conference on Real-Time System,* 2000

[5.] J.-Y. Chung, J. W. Liu, and K.-J. Lin. Scheduling periodic jobs that allows imprecise results. *IEEE Transactions on Computer*s, 39(9):1156–1175, Sep 1990.

[6.] M. K. Gardner and J. W.S.Liu. Performance of algorithms for scheduling real-time systems with overrun and overload. *Proceedings of the eleventh euromicro conference on real-time system*s, pages 287–296, Jun 1999.

[7.] K. Gilad and S. Dennis. Dover: an optimal on-line scheduling algorithm for overloaded uni-processor real-time systems. *Electronics Letter*s, 33(15):1301–1302, July 1997.

[8.] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learnin*g. Addison-Wesley, MA, 1989.

[9.] F. Remeo, *Simulated Annealing: Theory and Applications to Layout Problem*s. PhD thesis, Dept. Of Elec. Eng.& Comp. Sci., University of California, Berkeley, Mar. 1989.

[10.] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE Transactions on Computer*s, 44(4), 1443–1451, Dec.1995.

[11.] M. Hamdaoui and P. Ramanathan. Evaluating Dynamic Failure Probability for Streams with (m, k)-firm Deadline. *IEEE Transactions on Computer*s, 46(12), pp.1325–1337, Dec.1997.

[12.] W. lindsay and P. Ramanathan, DBP-M, A Technique for Meeting end-to-end (m, k)-firm Guarantee requirements in point-to-Point networks, *Procs of IEEE Conference on Local networks,* pp.294-303*,* 1999.

[13.] P. Ramanathan. Overload management in Real-Time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed System*s, 10(6):549–559, Jun 1999.

[14.] C.C. Han and H.-Y. Tyan. A better polynomial-time schedulability test for Real-Time fixed-priority scheduling algorithms. *Proceedings of the Real-Time Systems Symposium*, pp36–45, 1997.

[15.] G. Koren and D. Shasha. Skip-over: Algorithms and complexity for overloaded system that allows skips. *Proceedings of Real-Time Systems Symposiu*m, pages 110–117, Dec. 1995.

[16.] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. *Proceedings of 1989 IEEE Real-Time System Symposiu*m, pages 166–171, 1989.

[17.] A. Striegel and G. Manimaran, Dynamic Class-Based Queue Management for Scalable Media Servers, to appear in *Journal of Systems & Software*

[18.] A. Striegel, G. Manimaran, Packet Scheduling with Delay and Loss Differentiation, *Computer Communications,* vol.25, no.1, pp.21-31, Jan. 2002.

[19.] A. Striegel, G. Manimaran, Best-effort Scheduling of (m,k)-firm Real-time Streams in Multihop Networks, *Workshop of Parallel and Distributed Real-Time Systems*, Mexico, 2000.

[20.] C. C. Chou, K.G Shin, Statistical Real-Time Channels on Multi-access Bus Networks, *IEEE Transaction on Parallel and Distributed Systems,* Vol.7(8), 769-780, 1997

[21.] R. West and C. Poellabauer, Analysis of a Window-Constrained Scheduler for Real-Time and Best-Effort Packet Streams, *Proc. of 21st IEEE Real-Time Systems*

[22.] R. West, K. Schwan, and C. Poellabauer, Scalable Scheduling Support for Loss and Delay Constrained Media Streams, *Proc. of IEEE Real-Time Technology and Applications Symposium,* 1999

[23.] B. Choi, D. Xuan, C. Li, R. Bettati, and Wei Zhao, Scalable QoS Guaranteed Communication Services for Real-Time Applications*, Proc. of the IEEE International Conf. on Distributed Computing Systems,* April 2000.

[24.] S. Wang, D. Xuan, R. Bettati, and Wei Zhao, Providing Absolute Differentiated Services with Statistical Guarantees in Static Priority Scheduling Networks, *Prods of IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2001.

[25.] B. Gaujal, N. Navet and J. Migge, Dual-Priority versus Background Scheduling: A Path-Wise Comparison, to appear in Journal of Real Time System

[26.] R. L. Cruz, Quality of Service Guarantees in Virtual Circuit Switched Networks, *IEEE Journal of Selected Areas in Communication*, 1995.

[27.] R. L. Cruz, A Calculus for Network Delay, *IEEE Transactions on Information Theory* (1, 2), January 1991.

[28.] P.J.B King, Computer Communication Systems Performance Modelling. Prentice Hall, 1990

[29.] C. Buttazzo and Giorgio, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Application, Kluwer Academic Publisher, 1997

[30.] H. Takagi, Queuing Analysis of Polling System: An Update in Stochastic Analysis of Computer and Communication System, Elsevier Science North Holland, Amsterdam, 267-318, 1990

[31.] R. Rom, M. Sidi, Multiple Access Protocols_ Performance and Analysis, Springer Verlag new York, 1990