Solving the 2015 FMTV Challenge Using Response Time Analysis with MAST

Julio L. Medina, Juan M. Rivas, J. Javier Gutiérrez, and Michael González Harbour

Departamento de Ingeniería Informática y Electrónica, Universidad de Cantabria, 39005-Santander, SPAIN
{medinajl, rivasjm, gutierjj, mgh}@unican.es

Abstract

Abstract—This paper reports solutions and recommendations regarding the design of the systems proposed in the 2015 edition of the Formal Methods and Timing Verification Challenge (FMTV). It uses the modelling formalism and the schedulability analysis techniques provided by the MAST suite of tools. The paper starts by clarifying the role of the data provided against the design intent. Then, for each of the challenges proposed, it discusses the adequacy of the analysis models used to represent the corresponding timing characteristics as well as the strengths and limitations of the underlying analysis approaches, and presents the results obtained. Finally, we report the effort it took to solve them and the lessons learned. The solutions are also provided in electronic form to facilitate their assessment by the community.

1 INTRODUCTION[1]

This paper presents a response to the 2015 FMTV Challenge [6] which asked questions related to the timing verification of two concrete scenarios taken from a real industrial case study (fully described in [6]). This is an aerial video system to detect and track objects moving on land. Challenge 1 asks for the analysis of the video frame processing sub-system, which is formed by four tasks, each running in a separate hardware processing unit. Challenge 2 also analyses four tasks running in the processor that tracks objects and controls the camera orientation. Fig. 1 and Fig. 2 condense their timing specification.

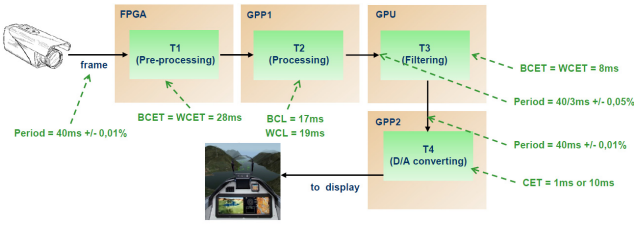

Fig. 1. . Challenge 1 Video frame processing

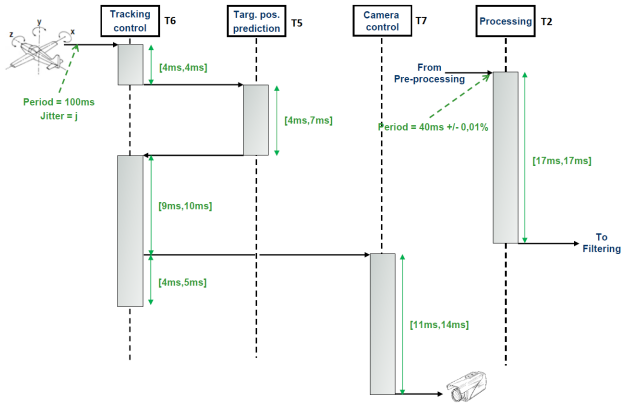

Fig. 2. . Challenge 2: Tracking and camera control

The paper faces the timing verification of these challenges by applying response time schedulability analysis (RTA). As will be discussed later, the questions asked for both challenges can be interpreted under the assumption that the design must prevent frame loss, which indirectly imposes hard real-time requirements that can be checked with schedulability analysis. If the interpretation of the challenges requirements is that it is acceptable that video frames can get lost, then the system would have soft real-time requirements. In this case, although the upper bounds obtained for the response times would continue to be valid, schedulability analysis might not be the right tool to evaluate the system performance.

The tools used for the analysis are offered by MAST [1], a Modeling and Analysis Suite for Real-Time Applications. MAST helps in moving the effort needed to do the analysis from the application of the mathematical models and the corresponding algorithms to a much higher level of abstraction. In turn, this demands for: (1) the correct characterization of the system in the analysis model, (2) the selection and use of the least pessimistic technique, and (3) the correct interpretation of the results provided by the tools. This approach is significantly much easier to carry out by real-time practitioners and greatly reduces the gap between the math in the analysis techniques and the

1. This work has been funded in part by the Spanish Government and FEDER funds under grant TIN2011-28567-C03-02 (HI-PARTES). This work reflects only the authors' views; the funding organisms are not liable for any use that may be made of the information contained herein.

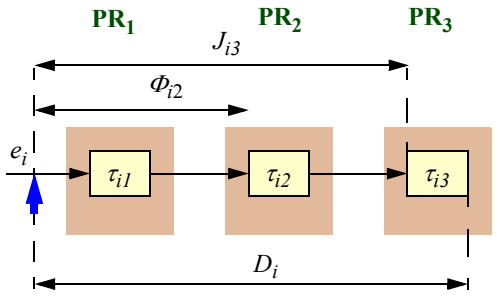

Fig. 3. MAST end-to-end flow model

software engineering oriented models that describe the timing behavior of the applications to be verified.

The rest of the paper is organized as follows. Section 2 describes the basic features supported by MAST and summarizes the various capabilities that can be used to solve the challenge. Section 3 and Section 4 discuss the modeling, analysis alternatives, and results obtained for FMTV'2015 Challenge 1 and Challenge 2, respectively. Finally Section 5 discusses the results, the effort required to obtain them and the lessons we think that practitioners may learn from this experience.

2 MAST Modeling and Analysis Approach

MAST provides an open source set of tools that enable engineers developing real-time applications to perform various kinds of schedulability analysis on them. Some aspects for further reading about MAST [4] are the following:

- A very rich model of the real time system is used [1][7]. It is an event-driven model that supports complex dependence patterns among the different tasks. For example, tasks may be activated with the arrival of several events, or may generate several events at their output. This makes it ideal for analyzing or simulating real-time systems designed in languages like UML/ MARTE [3][5].
- It includes offset-based response time analysis techniques [8][10][11][12] that can be applied to heterogeneous systems, combining different scheduling policies in the processors and networks[14].
- The toolset is open source and fully extensible. This has allowed other teams to provide enhanced versions [2].
- A priority assignment tool is included. The technique used by default is HOSPA (Heuristic Optimized Scheduling Parameters Assignment) [14].

The MAST system model assumes a real-time distributed system with multiple processing resources (CPUs and communication networks). This system has distributed *end-to-end flows* with periodic activations. Each end-to-end flow is released by a periodic sequence of external events, and contains a set of *steps.* A step represents the execution of a task in a processor, or a message transmitted in a network. Each periodic release of an end-to-end flow causes the execution of one instance of the set of steps, each step being released when the preceding one in its end-to-end flow finishes its execution. The model also allows mutual exclusion synchronization in the processors.

We assume that the steps are statically assigned to processors. The relative phasing of the activations of different end-to-end flows is assumed to be arbitrary. The events activating the flows may have release jitter and each step may also have an associated initial offset which is the minimum release time for the step, relative to the activation of the external event. We assume that both release jitter values and offsets may be smaller or larger than the period of its end-to-end flow.

Fig. 3 shows an example of an end-to-end flow with three steps, each executing in a different processing resource PR_x. The arrival of the external event that releases the end-to-end flow is represented by a thick vertical arrow labeled e_i. The thin horizontal arrows represent the release of the following steps in the end-to-end flow; a step cannot be executed before the preceding step has been completed. Each step has a worst-case execution time (WCET) and a best-case execution time (BCET). The model assumes no explicit limit for the event queue between a step and the next, although from the analysis results it is possible to obtain the actual size that ensures no event loss.

The timing requirements that we consider are end-to-end deadlines that start at the end-to-end flow instance's period, and must be met by the final step of interest in the flow, although intermediate deadlines are also possible. We allow deadlines to be larger than the periods. Deadlines are very important for two reasons: they allow checking the schedulability of the system, and they are also the optimization criteria to assign priorities.

MAST allows the tasks to use mutually exclusive resources under a real-time synchronization protocol such as the immediate priority ceiling. The tool provides automatic calculation of the priority ceilings of the resources and upper bounds on the blocking times of each task caused by the delays introduced by other lower priority tasks. These blocking times may be pessimistic in some cases, because precedence relations existing in an end-to-end flow are not taken into account for their calculation.

As a result of the schedulability analysis, each step has a worst-case response time (or an upper bound for it) (WCRT) and a best-case response time (or a lower bound for it) (BCRT). These response times are relative to the arrival of the external event. The WCRTs are calculated through response-time analysis, while BCRTs are just the sum of the BCETs of the tasks in the end-to-end flow. As is shown in [9] this lower bound on the best-case response times gives results that are quite tight in comparison with a more accurate analysis.

Among other scheduling policies, MAST supports fixed priorities allowing the user to specify the value of the priority of a thread or message stream. It also supports a boolean parameter called *Preassigned* with the following meaning: if this parameter is set to the value "No", the priority may be assigned by one of the priority assignment tools; otherwise, the priority is fixed and cannot be changed.

MAST also allows modeling different kind of servers based on fixed priorities. In particular, a polling server can be modeled through the *Polling Policy.* This server

represents a periodic task that polls for the arrival of its input event. Thus, the execution of the event may be delayed until the next period. Besides the parameters of the fixed priorities policy, the polling period and the overhead of the polling task can also be specified.

Finally, the model includes the possibility of having multipath end-to-end flows, where for example an event flow can be divided into several paths with a *fork* (multicast) construct.

The MAST model can be expressed as the combination of three complementary sub-models: (1) the platform model, which has processors, networks, schedulers, and threads (called servers), (2) the logic units model, which has operations and shared resources, and (3) the end-to-end flows or real-time situation model, which has the causal flows of externally triggered activities.

3 CHALLENGE 1: VIDEO FRAME PROCESSING

Before entering into the modeling of the problem we will clarify our interpretation of the specified timing requirements. The input stimulus for the end-to-end flow has a periodicity of 40 ms +/- 0.01%. The separate periods for tasks T3 and T4 suggest that the best model for these tasks is as polling servers. The polling period for task T4 is also 40 ms +/- 0.01% while the one for task T3 is 40/3ms +/- 0,05%. On the one hand, this variability is too high in absolute terms to be the drift of a hardware clock and thus it could be understood as a lack of resolution for the clock. On the other hand the specification states that the period is fully stable along the time. This brings up two interpretations of the variability of the clocks. In case (1-1) all three clocks use their nominal periods and the resolution acts as a release jitter; in case (1-2) each clock has a fully stable period, which can be any inside its range of variability. For each of these cases we need to use the worst case situation. All the time values will be expressed in milliseconds.

For case (1-2), the period of T1 is chosen as the one bringing in more work per time unit, which is the lower bound of the period interval: 39.996. The polling periods are chosen to introduce the longest delay, and thus are the upper bounds of the period intervals: 40.004 for T4, and 13.34 for T3. The worst case response time of the system is clearly unbounded since T4 is not able to catch up with all the frames produced by the camera and, disregarding the size of the queue, some frames will be lost.

For case (1-1) and for the same reason, even if we consider release jitter for T1, we need T4's polling period to be less than or equal to T1's period. We will use the most restricting values for them: 39.996ms.

3.1 Modeling

The platform of Challenge 1 has four processors, each with one schedulable resource (task). All of them use a fixed priority (FP) policy scheduler. The four tasks have an FP scheduling parameter assigned (a priority value). Tasks T4 and T3 are scheduled as polling servers, and so additional parameters are needed to include the polling period and polling overhead. There is no overhead associated to the polling servers, as there are no other tasks in their processors that could suffer it. Fig. 4 shows the platform model and the logic units model for this system.

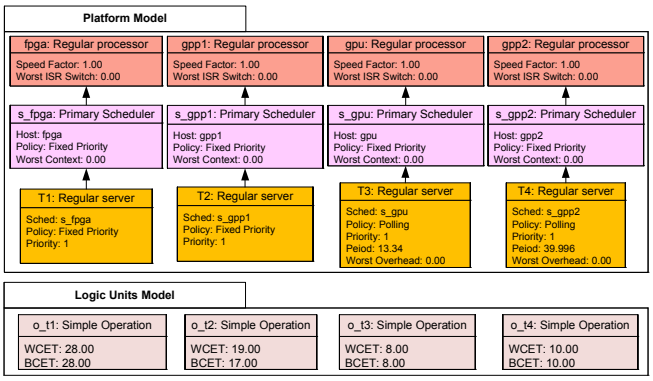

Fig. 4. . Platform and logic units models for Ch_1

The logic units model for Challenge 1 has four operations, each characterized by its worst and best case execution times (WCET and BCET). Since the interactions between tasks in this case do not involve mutual exclusion in the access to shared resources, there are no such elements in the model.

The real-time situation model for Challenge 1 is depicted in Fig. 5. T1 is activated periodically and T2 starts when T1 ends. T3 and T4 work periodically by processing the data provided at their input buffers.

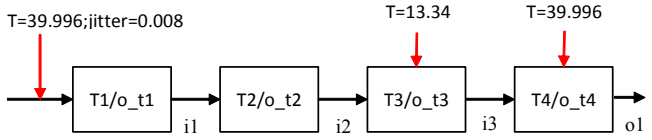

Fig. 5. . Real-time situation for Challenge 1

3.2 Analysis

When launching the MAST analysis it is possible to select the default tool, but in some cases it is possible to get a less pessimistic result by using the technique called offset-based with precedence relations [11]. For Challenge 1 this is the technique that provided the best results.

3.3 Results

MAST expresses the results as response times and output jitter values associated to the events that connect steps along the causal flow (i1, i2, i3 and o1 in this case). Their values are relative to the triggering external stimulus. The analysis results obtained for Challenge 1 case (1-1) are shown in Table 1.

Table 1Response times for Challenge 1 (ms)

	i1	i2	i3	o1
BCRT	28	45	53	63
WCRT	28.008	47.008	68.348	118.344
Max. Output Jitter	0.008	2.008	15.348	55.344

The questions posted in Challenge 1 are divided in two sets: 1A and 1B.

Challenge 1A: Compute:

1. The minimum (*b1a1*) and maximum (*w1a1*) latencies for a given frame from the camera output to the display input, for a buffer size n = 1
2. The minimum (*b1a3*) and maximum (*w1a3*) latencies for a given frame from the camera output to the display input, for a buffer size n = 3

Response. As stated at the beginning of this section, in the worst interpretation for case (1-2) the WCRT is unbounded for any finite buffer size; so w1a1 = w1a3 = unbounded.

For case (1-1), supposing correctly designed polling periods, the BCRT and WCRT for unlimited buffer size are 63 and 118.344 ms, respectively (see Table 1).

The MAST results model includes a result called Num_Of_Queued_Activations, which is defined as the maximum number of pending activations in the input queue of the activity preceding the referenced event. Currently, only the MAST simulator produces this parameter. We have an analytical method based on the response times to calculate that value, but it is not yet implemented in the analysis tool. The response times are calculated by assuming an unlimited buffer size and from these values we obtain the worst case latency of an internal event in the buffer. An upper bound for this buffer latency is the WCRT of the step consuming the event minus its WCET and minus the BCRT of the step injecting the event in the buffer. Finally, the number of queued activations is obtained as the ceiling of this buffer latency divided by the period.

By applying this method the buffer size should be n=2 for the period of 39.996ms in the frame production. Therefore, with n=1 we cannot assure that the system is schedulable; since frames arriving at a full buffer are discarded the response time for n=1 is bounded by the response time for unlimited buffer size, as the system would have less work to process, thus *w1a1= 118.344ms*. With n=3 there is no frame loss and therefore the results for unlimited buffer size (Table 1) apply: *b1a3= 63ms* and *w1a3=118.344ms*.

Challenge 1B: Due to the different clock drifts, all frame with a same index may be discarded at the entrance of the buffer at the input of the task T4. Compute:

1. the minimum time distance between two frames produced by the camera (*d1b1*) that will not reach the display, for a buffer size n = 1
2. the minimum time distance between two frames produced by the camera (*d1b3*) that will not reach the display, for a buffer size n = 3

Response. As mentioned in the introduction, we interpret these questions as finding the minimum distance between two frames such that there is no frame loss. In this way, if the time distance is just below the computed value then there might be a frame loss and a frame will not reach the display.

As said, if polling periods are allowed to be larger than the camera frame generation period the system is not schedulable, so here again our response will consider properly designed polling periods for T3 and T4.

As mentioned in the response to Challenge 1A, the buffer size should be n=2 for the period of 39.996ms in the frame production. The challenge states that all additional frames with the same index are discarded at the entrance of the buffer, so there is only one frame stored at this buffer per frame produced by the camera, thus resulting in a real period of 39.996ms for the event generated at the output of task T3. Notice that this period can be affected by the maximum output jitter of its event (i3), which is 15.348ms as shown in Table 1.

If we have a buffer size of n=1 then the minimum time distance between two frames produced by the camera should not be less than *d1b1=55.344ms* (according to our method for calculating the buffer size), which allows task T4 to process the frame awaiting in the buffer before the next frame is injected in the buffer by task T3. We assume that the rest of the characteristics of the system remain the same, in particular the activation periods of tasks T3 and T4 (13.34ms and 39.996ms, respectively).

In this challenge, considering the buffer size n=3 is equivalent to saying that we can produce frames at a higher rate. By simple inspection of task T1, we find a lower bound of 28ms for its period in order to have the utilization of the FPGA resource below 100%. We also assume that reducing the period of task T1 implies reducing the period of task T4 to the same value, to prevent frame loss. Keeping the timing consistency for task T3 we assume that its period is a third of this value. Taking all these changes into account, we repeat the calculation of response times and re-evaluate the size of the buffer resulting in a value of n=2. Therefore, n=3 will not help in reducing T1's period any further, and so *d1b3=28ms*.

4 Challenge 2: Tracking and Camera Control

The relevant characteristics for modeling Challenge 2 are: Each function is mapped to one task. All tasks are in the same processor using FP preemptive scheduling with priority order: T2 > T6 > T5 > T7. All tasks are triggered by the arrival of data at their inputs. T7 is invoked from an action in the middle of T6.

4.1 Modeling

Challenge 2 analyses processor GPP1. This platform is modeled by a processor and a scheduler, and has 4 tasks in it. This is shown in Fig. 6.

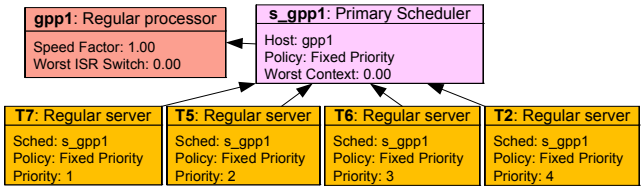

Fig. 6. . Platform model for Challenge 2

Challenges 2A and 2B differ in the use of a shared resource protected by the priority ceiling protocol between T5 and T2. Fig. 7 shows the logic units model for Challenge 2B.

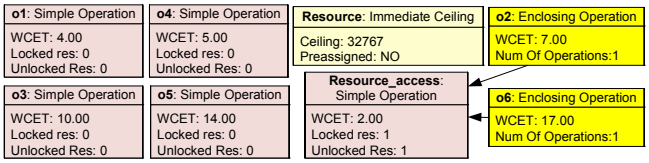

Fig. 7. . Logic units model for Challenge 2B

The real-time situation for Challenge 2 has two external stimuli, hence it is described by means of two end-to-end flows. Fig. 8 shows the model for the flow containing the processing task (T2), choosing the most demanding period of 39.996ms.

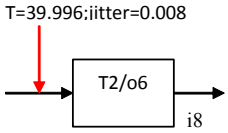

Fig. 8. . End-to-end flow for task T2 in Challenge 2

The second flow has a period of 100 (T100). To model the invocation of T7 from an action in the middle of T6 this task is broken up in two steps: T6/o3 before the invocation of T7 and T6/o4 after. The straightforward model includes a fork construct as it is shown in Fig. 9.

The current version of MAST uses the holistic analysis technique [15][13] to analyze this kind of multipath end-to-end flow containing fork elements. This analysis is more pessimistic than the offset-based techniques used to analyze linear end-to-end flows; for this reason two versions of the real-time situation are proposed. One directly models the scenario proposed in the challenge using a fork construct (multipath), and the other works with an equivalent linear version (linear) that assumes a concrete order for the execution of the steps that follow the fork, considering the priorities they have assigned and the fact that the flow executes in a single processor.

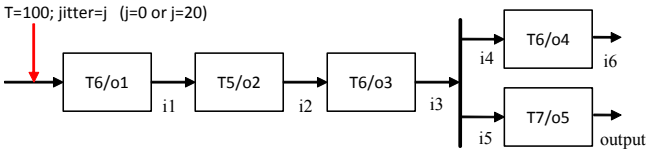

Fig. 9. . Direct model of T100 in Challenge 2 with a fork construct.

Since T6's priority is greater than T7's, o4 precedes o5 and it is possible to build the equivalent linear model for T100 shown in Fig. 10. If their priorities were the same we would need to build two equivalent linear models and use the worst results.

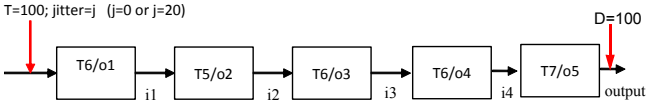

Fig. 10. . Linear model for T100 in Challenge 2.

The questions in Challenge 2 need two additional variants for T100, one with jitter 0 and another one with jitter 20.

4.2 Analysis

Challenge 2 not only requires analysis but also an optimum assignment of priorities. RTA has been applied to both, the multipath, and the linear models. For the multipath model only the holistic technique is available [15], while for the linear version three offset-based techniques have been essayed: The approximate with precedence relations (Approx_w_Pr) [11], Slanted [8], and Brute-Force [16]. The blocking time caused by the shared resource (in Challenge 2B) is automatically taken into account by all the analysis techniques in MAST.

For the assignment of priorities only the least pessimistic linear version of T100 is used. Since priorities are to be changed by the optimizer, and this change may lead to a different linear model, two scenarios of the model are analyzed. Each has sufficiently separated priority values preassigned to T6 and T7. Variant *A* is designed so that step o4 goes before o5 (T6>T7). Variant *B* uses the opposite order (T7>T6). The linear model for variant *B* is shown in Fig. 11; its deadline is attached to i4. The deadline for variant *A*, shown in Fig. 10, is at the output event.

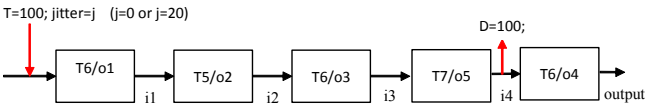

Fig. 11. . Linear model for priority assignment in T100, variant B

Additionally it is important to notice that for doing the priority assignment the tool requires the specification of deadlines (D) for the end-to-end flows of interest. The deadlines are the input specifying the optimization criteria. From the specification it seems that the latency of interest is only the one from the activation of T6 to the termination of T7, so for T100 let D=100ms (case h). In the understanding that this processor is just a part in a system we also searched the cases when T2 has D=1000ms (case m) and deadlines are equal to periods (case n).

4.3 Results

Table 2 summarizes the results of the analysis for Challenge 2A using the initial priority assignment, while Table 3 summarizes the results of the analysis for Challenge 2B.

Table 2 RTA for Challenge 2A, no shared resource (ms)

	Multipath Holistic		Offset-based Approx_w_Pr		Offset-based Slanted		Offset-based BruteForce	
jitter	0	20	0	20	0	20	0	20
BCRT	28	28	32	32	32	32	32	32
WCRT	726	831	74	94	463	498	463	498

To search for the priority assignment requested in Challenge 2B, we used the default MAST tool (HOSPA). The values used for T6 ad T7 priorities in variants *A* and *B* were 16 and 8, in a range of 1 to 32 (32 the highest). The

Table 3 RTA for Challenge 2B, with shared resource (ms)

	Multipath Holistic		Offset-based Approx_w_Pr		Offset-based Slanted		Offset-based BruteForce	
jitter	0	20	0	20	0	20	0	20
BCRT	28	28	32	32	32	32	32	32
WCRT	730	856	78	98	465	545	465	545

combinations of the two variants and three cases for the deadlines give way to the following six scenarios:

- (ScAh) Priority T6 > T7, T100 D=100
- (ScAm) Priority T6 > T7, T100 D=100, and T2 D=1000
- (ScAn) Priority T6 > T7, and D=T
- (ScBh) Priority T6 < T7, T100 D=100
- (ScBm) Priority T6 < T7, T100 D=100, and T2 D=1000
- (ScBn) Priority T6 < T7, and D=T

The results for all these scenarios including the obtained priority assignments are shown in Table 4.

Table 4 Priority assignment and WCRTs (ms)

		WCRT		Priority Assignment			
		T100	T2	T6*	T5	T7*	T2
j=0	Sc*A*h	42	57.008	16	27	8	1
	Sc*A*m	74	24.008	16	29	8	22
	Sc*A*n	74	19.008	16	22	8	29
	Sc*B*h	39	57.008	8	24	16	1
	Sc*B*m	54	31.008	8	24	16	12
	Sc*B*n	88	19.008	8	12	16	24
j=20	Sc*A*h	62	57.008	16	27	8	1
	Sc*A*m	94	24.008	16	29	8	22
	Sc*A*n	94	19.008	16	22	8	29
	Sc*B*h	59	57.008	8	24	16	1
	Sc*B*m	74	31.008	8	24	16	12
	Sc*B*n	106	19.008	8	21	16	27

* These priority values were preassigned in the model to make them consistent with the chosen variant, A or B

The concrete questions posted in Challenge 2 are:

Challenge 2A. With no shared resource, compute:

1. The best-case (*b2A0*) and worst-case (*w2A0*) end-to-end latencies from the activation of T6 to the termination of T7 for a jitter value j = 0ms
2. The best-case (*b2A20*) and worst-case (*w2A20*) end-to-end latencies from the activation of T6 to the termination of T7 for a jitter value j = 20ms

Response. *b2A0=32ms; w2A0=74ms;*
b2A20=32ms; w2A20=94ms;

Challenge 2B. With the 2ms mutually exclusive access to the shared resource between Tasks T2 and T5, compute:

1. The best-case (*b2B0*) and worst-case (*w2B0*) end-to-end latencies from the activation of T6 to the termination of T7 for a jitter value j = 0ms
2. The best-case (*b2B20*) and worst-case (*w2B20*) end-to-end latencies from the activation of T6 to the termination of T7 for a jitter value j = 20ms
3. The optimum priority assignment minimizing the worst-case latency for a jitter value J= 0ms (*priorder-2B0*) and J= 20ms (*priorder-2B20*)

Response. *b2B0=32ms; w2B0=78 ms;*
b2B20=32ms; w2B20= 98ms;

Minimizing the worst-case latency from the activation of T6 to the termination of T7, the priority order for both jitter values is:

priorder-2B0(wcrt 39ms)=priorder-2B20(wcrt 59ms)
= *T5>T7>T6>T2* (Scenario *B*h)

Notice that for that priority order T2 will have a latency larger that its period.

Minimizing the worst-case latency for T2, the best result is:

priorder-2B0=priorder-2B20 (T2 WCRT 19.008ms)
= *T2>T5>T6>T7* (Scenario *A*n)

5 CONCLUSIONS

This paper gives responses to all the questions posted in this challenge. This has been done using the MAST suite of tools for response time schedulability analysis. The interpretation of the system requirements that has driven the models and corresponding analyses has been that there should be no loss of video frames. All digital material with the input models as well as the results from the tools can be retrieved from http://mast.unican.es/waters15challenge/mastmodels.zip

Understanding the challenge took a PhD student about an hour and a half, let us say, the time to read it carefully. But identifying the correct models and tools, required several trials and organized experiments plus the experience of a senior practitioner. The actual time used for solving challenge 1 was about two labor days distributed along a week. Half a day for creating the models and running the tool, and the rest for understanding the results and iterating to bring up all the alternatives, put them in shape and formalize the response. Challenge 2 was much easier to understand but the fork issue made it more difficult to model and much more laborious for finding the best priority assignment. The actual time devoted for solving challenge 2 was about three days, one for modeling and getting the first results using different tools, the rest for iterating over the analysis in search of a proper priority assignment and for keeping all the documentation in order.

Besides the work for formalizing the paper, the most difficult part was trying to avoid inferring the reasoning behind the design choices... we just had to accept them. This surely comes from the fact that the challenge description does not include all the nasty constraints the actual Aerial Video System designer faces, but presents only an abstracted view that is meant to be sufficient for timing verification.

Some hints that may help designers to get less pessimistic worst case response times by schedulability analysis are: (1) Avoid using polling servers if direct activation is possible. This way of synchronization between tasks and processors is much more flexible and gets better response times. If not avoidable, ensure that the periods are faster than the minimum time between arrivals. (2) Use asynchronous calls (forks) only to trigger steps in other processors. For local interactions synchronous calls are easier to analyze and produce less

pessimistic results. (3) Translate user requirements into end-to-end deadlines. This helps to automate the assignment of priorities to tasks and the calculation of slacks, hence to optimize the use of resources. In the accompanying material a solution to these challenges is posted on a version of the given system whose design takes these hints into account. Another optimization could be using time offsets for the tasks of the end-to-end flows. This could help in minimizing jitter and thus increasing schedulability. MAST is able to use these offsets in the analysis.

Real-time systems theory has developed a large number of scheduling policies and analysis techniques. Engineers trying to develop industrial real-time systems need adequate skills to master those techniques, but also the tools that allow them to model their systems and apply the techniques with confidence. MAST was created to serve both as an engineering tool and as a research platform for developing such modeling and analysis techniques. If analysis tools such as MAST are to be used by engineers who do not master real-time analysis, it is necessary that the system architecture is automatically generated to ensure a smooth mapping between the software implementation and its real-time model.

REFERENCES

[1] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M.Drake, MAST: Modeling and Analysis Suite for Real-Time Applications, in Proc. of the *Euromicro Conference on Real-Time Systems*, June 2001.

[2] Marco Panunzio and Tullio Vardanega, Schedulability analysis of Ravenscar systems with MAST+. http://www.artist-embedded.org/docs/Events/2011/Models_for_SA/06-MAST+Marco_Panunzio.pdf.

[3] Object Management Group, UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, version 1.1, OMG document formal/2011-06-02, 2011.

[4] MAST home page: http://mast.unican.es/

[5] http://mast.unican.es/umlmast/marte2mast

[6] https://waters2015.inria.fr/files/2014/11/FMTV-2015-Challenge.pdf

[7] M. González Harbour, J.J. Gutiérrez, J.M. Drake, P. López Martínez, and J.C. Palencia, "Modeling distributed real-time systems with MAST 2," Journal of Systems Architecture 59(6), pp. 331-340, 2013.

[8] Jukka Mäki-Turja and Mikael Nolin, "Efficient implementation of tight response-times for tasks with offsets," Journal of Real-Time Systems. Volume 40 Issue 1, pp. 77 - 116, 2008.

[9] J. C. Palencia, J. J. Gutiérrez and M. González Harbour, "Best-Case Analysis for Improving the Worst-Case Schedulability Test for Distributed Hard Real-Time Systems", Proceedings of 10th Euromicro Workshop on Real-Time Systems, IEEE Computer Society Press, pp. 35-44, June 1998.

[10] J.C. Palencia and M. González Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets," Proc. of the 18th. IEEE Real-Time Systems Symposium (RTSS), Madrid, Spain, pp. 26-37, 1998.

[11] J.C. Palencia, and M. González Harbour, "Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems," Proc. of the 20th IEEE Real-Time Systems Symposium, USA, pp. 328-339, 1999.

[12] J.C. Palencia and M. González Harbour, "Offset-Based Response Time Analysis of Distributed Systems Scheduled under EDF," Proc. of the 15th Euromicro Conference on Real-time Systems (ECRTS), Porto, Portugal, pp. 3-12, 2003.

[13] K. Tindell, and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems," Microprocessing & Microprogramming, Vol. 50, Nos.2-3, pp. 117-134, 1994.

[14] J.M. Rivas, J.J. Gutiérrez, J.C. Palencia, and M.González Harbour, "Distributed Schedulability Analysis and Optimization of Heterogeneous EDF and FP Real-Time Systems," Proc. of the 23rd Euromicro Conference on Real-Time Systems (ECRTS), Porto, Portugal, pp. 195-204, 2011.

[15] J. J. Gutiérrez, J.C. Palencia and M. González Harbour, "Schedulability Analysis of Distributed Hard Real-Time Systems with Multiple- Event Synchronization," Proc. of 12th Euromicro Conference on Real-Time Systems, Stockholm (Sweden), pp. 15-24, 2000.

[16] K. Tindell, "Adding Time-Offsets to Schedulability Analysis", Technical Report YCS 221, Dept. of Computer Science, University of York, England, January 1994.