Verification of an Aerial Video System

Youcheng Sun
Dept. of Computer Science
University of Oxford, UK

Étienne André
LIPN, CNRS, UMR 7030
Université Paris 13, France

Giuseppe Lipari
CRIStAL - UMR 9189
University of Lille, France

***Abstract*—In this paper we propose solutions to the consolidated version of the WATERS 2015 industrial challenge of a distributed video processing system using the formalism of Parametric Timed Automata (PTA). The first challenge is harder because of the very large number of states to be analyzed, so we only provide upper bounds. The second challenge consists of a real-time scheduling problem for which we provide exact solutions by using a scheduling analysis based on the critical instant, and a PTA model.**

Furthermore, regarding the second challenge, we provide formal analysis for the frequency of possible temporal violations in the system, by applying the latest results from the so called weakly hard real-time schedulability analysis. This improvement contributes the major extension of this paper with respect to our solutions to the previous version of the challenge.

I. Introduction

With the increasing size and complexity of concurrent systems, the need for formal verification techniques has become higher and higher in the past decades. Systems mixing time and concurrency are especially subject to undesired behaviors (deadlocks, race conditions, etc.); often, their complexity makes the use of formal methods very challenging.

In this paper, we address the ”consolidated WATERS industrial challenge 2015” proposed by Thales (http://waters2017.inria.fr/challenge/#Challenge17) by extending our solutions [ALS15] (to its previous version) that use the formalism of *parametric timed automata* (PTA). Timed automata (TA) [AD94] are a well-known formalism for specifying and verifying concurrent real-time systems. TA extend finite-state automata with a set of clocks (real-time variables growing linearly) that can be compared with integer constants. TA are used in several powerful tools such as UPPAAL [LPY97] or PAT [SLDP09]. However, the binary answer (“yes” or “no”) output by model checking is not always satisfactory; indeed, it does not allow to change or optimize some values of the system constants, nor (in general) to evaluate the system robustness, *i.e.,* the infinitesimal variation of timing constants while preserving the reachability. PTA [AHV93] extend TA with rational-valued parameters allowed in place of constants. PTA are particularly well suited to verify systems where some timing delays are known with uncertainty. The natural drawback of PTA is the infamous state space explosion, that may prevent the verification to be truly scalable. We will use the tool IMITATOR [AFKS12] that takes PTA as an input formalism.

This work is partially supported by the ANR research program PACS (ANR-14-CE28-0002).

Improvements w.r.t. solutions in [ALS15] :

- The improvement comes from the solution to Challenge 2, where we provide also an analysis for the frequency of possible temporal violations in the video system. Although this is not explicitly specified by the challenge itself, we believe the result by natural fits into the interest of the challenge, and it offers a formal approach to evaluate the system performance.
- This extension is based on the so called weakly hard schedulability analysis, which estimates the number of deadline misses within an arbitrary time window.

Outline: We briefly review PTA and IMITATOR (Section II). Then, we recall the challenge (Section III). We derive solutions to Challenge 1A formally by using IMITATOR (Section IV). We provide a different PTA model for the much harder Challenge 1B (Section V) for which we derive an upper bound. Then, we analyze the problem in Challenge 2 (Section VI) and *the improvement in the new solution is at Section VI-C*. Finally, we conclude in the work (Section VII).

II. Parametric Timed Automata

Timed automata are finite-state automata augmented with clocks, *i.e.,* real-valued variables increasing uniformly, that are compared within guards and invariants with timing delays [AD94]. Parametric timed automata (PTA) [AHV93] extend timed automata with parameters, *i.e.,* unknown constants, that can be used in guards and invariants.

Given a set X of clocks (real-valued variables) and a set P of parameters (unknown rational-valued constants), a constraint C over X and P is a conjunction of linear inequalities on X and P[1]. Given a parameter valuation (or point) v, we write $v \models C$ when the constraint where all parameters within C have been replaced by their value as in v is satisfied by a non-empty set of clock valuations.

Definition 1: A parametric timed automaton (PTA) $\mathcal{A}$ is $(\Sigma, L, l_0, X, P, I, E)$ with Σ a finite set of actions, L a finite set of locations, $l_0 \in L$ the initial location, X a set of clocks, P a set of parameters, I the invariant assigning to every $l \in L$ a constraint over X and P, and E a step relation consisting of elements (l, g, a, R, l'), where $l, l' \in L$, $a \in \Sigma$, $R \subseteq X$ is the set of clocks to be reset, and the guard g is a constraint over X and P.

[1]Note that this is a more general form than the strict original definition of PTA [AHV93]; since most problems for PTA are undecidable anyway, this has no practical incidence, and increases the expressiveness of the formalism.

The semantics of a PTA $\mathcal{A}$ can be found in, *e.g.,* [AHV93], [AS13].

Most problems related to PTA (*e.g.,* the parametric reachability of a location) are undecidable [AHV93], [JLR15], with some decidable syntactic subclasses related to the use of the parameters [HRSV02], [BL09], [JLR15] or on the number of clocks [AHV93], [BO14], [BBLS15] (see [And16] for a survey). We do not consider it as drawback, as we use semi-algorithms that "often" terminate in practice; we will see this is also the case for solving challenge 1A.

IMITATOR [AFKS12] is a tool for modeling and verifying systems modeled using parametric timed automata. In its latest version (2.9), IMITATOR implements several algorithms, among which:

- parametric reachability analysis: "find all parameter valuations such that some state is reachable"
- parametric robustness analysis: "given a parameter valuation v, find other valuations for which the discrete (untimed) behavior is the same as v"
- behavioral cartography: "find all parametric subspaces in which the discrete behavior is uniform".

IMITATOR extends PTA with some useful constructions, such as constants, global discrete integer variables, strong broadcast synchronization between components, and stopwatches (*i.e.,* the power of stopping some clocks in some locations).

The input syntax of IMITATOR is a text file, which makes it possible to write models either manually or using scripts (*e.g.,* for large models derived from another formalism). For example, in the past several parametric schedulability analysis models were generated automatically using scripts.

III. A Brief Description of the System

We only briefly recall the systems and the challenges; the full description is available at the workshop Web page.

A. Challenge 1

There are four tasks T1, T2, T3 and T4, distributed in different processing units and performing respective functionalities. The task T1 periodically receives frames from the camera and pre-processes them. Task T2 embeds further tracking information into the video frame pre-processes by Task T1. Task T2 then inserts the video frame into a register, denoted as Register23. Then Task T3 reads the frame from the register, removes the noise and tries to put the resulting video frame into a buffer, denoted as Buffer34. In the end, Task T4 reads frames from the buffer, converts them from digital to analogue and sends the final frame to the display.

Tasks T1, T3 and T4 are periodic, but their triggering clocks are subject to drift. That is, their periods P_1, P_3 and P_4 are unknown constants. More specifically, $P_1 \in [40 - 40 \times 0.01\%, 40 + 40 \times 0.01\%]$ms, $P_3 \in [\frac{40}{3} - \frac{40}{3} \times 0.05\%, \frac{40}{3} + \frac{40}{3} \times 0.05\%]$, and $P_4 \in [40 - 40 \times 0.01\%, 40 + 40 \times 0.01\%]$ms. Task T2 is triggered by the completion of T1.

Each task has its Best-Case and Worst-Case Execution Time (BCET and WCET) or Latency (BCL and WCL): $\text{BCET}_1 = \text{WCET}_1 = 28$ms, $\text{BCL}_2 = 17$ms, $\text{WCL}_2 = 19$ms, $\text{BCET}_3 = \text{WCET}_3 = 8$ms. As for task T4, when it reads Buffer34 and there is no frame within the buffer, it performs an empty cycle with execution 1ms; otherwise, it executes 10ms and sends the result to display.

For such a video frame processing subsystem, we aim to tackle the following challenges.

Challenge 1A:
Compute the minimum and maximum latencies for a given frame from the camera output to the display input, for a buffer size $n = 1$ (challenge 1A.1) and $n = 3$ (challenge 1A.3).

Given the difficulty of Challenge 1A, we will focus on deriving the lower bound and upper bound of the end-to-end latency.

Challenge 1B:
Due to the different clock drifts, all frames with a same index may be discarded at the entrance of the buffer at the input of the task T4. Compute the minimum time distance between two frames produced by the camera that will not reach the display, for a buffer size $n = 1$ (challenge 1B.1) and $n = 3$ (challenge 1B.3).

B. Challenge 2

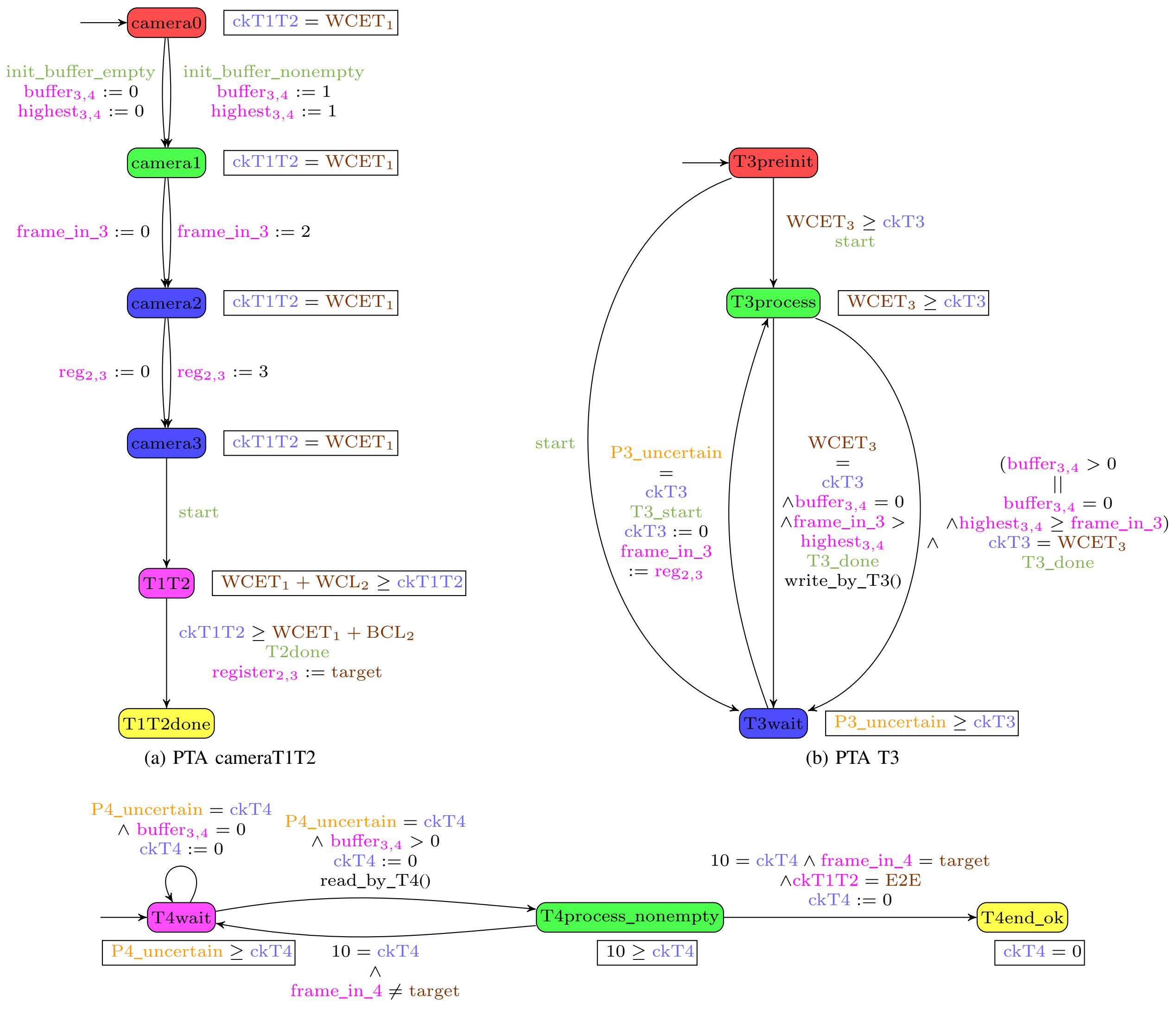
The complete system also includes a camera tracking subsystem that identifies objects on the camera images and commands the camera motors so to follow the objects. This subsystem consists of 3 additional tasks: Task T6 is a periodic task with period $P_6 = 100$ and it can start with a certain jitter J_6. Task T5 is activated by Task T6 with a synchronous call. Task T7 is activates asynchronously by Task T6 and controls the motors. Task T6 execution time is: $C_{6,1} = 4$ms before invoking Task T5; $C_{6,2} \in [9, 10]$ ms after the completion of Task T5 and before the invocation of Task T7; $C_{6,3} \in [4, 5]$ ms after the invocation to Task T7. Task T5 has an execution time of $C_5 \in [4, 7]$ ms. Task T7 has an execution time of $C_7 \in [11, 14]$ ms. All tasks execute on the same processor together with Task T2 (described in the first challenge), and they are scheduled by a fixed priority scheduler. Task T2 has a computation time of $C_2 = 17$ms. Finally, task priorities are assigned so that T2 > T6 > T5 > T7.

Challenge 2A:
1) Compute the best and worst-case end-to-end latencies from the activation of Task T6 to the completion of Task T7 when $J_6 = 0$.
2) Compute the best and worst-case end-to-end latencies when $J_6 = 20$ms.

Challenge 2B:
Tasks T2 and T5 have access to a shared mutually exclusive resource protected by the priority ceiling protocol. The access to the shared resource takes 2ms for both tasks. 1) Compute the best-case and worst-case end-to- end latencies from activation of T6 to termination of T7 for a jitter value $J_6 = 0$ms 2) compute the best-case and worst-case end-to-end latencies from activation of T6 to termination of T7 for a jitter value $J_6 = 20$ms. 3) The optimum priority assignment minimizing the worst-case latency for a jitter value $J_6 = 0$ms and $J_6 = 20$ms.

IV. Solving Challenge 1A Using IMITATOR

In this part, we formally solve the challenge using PTA. At first, we will solve the case with $n = 1$ for Buffer34. The


camera0
$\text{ckT1T2} = \text{WCET}_1$
init_buffer_empty
$\text{buffer}_{3,4} := 0$
$\text{highest}_{3,4} := 0$
init_buffer_nonempty
$\text{buffer}_{3,4} := 1$
$\text{highest}_{3,4} := 1$
camera1
$\text{ckT1T2} = \text{WCET}_1$
frame_in_3 := 0
frame_in_3 := 2
camera2
$\text{ckT1T2} = \text{WCET}_1$
$\text{reg}_{2,3} := 0$
$\text{reg}_{2,3} := 3$
camera3
$\text{ckT1T2} = \text{WCET}_1$
start
T1T2
$\text{WCET}_1 + \text{WCL}_2 \geq \text{ckT1T2}$
$\text{ckT1T2} \geq \text{WCET}_1 + \text{BCL}_2$
T2done
$\text{register}_{2,3} := \text{target}$
T1T2done
T3preinit
$\text{WCET}_3 \geq \text{ckT3}$
start
T3process
$\text{WCET}_3 \geq \text{ckT3}$
start
P3_uncertain = ckT3
T3_start
ckT3 := 0
frame_in_3 := $\text{reg}_{2,3}$
$\text{WCET}_3 = \text{ckT3}$
$\wedge \text{buffer}_{3,4} = 0$
$\wedge \text{frame\_in\_3} > \text{highest}_{3,4}$
T3_done
write_by_T3()
$(\text{buffer}_{3,4} > 0$
$||$
$\text{buffer}_{3,4} = 0$
$\wedge \text{highest}_{3,4} \geq \text{frame\_in\_3})$
$\wedge$
$\text{ckT3} = \text{WCET}_3$
T3_done
T3wait
P3_uncertain $\geq$ ckT3
P4_uncertain = ckT4
$\wedge\ \text{buffer}_{3,4} = 0$
ckT4 := 0
P4_uncertain = ckT4
$\wedge\ \text{buffer}_{3,4} > 0$
ckT4 := 0
read_by_T4()
$10 = \text{ckT4} \wedge \text{frame\_in\_4} = \text{target}$
$\wedge \text{ckT1T2} = \text{E2E}$
ckT4 := 0
T4wait
T4process_nonempty
T4end_ok
P4_uncertain $\geq$ ckT4
$10 = \text{ckT4}$
$\wedge$
$\text{frame\_in\_4} \neq \text{target}$
$10 \geq \text{ckT4}$
$\text{ckT4} = 0$

(a) PTA cameraT1T2

(b) PTA T3

(c) PTA T4

Fig. 1: Modeling the system of Challenge 1A for $n = 1$

corresponding PTA model is shown in Figure Fig. 1. Later, we show how to adapt this model to a larger buffer.

Experimental environment: We used the latest version of IMITATOR (v2.7-beta2, build 1073) with no specific modification of the tool. We just used a small Python script to parse the long list of intervals that IMITATOR outputs, and to produce a single minimum and maximum. The latest version (2.9) of the tool automatically does this, and the Python script is therefore not needed anymore. Sources, binary and models are available at [XP].

A. *Camera, Task T1, Task T2*

In order to reduce the state space, we model the camera, Task T1 and Task T2 into a single PTA (Fig. 1a). We also use this PTA to non-deterministically initialize the buffer and the frame currently processed by Task T4.

We choose an arbitrary frame with index target for end-to-end latency estimation and *we start from the exact point such that the target frame is handled from Task T1 to task T2*. A clock ckT1T2 is *initialised to be* WCET_1 and measures the end-to-end latency of target frame. Discrete variables frame_in_3 and frame_in_4 represent the index of frames in Task 3 and Task 4 respectively. The value 0 is used to denote that there is no frame in a task. $\text{reg}_{2,3}$ and $\text{buffer}_{3,4}$ are for frames within Register23 and Buffer34. While we assume frame index in the system is monotonically increased, $\text{highest}_{3,4}$ denotes the highest frame index among frames having been stored inside the buffer. For the register, buffer and Task 3, they may or may not initially contain a frame.

We do not model the period of the camera (or task 1), since we are only interested in a single frame. Let us now model the buffer. IMITATOR does not support other kinds of global variables than discrete integer variables. For a buffer with one slot ($n = 1$), its status can be modeled by using two discrete variables $\text{buffer}_{3,4}$ and $\text{highest}_{3,4}$.

- $\text{buffer}_{3,4}$ denotes the index of current frame inside Buffer34;
- $\text{highest}_{3,4}$ is the highest index recorded so far.

B. Task T3

The period of Task T3 is a *parameter* P3_uncertain, that is initialised as follows:

$$\text{P3\_uncertain} \in [40 - \text{P3\_delta}, 40 + \text{P3\_delta}]$$

where $\text{P3\_delta} = 0,05\,\% \times 40 = \frac{1}{150}$. Recall that parameters in PTA are unknown *constants*, *i.e.,* the value of which cannot evolve during the execution; this is exactly what we need to model P3_uncertain.

Task T3 is modeled by a periodic PTA in Fig. 1b. ckT3 is a clock variable for recording task 3's activation and execution. At the initial point, the PTA T3 is non-deterministically waiting for a new activation or executing. When T3 finishes execution, it writes into Buffer34 if the buffer is empty and its current frame has not been put into the buffer. We define a function call for this writing operation write_by_T3():

$$\text{buffer}_{3,4} := \text{highest}_{3,4} := \text{frame\_in\_3}$$

IMITATOR does not support function call in a model; here we still utilise the notation of function call for simplicity. Otherwise, task 3's writing fails. As shown in PTA T3, we utilise the operator "||" ("or") in the edge representing writing failure. Again, this is for saving some space.

C. Task T4

We use here a modeling mechanism similar to Task T3. The period of Task T4 is a *parameter* P4_uncertain, that is initialised as follows:

$$\text{P4\_uncertain} \in [40 - \text{P4\_delta}, 40 + \text{P4\_delta}]$$

where $\text{P4\_delta} = 0,01\,\% \times \text{P4} = 0.004$.

Task T4 is modeled by a periodic PTA as in Fig. 1c. A clock variable ckT4 is used for task 4's periodic activation and execution. When T4 activates, If the buffer is empty, T4 goes directly back to another waiting cycle. According to the system specification, there should an empty processing session for task 4. However, such an empty processing does not affect the end-to-end latency of any frame, and we omit it in PTA T4. If the buffer is not empty, task 4 reads a frame from the buffer by the function call read_by_T4():

$$\text{frame\_in\_4} := \text{buffer}_{3,4},\ \text{buffer}_{3,4} := 0$$

Task T4 takes 10ms to process a frame, after the processing, if its current frame is the target one, ckT4 is reset and T4 moves to an ending location. $\text{E2E} \geq 0$ is the parameter for representing all possible end-to-end latencies of the target frame.

D. Deriving the Latency for $n = 1$

As we have seen, in order to avoid exploring the exact configurations in the system, we target a single frame (which explains the non-cyclic behavior of the PTA modeling the camera, tasks 1 and 2) that is output from the task 1 at $t = \text{WCET}_1$. The main idea is that, at $t = \text{WCET}_1$, the initial state must be *arbitrary*, *i.e.,* encode all possible configurations that could happen in the system. However, such a model may be pessimistic for containing behaviors that cannot really happen in the system. Again, we aim to derive upper and lower bounds on end-to-end latency of an arbitrary frame.

After developing the model, we use IMITATOR to perform parametric reachability analysis of location T4end_ok, that is we ask IMITATOR to return all parameter valuations such that T4end_ok is reachable. Then, IMITATOR hides (using existential quantification) all parameters except E2E, and then returns a list of intervals for E2E. After some post-processing to unify intervals, we get

$$\text{E2E} \in [63, 145.008].$$

E. Deriving the Latency for $n = 3$

For the case of $n = 3$ for Buffer34, we can keep the same IMITATOR model, with the exception of the buffer modeling. We assume that access to elements in the buffer follows a FIFO manner.

Let us model Buffer34 for $n = 3$: besides the variable $\text{highest}_{3,4}$, we need 3 more discrete variables for frames within each slot in the buffer: $\text{buffer}^1_{3,4}, \text{buffer}^2_{3,4}$ and $\text{buffer}^3_{3,4}$. When non-deterministically initializing the buffer, we need to take into account all possible scenarios of frame occupation within it. When T3 writes into the buffer, it needs to find the first free position (say xth); thus the writing call becomes write_by_T3():

$$\text{buffer}^{\text{x}}_{3,4} := \text{highest}_{3,4} := \text{frame\_in\_3}.$$

Similarly, for T4 to read from the buffer, we call the adapted function read_by_T4():

$$\text{frame\_in\_4} := \text{buffer}^1_{3,4}$$

$$\forall x \in \{1, 2\}\ \text{buffer}^{\text{x}}_{3,4} := \text{buffer}^{\text{x}+1}_{3,4}$$

$$\text{buffer}^3_{3,4} := 0$$

Note that the buffer status is also updated by the reading operation such that the first slot always contains the oldest frame.

Then, we can apply parametric analysis on the model for $n = 3$ using IMITATOR. A projection of all possible values to parameter E2E gives the following result:

$$\text{E2E} \in [63, 225.016].$$

Finally, we conclude results for Challenge 1A in Table I.

Buffer34 size	min E2E	max E2E
$n = 1$	63 ms	145.008 ms
$n = 3$	63 ms	225.016 ms

TABLE I: E2E latency results for Challenge 1A

V. Solution to Challenge 1B

Due to the page limit, please refer to our solutions [ALS15] to the WATERS 2015 industrial challenge for more details.

VI. Solution to Challenge 2

A. Schedulability analysis

For Challenge 2A, when it goes to maximum end-to-end latency, we could employ the critical instant property [LL73], [LSAF14] to compute it. That is, the maximum latency happens such that

- T5, T6 and T7 always execute by their worst-case.
- T6 starts to execute coincidently with a release of T2.

We assume the period of T2 is exactly equal to $\mathrm{P}_2 = 40$ms; Then, we obtain the maximum latency 74ms for $J6 = 0$ and $74 + 20 = 94$ms for $J6 = 20$ms. Notice that they are both inferior to P_6, so there is no possibility that a new instance of T6 starts before the last instance of T7 completes. Possible variations in the period of T2 do not have any impact on the estimated end-to-end latency.

As for the minimum latency, a first intuition is that T5, T6 and T7 should execute by their best-case, so we compute $4 + 4 + 9 + 4 + 11 = 32$ms.

- If the initial offset between T2 and T6 is larger than 32ms, then the minimum latency is indeed 32ms.
- Otherwise, T2 preempts the execution of T5-T7 at least once, and the minimum latency would be $32+17 = 49$ms.

Therefore, we conclude that the latency of the chain T5-T7 is within $[49, 94]$.

In Challenge 2B, a mutually exclusive resource is shared between T5 and T2. In this case the minimum and maximum latencies do not change, whereas task T2 could suffer for a delay of 2 in accessing the shared resource, so its response time will vary in $[17, 19]$.

If we have the freedom to manipulate priorities we can further reduce the end-to-end latency. T2 is part of the chain for video frame processing, and to avoid interfering the verification result in Challenge 1, we still assume that T2 has higher priority than T5, T6 and T7. We can re-arrange the priorities among T5, T6 and T7 as T5 $>$ T7 $>$ T6, and the worst-case latency becomes 69ms for $J6 = 0$ and 89ms for $J6 = 20$ms. It is not difficult to see that this is the optimum priority assignment by enumeration.

B. PTA model for Challenge 2

We modeled the system following the scheduling framework for PTA proposed in [LSAF14]. We do not report here the full model for lack of space, it can be found at [XP]. We just summarize the main points.

- The scheduler is modeled as a separate automaton, generated automatically from the priority ordering, and it interacts with the task automata using synchronization.
- Initial task offsets are modeled as non-deterministic choices: the period of T6 is $p_6 \in [0, \mathrm{P}_6 - 32]$, whereas the period of T2 is $p_2 \in [0, \mathrm{P}_2 - 17]$.
- An observer automaton measures the end-to-end latency with a clock. Maximum and minimum latencies are modeled as parameters, exactly in the same way as the model developed for Challenge 1A.

In the end, we report the results for Challenge 2 in Table II. As we discussed, the existence of a shared resource between T5 and T6 does not affect the latency for the chain T5-T7.

	min latency	max latency
$J6 = 0$ ms	49 ms	74 ms
$J6 = 20$ ms	49 ms	94 ms

(a) The min/max latency

T2>T5>T7>T6	
	max latency
$J6 = 0$ ms	69 ms
$J6 = 20$ ms	89 ms

(b) The optimal priority assignment

TABLE II: Results for Challenge 2

C. The weakly hard analysis for Challenge 2

This part contains the improvement in our new solutions with respect to [ALS15]. Regarding the optimal priority assignment, the scheme in Section VI-A has to maintain that task T2 is assigned the highest priority, otherwise, T2 cannot be guaranteed to complete the execution before its next activation, which indicates *the violation of the (implicit) temporal constraint*. Different from the previous solution, this time we respect the optimal priority assignment scheme that minimizes the end-to-end latency for Challenge 2: T5 $>$ T7 $>$ T6 $>$ T2. Instead, we are going to formally analyze how often T2 violates its temporal constraint, and this is the so called *weakly hard analysis*.

The weakly hard analysis of a task is also named m-K analysis, which in principle aims to check if there will be more than m temporal violations out of arbitrarily K consecutive activations of a task.

Three kinds of the weakly hard analysis exist. [BBL01] is the first work that studies the m-K model, however, it relies on the explicit system initial status (*i.e.,* the first release time of every task is predefined) and the periodicity of a periodic task system for the weakly hard analysis. More recent works based on the *typical worst-case scenario* [QHE12] break the limitation on predefined initial task release times, and they are eligible for weakly hard analysis of systems in the case of the occasional sporadic workload.

Both the two approaches above do not fit into Challenge 2, as it is a periodic system with unknown task release times and there is no sporadic workload. Instead, we apply the tool

Weakly-hard Schedulability Analyzer (WSA) [WSA], which implements latest results for the m-K analysis. Different from other methods,

- WSA is able to conduct weakly hard analysis on general periodic systems, even when initial task release times are unknown;
- it supports tasks with release jitters;
- it supports the resource sharing, *e.g.,* the use of Priority Ceiling Protocol (PCP) [SRL90];
- its analysis is based on Mixed Integer Linear Programming, and is compatible with parametric task configurations (like the period of task T2).

Finally, for the weakly hard analysis of task T2, we abstract T5, T6 and T7 into a single task T6' that has period 100ms, best-case execution time 32ms, worst-case execution time 40ms and a jitter value = j (0ms or 20ms). T6' has a higher priority than T2. By applying WSA to T2, the weakly hard analysis results obtained are in Table III, where we list the maximum number (m^*) of temporal violations of T2, within an arbitrary sequence of K its activations. In this particular case, the jitter value does not change the weakly analysis result.

m^*	K
1	2
2	5
4	10
6	15

TABLE III: The weakly hard analysis results of Challenge 2

According to the results in Table III, there will be at most 2 temporal violations for T2 every 5 its activations and this figure is consistent with the result when $K = 10$ and 15. Also note that, in Table III, when $K = 2$, $m^* = 1$. This implies that there will be never two consecutive temporal violations that happen in a row.

VII. Conclusion

We proposed solutions, using formal methods based on PTA, to the consolidated version of the WATERS 2015 industrial challenge. Thanks to their expressiveness, PTA are especially convenient for modeling systems that contain some unknown but constant configurations. Challenges 1A and 1B were the most difficult, so we only provided upper bounds on results for them. The high complexity is mainly due to the large hyperperiods, therefore a complete model is intractable. We spent about one week for studying the problem and provide a model that could converge in a decent time.

IMITATOR is the software tool we rely on for performing parametric analysis. IMITATOR turned out to be particularly well-suited for the analysis of such systems with period constants but with some (unknown) imprecision.

Conversely, Challenge 2 is a scheduling problem, whose solution can be obtained quickly using schedulability analysis methods. The solution has been validated by a formal model based on PTA that took half a day to be built using the framework developed in [LSAF14], and only a few seconds to converge.

We further apply the weakly hard analysis tool WSA to evaluate the frequency of temporal violations that appear in Challenge 2, which helps elaborate more design choices for the system.

References

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFKS12] Étienne André, Laurent Fribourg, Ulrich Kühne, and Romain Soulat. IMITATOR 2.5: A tool for analyzing robustness in scheduling problems. In *FM*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer, 2012.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *STOC*, pages 592–601. ACM, 1993.

[ALS15] Étienne André, Giuseppe Lipari, and Youcheng Sun. Verification of two real-time systems using parametric timed automata. In *WATERS*, 2015.

[And16] Étienne André. What's decidable about parametric timed automata? In *FTSCS 2015*, volume 596 of *Communications in Computer and Information Science*, pages 52–68. Springer, 2016.

[AS13] Étienne André and Romain Soulat. *The Inverse Method.* FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc., 2013.

[BBL01] Guillem Bernat, Alan Burns, and Albert Liamosi. Weakly hard real-time systems. *Computers, IEEE Transactions on*, 50(4):308–321, 2001.

[BBLS15] Nikola Beneš, Peter Bezděk, Kim G. Larsen, and Jiří Srba. Language emptiness of continuous-time parametric timed automata. In *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*. Springer, July 2015. To appear.

[BL09] Laura Bozzelli and Salvatore La Torre. Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, 35(2):121–151, 2009.

[BO14] Daniel Bundala and Joël Ouaknine. Advances in parametric real-time reasoning. In *MFCS*, volume 8634 of *Lecture Notes in Computer Science*, pages 123–134. Springer, 2014.

[HRSV02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002.

[JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for timed automata. *IEEE Transactions on Software Engineering*, 41(5):445–461, 2015.

[LL73] C. L. Liu and James Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

[LSAF14] Giuseppe Lipari, Youcheng Sun, Étienne André, and Laurent Fribourg. Toward parametric timed interfaces for real-time components. In *SynCoP*, volume 145 of *Electronic Proceedings in Theoretical Computer Science*, pages 49–64, 2014.

[QHE12] Sophie Quinton, Matthias Hanke, and Rolf Ernst. Formal analysis of sporadic overload in real-time systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 515–520. IEEE, 2012.

[SLDP09] Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. PAT: Towards flexible verification under fairness. In *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 709–714. Springer, 2009.

[SRL90] Lui Sha, Ragunathan Rajkumar, and John P Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on computers*, 39(9):1175–1185, 1990.

[WSA] Weakly-hard Schedulability Analyzer (WSA). https://sites.google.com/site/theyoucheng/wsa.

[XP] Experiments. http://www.imitator.fr/static/FMTV15.