A Task Model for TDMA Communications

Shuai Li*†, Stéphane Rubini†, Frank Singhoff†, Michel Bourdellès*
*Thales Communications & Security, 4 av. des Louvresses, 92622 Gennevilliers, France
*Email: {first-name}.{last-name}@fr.thalesgroup.com
†Lab-STICC/UMR 6285, UBO, UEB, 20 av. Le Gorgeu, 29200 Brest, France
†Email: {last-name}@univ-brest.fr

***Abstract*—In this article a new task model, called DGMF, is proposed to improve scheduling analysis of dependent tasks in radio stations that embed a TDMA communication protocol. TDMA is a channel access protocol that allows several stations to communicate in a same network, by dividing time into several time slots. This protocol has an impact on task release times, execution times, and deadlines, which needs to be considered for less pessimistic scheduling analysis results. We experiment on software radio protocols from Thales, which are representative of the system we want to analyze.**

I. Introduction

This article proposes to improve scheduling analysis of systems with Time Division Multiple Access (TDMA) communications. In this kind of system, task release times, execution times, deadlines depend on the TDMA communication protocol. Furthermore, we focus on TDMA Software Radio Protocols (SRP) [7] where tasks have dependencies (shared resource and precedence).

A TDMA SRP is both a time-triggered [3] (TDMA) and event-triggered [3] (task precedence) system, with variable task parameters. Common scheduling analysis techniques are either limited to the time-triggered aspect [10], [17], event-triggered aspect [11], [13], [9], or do not allow task dependencies or variable task parameters [8], [4].

To specify more accurately tasks in such systems, a new task model is proposed: the Dependent General Multiframe (DGMF) model, which extends the General Multiframe (GMF) model [1] with task dependencies. This task model is applied to TDMA SRPs from Thales.

The rest of the article is organized as follows: in Section II our system architecture and assumptions are exposed. In Section III the DGMF task model is proposed. Section IV shows how to perform scheduling analysis of DGMF tasks. Evaluation of the proposed model is done in Section V. We conclude with future works in Section VI.

II. System Architecture and Assumptions

In this article, we consider SRPs embedded in radio stations. These radio stations communicate in a mobile ad-hoc wireless network. We assume that the effects of non-determinism in wireless networks, on scheduling analysis of a single station, are negligible. A SRP is a software that implements a communication protocol. TDMA is a common communication protocol in SRPs.

In this protocol, a TDMA frame is divided into several time slots of different types and durations. When several stations want to communicate, control and data flows are handled by different tasks. The tasks are constrained by the TDMA frame, e.g. a release depends on a slot start time, an execution times depends on a slot type, a deadline depends on a slot duration. Tasks may communicate (i.e. precedence dependency) and use shared resources protected by a mechanism [14].

Tasks run on an execution platform for which we make some assumptions. A task is allocated on a processor that is scheduled by a preemptive fixed priority policy. There are several processors, i.e. the system is partitioned.

III. Dependent General Multiframe

The task model we propose to specify and analyze a TDMA SRP is called DGMF, an extension of the GMF model.

A DGMF task G_i is a vector composed of N_i frames F_i^j, with $1 \leq j \leq N_i$. Each frame has some parameters.

E_i^j is the Worst Case Execution Time (WCET) of F_i^j, D_i^j the deadline, and P_i^j the minimum separation time between releases of F_i^j and F_i^{j-1} [1].

$[U]_i^j$ is a set of (R, S, B) tuples denoting shared resource critical sections. F_i^j allocates resource R after it has run S time units of its execution time, and then locks the resource during the next B time units of its execution time.

$[F_p^q]_i^j$ is a set of predecessor frames, i.e. frames from any other DGMF task that must finish before F_i^j can be released. We assume F_i^j can only have F_p^q in its predecessor set if G_i and G_p have the same sum of P_i^j.

$prio(F_i^j)$ is the priority of F_i^j and $proc(F_i^j)$ is the processor on which F_i^j is allocated on.

The first frame to be released by a DGMF task G_i is denoted F_i^1. We call r_i^1, the release time of first frame F_i^1. We call r_i, the release time of G_i and thus $r_i = r_i^1$. For any F_i^j, with $j > 1$, its release time is: $r_i^j = r_i^1 + \sum_{h=1}^{j-1} P_i^h$. For any F_i^j, we call value $r_i^j + D_i^j$ the global deadline of F_i^j.

A DGMF task set may have the following property:

Property 1 (Unique Predecessor): Let F_i^j be a frame of a task G_i, in a DGMF task set. Let $[F_p^q]_i^j$ be the set of predecessor frames of F_i^j. F_i^{j-1} is the previous frame of F_i^j in the vector of G_i, if $j > 1$. The set of frames that precede F_i^j is the set $[F_p^q]_i^j$ and F_i^{j-1} (if $j > 1$). A DGMF task set is said to respect the *Unique Predecessor* property if, for all frames F_i^j, there is at most one frame F_x^y, among frames that precede F_i^j, with a global deadline (i.e. $r_x^y + D_x^y$) greater than or equal

to the release time of F_i^j. Formally the *Unique Predecessor* property is defined as:

$$\exists_{\leq 1} F_x^y \in pred(F_i^j), r_x^y + d_x^y \geq \max(\max_{F_l^h \in pred(F_i^j)} (r_l^h + E_l^h), r_i^j) \quad (1)$$

Where $\exists_{\leq 1}$ means "there exists at most one", and the set $pred(F_i^j)$ is defined as:

$$pred(F_i^j) = \begin{cases} [F_p^q]_i^j \cup \{F_i^{j-1}\} & \text{if } j > 1 \\ [F_p^q]_i^j & \text{otherwise.} \end{cases} \quad (2)$$

We assume that TDMA SRPs are modeled with DGMF task sets having the *Unique Predecessor* property.

Consider the DGMF task set in Table I, modeling tasks constrained by a TDMA frame.

TABLE I
DGMF TASK SET

	E_i^j	D_i^j	P_i^j	$[U]_i^j$	$[F_p^q]_i^j$
G_1; $r_1 = 0$					
F_1^1	1	4	1		F_2^1
F_1^2	1	3	1		
F_1^3	1	2	6		
F_1^4	1	4	4		F_2^2
F_1^5	4	8	8	(R, 1, 3)	F_2^3
G_2; $r_2 = 0$					
F_2^1	1	4	8		F_{Tick}
F_2^2	1	4	4		
F_2^3	1	4	4		
F_2^4	2	4	4	(R, 0, 1)	
G_3; $r_3 = 4$					
F_3^1	1	2	2		F_4^1
F_3^2	1	2	18		F_4^2
G_4; $r_4 = 4$					
F_4^1	1	2	2		F_{Tick}
F_4^2	1	2	18		

All frames are allocated on *CPU1* except F_1^2, which is allocated on *CPU2*. Fig. 1 shows an example of a schedule produced by the task set, over 20 time units. In Fig. 1, there is a TDMA frame of 1 S slot, 2 B slots, and 3 T slots. G_2 is released at S and T slots. G_2 releases G_1 upon completion. G_4 is released at B slots. G_4 releases G_3 upon completion.

IV. DGMF SCHEDULING ANALYSIS

Original analysis techniques for GMF tasks [1], [16] cannot be directly applied to DGMF tasks due to task dependencies.

In [12] a GMF to transaction transformation algorithm is proposed. The author argues that scheduling analysis techniques for transactions can be applied to GMF tasks. We propose to use transactions, and their associated scheduling analysis techniques, to assess schedulability of DGMF tasks. The transformation in [12] is thus extended for DGMF.

A. Transaction Definitions

A transaction [17] (denoted Γ_i) is a set of periodic tasks (denoted τ_{ij}). A transaction is released by a periodic event that occurs every T_i. A particular instance of a transaction is called a job.

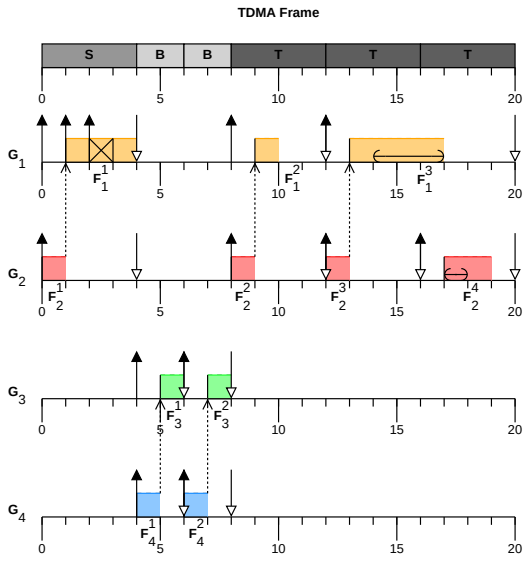

Fig. 1. DGMF Tasks in a TDMA SRP: Up arrows are frame releases; Down arrows are frame relative deadlines; Dashed arrows are precedence dependencies; Curved arrows are shared resource allocations; Crossed frame executes on different processor

An instance of a task in a transaction is released after the event that releases the job of the transaction; if the event that releases the p^{th} job of Γ_i occurs at t_0, then the p^{th} instances of its tasks are released at or after t_0. We call r_i the release time of the first job of transaction Γ_i and $r_i^j = r_i + O_i^j$ the release time of the first instance of τ_i^j.

Tasks in a transaction are related by precedence dependency. A precedence dependency between two tasks, denoted $\tau_{ip} \prec \tau_{ij}$, is a constraint that means that τ_{ip} (predecessor) must finish before τ_{ij} (successor) can be released.

A periodic task's shared resource critical section is a tuple (τ, R, S, B) where τ is the task using the resource, R the resource, S the resource allocation time and B the resource blocking time.

Each periodic task is defined by some parameters [17]. C_i^j is the WCET, O_i^j is the offset, d_i^j is the deadline (and value $O_i^j + d_i^j$ is called the global deadline), J_i^j is the jitter, B_i^j is the worst case blocking time [14], $prio(\tau_i^j)$ is the priority, and $proc(\tau_i^j)$ is the processor of τ_i^j.

B. DGMF To Transaction

To perform scheduling analysis of DGMF tasks, we transform them to transactions in several steps. The full transformation algorithm, with proofs, can be found in [6].

Step 1 of the transformation is an extension of [12]. Each DGMF task G_i is transformed into transaction Γ_i with period $T_i \leftarrow \sum_{j=1}^{N_i} P_i^j$ and release time r_i equal to the release time of G_i. Frames are transformed into periodic tasks. Frame parameters E_i^j, D_i^j, P_i^j are expressed with periodic task parameters $C_i^j \leftarrow E_i^j$, $O_i^j \leftarrow \sum_{h=1}^{j-1} P_i^h$, $d_i^j \leftarrow D_i^j$, $prio(\tau_i^j) \leftarrow prio(F_i^j)$,

$proc(\tau_i^j) \leftarrow proc(F_i^j)$.

In **Step 2**, critical sections in the DGMF model are expressed in the resulting transaction set from step 1.

In **Step 3**, precedence dependencies in the DGMF model, are modeled according to the transaction model [11]. First, precedence dependencies between frames of different DGMF tasks are expressed in the resulting transaction set. A precedence dependency is also added between two tasks representing consecutive frames of a same DGMF task (i.e. $\tau_i^j \prec \tau_i^{j+1}$, $j < N_i$). We then enforce successors to be released after predecessors: when $\tau_p^q \prec \tau_i^j$ and $r_p^q + C_p^q > r_i^j$ then offset and deadline are updated as: $O_i^j \leftarrow O_i^j + r_p^q + C_p^q - r_i^j$ and $d_i^j \leftarrow d_i^j - r_p^q + C_p^q - r_i^j$. Afterwards we ensure that precedence dependent tasks are delayed from the same event, with respect to the definition of transactions. Tasks in different transactions, are assigned into a same transaction Γ_m if:

$$\exists \tau_p^q, \tau_i^j \mid (\Gamma_i \neq \Gamma_p) \wedge \left(\tau_p^q \prec \tau_i^j \vee \tau_i^j \prec \tau_p^q\right)$$

Tasks τ_m^j in Γ_m were originally in Γ_o. The release time of the first job of Γ_m is then $r_m \leftarrow \min(r_m, r_o + O_m^j)$. The offset of each task τ_m^j is then $O_m^j \leftarrow r_o + O_m^j - r_m$. Finally for a specific task τ_i^j, the transformation reduces predecessors τ_i^q that have a global deadline (i.e. $O_i^q + d_i^q$) smaller than the offset O_i^j of τ_i^j. At least one predecessor of τ_i^j is kept.

C. Assessing Schedulability of Resulting Transactions

To enforce schedulability of the resulting transactions, we use a schedulability test [5] based on [13]. The schedulability test is applicable to tree-shaped transactions with *non-immediate tasks*. A *non-immediate task* is one that is not necessarily immediately released by its predecessor. We obtain tree-shaped transactions with *non-immediate tasks* from the DGMF to transaction transformation, according to the following theorem proved in [6]:

Theorem 1: A DGMF task set with the *Unique Predecessor* property (Property 1) is transformed into a transaction set without tasks that have more than one predecessor.

V. Experiments

The DGMF to transaction transformation is implemented in Cheddar [15], a real-time scheduling analysis tool. Experiments are conducted to assess the transformation's correctness, time performance, and the applicability of the approach on a real TDMA SRP from Thales.

A. Transformation Correctness

A total of **25600** DGMF task sets were randomly generated to evaluate the transformation's correctness. Each of them was transformed to a transaction set. Both DGMF and transaction sets were simulated and we observed that each DGMF set's schedule was strictly the same as the resulting transaction set's schedule. Along with the proofs in [6], this simulation further enforces the transformation's correctness.

B. Transformation Time Performance

We now verify the transformation's complexity which depends on two parameters: n_F the number of frames, and n_D the number of task dependencies (both precedence and shared resource). The complexity of the transformation is $O(n_D^2 + n_F)$. We check this complexity by varying n_D (resp. n_F) and setting n_F (resp. n_D) to a constant value.

Fig. 2 shows the transformation duration by the number of precedence dependencies. The number of frames is set to 1000, the number of DGMF tasks to 100, and the number of shared resource dependencies to 0. The minimum number of precedence dependencies starts at 900, due to precedence dependencies in the transaction model that represent consecutive frames of a DGMF task. From Fig. 2 we see that the transformation duration is polynomial when the number of precedence dependencies vary. This result is consistent with the complexity $O(n_D^2)$, when n_D is the varying parameter.

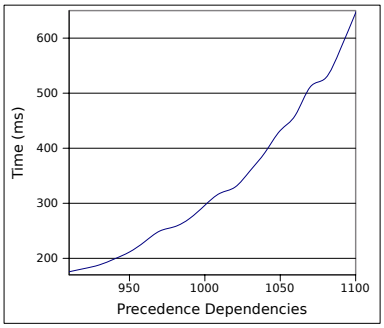

Fig. 2. Transformation Duration by Number of Precedence Dependencies

Fig. 3 shows the transformation duration by the number of frames. The number of precedence dependencies is set to 0 and the other parameters remains the same. From Fig. 3 we see that the duration is polynomial when the number of frames varies. One can think that this result is inconsistent with the transformation's complexity $O(n_F)$. In practice, the implementation in Cheddar introduces a loop to verify that a task is not already present in the system's task set. Thus the time complexity of the implementation is $O(n_F^2)$.

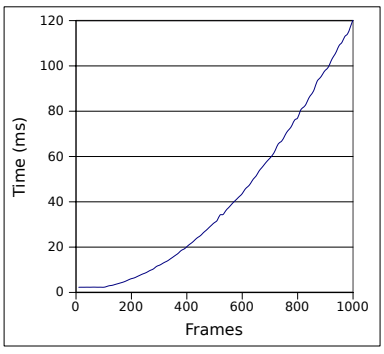

Fig. 3. Transformation Duration by Number of Frames

Overall we see that a system with no dependency, 1000 frames, and 100 DGMF tasks, takes less than 120ms to be

transformed on the PC used for the experiment (Intel Core i5 @ 2.40GHz). A system with 1100 precedence dependencies, 1000 frames, and 100 DGMF tasks, takes less than 650ms to be transformed. The transformation time is acceptable for our needs. Indeed for a typical TDMA frame handled by Thales, with 13 slots (1S, 4B, 8T) and 10 critical tasks, there would be a maximum of 130 frames (13×10), and 237 precedence dependencies ($10 \times 13 - 10 + 9 \times 13$) if all tasks are part of a same end-to-end flow released at each slot.

C. Experiment on a TDMA SRP

We now apply the DGMF task model to the modeling and scheduling analysis of a real TDMA SRP from Thales. The results given by DGMF analysis are compared to results given by GMF Worst Case Response Time (WCRT) analysis [16] and periodic task WCRT analysis [2].

The Cheddar model of the full case-study has 8 DGMF tasks and 44 frames. For sake of space, let us consider a partial TDMA frame with two slots called B and T. Slot B (of duration 4000μs) is followed by T (of duration 8000μs). In our system there are three tasks: G_1, G_2 and G_3. In the B slot G_3 releases G_1 which releases G_2. In the T slot G_3 releases G_1 and G_2 is not released in this slot. As shown by task parameters in Table II, task releases, execution times, deadlines are constrained by the TDMA frame. Task priorities are in highest priority first order. Execution times are in μs. Computed WCRTs are shown in Table III.

TABLE II
TASK SET OF TDMA SRP

(D)GMF					
		E_i^j	D_i^j	P_i^j	$[F_p^q]_i^j$
G_1, $prio(G_1) = 1$	F_1^1	955	4000	4000	F_3^1
	F_1^2	1874	8000	8000	F_3^2
G_2, $prio(G_2) = 2$	F_2^1	5722	12000	12000	F_1^1
G_3, $prio(G_3) = 3$	F_3^1	986	4000	4000	
	F_3^2	986	8000	8000	
Periodic Model					
		C_i	d_i	T_i	p_i
G_1		1874	4000	4000	1
G_2		5722	12000	12000	2
G_3		986	4000	4000	3

TABLE III
RESPONSE TIMES ("/" MEANS A DEADLINE IS MISSED)

		DGMF WCRT	GMF WCRT	Periodic WCRT
G_1	F_1^1	1941	/	/
	F_1^2	6523	/	
G_2	F_2^1	8649	7694	7694
G_3	F_3^1	986	986	986
	F_3^2	986	986	

From the WCRTs we see that DGMF analysis determines that no deadlines are missed. This is not the case for the other two analysis techniques. For G_2, GMF WCRT analysis gives a lower WCRT than DGMF analysis but this value is underestimated. Indeed GMF WCRT analysis considers that F_2^1 is only interfered by F_3^1 and F_3^2, without considering the fact that F_2^1 is released after F_1^1. In conclusion DGMF analysis determines a schedulable system, and precedence dependencies must be considered by the analysis, in order to not underestimate WCRTs.

VI. CONCLUSION

In this article we expected to improve scheduling analysis of systems with TDMA communications and task dependencies, using TDMA SRPs as an illustration. The DGMF task model was proposed to model such a system, and we proposed to transform DGMF tasks to transactions, for which an applicable scheduling analysis technique was chosen. Experiments on a real TDMA SRP from Thales showed that DGMF improves schedulability compared to both the GMF and periodic models.

In the future, we will integrate the proposed analysis technique in a SRP development process at Thales.

REFERENCES

[1] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *Real-Time Syst.*, 17(1):5–22, Jul, 1999.

[2] M. Joseph and P. Pandya. Finding response times in a real-time system. *Comput. J.*, 29(5):390–395, 1986.

[3] H. Kopetz. Event-triggered versus time-triggered real-time systems. In *Operating Systems of the 90s and Beyond*, volume 563 of *Lecture Notes in Computer Science*, pages 86–101. Springer Berlin Heidelberg, 1991.

[4] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer, 2001.

[5] S. Li, F. Singhoff, S. Rubini, and M. Bourdellès. Schedulability analysis of tree-shaped transactions with non-immediate tasks. Technical report, Lab-STICC/UMR 6285, 2014. Online: http://beru.univ-brest.fr/svn/CHEDDAR/trunk/docs/publications/li14_tree.pdf

[6] S. Li, F. Singhoff, S. Rubini, and M. Bourdellès. A task model for TDMA communication: application to software radio protocols Technical report, Lab-STICC/UMR 6285, 2014. Online: http://beru.univ-brest.fr/svn/CHEDDAR/trunk/docs/publications/li14_dgmf.pdf

[7] S. Li, F. Singhoff, S. Rubini, and M. Bourdellès. Applicability of real-time schedulability analysis on a software radio protocol. *ACM SIGAda Ada Lett.*, 32(3):81–94, Dec, 2012.

[8] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, Jan, 1973.

[9] J. Maki-Turja and M. Sjodin. Response-time analysis for transactions with execution-time dependencies. In *Proc. 19th Int. Conf. on Real-Time and Network Syst.*, pages 139–146, Nantes, France, 2011.

[10] N. Malcolm and W. Zhao. The timed-token protocol for real-time communications. *Comput.*, 27(1):35–41, Jan. 1994.

[11] J. Palencia and M. Harbour. Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In *Proc. 20th IEEE Real-Time Syst. Symp.*, pages 328–339, Phoenix, USA, 1999.

[12] A. Rahni. *(In French) Contributions à la validation d'ordonnancement temps réel en présence de transactions sous priorités fixes et EDF*. PhD thesis, Univ. Poitiers, Poitiers, France, 2008.

[13] O. Redell. Analysis of tree-shaped transactions in distributed real time systems. In *Proc. 16th Euromicro Conf. Real-Time Syst.*, pages 239–248, Catania, Italy, 2004.

[14] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: an approach to real-time synchronization. *IEEE Trans. Comput.*, 39(9):1175–1185, Sep, 1990.

[15] F. Singhoff, A. Plantec, P. Dissaux, and J. Legrand. Investigating the usability of real-time scheduling theory with the cheddar project. *Real-Time Syst.*, 43(3):259–295, Jun, 2009.

[16] H. Takada and K. Sakamura. Schedulability of generalized multiframe task sets under static priority assignment. In *Proc. 4th Intl. Workshop on Real-Time Computing Syst. and Applicat.*, pages 80–86, Taipei, Taiwan, 1997.

[17] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40(2-3):117–134, Apr, 1994.