QOS-DRIVEN SCHEDULING FOR MULTIMEDIA APPLICATIONS

Shaoxiong Hua and Gang Qu

Electrical and Computer Engineering Department and Institute of Advanced Computer Studies
University of Maryland, College Park, MD 20742
{shua, gangqu}@eng.umd.edu

ABSTRACT

Multimedia applications have intrinsic quality of service (QoS) requirements that may not be captured by the simple traditional completion ratio model. We have proposed a new quantitative QoS metric based on task completion ratio while differentiating firm and soft deadlines and taking data dependency into consideration. Using the decoding of MPEG movies as an example, we have shown that the proposed QoS metric is much better than completion ratio in measuring the quality of presentation (QoP) of the movies. Based on the new QoS metric, we present a set of new online algorithms that outperform popular scheduling algorithms (such as EDF, FCFS, and LETF) and enhance QoP significantly, particularly when the system is overloaded. All the proposed online algorithms have low computation overhead and can be easily integrated into real-time operating systems to improve multimedia embedded system's performance and/or to save system resources.

1. INTRODCUTION

In real-time multimedia and communication network research communities, quality of services (QoS) has been receiving wide attention because of the limited system resources (such as computing power and network bandwidth) and the tolerance to occasional data loss or deadline misses [5, 10]. Providing the required QoS guarantees becomes vital for the design of embedded systems that carry out real-time multimedia applications. The most popular way to specify time-related QoS requirements, such as synchronization and latency, is deadline. In hard real-time systems the deadlines are hard in the sense that missing deadlines will cause fatal errors of the system. However, in soft real-time systems such as multimedia systems, the occasional (firm or soft) deadline misses can be tolerated. Firm deadlines are timing constraints that must be satisfied for the system in order to get rewards. Missing soft deadlines, on the other hand, still can bring the system some rewards if the deadline failures are within an acceptable range. For instance, many MPEG video applications such as videoconferences require reliable communication and consistently high throughput, while being able to tolerate reasonable amount of packet error, jitter, or unsynchronization.

Task completion ratio [1, 4], which is equal to the percentage of completed tasks over all the requested tasks, has been widely used to measure QoS. However, it does not capture the firm/soft deadlines and data dependency that present in multimedia real-time applications. Therefore, it cannot accurately reflect the user-perceived quality of presentation (QoP). This leads us to define a new QoS metric that is a weighted sum of the rewards for completed tasks, the penalties for tasks completed after their soft deadlines, and the penalties for dropped tasks. We have recently shown that this new QoS metric describes QoP much more accurately than the completion ratio metric [7].

In this paper, we first modify several widely used on-line algorithms such as EDF (Earliest Deadline First), FCFS (First Come First Serve) and LETF (Least Execution Time First) [6] by using the drop policy based on the new QoS metric. However, the achieved QoS remains not so high because a significant portion of the completed tasks are actually computed incorrectly due to the factors such as data dependency. We then develop a set of new on-line scheduling algorithms to further improve the QoS. Experimental results show that the proposed schedulers such as IFF (important frame first) can enhance QoS and QoP significantly, particularly when the system is overloaded. They have low computation overhead and therefore can be easily integrated into multimedia embedded systems to deliver better QoS or to provide the same QoS with less system resources (CPU, power, memory, etc.).

The rest of the paper is organized as follows. In Section 2, we define the new quantitative QoS metric. Section 3 presents our on-line QoS-driven scheduling policies. In Section 4, we apply the general discussion to simulated MPEG movies and give the experiment results. We conclude the paper in Section 5.

2. A NEW QOS METRIC

We consider a system processing multimedia real-time applications. Each application consists of a sequence of tasks, and each task is characterized by $<a, d, e, f/s>$, where a is the arrival time, d is the deadline, e is the execution time that can be obtained a priori by pre-simulation or predicting [2, 3], and *f/s* specifies whether the deadline is firm or soft.

The system uses online schedulers to allocate system resources and processes the tasks in order to provide the required QoS. The traditional task completion ratio, which is defined as the ratio of the number of completed tasks over the total number of tasks according to the given scheduler, has been widely used in real-time systems. However, it may not give an accurate measure for the QoS due to the following reasons: 1) it does not distinguish the completion of tasks with firm deadlines and those with soft deadlines, on which the system may provide different QoS and get different rewards; 2) it does not distinguish tasks that are completed before their deadlines and those that are completed but miss their soft deadlines; and 3) it does not reflect data dependency among tasks because all deadline misses are treated

in the same way. Based on these observations, we define our new QoS for a scheduler S as follows:

$$Q(S) = \frac{\alpha_s K_s + \alpha_f K_f}{N} - \frac{\beta}{N}\sum \frac{\delta_i}{d_i - a_i} - \frac{\gamma}{N}\sum 1_i \Delta_i \qquad (1)$$

where N is the total number of tasks, K_s and K_f are the number of completed soft-deadline tasks and firm-deadline tasks, α_s and α_f are the weights for the completion of the soft-deadline task and firm-deadline task respectively (in general, $\alpha_s < \alpha_f$), β is the penalty parameter for deadline missing; δ_i is the difference between the task's deadline and completion time when the soft deadline is missed (if the task is completed before its deadline or eventually dropped, then δ_i is 0); d_i - a_i is the life time of the i-th task; γ is the penalty parameter for task dropping; 1_i =1 if the i-th task is dropped, otherwise 1_i = 0; Δ_i is the number of tasks that will be affected when the i-th task is dropped. In Equation (1), the first term rewards task completions; in the other terms, the first sum is taken over all the completed tasks that miss their soft deadlines; and the second sum is taken over all the dropped tasks regardless of their deadline type that captures data dependency.

In [7], we have shown that the traditional completion ratio does not measure user-perceived QoP properly and it cannot test different online schedulers. However, our new QoS metric is much closer to the QoP and it is necessary to develop low overhead online schedulers to maximize this new QoS metric in order to eventually improve QoP without using extra hardware.

3. ONLINE SCHEDULERS

In the overloaded systems, due to the uncertainty of the arriving tasks and the nature of online scheduling, it becomes unavoidable to drop tasks and hard to provide absolute QoS guarantees. Our objective is thus to develop online scheduling algorithms that give competitive average QoS. An online scheduler must have low complexity and should also specify its drop policy as the task drop becomes inevitable. In this section, we first give the drop lemma and then present a set of online scheduling heuristics based on the widely used EDF, FCFS and LETF.

Lemma (Drop Lemma): If a scheduler maximizes the QoS as defined in Equation (1), then it must

1) drop task <*a, d, e, f*> at time $t > d - e^*$
2) drop task <*a, d, e, s*> at time $t > \frac{\alpha_s + \gamma \times \Delta}{\beta}(d-a) + (d - e^*)$

where e^* is the task's remaining execution time, and $e^* = e$ for non-preemptive tasks.

[Proof] At time *t*, the earliest time that we can complete task <*a, d, e, f/s*> is $t+e^*$. If the task has a firm deadline *d*, it cannot be completed and will not contribute to QoS at time *t* when $t+e^* > d$. For the soft-deadline task, we will execute it when the benefit of completion (with deadline missing penalty if applicable) exceeds the penalty for dropping the task, that is, $\frac{\alpha_s}{N} - \frac{\beta}{N}\frac{\delta}{d-a} > -\frac{\gamma}{N}\Delta$ where $\delta = t + e^* - d$. A simple calculation leads us to 1) and 2) as above.

Intuitively, Drop Lemma suggests us to drop firm-deadline tasks as soon as we discover that we are unable to finish on time. However, for soft-deadline tasks, Drop Lemma implies that we should wait for an extra period because the soft deadline misses will still be beneficial to some extent. Clearly, the smaller is the deadline missing penalty parameter, and the larger is the weight of task completion or drop penalty, the longer we should wait.

3.1. EDF*, FCFS* and LETF*

The EDF, FCFS and LETF service strategies are among the most popular ones for real-time applications. On the completion of one task, they aggressively schedule the next task with the earliest deadline, the earliest arrival time and the least execution time respectively. However, neither of them distinguishes firm deadlines and soft deadlines and they may decide to execute the task that should be dropped according to the Drop Lemma. We integrate the Drop Lemma into these three scheduling policies and propose scheduling algorithms EDF*, FCFS* and LETF*.

Algorithm EDF*, FCFS* or LETF*:

(1) On the completion of the current task τ
(2) drop tasks according to the Drop Lemma;
(3) schedule the remaining tasks by using EDF, FCFS or LETF.

We are guaranteed that the completion of the task will either meet its deadline or still give positive contribution to the QoS even its soft deadline is missed. The reason is that the current task is the winner of all tasks in the previous round, which means that it survives the drop policy. During the drop policy checking in step 2, unlike the original schedulers, EDF*, FCFS* and LETF* will treat firm-deadline and soft-deadline tasks differently to maximize QoS. Finally, we argue that the drop policy checking takes only constant time. For example, in the implementation, we can first choose the task picked by EDF, FCFS or LETF, and check whether it meets the drop policy. If the drop policy is violated, we drop the task and ask EDF, FCFS or LETF for their next choice. Therefore, EDF*, FCFS* or LETF* will have the same run-time complexity as the original one.

3.2. S2F: Soft to Firm Deadline Conversion

From Drop Lemma, we see that task <*a, d, e, s*> and <*a*, $d + \frac{\alpha_s + \gamma \times \Delta}{\beta}(d-a)$, *e, f*> will always be dropped at the same time although they have different type of deadlines. Based on this observation we propose the following online scheduler:

Algorithm S2F:

(1) For each soft deadline task <a, d, e, s>
(2) change its deadline from d to $d + \frac{\alpha_s + \gamma \times \Delta}{\beta}(d - a)$;
(3) change its deadline type from soft to firm;
(4) apply EDF on the new set of firm-deadline tasks;

It converts soft deadline to firm and thus unifies task's deadline type. The advantage of this is that online scheduling algorithms do not need to treat different types of deadlines. Moreover, the Drop Lemma shows that whenever EDF achieves the best QoS, S2F also gives the best QoS.

3.3. IFF: Important Task (Frame) First

From Equation (1), we see that in order to maximize Q(s), we need to assign higher priority to important tasks. The larger are the task's dropping penalty and completion rewards, the more

important is that task. For example, missing firm deadline immediately erases the efforts that we have already put on the task completely. However, when we miss the soft deadline, we still get the chance to improve the QoS by finishing the task in a reasonable amount of extra time. Thus we should assign tasks with firm deadlines higher priority than those with soft deadlines. The IFF online scheduling algorithm is a variation of EDF based on this observation:

Algorithm IFF:

(1) On the completion of the current task τ
(2) drop tasks according to the Drop Lemma;
(3) untag the remaining tasks;
(4) select the untagged task τ' with the earliest deadline in the ready list;
(5) if τ' is not the most important untagged task in the ready list
(6) check the drop policy for untagged tasks that are more important than τ' by using time
t= current time + execution time of task τ';
(7) if there is a more important untagged task drop,
(8) unselect and tag τ' and goto step 4;
(9) schedule the current pick;

IFF is similar to EDF* with special treatment to important tasks such as firm-deadline tasks. A soft-deadline task will be processed only if its execution will not cause any firm-deadline task drops. Furthermore, IFF prioritizes the tasks that potentially contribute more to QoS among those with the same deadline types. Note that the calculation of the time at which the task will be dropped is a one-time effort and the operations in the loop (steps 4 - 8) are only simple additions and comparisons, therefore the computation overhead of IFF is low.

4. EXPERIMENTAL RESULTS

We have implemented the proposed QoS-driven online schedulers and applied them to decode a set of simulated MPEG movies [8]. In the simulation we compare the completion ratio (CR), which only considers the number of completed frames, our proposed new QoS metric, and quality of presentation (QoP), which can be conveniently measured by the number of correctly decoded frames by using different online schedulers. Our objective is to demonstrate that compared with some popular schedulers such as EDF, our proposed algorithms improve our new QoS metric and QoP dramatically.

Standard MPEG encoders generate three types of compressed frames: I frames, which are self-contained; P frames, which are dependent on the preceding I or P frames; and B frames, which are dependent on both preceding and succeeding I or P fames. In general, the encoders use a fixed GOP (Group of Pictures) pattern when compressing a video sequence. A typical GOP in display order and decoding order is shown as in Figure 1.

0	1	2	3	4	5	6	7	8	9	10	11	12	
I_0	P_1	B_2	B_3	P_4	B_5	B_6	P_7	B_8	B_9	I_{10}	B_{11}	B_{12}	decoding order
I_0	B_2	B_3	P_1	B_5	B_6	P_4	B_8	B_9	P_7	B_{11}	B_{12}	I_{10}	display order

Figure 1. A typical GOP pattern (I-to-I=12, I-to-P=3)

We simulate the frame information for movies, *Goldfinger, Wizard of OZ, Silence of the Lambs,* and *Star Wars,* based on the model presented by Krunz and Tripathi in [8]. In the simulation we assume that the execution time of MPEG decoding can be obtained a priori by predicting [2]. Furthermore, in some scenario such as the video-on-demand scenario we can get the exact information about the execution time directly from the user-data fields in the stream [3]. We also assume that the frames arrive in the decoding order and their inter-arrival times are independent with exponential distribution. The mean of the distribution is approximately equal to the reciprocal of frame display rate (in terms of fps or frame per second) to generate a balanced loaded system. We also simulate underloaded and overloaded systems by varying the fps requirement. The absolute deadline of each frame is monotonically increasing in its arrival time (in this case EDF and FCFS are the same). We use several standard display rates (in terms of fps) in our simulation: 15, 30, 45 and 60. The deadline type is assigned to each individual frame based on the frame dependency. We assign I and P frames firm deadlines and B frames soft deadlines to create tasks with mixed type of deadlines.

Each GOP can be viewed as one "application" independent of others as the correct decoding of all the frames in one GOP depends on the leading I frame. Each "application" consists of a set of tasks (frame decoding) and the drop of firm-deadline I and P frames will cause the incorrect decoding of the remaining frames in this "application". To better model the data dependency among "tasks", we assign different values Δ_I and $\Delta_{P,i}$, which are corresponding to the number of frames that will not be decoded correctly because of the dropped frame, to frames with firm deadlines. For example if I-to-I, the number of frames between two consecutive I frames (see Figure 1), is 12, then we assign Δ_I =11; $\Delta_{P,i}$ are assigned 10, 7, and 4 for the three P frames in the GOP pattern based on Figure 1; and $\Delta_B = 0$ because there is no frame depends on the B frame. As a result, I frames have higher priority than P and B frames; P frames have higher priority than B frames. This exactly matches the MPEG decoding mechanism. In sum, we use the following QoS, based on Equation (1) with consideration of MPEG application's characteristics, in our simulation:

$$Q_{MPEG}(S) = \frac{K_s + K_f}{N} - \frac{\beta}{N}\sum_{i=1}^{K_s}\frac{\delta_i}{T_d} - \frac{\gamma}{N}\left(m_I\Delta_I + \sum_{i=1}^{n_P} m_{P,i}\Delta_{P,i}\right) \qquad (2)$$

Where T_d *– the reciprocal of frame display rate;*

$\Delta_I, \Delta_{p,i}$ *– the number of tasks that will be affected if the I frame or P frame is dropped;*

$m_I, m_{P,i}$ *– the number of dropped I, P frames;*

n_P *– the number of P frames in a GOP pattern.*

$K_s, K_f, \beta, \gamma, \delta_i, N$ *are same as in* (1).

Note that this QoS measurement is calculated incrementally at run time and there are only a few arithmetic operations involved at each frame. The penalty parameter β and γ are stream-specific. For example, the β and γ for decoding Cartoon Video (e.g., 0.8) should be smaller than those for decoding Action Video (e.g., 1.0) because the human being are less sensitive to the artificial movements in Cartoon Video, but are very sensitive to the smoothness of the motions in Action Video [9]. The values of these parameters can be stored as user-defined data within the stream. In the simulation, β and γ are both set to be the default value 1.

We applied the proposed online scheduling algorithms to the simulated MPEG movies under different frame rates. For underloaded system with a frame rate of 15fps, the deadlines are relatively loose and we observe that almost all the algorithms achieve the maximal QoS and QoP (in the amount of 1) without task dropping and deadline missing. However, when the computation load increases, the system becomes balanced and overloaded eventually. Then we see, for instance in the movie of "Goldfinger" as shown in Figure 2, different online schedulers provide very different QoP which have the same trends as the new defined QoS measurement. In general we can rank them in the increasing order of QoP (or QoS): LETF*, EDF, EDF*, S2F, and IFF. When the system goes to the overloaded state (such as 45fps and 60fps), the algorithm IFF achieves significant higher QoS and user-perceived QoP comparing to other algorithms.

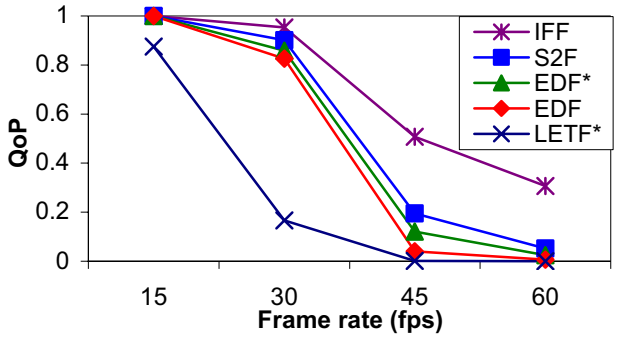

Figure 2. Comparisons of QoP under different online schedulers on movie "Goldfinger".

It is of our particular interest to study overloaded systems where task dropping and deadline missing become unavoidable. As an example Figure 3 gives the detailed reports on the completion ratio (CR), the new QoS metric as defined in Equation (2) and QoP, achieved by different schedulers on four movies in the frame rate of 30fps. We mention that the negative QoS comes from the fact of task dropping and deadline missing as well as their associated penalties. From the figure we can see that almost all the schedulers achieve similar performance for the CR, however, they behave very differently under the new QoS metric and QoP. The conclusion is that it is crucial to finish important tasks as many as possible, not the raw counter of task completions. It is mentioned that although LETF algorithm is 1/2–competitive in the completion ratio on our monotonic-absolute-deadline task system [1], LETF*, which is better than LETF in terms of QoP, achieves very bad user-perceived QoP. The reason is that in general, the execution times of B frames are shorter than those of I or P frames. Therefore, LETF* prefers to select B frames, which actually are the least important frames.

5. CONCLUSIONS

We have presented a new metric on how to measure the QoS provided by an embedded system for real-time multimedia applications with mixed firm and soft deadlines. Unlike from the traditional task completion ratio, the new QoS metric captures the deadline nature of such applications and models data dependency as well. We have shown that the new QoS metric can reflect user-perceived QoP much better than the task completion ratio metric. Based on the proposed QoS metric, we develop a set of online scheduling algorithms to maximize it. Simulations on MPEG movies show that compared with popular online scheduling policies such as EDF and LETF, most of the proposed algorithms achieve much better QoS and user-perceived QoP with low computation overhead and no extra hardware.

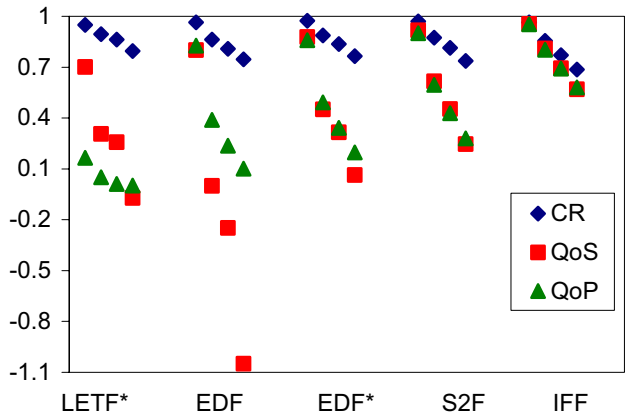

Figure 3: Comparisons of different online schedulers on four movies (for each scheduler, from left to right, Movie "Goldfinger", "Wizard of OZ", "Silence of Lambs", and "Star Wars") in the frame rate of 30fps (The vertical axis represents the values of different metrics).

REFERENCES

[1] S. K. Baruah, J. Haritsa, and N. Sharma, "On-Line Scheduling to Maximize Task Completions", *IEEE Real-Time Systems Symposium*, pp. 228-236, Dec. 1994.

[2] A. C. Bavier, A. B. Montz, and L. L. Peterson, "Predicting MPEG execution times", *SIGMETRICS'98*, pp.131-140, June 1998.

[3] L. -O. Burchard and P. Altenbernd, "Estimating Decoding Times of MPEG-2 Video Streams", *Proc. of International Conference on Image Processing,* pp. 560-563, Sep. 2000.

[4] G. Buttazzo, M. Spuri, and F. Sensini, "Value vs. Deadline Scheduling in Overload Conditions", IEEE *Real-Time Systems Symposium,* pp. 90-99, 1995.

[5] R. L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 3, No. 6, pp.1048-1056, Aug. 1995.

[6] A. Fiat, G.J. Woedinger (eds), "On-Line Algorithms: the State of the Art", Springer, Berlin, Germany, 1998.

[7] S. Hua and G. Qu, "A New QoS Metric for Hard/Soft Real-Time Applications", *ITCC,* pp. 347-351, April 2003.

[8] M. Krunz and S.K. Tripathi, "On the characterization of VBR MPEG streams", *SIGMETRICS*, pp.192-202, 1997.

[9] J. K. Ng, K. R. Leung, W. Wong, V. C. Lee and C. K. Hui, "Quality of Service for MPEG Video in Human Perspective", *RTCSA,* pp. 233-241, March 2002.

[10] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A Resource Allocation Model for QoS Management", *IEEE Real-Time Systems Symposium*, pp. 298-307, 1997.