On the schedulability analysis for dynamic QoS management in distributed embedded systems

L. Almeida[1], R. Marau[1], K. Lakshmanan[2], and R. Rajkumar[2]

[1] IEETA / IT - DEEC - Faculdade de Engenharia
University of Porto, Porto, Portugal
{lda,marau}@fe.up.pt
[2] Real-Time and Multimedia Systems Lab
Carnegie Mellon University, Pittsburgh, PA, USA
{klakshma,raj}@ece.cmu.edu

Abstract. Dynamic Quality-of-Service (QoS) management has been shown to be an effective way to make an efficient use of systems resources, such as computing, communication or energy. This is particularly important in resource-contrained embedded systems, such as vehicles, multimedia devices, etc.. Deploying dynamic QoS management requires using an appropriate schedulability test that is fast enough and ensures continued schedulability while the system adapts its configuration. In this paper we consider four utilization-based tests with release jitter, a particularly relevant feature in distributed systems, three of which were recently proposed and one is added in this work. We carry out an extensive comparison using random task sets to characterize their relative merits and we show a case study where multiple video streams are dynamically managed in a fictitious automotive application using such schedulability bounds.

Keywords: real-time scheduling, Quality-of-Service, distributed embedded systems

1 Introduction

Utilization-based schedulability tests are known since the 1970s [9] and they are extremely simple to compute given their linear time complexity. When used on-line in the scope of admission controllers of open systems, their time complexity can even be made constant by keeping track of the total utilization of the currently running tasks. However, this simplicity comes at a cost, which is a lower accuracy than other more complex schedulability tests, such as those based on response time or processor demand analysis. This disadvantage grows as the task model considers more aspects that are typical in real applications such as blocking, deadlines different from periods, offsets, non-preemption and arbitrary priority assignment policies [4].

Furthermore, recent developments in response time analysis brought to light new linear methods that are faster to compute, despite a small loss in accuracy [7] [3]. These tests are still more complex to evaluate than utilization-based tests but the difference became relatively small and their accuracy is still better.

Nevertheless, utilization-based tests are still the most adequate solution when it is important to manage utilization, for example in the scope of dynamic QoS (bandwidth) management, with tasks that allow variable worst-case execution times by switching between different algorithms or that can be executed at different rates. Therefore, those tests are an effective way to improve the efficiency in the use of system resources when the task set in the system can change on-line, either due to admission/removal of tasks or because the tasks exhibit multiple modes of operation.

In such scope, utililzation-based tests are the natural choice because they handle bandwidth directly. Thus, for example, when a task leaves the system, the bandwidth that is freed can be assigned to the remaining ones, according to an adequate policy, from fixed priority/importance, to elastic, weighted, even, etc. [5] [8]. This assignment cannot be done in a comparable manner using other kind of tests and it must be done in a more costly trial and error fashion.

When applying dynamic bandwidth management to distributed real-time systems there is one additional difficulty, which is the frequent occurrence of release jitter, both in the tasks and messages they use to communicate. In fact, messages are frequently released by tasks that do not execute periodically because of variable interference of higher priority ones in the processor where they reside and, similarly, tasks are frequently triggered by the reception of messages that do not arrive periodically for similar reasons [11].

The incorporation of release jitter in utilization-based tests was not done until very recently. In this paper we revisit the existing work on utilization-based schedulability tests that account for release jitter, we propose an enhancement to a previous test and we compare the performance of such tests with random task sets. Then we show a case study concerning a video system for driver assistance in a fictitious automotive application that illustrates the benefits of using such a schedulability test for dynamic management of communications in an Ethernet switch.

2 Related work

Despite all the work on utilization-based schedulability tests for periodic task models carried out in the past four decades, to the best of the authors' knowledge, there has never been an extension to include release jitter until very recently. In fact, the first published result in that direction seems to have been by Davis and Burns in 2008 [6] where they showed a pseudo-utilization-based schedulability test that accounts for release jitter based on the results in [1]. In fact, it is shown that the release jitter that affects each task can be subtracted from its period, or deadline if shorter than the period, and used directly in the denominator of the respective utilization term. The total sum gives a pseudo-utilization value that can be compared against the Liu and Layland bounds. The same work has an extensive comparison among different tests, focusing on the relative performance of response-time based tests.

On the other hand, the authors have previously developed a different analysis applicable to both Rate-Monotonic and Earliest Deadline First scheduling that accounts for release jitter as an extra task [10]. This analysis results in a set of n conditions for n tasks. However, the authors also showed how such conditions could be reduced to just one single test that upper bounds the n conditions.

The comparison between the n conditions test and the test in [6] is not obvious, as it will be shown further on, since none dominates the other in all situations. Nevertheless, the latter does not directly reflect bandwidth, which makes it harder to use within the scope of dynamic bandwidth management schemes. Moreover, the comparison becomes even less obvious given the different underlying priority assignment policies. In fact, while the former assumes usual *rate-monotonic* priorities, the latter assumes a *'deadline minus jitter'-monotonic* policy. Naturally, either test can be optimistic whenever used with a priority assignment different from the assumed one.

In this paper we propose a different simplification of the n conditions test proposed in [10] that also results in a single bandwidth-based condition but which is more accurate than the one presented therein at the cost of additional overhead. The remainder of the paper will focus on a comparison of the presented tests.

3 Task model and previous analysis

We consider a set Γ of n periodic tasks $\tau_i (i = 1..n)$, with period T_i and worst-case execution time C_i, which may suffer a jittered release of at most J_i. The deadlines are currently considered to be equal to the respective periods. We consider the tasks in the set to be sorted by growing period, except when explicitly referred. In the context of Rate-Monotonic scheduling (RM) τ_1 will be the highest priority task and in the context of Earlist Deadline First scheduling (EDF) τ_1 will be the task with the highest preemption level. For the moment we also consider tasks to be fully preemptive and independent.

The test in [6] copes with deadlines equal to or less than the periods and blocking but, for the sake of comparison with the other tests, we will constrain it to our model. In this case we will assume the task set sorted by growing 'period minus jitter'. The test then states that schedulability of the task set is guaranteed if condition (1) is satisfied. The test in [6] considers fixed priorities, only, but the same reasoning expressed therein allows to infer that it should also be applicable to EDF scheduling. Thus, we will also consider it in such case. $U^{lub}_{RM,EDF}(n)$ refers to the least upper bound on utilization for RM scheduling with n tasks and for EDF, i.e., $n(2^{\frac{1}{n}} - 1)$ and 1, respectively.

$$\sum_{i=1} \frac{C_i}{T_i - J_i} \leq U^{lub}_{RM,EDF}(n) \tag{1}$$

On the other hand, the test in [10] states that if the n conditions in (2) are satisfied, then the task set is schedulable.

$$\forall_{i=1..n} \sum_{j=1}^{i} \frac{C_j}{T_j} + \frac{\max_{j=1..i} J_j}{T_i} \leq U^{lub}_{RM,EDF}(i) \tag{2}$$

The work in [10] also provides a simplified test based on a single condition as in (3).

$$\sum_{i=1}^{n} \frac{C_i}{T_i} + \frac{\max_{i=1..n} J_i}{T_1} \leq U^{lub}_{RM}(n) \tag{3}$$

This condition may become very pessimistic in cases with a wide dispersion of tasks periods. In fact, a task with a longer period can accommodate a longer release jitter, leading to a term $\max_{i=1..n} J_i$ that can be close to, or longer than, T_1. This situation can be improved using a new simplified test with a single condition (4) that is more accurate than (3) despite more costly to compute.

$$\sum_{i=1}^{n} \frac{C_i}{T_i} + \max_{i=1..n} \frac{\max_{j=1..i} J_j}{T_i} \leq U^{lub}_{RM,EDF}(n) \tag{4}$$

It can be trivially verified that if condition (4) holds then all n conditions in (2) will also hold and thus this test also implies the schedulability of the task set. Note, also, that the term $\max_{i=1..k} J_i$ is now divided by $T(k)$ that grows with k and thus the potentially longer jitter of slower tasks does not have such a strong effect as in condition (3).

4 Comparing the tests

In this section we compare the four conditions presented above in terms of schedulability accuracy, i.e., level of pessimism, and execution time. For both cases, we include one accurate analysis as a reference, namely a response time test for RM as in (5) [2] and a CPU demand test for EDF based on the analysis presented in [11] as in (6). For the sake of presentation, we will also refer to the test of condition (1) as RM whenever applied to fixed priorities, although these being assigned in 'deadline minus jitter' order.

$$\forall_{i=1..n} Rwc_i \leq T_i \textbf{ with } Rwc_i = R_i + J_i \textbf{ and } R_i = \sum_{j=1}^{i-1} \left\lceil \frac{R_i + J_j}{T_j} \right\rceil * C_j + C_i \tag{5}$$

$$\forall_{t\in\Psi} h(t) \leq t \textbf{ with } h(t) = \sum_{i=1..n} \left\lfloor 1 + \frac{t-(T_i-J_i)}{T_i} \right\rfloor * C_i \quad (6)$$

$$\textbf{and } \Psi = \{t \in [0,L] \wedge t = m * T_i + (T_i - J_i), \forall i = 1..n, m = 0,1,2...\}$$

$$\textbf{with } L = \sum_{i=1..n} \left\lceil \frac{L+J_i}{T_i} \right\rceil * C_i$$

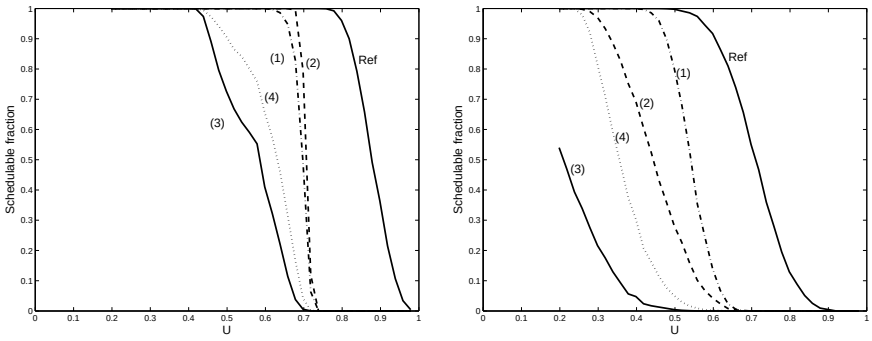
For the comparison we use synthetic task sets with random tasks. The generation method is the following. We start by generating a target global utilization (U), then we generate random tasks one by one with a period (T_i) uniformly distributed within $[1, 10]$ and utilization factor (U_i) within $(0, 0.2]$. The utilization of the last task is adjusted if the total utilization is more than 1% above the desired target U. Then, we compute the respective execution times ($C_i = T_i * U_i$) and we generate the values of release jitter (J_i) randomly with uniform distribution.

The values of U go from 0.2 to 0.98 in steps of 0.02 and for each particular point we generate 5000 random task sets. We gerenated two sets of experiments with different patterns of jitter. In one case we considered the same interval in the generation of jitter for all tasks, namely randomly distributed within $(0, 0.3]$. This is a relatively small jitter that impacts mainly the tasks with shorter periods. We call this the *flat* jitter profile. On the other hand, we generated another profile with a stochastically growing value of jitter according to the taks periods. Basically, the jitter J_i is randomly generated witin $(0, 0.5 * T_i]$. This means that longer tasks can suffer longer release jitter. We call this, the *linear* jitter profile.

The simulation experiments were carried out using Matlab on a common laptop with a Centrino CPU operating $1.2GHz$ and running the Windows$_{XP}$ operating system. The execution times of the analysis that are shown next were not optimized and serve mainly for relative assessment.

Figure 1a shows the results obtained using the RM scheduling and the flat jitter profile. The rightmost curve shows the ratio of task sets that meet the response time test in (5) with both 'deadline minus jitter' priorities and rate-monotonic priorities. In this case, given the low amount of jitter, its impact in the priority assignment is minimal and, consequently, the response time test delivers very similar results with both policies (Ref1/2). This reference test assures that all task sets with this jitter profile and utilization roughly up to 74% are schedulable. Then we track how many of the task sets found schedulable by the reference test were also deemed schedulable by tests (1) to (4). This gives us a good measure of the level of pessimism embedded in these tests.

The values obtained for RM scheduling with the flat jitter profile are shown in Table 1. Test (2) performed best in this case, finding 75% of the sets considered schedulable by the reference test. Test (1) performed very closely, 73%. Then come tests (4) and (3) fiding 62% and 55% of the reference schedulable sets, respectively. It is expected that these two tests perform poorer than test (2) since they are a simplification of it and it is also expected that both tests perform similarly since with the flat jitter profile there is a non-negligible probability that



(a) with the flat jitter profile. (b) with the linear jitter profile.

Fig. 1: Performance of different tests under RM

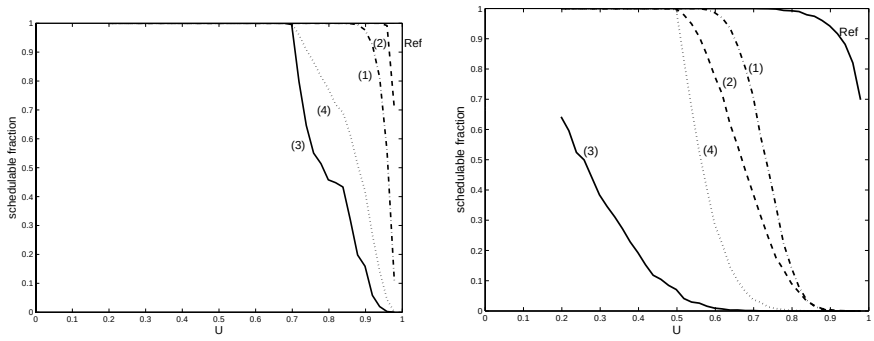
the maximum jitter will affect the task with the shortest period, which leads to a similar result with both tests. Table 1 also shows the average and maximum time taken by each analysis with each set.

	Ref 1/2	Test (1)	Test (2)	Test (3)	Test (4)
% of schedulable task sets found by the U-based tests	100%	73%	75%	55%	62%
analysis exec time (avg)	14.8ms	1ms	9ms	1.4ms	4.7ms
analysis exec time (max)	96ms	47ms	79ms	47ms	64ms

Table 1: Summary of results with the flat jitter profile under RM

Interestingly, the situation changes substantially when we use the linear jitter profile, showing the expected strong impact of release jitter. In this case, the tasks with longer period may be affected by a longer release jitter in absolute terms, particularly, longer than the shorter periods in the system. Therefore, the imapct of the term $\max J$ in either tests (2), (3) and (4) can be significant, specially in test (3) in which such term is just divided by the shortest period. Consequently, this test becomes useless in practice whenever there are large release jitters in the system when compared to the shortest period. Conversely, the new test (4) that we present in this paper achieves a performance inferior but closer to the original n conditions test (2), since the stochastically growing terms of jitter are also divided by growing periods. The best performance with this jitter profile was, however, achieved with test (1).

The results can be observed in Figure 1b and in Table 2. Note that in this case, the larger jitter already causes a very slight difference in the reference test depending on whether it uses 'deadline minus jitter' (Ref1) or rate-monotonic priorities (Ref2). Such difference favours the former case given the optimality of that priority assignment in the presence of jitter [1]. In the Table, the results of



(a) with the flat jitter profile. (b) with the linear jitter profile.

Fig. 2: Performance of different tests under EDF

test (1) are shown with respect to those of reference test Ref1 and those of tests (2) to (4) with respect to reference test Ref2.

	Ref1 & Ref2	Test (1)	Test (2)	Test (3)	Test (4)
% of schedulable task sets found by the U-based tests	100%	68%	50%	11%	34%
analysis exec time (avg)	14.1ms	0.9ms	6.8ms	1.5ms	4.3ms
analysis exec time (max)	94ms	47ms	78ms	47ms	48ms

Table 2: Summary of results with the linear jitter profile under RM

The results for EDF scheduling are shown in Figures 2a and 2b for the flat jitter and linear jitter profiles, respectively. In this case, just one reference test was used (Ref) since the test in 6 is independent of the order in which the tasks are considered. The numerical results for both cases are shown in Tables 3 and 4. Basically, we achieved very similar results to those of the RM scheduling case but with a shift to the right due to the higher schedulability capacity of EDF, thus with an increment in the percentages of schedulable configurations found. Anyway, the relative performances of the diverse tests are similar to those achieved with RM.

	Reference	Test (1)	Test (2)	Test (3)	Test (4)
% of schedulable task sets found by the U-based tests	100%	96%	99%	77%	84%
analysis exec time (avg)	47ms	1ms	8.8ms	1.5ms	4.9ms
analysis exec time (max)	159ms	47ms	79ms	47ms	62ms

Table 3: Summary of results with the flat jitter profile under EDF.

	Reference	Test (1)	Test (2)	Test (3)	Test (4)
% of schedulable task sets found by the U-based tests	100%	69%	62%	13%	49%
analysis exec time (avg)	76ms	1ms	8.3ms	1.5ms	4.7ms
analysis exec time (max)	204ms	47ms	110ms	47ms	62ms

Table 4: Summary of results with the linear jitter profile under EDF

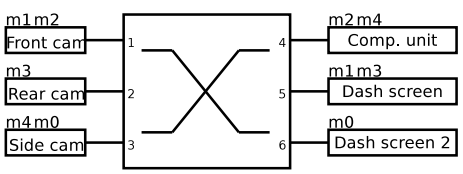
As a concluding remark, we can say that test (1) seems to perform better in the general case, only losing to test (2) with relatively low release jitter (flat jitter profile). This is expected since as jitter disappears, all single condition tests converge to the same result, which is more pessimistic than that achieved with the n-conditions test. Test (3) seems usable for situations with low release jitter and similar periods, only. In the general case, with a linear jitter profile, tests (2) and (4) seem to apply, presenting a tradeoff between pessimism, less with test (2), and computing overhead, less with test (4). As the release jitter becomes smaller and closer to the flat jitter profile, the performance of all tests improves substantially while their overheads stay approximately constant. Finally, it is also interesting to note the stronger impact that jitter has on RM scheduling with respect to EDF, which was also expected given the higher scheduling capacity of the latter.

5 Application example

In order to illustrate the use of such fast schedulability tests in the context of dynamic QoS management, we will consider a case study developed around a fictitious automotive application for driver assistance relying on video transmissions [12]. To cope with the required bandwidth the system uses switched Ethernet [13] enhanced with the FTT-SE protocol, Flexible Time-Triggered communication over Switched Ethernet, to provide deterministic dynamic adaptation and reconfigurable virtual channels with guaranteed QoS. The purpose is to redistribute bandwidth released by cameras that are temporarily switched off among the remaining active cameras, improving the QoS that they get at each moment by maximizing the resource usage (links bandwidth).

In this application scenario several video sources are connected to the Ethernet switch, see Fig. 3a. The front cameras node produces two streams (m_1 and m_2) that are used for obstacle detection, lane departure warning and night vision, and which are connected to the dashboard screen and the video processing unit, respectively. The side cameras (streams m_0 and m_4) are used for obstacle detection in the blind spot, and are connected to the processing unit that may then trigger alarms, and to a secondary screen in the dash board, respectively. The rear view camera (stream m_3) connects to the dash board screen for driving assistance, when reversing.

The bandwidth distribution mechanisms follows a greedy approach that distributes the available bandwidth in a fixed priority order (according to parameter



(a) Architecture.

	$C_i \times 100Mbps$	T_i	$P_{QoS,i}$	S_i	R_i
m_0	90kB-200kB	40ms	0	3	6
m_1	100kB-200kB	40ms	1	1	5
m_2	100kB-200kB	40ms	2	1	4
m_3	100kB-200kB	40ms	3	2	5
m_4	100kB-200kB	40ms	4	3	4

(b) Requirements.

Fig. 3: Application scenario.

$P_{QoS,i}$). Each of the video streams operates under a constant frame rate of 25fps ($1/T_i$). The channel width ($C_i^w = C_i \times$ 100Mbps) can vary between 100kB (90kB for m_0) and 200kB. These values correspond to a bandwidth per channel that can vary between 20Mbps (18Mbps for m_0) and 40Mbps. Table 3b summarizes the streams properties.

The links have a capacity of 100Mbps each but the traffic scheduling model of FTT-SE imposes several constraints and overheads that reduce that capacity to 90Mbps. Nevertheless, it allows using any traffic scheduling policy, and considering full preemption. Therefore, we will use EDF, which allows us to make a full use of the 90Mbps of each link.

The streams generation model is periodic without jitter, which allows a direct use of the Liu and Layland utilization bounds [9] to assess the schedulability of the traffic in the uplinks (nodes to switch connection). Using EDF, the bound equals the link capacity, i.e., 90Mbps, and since this is higher than the worst-case bandwidth requirement in any uplink, which is 2*40Mbps, the traffic is schedulable in any case.

However, as explained in [10], when there are several streams sharing an uplink but going to different downlinks, the traffic in the downlinks may arrive jittered. In fact, despite having similar periods, there is no synchronization between the cameras which send frames in self-triggered mode. This means that the frames of the streams can arrive at the switch with any (and variable) relative phase thus generating a variable interference pattern that leads to the referred jittered release in the downlinks. Consequently, the traffic schedulability in the downlinks must be assessed with the tests that account for release jitter as those discussed earlier in the paper.

In this case, we must start by determining the amount of jitter that may affect each stream. We can see that stream m_3 does not suffer interference on its uplink, arriving periodically at the switch, thus $J_3 = 0$. However, all other streams suffer interference of another stream in the respective uplink, thus creating an interference as large as one frame. Table 5 shows the release jitter that can affect each stream on downlinks $j = 4..6$. Note that when C_i changes so does J_i created by that stream.

A direct application of the schedulability tests (1) through (4), with the minimum requirements, will deliver a positive result in all downlinks. This means

	J_0	J_1	J_2	J_3	J_4
$j = 4$	x	x	C_1	x	C_0
$j = 5$	x	C_2	x	0	x
$j = 6$	C_4	x	x	x	x

Table 5

that the system can always meet those requirements. However, when testing schedulability with the maximum requirements all tests deliver a negative result in downlinks 4 and 5. This means it is not possible to satisfy the maximum requirements of all streams simultaneously and some adaptation must be put in place.

In this case, we use a simple greeady approach according to which we start reducing the QoS of the least priority streams until we can meet the resources capacity. At this point, a significant difference between tests (1) and (2) on one hand, and tests (3) and (4) on the other, is that with the former we do not know exactly the bandwidth in excess so that we directly determine the new distribution of bandwidth that allows us to meet the resource capacity. Basically, test (1) uses terms that do not directly represent bandwidth and test (2) uses n conditions and thus we cannot determine the bandwidth overload that must be removed from the system. Using these tests to carry out dynamic bandwidth management requires a trial and error approach that can be costly in number of iterations and thus implying high overhead. This, nevertheless, requires further assessment since it will substantially depend on the specific search technique used.

Therefore, we will use test (3), which, in a set with equal periods, is equivalent to test (4). In this case, the total utilization plus the virtual utilization corresponding to the jitter terms in both links 4 and 5 reaches 120Mbps, i.e., 30Mbps above the links capacity. In order to eliminate this overload we start reducing the QoS of m_0 to the minimum (18Mbps) which causes J_4 to be reduced accordingly (Table 5).

This has no impact on the overload and thus we need to reduce the QoS of m_1 to the minimum (20Mbps), leading to a corresponding reduction in J_2. This reduction from 40Mbps to 20Mbps impacts directly both links 4 and 5 and present now an overload of 10Mbps beyond the links capacities.

Thus, we still need to adjust the QoS of the following stream m_2 in order to reduce the 10Mbps overload from its QoS. Now, note that m_2 affects link 4 directly and link 5 indirectly vis J_2. Thus, by simply reducing the bandwidth of this stream from 40 to 30Mbps allows meeting exactly the link capacity of 90Mbps in both links 4 and 5. The final QoS assigned to each stream is given in Table 6.

Now, imagine that at a certain point in time both streams m_0 and m_1 are temporarily switched off. This implies that the associated jitter terms, J_0, J_1, J_2 and J_4, are nullified, which means the system becomes free of jitter. This is easy to see in Figure 3a since in such a configuration there will be one single stream coming from each node in each uplink. Running the schedulability test in all links will yield a positive result for the maximum requirements of the active

	$P_{QoS,i}$	$U_{i,min}$	$U_{i,extra}$	U_i
m_4	4	$20Mbps$	$20Mbps$	$40Mbps$
m_3	3	$20Mbps$	$20Mbps$	$40Mbps$
m_2	2	$20Mbps$	$10Mbps$	$30Mbps$
m_1	1	$20Mbps$	$0Mbps$	$20Mbps$
m_0	1	$18Mbps$	$0Mbps$	$18Mbps$

Table 6

streams, which can thus operate at highest QoS, as shown in Table 7. Links 1, 2, 3 and 5 will be used at 40Mbps, link 4 will be used at 80Mbps and link 6 will be unused.

	$P_{QoS,i}$	$U_{i,min}$	$U_{i,extra}$	U_i
m_4	4	$20Mbps$	$20Mbps$	$40Mbps$
m_3	3	$20Mbps$	$20Mbps$	$40Mbps$
m_2	2	$20Mbps$	$20Mbps$	$40Mbps$

Table 7

If one of the disconnected streams is connected again, the schedulability test must be executed for all links to determine whether that would create an overload and, if necessary, adjust the QoS of the active and arriving streams in order to make room for the arriving one while maximizing the usage of the links capacity and avoiding the overload.

6 Conclusion

Growing interest on improving the resource efficiency in embedded systems is pushing towards the use of on-line adaptation and reconfiguration. For example, dynamic bandwidth management, or more generally dynamic QoS management, may allow maximizing the use of a given resource, e.g. cpu or network, by a dynamic set of entities. When these systems have real-time requirements, the on-line adaptations must be carried out with the assistance of an appropriate schedulability analyzer that assures a continued timely behavior.

In this paper we have analyzed several utilization-based schedulability tests, one of which proposed herein, that incorporate release jitter. This feature was introduced very recently and allows their use in distributed systems where release jitter appears naturally. Moreover, they are particularly suited to be used on-line given their low computational overhead.

Therefore, we carried out an extensive simulation with random task sets to characterize the level of pessimism and computational overhead of the schedulability tests. We determined the most adequate utilization scenarios for each of the remaining tests. Finally, we illustrated the use of these tests in a fictitious automotive application involving the scheduling of several video streams.

Acknowledgments

This work was partially supported by the iLAND project, call 2008-1 of the EU ARTEMIS JU Programme and by the European Community through the ICT NoE 214373 ArtistDesign.

References

1. A. Zuhily, A. Burns : Optimality of (D-J)-monotonic Priority Assignment. Information Processing Letters 103(6), 247–250 (Sept 2007)
2. Audsley, N., Burns, A., Richardson, M., Tindell, K., Wellings, A.J.: Applying New Scheduling Theory to Static Priority Pre-Emptive Scheduling. Software Engineering Journal 8, 284–292 (1993)
3. Bini, E., Baruak, S.K.: Efficient computation of response time bounds under fixed-priority scheduling. In: Proc. of 15th Int. Conf. on Real-Time and Networked Systems (RTNS'07) (Mar 2007)
4. Buttazzo, G.C.: Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series). Springer-Verlag TELOS, Santa Clara, CA, USA (2004)
5. Buttazzo, G.C., Lipari, G., Caccamo, M., Abeni, L.: Elastic Scheduling for Flexible Workload Management. IEEE Trans. on Computers 51(3), 289–302 (2002)
6. Davis, R.I., Burns, A.: Response Time Upper Bounds for Fixed Priority Real-Time Systems. In: Proc. of the 29th Real-Time Systems Symposium (RTSS'08). pp. 407–418. IEEE Computer Society, Washington, DC, USA (2008)
7. Fisher, N., Nguyen, T.H.C., Goossens, J., Richard, P.: Parametric Polynomial-Time Algorithms for Computing Response-Time Bounds for Static-Priority Tasks with Release Jitters. In: Proc. of the 13th IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA'07). pp. 377–385. IEEE Computer Society, Washington, DC, USA (2007)
8. Ghosh, S., Rajkumar, R.R., Hansen, J., Lehoczky, J.: Scalable QoS-Based Resource Allocation in Hierarchical Networked Environment. In: Proc. of the 11th IEEE Real Time on Embedded Technology and Applications Symposium (RTAS'05). pp. 256–267. IEEE Computer Society, Washington, DC, USA (2005)
9. Liu, C.L., Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the Association for Computing Machinery 20(1), 46–61 (1973)
10. Marau, R., Almeida, L., Pedreiras, P., Lakshmanan, K., Rajkumar, R.: Utilization-based Schedulability Analysis for Switched Ethernet aiming Dynamic QoS Management . In: Proc. of the 15th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA'10) (Sep 2010)
11. Palencia, J.C., Harbour, M.G.: Response time analysis of EDF distributed real-time systems. J. Embedded Comput. 1(2), 225–237 (2005)
12. Rahmani, M., Steffen, R., Tappayuthpijarn, K., Steinbach, E., Giordano, G.: Performance analysis of different network topologies for in-vehicle audio and video communication. In: 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks (IT-NEWS'08). pp. 179–184 (13-15 2008)
13. Sommer, J., Gunreben, S., Feller, F., Kohn, M., Mifdaoui, A., Sass, D., Scharf, J.: Ethernet - A Survey on its Fields of Application. Communications Surveys Tutorials, IEEE 12(2), 263–284 (second 2010)