Parametric Analysis of Distributed Firm Real-Time Systems: A Case Study*

Thi Thieu Hoa Le, Luigi Palopoli, Roberto Passerone, Yusi Ramadian
DISI - University of Trento, Trento, Italy

Alessandro Cimatti
Fondazione Bruno Kessler (FBK), Trento, Italy

Abstract

A new generation of distributed real-time systems (DRTS) is based on heterogeneous models of computation and communication and is associated with flexible real-time constraints. Classical design flows based on real-time scheduling theory display important limitations related to the restrictive assumption on the system model. On the other hand, formal verification of timed automata is far more general, but it suffers a different limitation: it does not provide any guide on how to choose the design parameters, nor does it permit to gauge the robustness of the design against unknown parameters. In this paper, we advocate the use of formal verification of parametric timed automata as a means to combine the best of the two approaches. The feasibility of the idea is shown on a significant industrial case study.

1. Introduction

A distributed real-time system (DRTS) is a collection of applications that execute across different computing nodes and interact through a networked communication infrastructure. By and large, in a DRTS applications can be decomposed in a set of computing activities (tasks), and of communication activities. Each task is executed on a computing node, and it typically spawns a virtually infinite sequence of execution instances (jobs). There are two frequent ways for activating a job: by a timer (time-triggered activation) or by an event (event triggered activation). Time-triggered activations are typically used to implement feedback control loops, or to process multimedia data (e.g., in video encoder/decoder scheme). The event triggered activation can be associated with the presence of a new input or with a specific request from another task (*via* an appropriate system call). The communication takes place exchanging elements of information (packets), which are funneled through the network links.

The increasing complexity of the automated functionalities requested by a large class of modern industrial systems is propelling the development of DRTS to an unprecedented level. The future generation of DRTS is going to be heterogeneous in several respects. First, applications often require the co-existence of time-triggered components (e.g., to actuate a digital control loop) with event triggered components (e.g., to respond to events or mode change requests). Second, the strict compliance with every deadline is not always required, either because the system itself is soft real-time (e.g., a multimedia system) or because a moderate and controlled presence of timing failures is deemed acceptable in a control system if rewarded by a radical efficiency gain [7, 20]. Finally, the scheduling policy to manage shared computation and communication resources can be different for the different resources. For instance, computing nodes are more often than not scheduled preemptively, whereas network links are inherently non-preemptive. In the face of this complexity, classic design paradigms (based on very stringent assumptions) are doomed to a quick obsolescence. Still, there is an urging need for a design procedure guiding the choice of the free design parameters and enabling, at the same time, the assessment of the robustness margins for a candidate design.

Paper contribution. In our previous work [8], we have laid the foundation for a novel methodology. We considered a set of real-time tasks sharing a single CPU through a fixed priority preemptive scheduler. Moving in the track opened by Wang Yi and co-workers [11], we modelled task activations and the scheduler by means of parametric timed automata. The subsequent use of model checking and sensitivity analysis enabled the construction of the subset of the parameters' space that correspond to feasible hard real-time schedules. Our goal was to combine the generality offered by the application of timed automata with the exploration of the parameter space offered by the standard real-time scheduling analysis.

In this paper, we take a step further in the direction of proving the generality of our approach. We considered an industrial case study, suggested by the avionic industry characterized by: 1) a complex network topology that interconnects a potentially large number of nodes, 2) the co-existence of preemptive and non preemptive scheduling algorithm (to manage different components of the system), 3) the presence of activities characterised by soft

---

real-time constraints. Our first contribution is to show a complete model of the system based on the use of timed automata. The model has been validated by using standard verification tools (the UPPAAL tool-suite) by grounding the parameters to fixed choices. The second contribution has been the derivation of the parametric feasibility region on a simplified yet meaningful subset of the system by adapting the methodology and the tool that we previously developed [8].

State of the art. Ever since the seminal work of Liu and Layland [17], the real-time scheduling theory has identified conditions for a set of real-time task to be schedulable (i.e., to execute within the timing constraints) on a single processor. Such conditions are defined on the task activation periods and computation time, under specific choices for the scheduling algorithm. A particular attention has been placed on fixed priority preemptive schedulers, for which the response time analysis [13] and the time demand analysis [16] have proven effective tools to evaluate the schedulability of a task set on a single CPU. An interesting development of these techniques has been proposed by Bini et al. [5], who have applied the time demand analysis to assess the sensitivity of the schedulability result w.r.t. variations in the task parameters. A very interesting point of these techniques is that they are based on analytical tests and are therefore very efficient, to a point where they are often utilized *on-line* to decide or deny admission of a new task in a system. The downside is that they are usually conservative (they assume the worst case phasing for the task activation) and, more importantly, operate within a restricted assumption space: single processor, fixed priority preemptive scheduler, periodic activation and strict respect of every deadline (hard real-time hypotheses). In recent years, commendable extensions to the real-time scheduling theory have been made in the direction of multiprocessor systems [4, 3], of relaxed timing constraints [1, 18] and of distributed real-time systems [12, 23, 9]. Despite the relaxation of some of the hypotheses of the classic real-time task model, the applicability for real-time scheduling theory remains restricted to the adoption of specific models of computation and of homogeneous scheduling algorithms.

A greater level of generality can be attributed to the application of formal methods to the verification of timed systems, pioneered by Alur et al. [2]. The adoption of the timed automata formalism enables the specification of a large class of real-time systems, underlied by heterogeneous models of computation and communication. Complex timing properties can then be verified in a reasonable time by the use of such optimized model checkers as UPPAAL [14]. This idea has been successfully applied to the verification of real-time schedulability for task sets assuming both non-preemptive [19] and preemptive [11] schedulers. Two important limitations of these approaches are: 1) they require a complete specification of the system, 2) they only return a positive/negative to the verification problem without any feedback to the designer on the

adjustment to make on the parameters or on the robustness margins. In other words, they do not deal properly with the design of parametric systems. On the other hand, complementing the application of formal methods with a binary search on the parameter space [22] can be hardly a scalable solution when the number of parameters is considerable and it assumes a "monotonic" relation between parameters and feasibility, which is not necessarily true in the general case.

2. A Case Study

We have taken as a case study a simplified version of a Heterogeneous Communication System (HCS), such as one that could be found on board of aircrafts [10]. The architecture of the system is shown in Figure 1. The
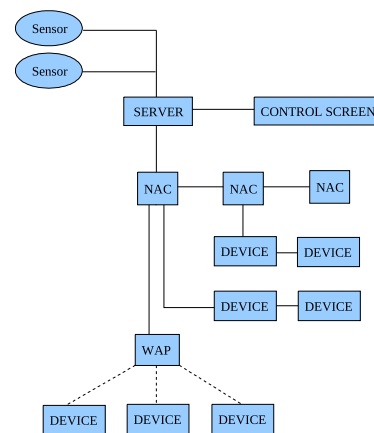

Figure 1. Heterogeneous Communication System (HCS)

HCS system contains a common server, wired and wireless communication networks and a number of devices. The components of the network communicate through Network Access Controllers (NAC), which perform gateway function and routing, and are connected in a daisy chain topology. In this case study we focus on audio devices, which are required to distribute music and audio announcements to the main cabin. The audio stream is transmitted by the server through the network. To avoid echo effects, the devices must reproduce the audio at synchronized instants. For this reason, the network, server and devices implement also the Precision Time Protocol (PTP) [21] to synchronize their clocks.

The communication between the server and device is asynchronous. The server sends an audio packet every $audPeriod$ ms; audio packets are characterized by two parameters: a sequence number i and a timestamp ti denoting the time the packet has to be played at the device. Due to varying network conditions, packets arrive at the device (except if they are lost) with a minimal latency $Lmin$ and a maximal latency $Lmax$. The NACs simply forward the incoming packets to the devices. Packets

passing through the NACs experience delay of $Lnac$ during which they are preprocessed by the NACs. The device processing time for each audio packet is τ, after which the device is ready to receive the next audio packet. The PTP protocol runs on the server, the devices and NACs, and is used to synchronize the respective clocks. Figure 2 depicts the message sequence of clock synchronization between a device (slave) and the master clock. Various tim-
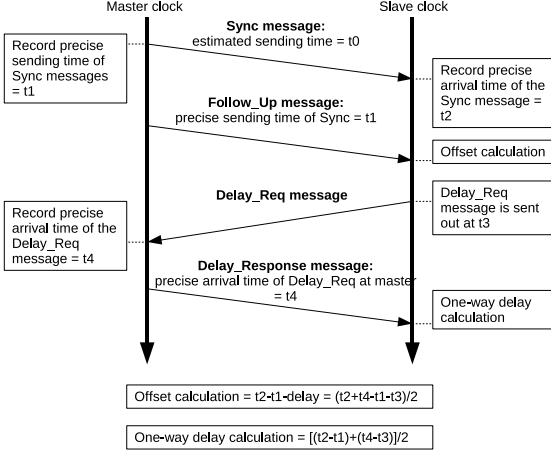

Figure 2. Time sequence diagram of message exchanged between master and slave to achieve clock synchronization

ing delays are to be guaranteed, for example, in a scenario where two devices are connected to the server, it should be guaranteed that both devices are synchronized within an error of 0.1 ms (synchronization precision).

Our objective is to identify the largest region of the parameter space in which the the correct functioning of audio streaming and clock synchronization can be guaranteed. To do so, we employ parametric timed automata. However, HCS is too complex to be parametrically-modeled completely. Therefore, some of the above requirements have to be relaxed before we attempt to parametrically model the system. The simplified system would contain only one server and many NACs and devices where one NAC may be associated to at most one device and one other NAC. Also, only the PTP parts on the server and devices are modeled. Additionally, every packet will have to experience the maximal delay $Lmax$ when traversing the medium. Furthermore, we only partially model the unreliability of the network. The system high-level description can be viewed logically as in Figure 3. Lastly, the transmission priority of PTP messages are assumed to be higher than that of audio packets, however, an ongoing transmission of an audio packet will not be preempted by a PTP message.

A complete set of models for this simplified system was developed in UPPAAL [14] as a network of 13 timed automata [15]. In UPPAAL, HCS is modeled as a network of extended timed automata with global real-valued
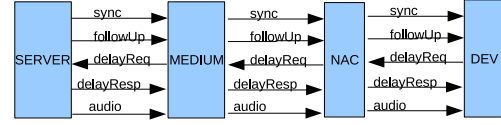

Figure 3. Logical model of HCS

clocks and integer variables. Clock value retrieval which is essential to the PTP protocol is not supported by UP-PAAL, hence the introduction of integer clocks. There are four error states in the automata network, corresponding to error conditions. One is reached when an audio packet arrives after its time-to-play. Three others are reached when buffer overruns occur in the audio buffer, the NAC input buffer and the NAC output buffer. To verify that the system is schedulable, we must show that these four error states are never reachable. Three test cases have been performed to test the schedulability of the system with different assignments to the parameters [15]. In the first, the system is guaranteed to be schedulable because all properties are satisfied while it is not in the second case due to the violation of the first property. The last test case points out the possibility of deadlock when the third property is not satisfied. Our objective in this paper is to replace this manual exploration of the parameter space with an automatic procedure.

3. Background on parametric verification of timed systems

In our previous work [8], we proposed a methodology for parametric analysis of real-time systems. The cornerstones of our construction are the notion of Parametric Timed Automata and an algorithm to infer the region of feasible parameters. For the sake of completeness, we report here a brief summary of that work, referring the interested reader to the cited paper for additional details.
Parametric Timed Automata Parametric Timed Automata (PTA) are an extension of the classical notion of timed automata [2], defined as a tuple $\langle L, L_0, \Sigma, X, P, \Gamma, I, E \rangle$, where
- L is a finite set of locations
- $L_0 \subseteq L$ is the set of initial locations
- Σ a finite set of labels
- X is a finite set of variables
- P is a finite set of parameters
- $\Gamma \subseteq \mathcal{B}(P)$ is the parameter space
- $I : L \rightarrow 2^{\mathcal{C}(X \cup P)}$ is the invariant map
- $E \subseteq L \times \Sigma \times \mathcal{C}(X \cup P) \times 2^{\mathcal{U}(X)} \times L$ is the set of switches.

The meaning of location, switch, clock is the same as in timed-automata. The PTA is characterized by two additional sets: the parameters P and the additional state variables X_s. The state variable of the system X are then the disjoint union of X_c (clocks) and X_s (state variables).

Parameters and state variables are defined as set of symbols with valuation over the rationals. In a transition

the value of the variables in X is updated according to the function $\lambda : X \times P \rightarrow X$. The value of clocks can either be updated on each transitions, or it can grow linearly in time in each location. The value of state variables can be changed only as a result of a reset action when a transition is taken and clocks may have a linear polynomial in the right hand side of the assignment. We can obtain standard timed automata if the set of parameters and of state variables are empty, and the update constraints have the form $x := 0$. If update constraints have the form $x := x - c$, with $c \in \mathbb{Q}$, we obtain the TA with subtraction [11]. PTAs can be composed using the standard notion of product between time automata [2].

Construction of the feasibility region Once a real-time system is modelled as a network of PTA, the violation of a timing constraint can be associated with an error location. The problem of identifying the feasibility regions is then formulated as finding the assignments of parameters that make the error location reachable.

The basic idea can be described as follows. A model checker identifies a counter-example, i.e., an execution trace that terminates into the error state for a given assignment of parameters. A sensitivity analysis is then carried out that identifies the subregion of the parameters validating the trace. This subregion is given by a conjunction of linear constraint on the parameters and is subtracted from the search space. The procedure is iterated until no error trace can be identified. The union of all the subregions found in this way identifies the set of parameters associated to unfeasible schedules (the feasible region is obviously found by complementation).

Algorithm 1 Iterative algorithm for PTA schedulability region analysis

---

Require: PTA describing activations and scheduling of n tasks
Ensure: Schedulability Region
1: **for** $i = 1$ to n **do**
2:    PTA.init(ParamSchedProblemForTask(i))
3:    j = 0
4:    **while** PTA.reachable(Error) **do**
5:       trace = PTA.get_trace()
6:       Unfeasible[j] = PTA.get_parameter(trace)
7:       PTA.add_constraints(negate(Unfeasible[j]))
8:       j++
9:    **end while**
10:    Feasible[i] = not(big_or(0, j, Unfeasible))
11: **end for**
12: **Return** big_and(0, n, Feasible)

The algorithm implementing this idea is shown in Algorithm 1 and it relies on the symbolic approach for representing and model-checking timed automata. A PTA is described by a conjunction of formulas in the theory of rationals, representing transitions, guards, invariants and reset maps. The use of a model checker allows symbolic processing of the model to identify all the possible traces,

and particularly the ones that end in the error state. This step can be carried out by any off-the-shelves symbolic model checkers. In our case, we have used NuSMT [6], a system for the verification of symbolically described infinite state systems. In particular, we used bounded model checking to carry out reachability analysis. With this approach, a trace is a sequence of truth assignment for binary variables (associated to the transitions) and of linear constraints on the real variables representing the time elapsed in each state. By uniting this information with the symbolic model, we are able to construct a set of linear constraints on the parameters and on the state variables that validate the trace. The subregion of interest is then found by a simple projection of this set on the parameter space. To identify a termination criterion (i.e., a situation in which no new error trace can be found), we have to complement bounded model checking with other techniques such as k-induction.

4. Parametric Modeling

The system described in Section 2 has been modelled in full detail and analyzed, for ground parameters, using UPPAAL [15]. The next step was to carry out the parametric verification described in Section 3. Because of its considerable complexity, we have worked out an abstraction of the system to limit the state space and to concentrate in isolation on each outstanding issue (the non preemptive scheduler, the different criticality of the timing constraints and so on). The parametric analysis of the complete system is reserved for future work. Our strategy is to first analyze the timing constraints of the audio stream and PTP, under certain assumptions on the PTP synchronization accuracy, and determine the acceptable clock drift relative to other parameters. In a second step, which is part of our future work, we will analyze the PTP operation to ensure that the synchronization accuracy meets the constraints defined in the first phase.

4.1. Abstract Models

We divide the abstract model of the system into two parts. The first corresponds the release of the packets on the network according to a periodic pattern. The second models the network and device, including the scheduling policy and the real-time constraints. For the latter, we have considered both the classic hard real-time constraints and firm real-time constraints.

We model the release of packets as activation automata, shown in Figure 4. Each stream of packets is characterized by the offset for the first release (transition from initial state to the second state), and is then periodic afterwards (self transition on the second state). A release signal is emitted every time a transition is taken, and is used to synchronize the automaton with the rest of the system. In the following, for consistency with our previous work, we will refer to these automata also as "tasks".

The remaining part of the system is modeled as a set

Figure 4 (a):

c==ptpPeriod
Release_PTP!
c=0

c==ptpOff
Release_PTP!
c=0

Wait_for_offset
c<=ptpOff

Wait_for_period
c<=ptpPeriod

(a)

Figure 4 (b):

c==audPeriod
Release_Audio!
c=0

c==audOff
Release_Audio!
c=0

Wait_for_offset
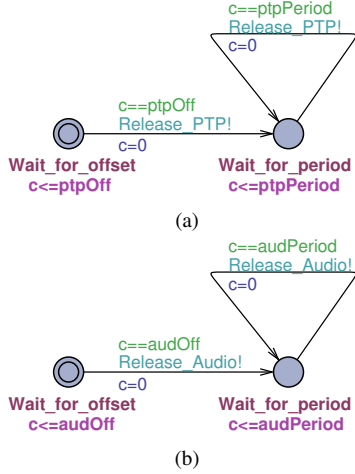c<=audOff

Wait_for_period
c<=audPeriod

(b)

Figure 4. PTP and audio task activation automata

of schedulability checkers [8]. Unlike our previous work, the checkers for the two tasks are modified to model a *non-preemptive* scheduling algorithm, i.e., a transmission will not be interrupted if it has already started. The scheduler is also *prioritized*, so that when there is no ongoing transmission and many packets are ready, the PTP packets go first and the audio packets back off.

The scheduler checker for PTP packets is shown in Figure 5. D_1 is the deadline of PTP packets, which is less
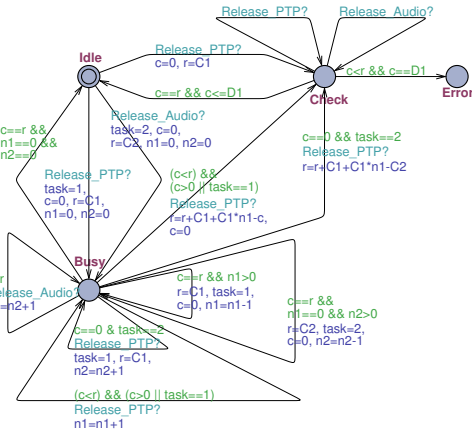
Figure 5:

Release_PTP?   Release_Audio?

Idle

Release_PTP?
c=0, r=C1

c==r && c<=D1

c<r && c==D1   Check   Error

c==r &&
n1==0 &&
n2==0

Release_Audio?
task=2, c=0,
r=C2, n1=0, n2=0

c==0 && task==2
Release_PTP?
r=r+C1+C1*n1-C2

Release_PTP?
task=1,
c=0, r=C1,
n1=0, n2=0

(c<r) &&
(c>0 || task==1)
Release_PTP?
r=r+C1+C1*n1-c,
c=0

Busy

c<r
Release_Audio?
n2=n2+1

c==0 & task==2
Release_PTP?
task=1, r=C1,
n2=n2+1

c==r && n1>0
r=C1, task=1,
c=0, n1=n1-1

c==r &&
n1==0 && n2>0
r=C2, task=2,
c=0, n2=n2-1

(c<r) && (c>0 || task==1)
Release_PTP?
n1=n1+1

Figure 5. Schedulability checker for PTP packets

than or equal to the PTP period, while C_1 and C_2 are the transmission time of PTP and audio packets, respectively. The PTP task has a higher priority than the Audio task as specified in Section 2. In addition, the execution time of the former accounts for the total PTP load that the devices could bear and that of the latter accounts for the total delay of traversing through the medium and the NACs of audio packets. Furthermore, we introduce five additional variables. The variable $task$ denotes the currently-executed

task (i.e., the current on-going transmission), n_1 and n_2 record the number of PTP and audio packets released during the current execution, c is a clock accumulating the time since the task queues were last idle and r is a data variable used to sum up the time needed to complete all tasks released since the checker was last idle. The transitions of the checker are intuitively interpreted as follows:

- The transitions to Idle are taken when the task instance being checked in Check or a sequence of tasks arrived in Busy, has finished execution.

- The transitions to Busy are taken when an instance of task PTP or Audio is released. Self-loops are taken to queue the newly-released instances and to retrieve them when the current execution has finished.

- The transitions to Check are taken when a PTP instance is (non-deterministically) chosen for checking. Before verifying the deadline, the execution time of all other PTP instances in the queue must be taken into account as they would be scheduled before the current instance, that is r should be updated to $r + C_1 + C_1 * n_1 - c$, or $r + C_1 + C_1 * n_1 - C_2$. New PTP and Audio instances in Check are ignored as they are already considered in location Busy.

- The transition to Error is taken when the currently-executed instance misses its deadline.

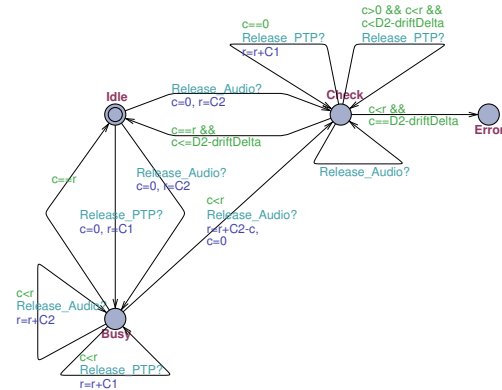The scheduler checker for audio packets is shown in Figure 6. The Audio checker is similar to but simpler than

Figure 6:

c>0 && c<r &&
c<D2-driftDelta
Release_PTP?

c==0
Release_PTP?
r=r+C1

Idle

Release_Audio?
c=0, r=C2

c==r &&
c<=D2-driftDelta

c<r &&
c==D2-driftDelta   Check   Error

c==r

Release_Audio?
c=0, r=C2

Release_PTP?
c=0, r=C1

c<r
Release_Audio?
r=r+C2-c,
c=0

Release_Audio?

c<r
Release_Audio?
r=r+C2

Busy

c<r
Release_PTP?
r=r+C1

Figure 6. Schedulability checker for audio packets (hard deadline)

the PTP checker because task audio has a lower priority. D_2 is the relative deadline of audio packets and Δ is introduced to account for the offset time of the local clock compared to the server clock. The worst case happens when the local clock is substantially slower than the server clock and thus when an audio packet is received, the actual deadline to be verified would be $D_2 - \Delta$ instead of D_2.

In fact, the requirement of no deadline miss (hard deadline) is difficult to obtain in real-time environments.

Therefore, in order to make the analysis more practical, the requirement is relaxed by allowing an audio packet to sometimes miss its deadline (*firm real-time constraint*). A firm real-time constraint is given by a deadline and by a couple (m, n) meaning that m deadlines can be missed every n jobs. [18]. In our case study, a packet may miss its deadline as long as the previous packet has not already missed the deadline ($m = 1, n = 2$). The checker adapted for this new requirement is shown in Figure 7. We intro-
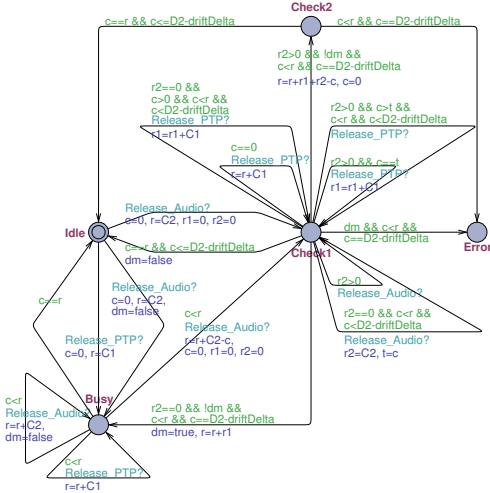
	Experiment 1	Experiment 2
$ptpOff$	0	5
$ptpPeriod$	40	40
D_1	10	10
$audOff$	0	0
$audPeriod$	10	10
D_2	10	10

Table 1. Fixed parameter values in the two experiments

	Experiment 1	Experiment 2
$Checker_{PTP}$	726	1105
$Checker_{Audio}$	8	12

Table 2. Running time in minutes in two experiments

Figure 7. Schedulability checker for audio packets (firm deadline)

duce four new variables: dm is a boolean variable used to capture the fact that one deadline miss has already happened ($dm = true$), r_1 is a real variable used to record the total execution time of all PTP instances released after the currently-checked instance and before a deadline miss or the next audio arrival, r_2 marks the next audio arrival whose time is marked in t.

Transitions entering Check1 from Idle or Busy are taken when an audio instance is (non-deterministically) chosen for checking. If another PTP instance is already released, the PTP will be executed first ($r = r + C_1$). The self-loops in Check1 are taken to accumulate the execution time of all PTP instances released afterwards until another audio instance is released or the current Audio instance misses its deadline. If one deadline miss had happened before ($dm = true$), the location *Error* is reached because of two successive deadline misses. Else, if another audio instance has already been released ($r_2 > 0$), the transition from Check1 to Check2 is taken in order to verify if the next deadline is missed again. Otherwise, the variable dm is updated to $true$ and the transition from Check1 to Busy is taken to tolerate the first deadline miss.

5. Parametric Analysis

We have performed several experiments to compute the feasibility region of the system under a diverse set of pa-

rameters. We present here the results of two such experiments, which differ in the amount of offset by which packets are issued to the network. We have chosen as free parameters the transmission times C_1 and C_2 and the desired accuracy Δ (or drift) of the clocks ($\Delta = 0$ corresponds to infinite accuracy), given by the application of the PTP protocol. The values of the fixed parameters for each of the experiments are shown in Table 1. The free parameters, on the other hand, are constrained to vary in the following intervals:

$$0 < C_1 \le D_1, \qquad 0 < C_2 \le D_2$$
$$0 \le \Delta$$

For reference, the running time results of the two experiments are summarized in Table 2. The computer used in the experiments has 3481MiB RAM and AMD Athlon 64 Dual Core Processor 5000+. The PTP checker generally runs much slower than the Audio checker which may be because the path leading to the error state of the former is generally longer than that of the latter. The running time also depends on the bound used to model check the system. Using a large bound can help find more traces to the error state, hence the feasibility region is more correct. However, the larger the bound, the longer the running time. So in finding the schedulable region, one must trade off between the number of bounds and computation time.

5.1. Audio feasibility and error regions

We perform experiments on both Audio hard-deadline and firm-deadline checkers. Figure 8 and 9 graphically show the Audio feasibility (not shaded) and error (shaded) regions for $\Delta = 0, 3, 5, 7$ in two experiments. The feasibility regions are expressed by set of constraints, for example, with $\Delta = 5$ the feasibility region for the Audio firm-deadline checker in Experiment 1 is expressed by a conjunction of three constraints as shown below. A first evident result is that the relaxation of timing constraints produces a larger feasibility region. This rather intuitive

1: $\neg[(20 < C_1 + 2C_2) \wedge (0 < C_1 \le 10) \wedge (0 < C_2 \le 10)] \wedge$
2: $\neg[(15 < C_1 + 2C_2) \wedge (10 < C_1 + C_2) \wedge (0 < C_1 \le 10) \wedge (0 < C_2 \le 10)] \wedge$
3: $\neg[(C_1 + C_2 \le 10) \wedge (0 < C_1 \le 10) \wedge (5 < C_2 \le 10)]$

fact can be exactly quantified and even pictorially represented thanks to our methodology. The second insight offered by this analysis is to quantify the impact of the synchronization accuracy on the correctness of the system. Indeed, the feasibility region shrinks as Δ increases and the temporal behaviour of the system can easily be jeopardized. This result can be used as a specification for the PTP. Finally, our analysis can be used to guide the choice of activation offsets, an important degree of freedom to simplify schedulability of highly loaded systems.
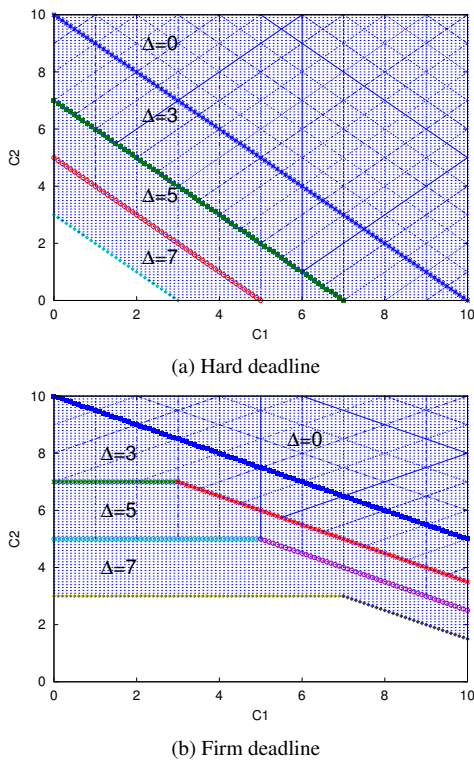

(a) Hard deadline

(b) Firm deadline

Figure 8. Audio feasibility and error regions for $\Delta = 0, 3, 5, 7$ in Experiment 1

5.2. PTP feasibility and error regions

The PTP feasibility region in the two experiments is also expressed in terms of sets of constraints. Figure 10 graphically shows the PTP feasibility (not shaded) and error regions (shaded) in the two experiments. By joining the PTP and Audio feasibility regions together, we can obtain the final region in which the whole system is guaranteed to work properly. For example, the feasibility region for the whole system in Experiment 1 is actually that for task Audio. If the device clock does not drift ($\Delta = 0$), point ($C_1 = 10, C_2 = 5$) is a schedulable point because

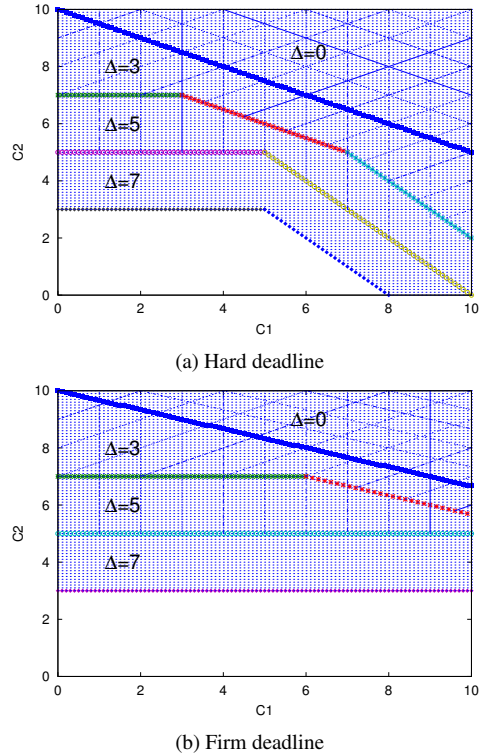(a) Hard deadline

(b) Firm deadline

Figure 9. Audio feasibility and error regions for $\Delta = 0, 3, 5, 7$ in Experiment 2

the first PTP released at time 0 does not miss its deadline ($C_1 = D_1 = 10$), then the first Audio deadline miss happens ($C_1 + C_2 > D_2$) but the second Audio deadline is respected ($C_1 + 2 * C_2 = 2 * D_2$). Other Audio instances released at time 20 and 30 are not preempted, thus able to meet their deadlines. At time 40, the task arrival pattern is repeated exactly the same as time 0. However, point ($C_1 = 10, C_2 = 5$) is no longer a feasible point when $\Delta = 5$. Because the first Audio instance misses its deadline as it does when $\Delta = 0$ and so does the second Audio instance. So, point ($C_1 = 10, C_2 = 5$) should be in the error region which is verified easily by looking at Figure 8(b).

6. Conclusions

We have shown how to apply parametric analysis techniques [8] to a distributed Heterogeneous Communication System (HCS). In this work we have extended this technique to account for non-preemptive scheduling and soft real-time constraints. In addition, we have shown how to use this technique in the context of a network, rather than the more traditional processor scheduling problem. Starting from the full specification, we have derived a simplified model which still exhibits the required characteristics. The analysis identifies non-trivial feasible regions for various values of the desired synchronization accuracy.

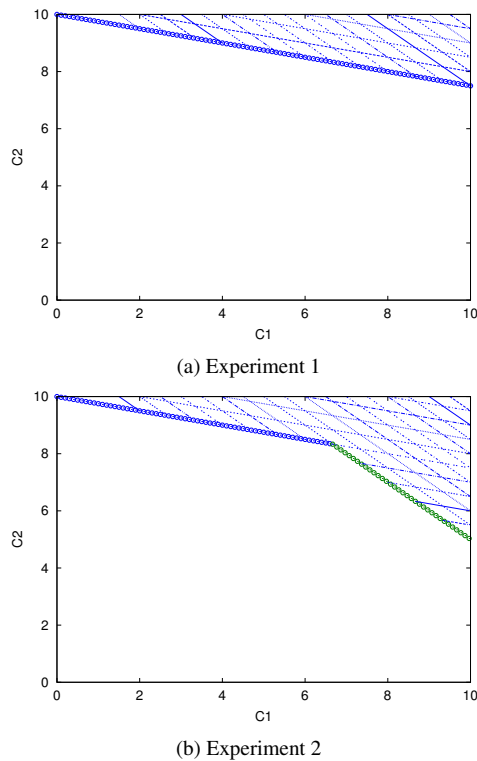Our future work includes a more complete paramet-

(a) Experiment 1

(b) Experiment 2

Figure 10. The PTP feasibility and error regions in two experiments

ric analysis of the system. To this end, we are working towards improving the runtime efficiency of the tool by employing explicit state techniques. Preliminary results show savings in the range of two orders of magnitude, when the parameter space is randomly explored, and then generalized, using UPPAAL and symbolic computation.

References

[1] L. Abeni and G. Buttazzo. Qos guarantee using probabilistic deadlines. In *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, pages 242 –249, 1999.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] S. Baruah. Techniques for multiprocessor global schedulability analysis. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 119 –128, dec. 2007.

[4] M. Bertogna, M. Cirinei, and G. Lipari. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *Parallel and Distributed Systems, IEEE Transactions on*, 20(4):553 –566, april 2009.

[5] E. Bini and G. Buttazzo. Schedulability analysis of periodic fixed priority systems. *Computers, IEEE Transactions on*, 53(11):1462 – 1473, nov. 2004.

[6] R. Cavada, A. Cimatti, A. Franzén, K. Kalyanasundaram, M. Roveri, and R. K. Shyamasundar. Computing Predicate Abstractions by Integrating BDDs and SMT Solvers. In *FMCAD'07*, pages 69–76, Washington, DC, USA, 2007. IEEE Computer Society.

[7] T. Chantem, X. S. Hu, and M. Lemmon. Generalized elastic scheduling for real-time tasks. *Computers, IEEE Transactions on*, 58(4):480 –495, april 2009.

[8] A. Cimatti, L. Palopoli, and Y. Ramadian. Symbolic computation of schedulability regions using parametric timed automata. In *Real-Time Systems Symposium, 2008*, pages 80–89, Dec. 3 2008.

[9] T. Cucinotta and L. Palopoli. Qos control for pipelines of tasks using multiple resources. *Computers, IEEE Transactions on*, 59(3):416 –430, march 2010.

[10] EADS Innovation Works. Case study on distributed heterogeneous communication systems (HCS), 2009, `http://www.combest.eu/home/?link=Application1`.

[11] E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. Schedulability analysis using two clocks. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 224–239. Springer, 2003.

[12] R. Gerber, S. Hong, and M. Saksena. Guaranteeing end-to-end timing constraints by calibrating intermediate processes. In *Real-Time Systems Symposium, 1994., Proceedings.*, pages 192 –203, dec 1994.

[13] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390, 1986.

[14] K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):134–152, 1997.

[15] H. T. T. Le, L. Palopoli, R. Passerone, and Y. Ramadian. Modeling a distributed heterogeneous communication system using parametric timed automata. Technical Report DISI-10-031, Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, April 2010.

[16] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Real Time Systems Symposium, 1989., Proceedings.*, pages 166 –171, dec 1989.

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

[18] A. Marchand and M. Silly-Chetto. Qos scheduling components based on firm real-time requirements. In *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*, page 141, 2005.

[19] C. Norstrom, A. Wall, and W. Yi. Timed automata as task models for event-driven systems. *Real-Time Computing Systems and Applications, International Workshop on*, 0:182, 1999.

[20] L. Palopoli, L. Abeni, G. Buttazzo, F. Conticelli, and M. Di Natale. Real-time control system analysis: an integrated approach. In *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pages 131 –140, 2000.

[21] A precision clock synchronization protocol for networked measurement and control systems. Ieee standard 1588-2002, November 2002.

[22] R. Racu, M. Jersak, and R. Ernst. Applying sensitivity analysis in real-time distributed systems. In *RTAS '05: Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium*, pages 160–169, Washington, DC, USA, 2005. IEEE Computer Society.

[23] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40(2-3):117 – 134, 1994. Parallel Processing in Embedded Real-time Systems.