Modeling Fixed Priority Non-Preemptive Scheduling with Real-Time Calculus

Devesh B. Chokshi and Purandar Bhaduri
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati, Guwahati - 781039, India
{*chokshi, pbhaduri*}*@iitg.ernet.in*

Abstract

Modern real-time embedded systems are highly heterogeneous and distributed. As a result, compositional methods play an important role in the design and analysis of such complex systems. One such compositional analysis method is based on Real-Time Calculus [4, 14]. In this paper, we present an analysis of fixed priority non-preemptive scheduling with the Real-Time Calculus. Although fixed priority non-preemptive scheduling was modeled with the Real-Time Calculus previously [7], we show that the model gives overly pessimistic results. We also compare our analysis with the existing holistic scheduling analysis [10, 3] through an example of a system using a Controller Area Network (CAN) bus [5]. The proposed method can be automated by incorporating it in the RTC Toolbox [13].

1. Introduction

The constraints imposed by application requirements in domains like automotive and avionics very often lead to complex distributed real-time embedded systems. Such systems are heterogeneous in terms of the underlying execution platform, different activation rates and execution demands of tasks and different scheduling and arbitration policies of processing and communication resources. Based on abstractions of the given architecture and application, various analytic methods can be used to obtain hard bounds on properties of such systems like end-to-end delays, buffer requirements and throughput. But one of the drawbacks of these methods is their limited modeling capacity which can lead to pessimistic (but still correct) results.

One such method is known as "holistic scheduling analysis" [12, 10] and is based on classical scheduling theory which uses classical models of task arrivals such as periodic, periodic with jitter, etc. Richter et al. [11] gave a more general and modular approach to extend the concepts of classical scheduling theory to heterogeneous distributed systems. But it is also based on standard event models and hence is not able to model arbitrary arrival patterns.

A different modular performance analysis approach, Real-Time Calculus [4], was proposed by Thiele et al. Real-Time calculus extends the basic concepts of network calculus [2]. The main advantage of this framework is that it supports any arbitrary arrival pattern of events and thus is not restricted to standard event models.

As discussed in [7], fixed priority non-preemptive scheduling (FPNS) often finds a role in embedded systems. For example, the Controller Area Network (CAN) [5] in which messages between electronic control units (ECUs) are sent on the bus according to FPNS, is used extensively in automotive applications, with more than 400 million CAN enabled microcontrollers manufactured each year.

In this paper, we present a method to model the execution of a set of fixed priority non-preemptive tasks on a resource with full availability [14] in the framework of Real-Time Calculus. Our method can be applied for compositional analysis of a distributed system involving FPNS, and thus can be used to compute various performance attributes such as end-to-end latencies and buffer requirements. Note that [7] also discusses the modeling of FPNS in Real-Time Calculus, but the results are too pessimistic. The bounds that we provide in this paper are tighter in comparison.

2. The framework of Real-Time Calculus

The key concepts in the framework of Real-Time Calculus [4, 14] are (i) the modeling of the arrival pattern of tasks (or the *event model*) which generates demands on the resources, (ii) the modeling of the service offered by the resources to the tasks (i.e., the *resource model*). It can be used to derive hard upper and lower bounds of various performance criteria such as maximum end-to-end delay experienced by an event stream or buffer requirements.

Arrival Curves: The event model is captured by the notion of arrival curves. Let $R[s,t)$ be the number of events that arrive in the time interval $[s,t)$. Then R, the upper arrival curve α^u and the lower arrival curve α^l are related by $\alpha^l(t-s) \leq R[s,t) \leq \alpha^u(t-s), \forall s \leq t$ with

$\alpha^l(0) = \alpha^u(0) = 0$. We write $\alpha = [\alpha^u, \alpha^l]$ and call it the arrival curve.

Service curves: The resource model is captured by the notion of service curves. Let $S[s,t)$ be the number of events that a resource can service in the time interval $[s,t)$. Then S, the upper service curve β^u and the lower service curve β^l are related by $\beta^l(t-s) \leq S[s,t) \leq \beta^u(t-s), \forall s \leq t$ with $\beta^l(0) = \beta^u(0) = 0$. We write $\beta = [\beta^u, \beta^l]$ and call it the service curve.

Arrival and service curves can also be described in terms of the amount of resources, such as the number of processing or communication cycles, instead of number of events as above. The *resource based service curve* $\overline{\beta}(\Delta)$ denotes the resource units available in any time interval of length Δ. Similarly, the *resource-based arrival curve* $\overline{\alpha}(\Delta)$ denotes the requests in terms of resource units that arrive in any time interval of length Δ.

Greedy Processing Component: In Real-Time Calculus, the greedy processing component (GPC) [7, 14, 4] is an abstract component that is triggered whenever an event is available on the input event stream (described by the arrival curve α) and produces a single output event stream (described by the arrival α'). At every event arrival, a task is instantiated to process the incoming event. Events are processed in a greedy fashion in first-in-first-out order, while being restricted by the availability of processing resources described by the service curve $\overline{\beta}$. Let $E^{\max}$ and $E^{\min}$ denote the execution demand in terms of the maximum and minimum resource units required for processing one event. Then the GPC can be modeled as: $\alpha'^u = \lceil \min\{(\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u\} \rceil$, $\alpha'^l = \lfloor \min\{(\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l\} \rfloor$, $\overline{\beta}'^u = \max\{(\overline{\beta}^u - \overline{\alpha}^l) \overline{\oslash} 0, 0\}$, and $\overline{\beta}'^l = (\overline{\beta}^l - \overline{\alpha}^u) \overline{\otimes} 0$, where $\overline{\beta}'$ denotes the remaining service available to process other event streams, and the workload transformations are $\beta^u = \overline{\beta}^u / E^{\min}$, $\beta^l = \overline{\beta}^l / E^{\max}$, $\overline{\alpha}^u = E^{\max}.\alpha^u$, and $\overline{\alpha}^l = E^{\min}.\alpha^l$. See [2] for the definitions of $\otimes$, $\oslash$, $\overline{\otimes}$, and $\overline{\oslash}$. Refer to [4, 14] for a discussion of these results on the GPC.

The worst case response time WR (time between activation and completion) experienced by any event on the event stream and the maximum number of events in the input queue $Buffer$ can be determined as follows [4, 14]:

$$WR \leq Del(\alpha^u, \beta^l) \tag{1}$$

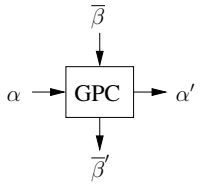

Figure 1. Greedy processing component

$$Buffer \leq Buf(\alpha^u, \beta^l) \tag{2}$$

where $Del(\alpha^u, \beta^l) = \sup_{\Delta \geq 0}\{\inf\{\mu \geq 0 : \alpha^u(\Delta) \leq \beta^l(\Delta + \mu)\}\}$ and $Buf(\alpha^u, \beta^l) = \sup_{\lambda \geq 0}\{\alpha^u(\lambda) - \beta^l(\lambda)\}$.

The end-to-end response time r experienced by an event with upper arrival curve α^u that is processed on N consecutive GPCs with lower service curves $\beta_1^l, \ldots, \beta_N^l$ is (see [14]):

$$r = Del(\alpha^u, \beta_1^l \otimes \ldots \otimes \beta_N^l) \tag{3}$$

3. Modeling FPNS with Real-Time Calculus

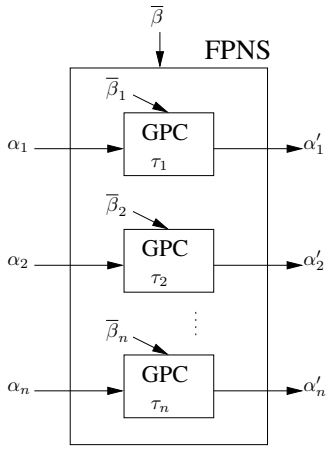

Figure 2. Modeling FPNS in Real-Time Calculus

Assume a set $\mathcal{T}$ of n tasks $\tau_1, \tau_2, \ldots, \tau_n$ with unique, fixed priorities (τ_1 has the highest priority while τ_n has the lowest priority) executing on a resource with service curve $\overline{\beta}$. The input event stream of task τ_i is described by α_i. We also assume that each job of task τ_i requires a fixed E_i units of resource or equivalently, C_i units of time for execution. Each task τ_i produces an output event stream α'_i given by the equations modeling the GPC in the previous section. The main focus of this paper is to find $\overline{\beta}_i$, the service available to task τ_i. Figure 2 depicts the situation. Note that this is more complicated than fixed priority preemptive scheduling (FPPS) where the service $\overline{\gamma}_i = [\overline{\gamma}_i^u, \overline{\gamma}_i^l]$ available to task τ_i is given by (see [14])

$$\overline{\gamma}_i^u = \max\{(\overline{\beta}^u - \sum_{j=1}^{i-1} \overline{\alpha}_j^l) \overline{\oslash} 0, 0\} \tag{4}$$

$$\overline{\gamma}_i^l = (\overline{\beta}^l - \sum_{j=1}^{i-1} \overline{\alpha}_j^u) \overline{\otimes} 0 \tag{5}$$

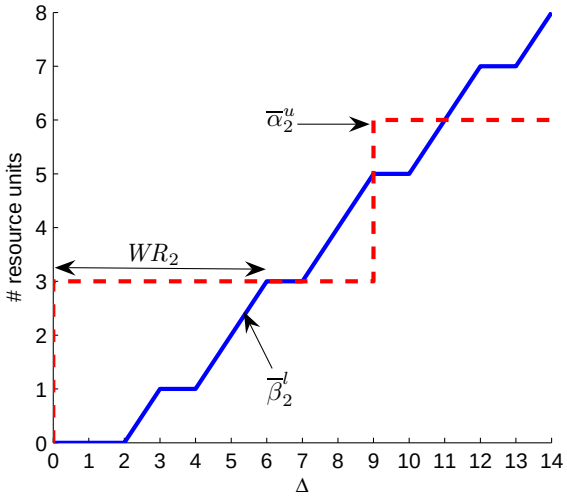

Figure 3. Computation of $\overline{\beta}_2^l$ according to Eq. 7 and WR_2 for task set $\mathcal{T}_1$

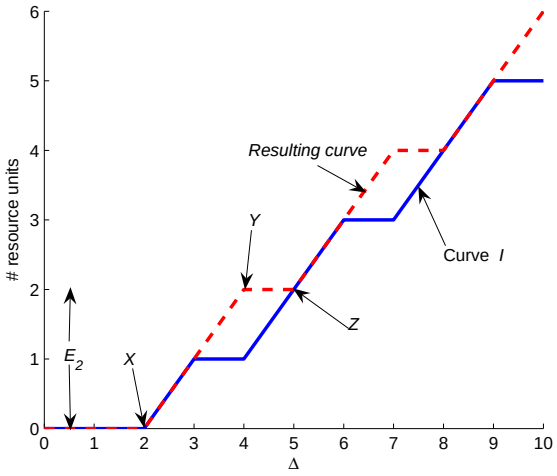

Figure 4. Computation of $\overline{\beta}_2^l$ (resulting curve) from $\overline{\gamma}_{2,1}^l$ (curve I) for task set $\mathcal{T}_2$ according to our method

According to [7], FPNS can be modeled as depicted in Figure 2 with the service curves as:

$$\overline{\beta}_i^u = \min\{\overline{\beta}^u, \overline{\gamma}_i^u + E_i\} \tag{6}$$

$$\overline{\beta}_i^l = \max\{0, \overline{\gamma}_i^l - \max_{i<j\leq n} E_j\} \tag{7}$$

We show that the upper service curve $\overline{\beta}_i^u$ (given by Eq. 6) offers more service and the lower service curve $\overline{\beta}_i^l$ (given by Eq. 7) offers less service than what is actually offered by the resource to task τ_i, thus leading to pessimistic results. In the following section, we give a procedure for finding $\overline{\beta}_i^u$ and $\overline{\beta}_i^l$ which are tighter than the equations 6 and 7.

First, we show that $\overline{\beta}_i^l$ in Eq. 7 gives pessimistic results. Consider the task set $\mathcal{T}_1$ consisting of three periodic tasks τ_1, τ_2 and τ_3 executing on a resource, with computation times C_i of 1, 3 and 1, and periods T_i of 3, 9 and 4 respectively. Let us assume that it takes x resource units to process a task of x time units. Thus, $E_i = C_i$ in this case. Then, the lower service offered to the task τ_2, i.e., $\overline{\beta}_2^l$ (using Eq. 7) and the upper arrival curve of τ_2, i.e., $\overline{\alpha}_2^u$, are shown in Figure 3.

The actual worst case response time WR_2 for task τ_2 is 5, assuming the continuous scheduling model and its analysis presented in [3]. Hence, in any time interval of duration $WR_2 = 5$, task τ_2 should get at least $E_2 = 3$ resource units. But according to Figure 3, in any time interval of 5 units, task τ_2 gets a lower bound of 2 resource units, and the WR_2 using Eq. 1 comes out to be 6. Thus, $\overline{\beta}_i^l$ according to Eq. 7 leads to pessimistic results. It turns out that $\overline{\beta}_i^u$ according to Eq. 6 also leads to pessimistic results, since the computation of α' depends on $\overline{\beta}$.

3.1. FPNS assuming discrete scheduling

In this section, we give a procedure for computing the service curves $\overline{\beta}_i^l$ and $\overline{\beta}_i^u$ for FPNS assuming the discrete scheduling model [1, 6], given the service curves $\overline{\gamma}_i^l$ (Eq. 5) and $\overline{\gamma}_i^u$ (Eq. 4) for FPPS. We do not know whether a closed form analytic expression can be given for $\overline{\beta}_i^l$ and $\overline{\beta}_i^u$.

In the discrete scheduling model, the maximum blocking time experienced by task τ_i due to lower priority tasks being executed when task τ_i is activated is given by $b_i = \max_{i<j\leq n}(C_j - dtu)$, for $1 \leq i < n$ and $b_n = 0$, where dtu is the discrete time unit. In terms of resource units, let this blocking be B_i, i.e., $B_i = b_i.E_i/C_i$.

To obtain $\overline{\beta}_i^l$, compute $\overline{\gamma}_{i,1}^l = \max\{0, \overline{\gamma}_i^l - B_i\}$ and apply Procedure 1 shown below with $I = \overline{\gamma}_{i,1}^l$. To obtain $\overline{\beta}_i^u$, apply Procedure 1 with $I = \overline{\gamma}_i^u$.

Procedure 1: Let the input curve be I.
Let point Z be the origin.

1. Start with point Z. Move along the time interval axis to find the point where I starts increasing. Let this point be X. If the curve I increases at a constant slope from point X extending infinitely to the right, then exit the procedure. Otherwise, there is a finite segment of the curve which is increasing at a constant slope.

2. If the service offered by the increasing curve segment

starting at point X is not a multiple of E_i, then extend that segment (with the same slope E_i/C_i) by the least amount such that the resulting segment offers a service which is a multiple of E_i. Suppose this extension is up to point Y (see Figure 4). From point Y make the resulting curve constant up to the point where it touches I. Let this point be (the new) Z. Now, go to step 1.

3. If the service offered by the increasing curve segment starting at point X is a multiple of E_i, then let the end-point of that increasing curve segment be the (new) value of point Z. Now, go to step 1.

To illustrate the above procedure, consider the task set $\mathcal{T}_2$ consisting of three periodic tasks τ_1, τ_2 and τ_3 with computation times C_i of 1, 2 and 2, and periods T_i of 3, 4 and 15 respectively, with the discrete time unit as 1. Figure 4 shows how we obtained $\overline{\beta}_2^l$ from $\overline{\gamma}_{2,1}^l$.

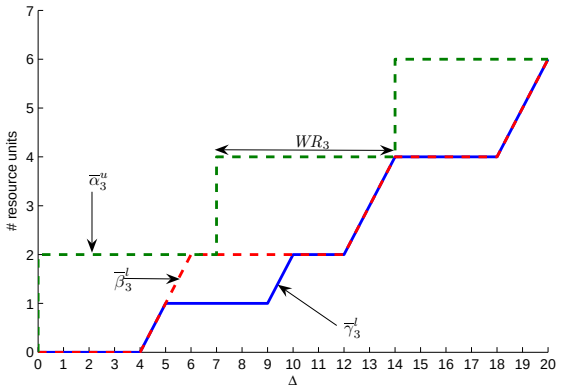

Figure 5. Worst case response time of task τ_3 for task set $\mathcal{T}_3$

As another example, consider the task set $\mathcal{T}_3$ consisting of three periodic tasks τ_1, τ_2 and τ_3 with computation times C_i of 2, 2 and 2, and periods T_i of 5, 7 and 7 respectively with the discrete time unit as 1. In this case, the worst case response times come out to be 3, 5 and 7 respectively. These results are identical to those obtained by using the analysis of [6]. Figure 5 shows how we calculate the worst case response time WR_3 for task τ_3.

3.2. FPNS assuming continuous scheduling

In this section, we give a procedure for computing the service curves $\overline{\beta}_i^l$ and $\overline{\beta}_i^u$ for FPNS assuming the continuous scheduling model [3], given the service curves $\overline{\gamma}_i^l$ (Eq. 5) and $\overline{\gamma}_i^u$ (Eq. 4). As in the discrete scheduling case, we do not know whether a closed form analytic expression can be given for $\overline{\beta}_i^l$ and $\overline{\beta}_i^u$.

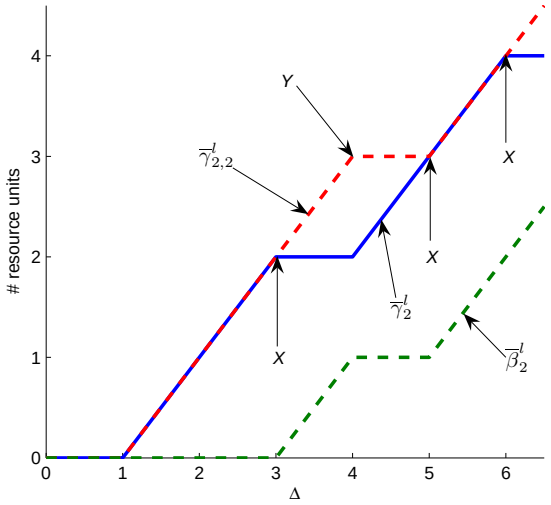

Figure 6. $\overline{\gamma}_2^l$, $\overline{\gamma}_{2,2}^l$, and $\overline{\beta}_2^l$ for task set $\mathcal{T}_4$

In the continuous scheduling model, the maximum blocking time experienced by task τ_i due to lower priority tasks is given by $b_i = \max_{i<j\leq n}(C_j)$, for $1 \leq i < n$ and $b_n = 0$. In terms of resource units, let this blocking be B_i, i.e., $B_i = b_i.E_i/C_i$.

Strictly speaking, this blocking time is a supremum (and not a maximum) for tasks $\tau_i, 1 \leq i < n$ because the blocking lower priority task has to start an amount of time ϵ before the release of task τ_i. So, the actual maximum blocking time for task $\tau_i, 1 \leq i < n$ is $b_i - \epsilon$ where $\epsilon > 0$ and ϵ is infinitesimally small.

To obtain $\overline{\beta}_i^l$ for $1 \leq i < n$, apply Procedure 2 shown below with $I = \overline{\gamma}_i^l$ and let the resulting curve be $\overline{\gamma}_{i,2}^l$. Then, $\overline{\beta}_i^l = \max\{0, \overline{\gamma}_{i,2}^l - B_i\}$.

Procedure 2: Let the input curve be I.

1. Start with the origin of I. Move along the time interval axis to find the first point where the curve offers a service of B_i. Let this point be X.

2. If the curve I increases at a constant slope from point X extending infinitely to the right, then exit the procedure. Otherwise, there is a finite segment of the curve which is increasing at a constant slope.

3. If the curve I is not increasing from point X, offer an E_i amount of service starting at point X (with the slope E_i/C_i). Let the end-point of this newly added service be point Y (see Figure 6). From Y, make the resulting curve constant till it touches I. Let this point be the (new) value of point X. Go to step 2.

4. If the curve starts increasing from point X and if the service offered by the increasing curve segment is not a multiple of E_i, then extend that segment (with the slope E_i/C_i) by the least amount such that the resulting segment offers a service which is a multiple of E_i. Suppose this extension is up to point Y . From point Y, make the resulting curve constant up to the point where it touches I. Let this point be the (new) X. Go to step 2.

5. If the curve starts increasing from point X and if the service offered by that increasing curve segment is a multiple of E_i, then let the end-point of that increasing curve segment be the (new) value of point X. Go to step 2.

To obtain $\overline{\beta}_n^l$, apply Procedure 1 (see Section 3.1) with $I = \overline{\gamma}_n^l$. To obtain $\overline{\beta}_i^u$ for $1 \leq i \leq n$, apply Procedure 1 with $I = \overline{\gamma}_i^u$.

To illustrate the above procedure, consider the task set $\mathcal{T}_4$ consisting of five periodic tasks τ_1, τ_2, τ_3, τ_4 and τ_5 with computation times C_i of 1, 1, 2, 2, and 0.5 and periods T_i of 3, 4, 10, 10, and 50 respectively. Figure 6 shows how we obtained $\overline{\gamma}_{2,2}^l$ from $\overline{\gamma}_2^l$, and finally $\overline{\beta}_2^l$. The worst case response times come out to be 3, 4, 8, 9.5, and 59.5 respectively. These results are identical to those obtained by using the analysis of [3].

4. Example based on CAN Bus

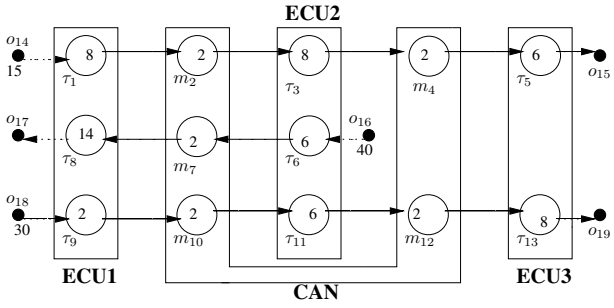

Figure 7. Example network of ECUs communicating through a CAN bus [9]

In this section, we apply the results of the previous section for a compositional analysis of a system using a CAN bus. The system we analyze is shown in Figure 7. It is adapted from [9] and consists of three ECUs, one CAN bus, eight tasks (τ_i) mapped on different ECUs and five messages (m_i) mapped on CAN bus as shown in Figure 7. Tasks and messages are referred to as objects. Tasks on ECUs execute according to the FPPS policy whereas messages are transmitted on the CAN bus according to the FPNS policy. If tasks τ_i and τ_j are executing on the same ECU, then task τ_i has higher priority than task τ_j iff $i < j$. Similarly, message m_i has higher priority than m_j iff $i < j$. The computation times for tasks and transmission times for messages are shown inside the circles in Figure 7. Three computation paths are defined, $o_{14} - o_{15}$, $o_{16} - o_{17}$ and $o_{18} - o_{19}$. The objects follow an event-based activation mode, i.e, the activation of an object is always driven by the completion of its predecessor, except the first tasks in the paths (which are τ_1, τ_6, and τ_9). For example, task τ_1 is activated periodically at every 15 time units, and its completion causes activation of message m_2, whose completion in turn causes activation of task τ_3.

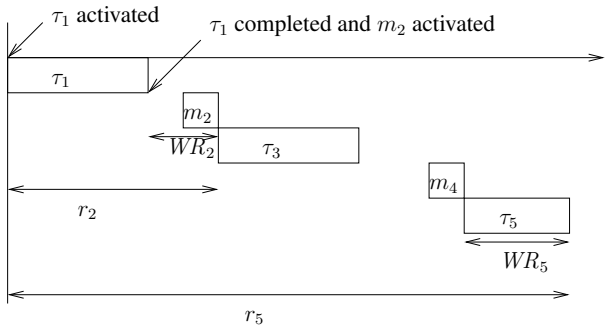

Figure 8. r_i and WR_i

The paper [10] discusses the holistic scheduling analysis of FPPS and applies it to a distributed system. Note that it gives tighter bounds than the original holistic scheduling analysis proposed in [12]. The schedulability analysis of FPNS is discussed in [3] and is applicable to the CAN bus. The schedulability analysis of CAN is also discussed in [5] which is based on a pessimistic variant of [3].

Recall from Section 2 that r_i is the worst case response time of the object i with reference to the first object in the path, i.e., it is the end-to-end latency experienced by an event between the first object in the path and the object i. Also recall that WR_i is the worst case response time of object i with reference to its activation. Figure 8 explains the notions of WR_i and r_i for the topmost path $o_{14} - o_{15}$ in Figure 7.

Applying the holistic scheduling analysis of [10, 3], the values of T_i (period), J_i (jitter) and r_i for each object i are shown in columns 2, 3 and 4 of Table 1. Note that these results are not the same as that of [9] as the latter are based on [12] and are thus pessimistic.

In the framework of Real-Time Calculus, we use the analysis of FPPS for the ECUs and the method presented in the previous section for CAN bus for the compositional analysis of the system shown in Figure 7. The number of buffers, WR_i and r_i for each object i are computed accord-

	Holistic analysis			Real-Time Calculus		
Object i	T_i	J_i	r_i	$Buffer_i$	WR_i	r_i
τ_1	15	0	8	1	8	8
m_2	15	0	12	1	4	12
τ_3	15	2	20	1	8	20
m_4	15	2	26	1	6	26
τ_5	15	6	32	1	6	32
τ_6	40	0	22	1	22	22
m_7	40	16	30	1	8	30
τ_8	40	22	60	2	42	60
τ_9	30	0	130	5	130	130
m_{10}	30	128	140	4	14	140
τ_{11}	30	136	168	6	104	168
m_{12}	30	158	190	3	22	190
τ_{13}	30	178	210	3	44	210

Table 1. Comparison between analysis results using the methods of [10, 3] and that of Real-Time Calculus. Note that the results of holistic analysis are not the same as that mentioned in [9].

ing to equations 2, 1 and 3 and are shown in columns 5, 6 and 7 of Table 1. Note that our results are the same as the one obtained by using the holistic scheduling technique of [10, 3]. This leaves open the question whether the two methods give identical results in all cases. This investigation is part of ongoing work.

5. Conclusions

In this paper, we modeled the execution of tasks on a resource with fixed priority non-preemptive scheduling. The proposed analysis can be used in the Real-Time Calculus (RTC) Toolbox for modeling non-preemptive fixed priority scheduling. The resource considered in this paper is a resource with full availability. Modeling hierarchical scheduling such as FPNS/TDMA (Time Division multiple access) which is used in TT-CAN [8] is part of future work.

Acknowledgements: We thank S. Ramesh, Manoj Dixit and Prahlad Sampath for discussions and for providing useful feedback on the initial draft of this paper. The second author was partially supported by a grant from GM R&D, Bangalore, India.

References

[1] S. K. Baruah, L. E. Rosier, and R. R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, 1990.

[2] J.-Y. L. Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. LNCS 2050, Springer-Verlag, 2001.

[3] R. J. Bril, J. J. Lukkien, and W. F. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In *Proc. 19th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 269–279, 2007.

[4] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proc. 6th Design Automation and Test in Europe (DATE)*, pages 190–195. IEEE Press, 2003.

[5] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, Apr. 2007.

[6] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uni-processor scheduling. Technical Report 2966, INRIA, France, Sep 1996.

[7] W. Haid and L. Thiele. Complex task activation schemes in system level performance analysis. In *Proc. 5th Intl. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 173–178. ACM Press, 2007.

[8] G. Leen and D. Heffernan. TTCAN: a new time-triggered controller area network. *Microprocessors and Microsystems*, 26(2):77–94, 2002.

[9] M. D. Natale, W. Zheng, C. Pinello, P. Giusto, and A. Sangiovanni-Vincentelli. Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems. In *Proc. 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2007.

[10] J. C. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS)*, 1998.

[11] K. Richter, M. Jersak, and R. Ernst. A formal approach to MpSoC performance verification. *IEEE Computer*, 36(4):60–67, 2003.

[12] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40:117–134, April 1994.

[13] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. http://www.mpa.ethz.ch/Rtctoolbox.

[14] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis - a case study. *Software Tools for Technology Transfer*, 8(6):649 – 667, Oct. 2006.