Combining computational and analytic model descriptions for evaluating embedded real-time systems

Kai Lampka† and Lothar Thiele†

† {lampka,thiele}@tik.ee.ethz.ch
Computer Engineering and Communication Networks Lab.,
ETH Zurich, Switzerland

TIK-Report 296,
ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-296.pdf

Abstract

Industrial embedded real-time systems such as cabin communication systems (CCS) of modern (passenger) aircrafts [20], are highly complex. Verification of such complex systems is either hampered by the state space explosion problem, –attached to state-based verification,– or lacks a high-degree of detail, –due to the usage of some analytic method. For achieving a detailed analysis of system components, but still maintaining scalability of the procedure for computing overall properties of a system, our current research tackles the combination of executable and analytic model descriptions for the joint analysis of embedded real-time systems. As first step this report focuses on a combination of Timed Automata [2] with the MPA-method [8], where the latter is a relatively new compositional, analytical performance evaluation method based on the well-known Network Calculus [7]. As major outcome we present an efficient approach for converting analytic (sub-)model descriptions of the MPA-method into networks of interacting Timed Automata.

1 Introduction

1.1 Motivation

As our daily lives highly depend on complex hard- and software systems, it is crucial to assert that these systems behave as expected. Expected behavior does not only encompass qualitative aspects, functional correctness resp., but also quantitative aspects, such as performance.

One of the industrial case studies in the COMBEST project is a cabin communication systems (CCS) for a modern (passenger) aircraft. Its functional and non-functional complexity results from a huge number of time-constrainted functionalities to be implemented. Verifying the qualitative and quantitative correctness of such systems is not only imposed by legislation, but is also vital for the functionality of the overall system. For example, the specification provided by the industrial partners involves sensor components whose data need to be forwarded to the cockpit, such that the crew is informed about the status of the aircraft, whereas the same time the communication bus also routes audio and video data to each of the passenger's seats.

Parts of this COMBEST industrial case study may require now a high-degree of detail, when it comes to their modeling, whereas other parts can only be viewed as traffic generators, the complexity of which can be abstracted away. For exemplification one may think of the two servers of the COMBEST cabin communication systems (CCS) and of the fire-alarm sensors monitoring the lavatories. The server obviously must fulfil a huge set of functionalities, such as checking a database for new sensor data, sending the audio streams to the respective subnets, just to name a few.

Contrary to this, the functionality of a particular sensor may not be of interest when it comes to the evaluation of the COMBEST CCS system. It might be possible to abstract over its implementation detail, the only thing which might matter is the fact, that each sensor produces a specific amount of data within given time periods, where each data packet routed to the server and triggering some action there. On the other hand specific requirements such as message latencies and communication buffer over-/under-flows must be guaranteed for fulfilling the technical specification. In summary, such scenarios, where specific requirements need to be verified seems highly suited to be evaluated by a set of methods, each introducing its own degree of detail.

Timed automata [2] are well accepted for analyzing systems with time constraints, let it be for the verification of simple properties as deadlock

freeness or for verifying more complex progress properties specified by some LTL- or CTL formulae,– a comprehensive introduction to model checking can be found in [3].

However, it is well known, that state-based methods suffer from the so called state space explosion problem, e.g. the region graphs underlying a network of timed automata grows exponential in the number of regions with respect to the number of employed clocks and constants the clocks are compared with. The state space explosion problem makes a detailed analysis of systems in practice often infeasible if not impossible at all.

Contrary to this analytical methods often better scale with the size of the system model to be analyzed. But this advantage has its flaw: the proposed methods are limited to the computation of specific system measures, where verification of arbitrary system properties is not possible.

1.2 Contribution

This COMBEST project report introduces an heterogeneous approach for the evaluation of complex embedded real-time systems such as the COMBEST Cabin Communication System (CCS) illustrated above.

At its core the new approach combines analytical methods as known from real-time scheduling analysis with timed automata for encompassing complexity of nowadays embedded systems. Thus when detailed qualitative and quantitative properties must be verified and/or the component's configuration forbids an analytic evaluation or yields at least very rough bounds only, the approach supports a detailed state-based modeling. In other cases, where one may abstract over state-based and detailed system behaviors, the proposed approach applies the analytical method as introduced in [8].

Thus state graph generation and explosion is limited to some dedicated components, where the overall system is treated according to an analytical method for maintaining scalability of the approach. Finally, the approach is applied to a part of the COMBEST CCS system.

1.3 Organization

In Sec. 2 we present related work and contemporary approaches covering similar ideas. Sec. 3 provides details on the Real-Time Calculus (RTC) and timed automata (TA), thus clarifying the theoretical foundation of the ideas developed here. Sec. 4 will introduce our new ideas of coupling RTC- and TA-based models within a unique framework. Sec. 5 concludes the project report, where also the next steps are indicated.

2 Related work

Real-time Calculus (RTC) The analysis method developed in [8] is based on the network calculus and it concentrates on the evaluation of best - and worst-case system behavior. In order to deal with highly complex systems, the method is organized in a modular fashion: its basic building blocks are components or nodes, which describe the interaction between tasks and resources. Tasks are triggered by event arrivals. The timing properties of streams of events are characterized by (event-) arrival curves $\alpha(\Delta)$. In case of a greedy processing component tasks proceed with event processing, as soon as there are sufficiently enough consumable resources available. Resource availability is modeled by streams of abstract resource units, where mainly each resource is characterized by its own resource or service curve $\beta(\Delta)$. Each component of a system model takes arrival and service curves as input and allows the computation of streams of outgoing events and remaining resource or service units. The timing properties of these streams are once again captured by the respective curves, where in case of an outgoing event stream one speaks of the outgoing arrival curve $\tilde{\alpha}(\Delta)$ and where in case of an outgoing (or remaining) stream of resources one speaks of the outgoing service curve $\tilde{\beta}(\Delta)$.S Similar to the network calculus [7] the RTC employs arrival and service functions which are defined on time intervals Δ, rather than on the (real-)time domain. These functions either characterize best-case or worst-case behavior of the associated arrival - and service streams. In total arrival - and service streams are bounded in such a way, that the resp. arrival and service functions describe upper and lower bounds on the amount of incoming/outgoing events or available/remaining resource units in any time interval of length Δ. The corresponding arrival and service curves are denoted as **u**pper and **l**ower arrival curves $\alpha^{\{u,l\}}(\Delta)$, **u**pper and **l**ower service curves $\beta^{\{u,l\}}(\Delta)$ and the functions $\tilde{\alpha}^{\{u,l\}}(\Delta)$ and $\tilde{\beta}^{\{u,l\}}(\Delta)$ addressing the streams of outgoing events or resource units.
Different tasks may require access to the same resource, s.t. a resource sharing policy is applicable. Each scheduling strategy may define the rules by which the tasks share the available resource. The computation of outgoing arrival- and service curves must than follow the respective scheduling strategy, i.e. when transforming incoming arrival- and service functions into their outgoing counterparts each component must obey the applicable resource sharing strategy.
Besides this, the order of computation also depends on the layout of the communication infrastructure, i.e. on the paths the events are transmitted over, which may impose more conditions on the RTC-operators to be used,

e.g. cycles are resolved by fixed point computations with respect to a set of affected components and a valid starting point. As drawback one may therefore point-out, that

(1) new types of system configurations need in principle the development of new closed-form solutions for obtaining tight best-/worst case bounds on the system properties to be evaluated.

(2) The method is not easily applicable to systems, where state-dependent behavior is essential. For exemplification one may think of systems with state-dependent scheduling strategies, systems where the clock speed depends on the workload or specific cyclic dependencies among the components, to name only few of them.

(3) Best-/worst-case system properties to be evaluated are limited to those measures, which can be derived form incoming/outgoing upper and lower arrival, service curves respectively. Examples of such measures are buffer sizes (backlogs), event delays and resources utilizations.

Given these draw-backs, there is a major benefit of the evaluation method as illustrated above: A compositional RTC-based method nicely scales well to large and distributed real-time systems. Contrary to the state-based methods it allows therefore the analysis of large systems as known from industrial practice.

Timed automata (TA) Since the original contribution of Alur and Dill [2], the area of timed automata (TA) was subject of intense research activities, where many fruitful results have been produced. In the following we limit the discussion to results which are of great value in particular for the research carried out here: In [10] it was shown, how TAs can be used for solving min/max delay problems in real-time systems. The authors of [1] introduced the concept of stop-watches. This allows to naturally model systems, where jobs can be preempted and may resume with their computation. The works [17] and [11] show how TA can be used for specifying periodic event streams with jitter and min. distance between events. The complexity of the resulting event generating automata depends on the type of the employed model, e.g. if period $>$ jitter, if min. distance $= 0$, etc.. Once the event generating automata were defined, the authors of [17] and [11] employed them within complex systems to be analyzed, where the former automata generated the input stimuli of the system.

Combined approaches How to derive event traces form event curves is introduced in [13]. This work discusses a two mode algorithm, which either follows the upper arrival curve of events $\alpha^u(\Delta)$ ("on" mode) or follows the lower arrival curve $\alpha^l(\Delta)$ ("off" mode). The main difficulty is that for any point in time one must decide if the absence or generation of an event interferes with the specified input curves. These checks require that the approach stores the history of the produced event stream and compute the corresponding event curves on-the-fly, which makes the approach computational expensive. However, the approach allows to generate stimuli from an analytic (input) model, triggering behavior of a state-based model, which is something we will tackle in this report as well. However, contrary to our approach [13] generates only a single traces of possible input stimuli, where the original analytic input function covers infinite many. In total the approach is therefore suited to be applied in the context of simulation models, the description of which can be as close to a desired implementation as wished.

In [18] a combination of state-based and analytical performance evaluation of real-time systems is presented. To do so, the authors bridge the gap between event count automata (ECA) [9] and the modular performance evaluation scheme [8] mentioned above. Within the ECA-formalism the user must specify the minimum and maximum number of event arrivals taking place in the different locations of the ECA, i.e. one specifies this min./max. number for each of the locations. A location is left by choosing (non-deterministically) from those outgoing transitions, the guards of which evaluate to true. In total the ECAs seem to be designed in particular for combining RTC and its event functions with state-based modelling. Therefore the expressive power of ECA remains unclear, since contrary to advanced state-based formalism like TA, ECA do not include local clocks for triggering or suppressing actions within the modelled system. The authors of [21] present an approach, where a system to be analyzed is mapped to a process network. The properties to be verified are defined by some high-level description technique, such as live sequence charts. The suggested procedure follows than four stages:

(1) Application of a scheduling analysis, which is based on the structure of the process-network, specified input functions and user-defined execution times. As result one obtains minimum and maximum response times [19].

(2) Periodic event stream models with min./max. period and a fixed jitter and the previously computed min./max. response times serve as parameter for pre-defined TA, where cross-product computation of these

automata yields the system model.

(3) The high-level descriptions of system properties to be checked are transformed into TA.

(4) Application of standard model checking procedures allows now to check, whether the system model fulfil the desired properties or not.

The major drawback of this approach can been seen in the fact that the system model to be checked is derived from pre-defined TA, solely parameterized by the intermediately computed results. The only connection between these automata and the user-defined process network is the connection structure, input functions of periodic traffic models with jitter and the computed response times. Thus functional, i.e. state-based behavior is not explicitly taken into account restricting the approach to the verification of pure performance measures only. This lets one wonder, why one should use a detailed state-based framework, the analysis of which is computational expensive, if one is only able to verify quantitative behavior.

The authors of [12] also addressed the topic of the here presented work, namely of a method for combining TA-based model description with the RTC. However their work will be discussed in detail in Sec. 5.1, so that we can point out major differences to our approach.

3 Background Theory

3.1 Analytic analysis of real-time system

In this section we will re-capitulate the basics of a general approach for analysing real-time systems [8] and give details about the scenarios to be dealt with.

3.1.1 The Real-time Calculus

Contrary to other existing techniques the approach is not stochastic, thus it delivers hard bounds on queue sizes, job delays and hardware utilization, which is essential when it comes to the analysis of time-constrained real-time systems. The method itself is based on network calculus [7], thus one needs to apply algebraic operators on input functions for evaluating specific system properties. Consequently its applicability is limited to system configurations

(1) where certain scheduling strategies apply, e.g. fixed priority, time division multiple access, FIFO or EDF scheduling and

(2) with fixed and limited communication structures, s.t.

  (2.a) the routing of events in the system is known on beforehand, –each layout requires its own computation of measures and s.t.

  (2.b) the system posses a specific structure, –until now one can only handle certain kinds of feedback systems.

Overall the system description is analytic and thus state-less, i.e. one solely provides analytic descriptions of resource consumption and workloads, besides the layout of the infrastructure and the employed scheduling policies. This restricts the expressiveness of the method, as outcome one solely obtains analytic descriptions of un-used resource capacities and outgoing event streams from which quantities such as end-to-end delays and buffer backlogs can be computed.

As its core the approach feeds an event-stream and a resource stream into a single node of the system. As outcome one obtains an outgoing event-stream and the stream of remaining resources. However, contrary to other techniques the streams are not defined on a straight time-line, but for time intervals of length Δ. Formally we define upper bounding function $\alpha^u(\Delta)$ and lower bounding function $\alpha^l(\Delta)$ for a(n) (event) counting function R as follows:

$$\alpha^l(t-s) \leq R(t) - R(s) \leq \alpha^u(t-s), \text{ for } 0 \leq s \leq t.$$

Consequently α^u and α^l give the maximal and minimal number of events arriving in any time interval of length Δ, where it is assumed that this number can be bounded by a right-continuous and sub-additive function:

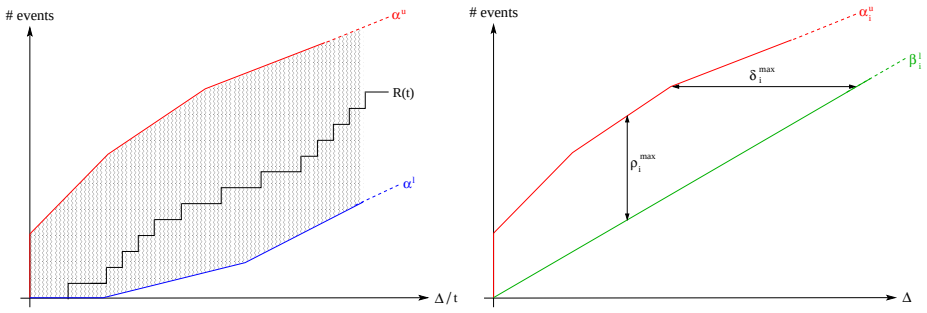
Definition 3.1: Super-additive and sub-additive functions

An upper arrival function α^u is denoted sub-additive $\iff$

$$\alpha^u(s) + \alpha^u(t) \geq \alpha^u(s+t) \text{ for } s, t \in \mathbb{R}_+$$

A lower arrival function α^l is denoted super-additive $\iff$

$$\alpha^l(s) + \alpha^l(t) \leq \alpha^l(s+t) \text{ for } s, t \in \mathbb{R}_+.$$



(a) Upper and lower arrival curves α^u and α^l and a bounded cumulative arrival function $R(t)$

(b) Performance measures of interest

Figure 1: RTC-compliant curves and related measures

In the following the will denote such functions as being *RTC-conformant*. For exemplification one may refer to Fig 1.a, which illustrates an upper and lower arrival curve α^u and α^l, as well as an example of a cumulative event count curve $R(t)$ bounded by them. In practice upper and lower (event) arrival functions can be obtained from (a) technical specifications, (b) traffic models such as periodic with jitter and min. distance (PJD-models), or (c) from practical measurements obtained from execution - / simulation traces. To obtain an upper or lower arrival curve $\alpha^{\{u,l\}}$ in these different cases one either simply specifies the minimal/maximal number of events to arrive in an interval Δ manually, by evaluating an analytic expression, i.e. analytically compute the number, or slide windows of size Δ over the real execution- or simulation-trace.

Analogous to the above description of workloads by means of upper and lower (event) arrival curves, one may also define an upper and lower bound for the availability of resources. Let $C(t)$ be the counting function of available resource units in the interval $[s, t)$, it follows:

$$\beta^l(t-s) \leq C(t) - C(s) \leq \beta^u(t-s), \text{ for } 0 \leq s \leq t.$$

For computing the bounding functions on outgoing events $\tilde{\alpha}^{\{u,l\}}$ and remaining resources $\tilde{\beta}^{\{u,l\}}$, via combining $\alpha^{\{u,l\}}$ with $\beta^{\{u,l\}}$, one must guarantee that the functions $\alpha^{\{u,l\}}$ and $\beta^{\{u,l\}}$ are defined on the same domain (Δ) and the same range ($\#events$). This can often be achieved by scaling $\beta^{\{u,l\}}$ appropriately or alternatively by scaling the in-going $\alpha^{\{u,l\}}$ and the outgoing $\tilde{\alpha}^{\{u,l\}}$ functions. In the above setting in-/outgoing event streams of a greedy

processing component are connected in the following way:

$$\begin{aligned}
\tilde{\alpha}^l(\Delta) &:= \min\left(\inf_{0\le\mu\le\Delta}\left\{\sup_{0<\lambda}\langle\alpha^l(\mu+\lambda)-\beta^u(\lambda)\rangle+\beta^l(\Delta-\mu)\right\},\beta^l(\Delta)\right)\\
\tilde{\alpha}^u(\Delta) &:= \min\left(\sup_{0<\lambda}\left\{\inf_{0\le\mu<\lambda+\Delta}\langle\alpha^u(\mu)+\beta^u(\lambda+\Delta-\mu)\rangle-\beta^l(\lambda)\right\},\beta^u(\Delta)\right)
\end{aligned}$$

Analogously the function of the remaining service units for the time intervals of length Δ can be computed by evaluating the expressions:

$$\begin{aligned}
\tilde{\beta}^l(\Delta) &:= \sup_{0\le\lambda\le\Delta}\langle\beta^l(\lambda)-\alpha^u(\lambda)\rangle\\
\tilde{\beta}^u(\Delta) &:= \min\left(\inf_{\Delta<\lambda}\langle\beta^u(\lambda)-\alpha^l(\lambda)\rangle,0\right)
\end{aligned}$$

As performance results one may compute now the experienced delay δ_i as well as the max. queue size ϱ_i experienced at node i, which can be obtained as the vertical, resp. horizontal maximal distance between upper arrival and lower service curve (cf. Fig. 1.b):

- experienced delay δ_i for an event at node i:

$$\delta_i \le \sup_{0\le t}\left\{\inf\langle\tau\ge 0:\alpha_i^u(t)\le\beta_i^l(t+\tau)\rangle\right\},$$

- the max. queue size ϱ_i at node i:

$$\varrho_i \le \sup_{0\le t}(\alpha_i^u(t)-\beta_i^l(t)) \text{ and}$$

- the utilization ρ_i of node i:

$$\rho_i := \lim_{\Delta\to\infty}\frac{\beta_i^u(\Delta)-\tilde{\beta}_i^l(\Delta)}{\beta_i^u(\Delta)}.$$

For a given system description consisting of tasks, hardware units, a fixed communication structure among the tasks and a mapping of tasks to hardware units the above formulae can be exploited for determining the end-to-end system properties. In case tasks compete over a hardware unit, a fixed scheduling scheme is assumed. This is necessary for fixing the flow of the resource streams, where the communication structure gives the layout of piping the outgoing event function to the resp. successor node.The overall

delay of an event within the system can than be computed by summing up the individual delays experienced at node i. In case of alternative routes within the system, the max. must be built over the individual routes.

3.1.2 Approximating concave,convex piece-wise linear event curves

In the following we will develop a pattern for modelling a RTC-conformant curve γ by a set of interacting timed automata. The RTC-conformant curve γ represents the characteristics of a stream of incoming events or of a stream of available resources. In the following we will therefore speak generically of event streams, which implies either the modelling of arrival curves or to the modelling of resource availabilities by the resp. stream of service units.

> In the following we assume that γ can be described by a set of piece-wise linear functions, s.t. $\gamma^{\{u,l\}}$ is computable as the min., max.-function resp. on a set of linear functions as follows:

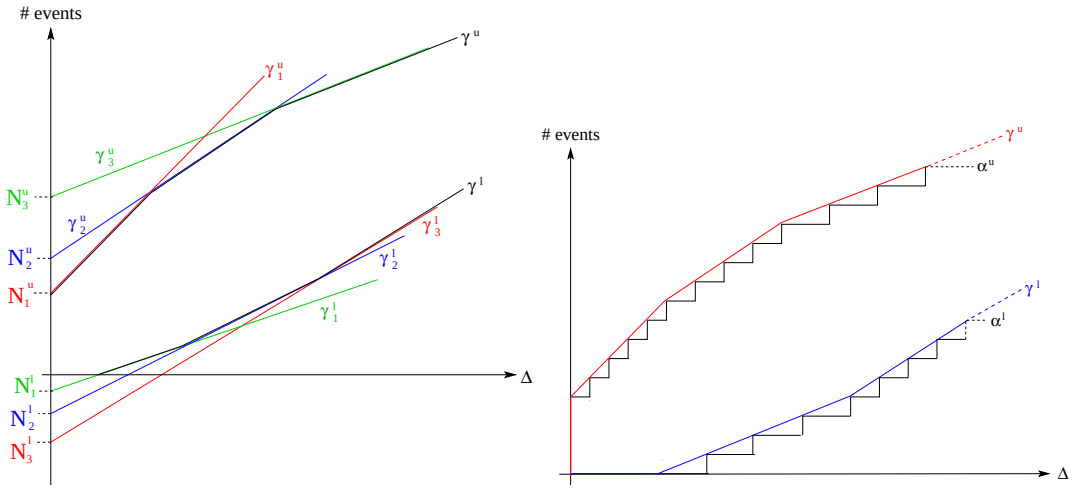
$$\begin{aligned} \gamma^u(\Delta) &:= \min_i(\gamma_i^u(\Delta)) \text{ with } \gamma_i^u(\Delta) := N_i^u + \tfrac{\Delta}{\delta_i^u} \\ \gamma^l(\Delta) &:= \max_i(0, \gamma_i^l(\Delta)) \text{ with } \gamma_i^l(\Delta) := N_i^l + \tfrac{\Delta}{\delta_i^l} \end{aligned} \tag{1}$$

For exemplification one may refer to Fig. 2.a. Parameter N_i^u in the above equations can be understood as (burst) capacity, which describes the number of events producible in zero time and $\delta_i^{\{u,l\}}$ as minimum/maximum distance of two successive events. Parameter N_i^l is smaller than 0, thus it lacks a intuitive physical explanation, its absolute value can be understood as the fictious number of events to be produced, before actual event emission takes place.

> As another major restriction, it is also assumed that $\gamma^{\{u,l\}}$ are not only sub-additive, but also concave,convex respectively.

In the following we also deal with complete amounts of events, i.e. the number of events are integer values. Thus we will approximate an arbitrary concave,convex RTC-conformant curve γ by staircase function α. This can be achieved by defining the individual staircase function $\alpha_i^{\{l,u\}}$ as floor or ceil of the resp. $\gamma_i^{l,u}(\Delta)$ function:

$$\alpha_i^{\{u,l\}}(\Delta) := N_i^{\{u,l\}} + \left\lfloor \frac{\Delta}{\delta_i^{\{u,l\}}} \right\rfloor \text{ for } i \in \{1, \ldots, n\} \tag{2}$$



(a) Piece-wise linear sets of concave, convex arrival curves

(b) staircase -functions for modelling arrival curves

Figure 2: Scenario to be handled in this report

The fact that the underlying $\gamma^{\{u,l\}}$ functions are furthermore assumed to be concave, convex resp. yields that the upper staircase -function α^u has increasing staircase - or step sizes, i.e. the distances between two jump points grows with Δ and that the lower staircase function α^l posses decreasing staircase sizes, i.e. the distances between two jump points shrinks for larger values of Δ. This situation is illustrated in Fig. 2.d and it will exactly be this scenario to be treated in the following. One may note that it is irrelevant for the staircase functions to be above or below the underlying RTC-conformant curve $\gamma^{\{u,l\}}$, since one simply may adjust the parameters $N_i^{l,u}$ accordingly, also the heights of the staircase s can be chosen at an arbitrary level of granularity.

3.2 State-based analysis of real-time systems

In this section we focus on the a state graph based analysis of real-time systems. In the past decades many of such methods have been developed, their commonness is that high-level descriptions are mapped to a finite, labelled transition systems, where the latter allows a detailed analysis of the system under study. Due to the importance of timed automata, the discussion will be limited to this description method. However, many other methods exist,

for example timed Petri net [15], the BIP-framework [4], among many others.
A timed automaton $\mathfrak{t}a$ is a directed action-labelled graph, which is extended with non-negative clocks. Conditions imposed on the clocks (time constraints) steer the execution of the TA.

Definition 3.2: Timed automata

A timed automata $\mathfrak{t}a$ is defined as a tuple $(\mathfrak{L}oc, \mathfrak{L}oc_0, \mathcal{C}, \mathcal{A}ct, \mathcal{E}, \texttt{Inv})$ with the following components:

- $\mathfrak{L}oc$ as set of location with $\mathfrak{L}oc_0$ as the set of initial locations,
- $\mathcal{C}$ as set of clocks,
- $\mathcal{A}ct$ as set of action labels, including internal τ-actions.
- $\mathcal{E} \subseteq \mathfrak{L}oc \times \mathcal{A}ct \times \mathcal{CC} \times 2^{\mathcal{C}} \times \mathfrak{L}oc$ as set of action-labelled edges and
- $\texttt{Inv} : \mathfrak{L}oc \mapsto B(C)$ as the function mapping locations to (time) invariants, where the latter are expressed as conjunctions over (atomic) time constraints of the kind $x \odot c$, with $x \in \mathcal{C}$, $c \in \mathbb{N}$ and $\odot \in \{<, \leq, =, \geq, >\}$.

As pointed out above, each edge of a TA is equipped with an action-label $a \in \mathcal{A}ct$, a clock condition or guard $g \in \mathcal{CC}$ and a set of clocks $C \in \mathcal{C}$ to be reset, once the edge is taken. With the locations $\mathfrak{l}_1$ and $\mathfrak{l}_2$ this yields the notation:

$$\mathfrak{l}_2 \overset{a,g,C}{\rightharpoonup} \mathfrak{l}_1.$$

In this context one may already note, that taking an edge, the guard of which evaluates to true, i.e. the time constraint of which is satisfied by the current values of the clocks, is optional, not mandatory. Furthermore, the edge can solely be taken, if the invariant of the target location is satisfied, once the specified clocks are reset. However, before the semantics of a TA is formalized, it is required to define valuations of clocks. For this purpose we define the function $\eta : \mathcal{C} \mapsto \mathbb{R}_{\geq 0}$, where $\forall x \in \mathcal{C} : \eta_0(x) = 0$ holds. Let $\texttt{Eval} \subseteq \mathbb{R}_{\geq 0}^{\mathcal{C}}$ be the set of all clock evaluations. This allows one to construct a satisfaction relation $\models \subseteq \texttt{Eval} \times \mathcal{CC}(\mathcal{C})$ between clocks and their constraints as follows:

- $\eta \models \mathit{true}$
- $\eta \models x < c \Leftrightarrow \eta(x) < c$

- $\eta \models x \leq c \Leftrightarrow \eta(x) \leq c$
- $\eta \models \neg g \Leftrightarrow \eta \not\models g$
- $\eta \models g \wedge g' \Leftrightarrow \eta \models g \wedge \eta \models g'$ where $g, g' \in \mathcal{CC}(\mathcal{C})$

From a TA one can now derive a labelled transition system TS, which is a tuple $(\mathfrak{S}, \widetilde{\mathcal{A}ct}, \rightarrow, \mathfrak{s}_0)$ and gives the semantic model of the TA. In this transition system $\mathfrak{S} \subseteq \mathfrak{Loc} \times \mathtt{Eval}(C)$ is the set of states, with the initial states $\mathfrak{s}_0 := \{[\mathfrak{l}_0, \eta] \mid \mathfrak{l}_0 \in \mathfrak{Loc}_0 \wedge \forall x \in \mathcal{C} : \eta(x) = 0\}$. The set $\widetilde{\mathcal{A}ct} \subseteq \mathcal{A}ct \times \mathbb{R}_{\geq 0}$ is the set of transition labels, for the connection relation $\rightarrow$ the following two rules are decisive:

(1) *discrete transition*: $[\mathfrak{l}, \eta] \xrightarrow{a} [\mathfrak{l}', \eta']$ if the following conditions hold: There exists an edge $\mathfrak{l} \overset{a,g,C}{\rightarrow} \mathfrak{l}'$ in the TA. The invariant possibly associated with location $\mathfrak{l}'$ must hold, once the transition is taken, and all clocks are re-evaluated to valuation η', where the latter also includes the reset clocks, the value of which is 0.

(2) *delay transition*: $[\mathfrak{l}, \eta] \xrightarrow{d} [\mathfrak{l}, \eta+d]$ $\forall d \in \mathbb{R}_{\geq 0}$ and the clock evaluation $\eta + d$ still satisfies the invariants associated with location $\mathfrak{l}$. For simplicity we used the notation $\eta + d$, which refers to the fact, that all clocks have advanced with d time units.

The above construction yields that in a TA time advances only while being in a location, transitions are instantaneous, where from a state infinitely many delay transitions can emanate. Consequently in a TA at a single time instant, several actions take place and the underlying TS consists of an infinite number of states. The main idea is now to define an equivalence relation on the states, by exploiting clock valuations. Once such an equivalence is established, the analysis can be restricted to the resulting quotient graph.

As first step clock evaluations are divided into a fractional and an integral part: (a) The integral part of a clock x is the largest integer with $d \leq \eta_a(x)$. As common the notation $\lfloor x_a \rfloor$ will be employed, where $\lfloor x_a \rfloor = \max(\{\eta_a(x) \in \mathbb{N}_0 \mid \eta_a(x) \leq r\})$. (b) The fractional part is then defined as the difference of $\eta_a(x)$ and $\lfloor x_a \rfloor$: $frac(x_a) := \eta(x_a) - \lfloor x_a \rfloor$. The index a in the above notation emphasizes that one speaks of a specific valuation η_a of a clock x. Now one is enabled to partition the set of clock valuations, where two clock valuations are considered equivalent *iff* the valuation of a specific clock is either larger as the largest constant the clock is compared to or if

the integral parts of the clock is equal in the valuations and the fractional parts are either 0 or obey the same orderings in the valuations. Let c_x be the largest constant clock x is compared with, the above partitioning can then be formalized as follows: Two clock valuations η_a and η_b are equivalent ($\eta_a \equiv \eta_b$) if and only if one of the following two conditions apply:

(1) $x \in \mathcal{C} : \eta_a(x) > c_x \wedge \eta_b(x) > c_x$ or

(2) $\forall x, y \in \mathcal{C}$ it holds that
   (i) $\lfloor x_a \rfloor = \lfloor x_b \rfloor$ and
   (ii) $frac(x_a) = 0 \Leftrightarrow frac(x_b) = 0$ and
   (iii) $frac(x_a) \leq frac(y_a) \Leftrightarrow frac(x_b) \leq frac(y_b)$

Condition one gives, that the number of equivalence classes is finite, since once a clock x is larger than the largest constant c_x it is compared to, it affects the guards of all edges in the same manner. Contrary to this condition two deals with the case, that the valuation agree on the integral parts of a clock. In such cases the fractional part must either be 0 or give the same orderings of the clocks in the valuations. This guarantees that the valuations of a specific class enables the same edges of the TA, since the order of the clocks gives an ordered sequence of guards satisfied by the clocks, which makes it clear why in TA clocks are only allowed to be compared with integer values.

The partitions of the clock valuation following the above conditions are denoted as clock regions. Following that all states of the transition system of a TA which carries the same location and have equivalent clock evaluation form also an equivalence class, denoted as state region. Based on this we give the following definitions:

Definition 3.3: Region graph

(1) A clock region of a TA $\mathfrak{t}a$ is defined as follows:

$$[\eta_a] := \{\eta_b \in \texttt{Eval}(\mathcal{C}) \mid \eta_a \equiv \eta_b\},$$

where $\texttt{Eval}(\mathcal{C})$ refers to all clock evaluations of $\mathfrak{t}a$ and $\equiv$ is the equivalence relation established above.

(2) A state region of a TA $\mathfrak{t}a$ is defined as follows:

$$[\langle \mathfrak{l}, [\eta]\rangle] := \{\langle \mathfrak{l}, \eta_b\rangle \mid \eta_b \in [\eta]\}$$

This gives now, that a TA can be mapped to a quotient TS, which is finite in the number of states. However, as it turns out the number of the clock regions $|\texttt{Eval}/\equiv|$ and thus the number of state regions of the original TS is still exponential with respect to the number of clocks and clock constants:

$$|\mathcal{C}|! \cdot \prod_{x\in\mathcal{C}} c_x \leq |\texttt{Eval}/\equiv| \leq |\mathcal{C}|! \cdot 2^{|\mathcal{C}|-1} \prod_{x\in\mathcal{C}} (2c_x + 2).$$

In practice the analysis of complex models is therefore limited, but still feasible as long as the number of clocks and constants is truly limited.

For approaching the complexity of systems, their modularization seems appropriate. This requires composition operators which allow the construction of complex system models from individual components, TA resp.. A well established procedure is the synchronization of TA, where dedicated edges can only be jointly taken and where non-synchronized edges have to be taken sequentially, yielding their shuffled appearance within the resulting product automaton. For keeping the semantics of a composition operator as simple as possible, it is required that automata to be synchronized do not share clocks and that edges to be jointly taken carry the same label, leading to the following rules when combining 2 TA $\mathfrak{t}a_a$ and $\mathfrak{t}a_b$:

(1) Synchronizing edges:

$$\frac{\mathfrak{l}_a \xrightarrow{\alpha,g,C_a} \mathfrak{l}'_a \wedge \mathfrak{l}_b \xrightarrow{\alpha,h,C_b} \mathfrak{l}'_b}{\langle \mathfrak{l}_a, \mathfrak{l}_b\rangle \xrightarrow{\alpha,g\wedge h,C_a\cup C_b} \langle \mathfrak{l}'_a, \mathfrak{l}'_b\rangle}$$

(2) Independent edges:

$$\frac{\mathfrak{l}_j \xrightarrow{\alpha,g,C} \mathfrak{l}'_j}{\langle \mathfrak{l}_k, \mathfrak{l}_j \rangle \xrightarrow{\alpha,g,C} \langle \mathfrak{l}_k, \mathfrak{l}'_j \rangle} \; k, j \in \{a, b\}, \text{ and } k \neq j$$

Following the definition of TA as employed in the tool Uppaal [6] we extend TA by

(1) bounded integer variables, which can be employed in expressions defining guards of edges and invariants of locations. Their value assignments take place, once a specific edge is followed. Thus one not only resets a set of clocks, but also may update a set of variables, once following an enabled edge, i.e. an edge the guard of which evaluates to true. The invariant of the target location must evaluate to true, with respect to the new valuation of the clocks and variables.

(2) synchronization: by default Uppaal applies binary synchronization, i.e. from a set of enabled edges carrying the same activity label one non-deterministically chooses a sending and receiving edge for joint execution. Sending is indicated with an activity-label followed by an exclamation mark, whereas receiving is indicated by an activity-label followed by a question mark. One may note, that for each sender there must be a receiver and vice versus, willing to participate in the synchronization. If this is not the case the respective edges are blocked from execution. Binary synchronization can be extended to non-blocking broadcasting semantics, i.e. one sender synchronizes with 0 up to n receivers.

(3) urgent synchronization: binary and broadcasting communication can be made urgent. This gives that enabled sending and receiving edges carrying the same activity-label are immediately scheduled for joint execution, where in case of the binary synchronization both, i.e. sender and receiver must be enabled for participating in the synchronization.

(4) urgent locations force the system to proceed as soon as they are reached, i.e. the system is not allowed to spent any time in an urgent location. Alternatively such a feature can be implemented by adding an extra clock z, reset this clock on all incoming edges and equip the target location with the clock invariant $z \leq 0$.

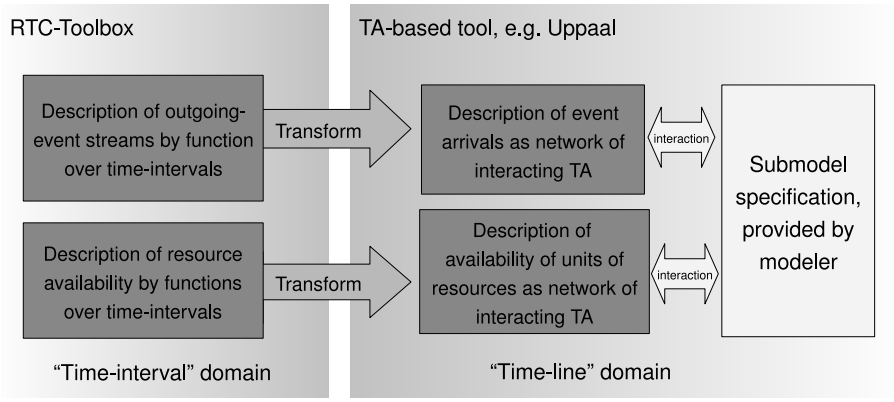


Figure 3: Combining analytic and state-based models

4 TA-based modelling of event curves

In the following we will develop a pattern for modelling a RTC-conformant curve representing an arrival of service curve with a network of interacting TA. In total the presented approach allows one to generated every event stream within a TA system model and as bounded by the respective upper and lower RTC-conformant curves. The event generating network of TA can than be used for stimulating or suppressing behavior of a down-streamed TA, where the latter may represent some piece of system behavior, e.g. some communication task mapped to a specific resource. For exemplification one may refer to Fig. 3

Before we proceed one may recall, that Sec. 3.1.2 suggested, that each convex, concave and piece-wise-linear RTC-conformant function γ is approximated by a staircase function α and its associated individual staircase functions α_i. In accordance with Eq. 1 the arrival curve $\alpha^{\{u,l\}}$ to be modelled by a set of TA can be defined as follows:

$$\alpha^{\{u,l\}}(\Delta) := \begin{cases} \alpha_1^{\{u,l\}}(\Delta) & \Leftrightarrow 0 \leq \Delta < K_1^{\{u,l\}} \\ \vdots & \vdots \\ \alpha_i^{\{u,l\}}(\Delta) & \Leftrightarrow K_{i-1}^{\{u,l\}} \leq \Delta < K_i^{\{u,l\}} \\ \vdots & \vdots \\ \alpha_n^{\{u,l\}}(\Delta) & \Leftrightarrow K_{n-1}^{\{u,l\}} \leq \Delta \end{cases} \tag{3}$$

where $\alpha_1^l := 0$ and the $\alpha_i^{\{u,l\}}$ are defined by Eq. 2. As common it is required that the RTC-conformant curves $\gamma^{\{u,l\}}$ to be approximated by the staircase function $\alpha^{\{u,l\}}$ is wide-sense increasing, which gives here

Definition 4.1: Wide-sense increasing (decreasing) functions

A staircase function α is denoted wide-sense increasing (decreasing) $\Longleftrightarrow$ $\forall \Delta, \Delta' \in \mathbb{R}_+$ holds

$$\alpha^{\{u,l\}}(\Delta) \leq \alpha^{\{u,l\}}(\Delta') \text{ for } \Delta \leq \Delta'.$$

Since $\gamma^{\{u,l\}}$ is concave, convex respectively, for the parameters of the $\alpha_i^{\{u,l\}}$ functions it must hold:

$$|N_i^{\{u,l\}}| < |N_{i+1}^{\{u,l\}}|, 0 < \delta_i^u < \delta_{i+1}^u \text{ and } \delta_i^l > \delta_{i+1}^l > 0 \text{ for } i \in \{1, \ldots, n-1\} \quad (4)$$

In the following we will denote a staircase function α fulfilling the above definition as wide-sense convex, concave respectively, formally:

Definition 4.2: Wide-sense convex (concave) functions

A wide-sense increasing (decreasing) staircase function α is denoted wide-sense convex (concave) $\Longleftrightarrow$ $\forall t, t' \in \mathbb{R}_+$ with $t \leq t'$ and step widths δ_i holds that

$$\delta_i \overset{(\geq)}{\leq} \delta_{i+1} \text{ for } i \in \mathbb{N}_0$$

In the following we will now solely refer to wide-sense concave, convex staircase functions $\alpha^{\{u,l\}}$ and their associated staircase (sub-)functions $\alpha_i^{\{u,l\}}$.

According to the above discussion the $K_i^{\{u,l\}}$ in Eq. 3 are defined as those jump points, where for the step width δ_i and δ_{i+1} $\delta_i > (<)\delta_{i+1}$ holds. For an arbitrary convex, concave piece-wise-linear RTC-conformant curve $\gamma^{\{u,l\}}$ we can now compute the piece-wise linear functions $\gamma_i^{\{u,l\}}$ and from them each piece-wise staircase function $\alpha_i^{\{u,l\}}$ and thus obtain the overall piece-wise staircase functions $\alpha^{\{u,l\}}$. How to do this is indicated in Fig. 4.

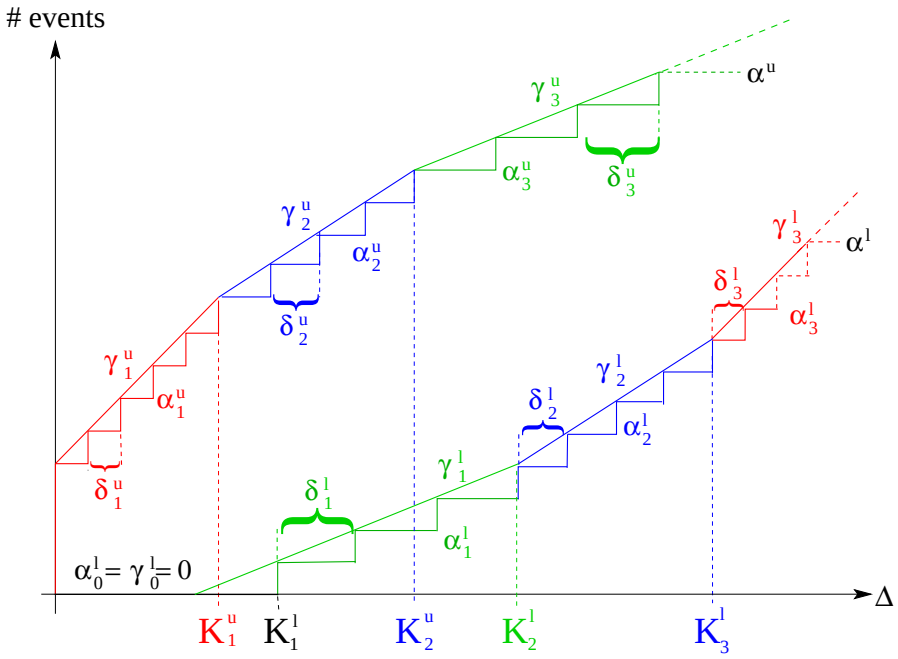


Figure 4: Piece-wise linear upper and lower arrival curves and the resp. staircase functions

4.1 The pattern

The main idea of the approach is taken from the leaky bucket mechanism for shaping network flows. In the context of this work this means, that the number of events producible in the near future is determined by the status of a set of counters $\mathcal{C}$, –one may recall that we are dealing with discrete number of events. In our approach we model each of the above $\alpha_i^{\{u,l\}}$ curves by its own TA i, where a network of such automata models all traces bounded by $\alpha^{\{u,l\}}$. Each automaton contains its local clock x_i and its local counter c_i. Each counter increases periodically as time progresses, i.e. each time $x_i^{\{u,l\}} = \delta_i^{\{u,l\}}$ the local counter c_i is incremented. For periodically incrementing a local counter c_i, we simply rest local clock x_i after the increment of c_i has been executed, where the execution of the respective transition is enforced by a location invariant $x_i^{\{u,l\}} \leq \delta_i^{\{u,l\}}$ and the transition guarded by the expression $x_i^{\{u,l\}} = \delta_i^{\{u,l\}}$. Contrary to this, the generation of an event yields a decrement of all counters (not only of a single one!). The joint decrement of the local counters is implemented by fully synchronizing

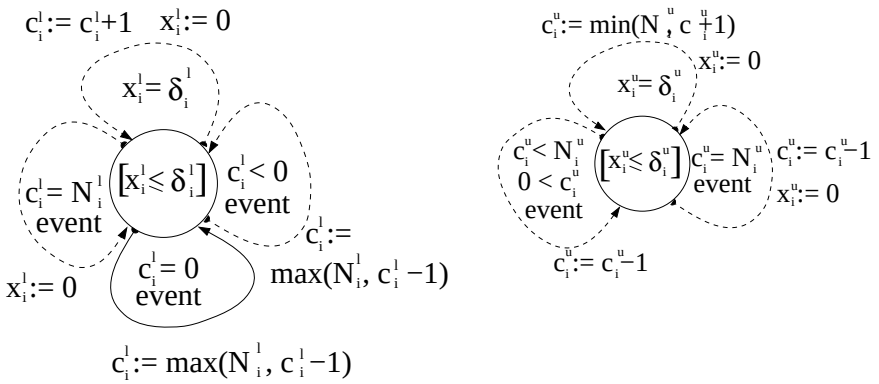


Figure 5: TA modelling arrival curve $\alpha_i^{\{u,l\}}$

the individual TA.

Fig. 5 shows our approach. The TA shown on the left models α_i^l, the one on the right α_i^u. The dashed lines indicates that the associated transitions might be taken as soon as their guards evaluate to true. Contrary to this, the solid line is associated with an urgent transition, meaning that this transition has to be executed (!) as soon as its guard is evaluated to true, which can be achieved by assigning a high priority or in case of Uppaal an urgent synchronization to the resp. transition. The guards of the different transitions are given inside the looped arcs, whereas clock resets and variable updates are specified outside. One may note, that in the above setting full synchronization is mandatory. This means that either all TAs execute an event generating transition or none. This is why the transition label "event" occurs next to the guards. Thus it is assumed that transitions with the same label always (fully) synchronize, i.e. if not all partners can perform the synchronizing transition it is blocked until this is the case:

$$\frac{\exists\,\mathrm{TA}\; i \in \mathcal{M} : l_a^i \not[\!\!\succ\; \alpha}{\not\exists \langle l_a^1, \ldots, l_a^n \rangle \xrightarrow{\alpha, g_\alpha^1 \wedge \cdots \wedge g_\alpha^n, C_a^1 \cup \cdots \cup C_a^n} \langle l_{a'}^1, \ldots, l_{a'}^n \rangle \in TS(\mathcal{M})} \; \alpha \in \mathcal{A}ct_S$$

In the above formula a model $\mathcal{M}$ is assumed to consist of n TA to be synchronized. $l_a^i \not[\!\!\succ\; \alpha$ states that the guard of edge labelled with α emanating from location l_a^i does not evaluate to true, i.e. transition α is not enabled. As drawback full synchronization gives that the network of synchronizing TAs may deadlock, due to the fact that not all TA can execute an event

generating transition, –we assume, that an enabled urgent transition prevents execution of any other enabled non-urgent transition. In our approach such a situation can only happen, if the modeler has chosen an inconsistent set of parameters for the TAs modelling the upper and lower arrival curve. Inconsistency exists if the properties of Eq. 4 are violated and/or $\alpha^u < \alpha^l$. –In the actual implementation of the approach within Uppaal we will make use of a dedicated error location, which enforces a deadlocking of the system, which allows us to detect inconsistent parameter specifications in a straightforward manner. Finally one may note, that each local clock x_i must be reset once an event has been generated and c_i currently holds (a) its maximum N_i^u in case of a TAs modelling an upper curve or (b) its minimum N_i^l in case a TA modelling a lower curve. This is necessary, due to the fact, that the maximal number of events producible by TA i within 0 time is bounded by N_i^u, and that the maximum distance between two events can be large as $|N_i^l| \cdot \delta_i^l$.

Now it should also be clear why we decided to discretize the number of events, rather than to make use of continuous flows as modelled by the original RTC-conformant γ curves as introduced in Sec. 3.1.1: Within a time - and(!) state continuous setting, i.e. within a setting where not only time is continuous, but also the number of events, it is required to employ continuous counters for tracking event production. However, for staying in the class of TA we cannot allow the manipulation of clocks, except for resets and we can also not allow continuous counters which dynamically change over times. This later case would refer to modelling environments where clocks may run at different speeds, which truly goes beyond the concept of TA.

4.2 Proof of correctness and completeness

In the following we will show, that the above pattern describes all event traces bounded by $\alpha^{\{u,l\}}$. To do so, we will first show, that the network of TAs (NTA) defined above does not produce event traces violating $\alpha^{\{u,l\}}$. This will be followed by a discussion on the completeness of the approach, i.e. if all traces bounded by $\alpha^{\{u,l\}}$ are considered by the NTA.

For convenience we give the following definitions:

- Let $\mathcal{C}^{\{u,l\}}$ be the set of all local counters c_i of the TAs,
- $\mathfrak{S}$ the set of all reachable system states

- $\eta_s(c_i)$ the evaluation of counter c_i with respect to state s and $\eta(c_i, t)$ the evaluation of counter c_i at time t. –For simplicity we will not always make a clear distinction between counters and their evaluations, as long as this is clear from the context. Often we will also use the notation $c_i(t)$ when referring to the evaluation of counter c_i at time t.–

- and let e_j be an event generating transition of TA j with $\mathcal{E}^{\{u,l\}}$ be the set of all such transitions.

For convenience we introduce the following notation: A TA $\mathfrak{t}a_i$ modelling an upper arrival curve α_i^u is denoted as U-TA $\mathfrak{t}a_i$ and the TA $\mathfrak{t}a_i$ modelling a lower arrival curve $\alpha_i^l$is denoted as L-TA $\mathfrak{t}a_i$. In the following we will now investigate the U-TAs and L-TAs separately.

4.2.1 Upper-TA

For the network of U-TAs modeling the upper event curve α^u in co-operation the following features and definitions are relevant:

- In isolation U-TA i can generate at most N_i^u events in a single instant of time, since once c_i is zero all event generating transitions are disabled as long as $x_i < \delta_i$ holds. Thus the max. number of events producible in an interval smaller than δ_i^u is bounded by N_i^u.

  - The number of events producible by U-TA i is bounded by

    $$\alpha_i^u(\Delta) := N_i^u + \left\lfloor \frac{\Delta}{\delta_i^u} \right\rfloor,$$

    since once c_i equals zero, solely every δ_i time units an event can be generated, one may recall that for $c_i = N_i^u$ and an event generation a reset of x_i to 0 is done.
  - For all the U-TAs the max. number of events producible in an interval $\Delta < \delta_1^u$ is bounded by N_1^u, see also Eq. 4.

- Upper Synchronization constraint: Once the counter of any U-TA i is zero, it blocks all event generating transitions of the other U-TAs due to synchronization:

  $$\forall s, t \in \mathfrak{S} \ : \ \text{if} \ \exists c_i \in \mathcal{C}^u : \eta_s(c_i) = 0 \ \text{then} \ \nexists e_j \in E \ \text{s.t.} \ s \xrightarrow{e_j} t \ \text{holds.}$$

  The blocking of event generating transitions stops at point $t_k \iff$ there $\exists c_j \in \mathcal{C}^u$, s.t. $c_j(t_k) = 0$ holds.

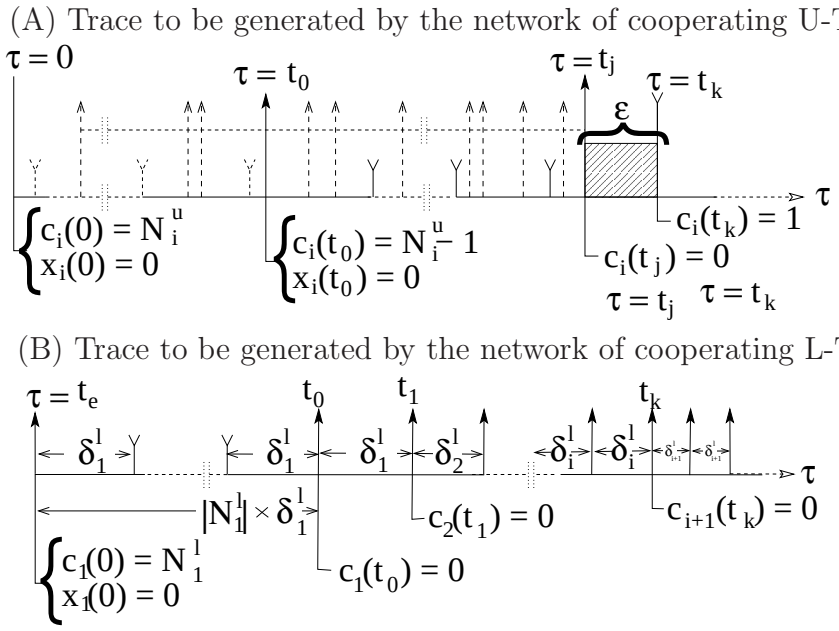


Figure 6: Time-line and values of c_i at different points in time

- Let t_j be the earliest point in time and let i be smallest index for which $c_i(t_j) = 0$ holds. The earliest point in time t_k, where the next event can be generated is than given as

$$\max_{\forall i: c_i(t_j)=0} \left(\inf_{t_k > t_j} \left(\{ t_k |\, c_i(t_k) \neq 0 \} \right) \right) \tag{5}$$

  i.e. one may execute an event generating transition not before counter c_i is greater than 0. Given that at t_0 the local clock x_i experiences a reset, i.e. there is an event generation once $c_i = N_i^u$ it must hold $(t_k - t_0) \bmod \delta_i^u = 0$.

Before we go on one may refer to Fig. 6.A which illustrates some generic trace to be produced by the network of U-TAs. The arrow-headed vertical lines mark the generation of events, where the solid lines indicate the event generation at some fixed points in time and the dashed lines simply illustrates the non-determinism related to the event generation, i.e. for the discussion to follow these points in time are irrelevant. The lines with arrow-ends mark the point in times, where counter c_i of U-TA i is incremented. Once again only the solid lines are concrete point in times, increments associated with dashed lines are irrelevant for the time being. For convenience

the following points in time on this trace tr should be defined, where we will also point out the values of the associated counter c_i and the local clock x_i.

- Let τ be the global time progressing from left to right on the illustrated trace.
- At $\tau = 0$, which is the starting point for all traces each counter c_i is set to its maximum N_i^u and each local clock x_i is initialized with 0.
- $\tau = t_j$: At t_j an event is generated leading to a decrement of c_i, which is supposed to be 0 after this decrement ($c_i(t_j) = 0$) with $c_i(0) = N_i^u$ we can have generated at most

$$N_i^u + \left\lfloor \frac{t_j}{\delta_i^u} \right\rfloor$$

  events. After t_j U-TA i blocks the generation of events until c_i is incremented once again, which is due to the upper synchronization constraint. This increment is executed at t_k, s.t. from this point on a new event can be generated. Here we choose i in such a way, that it addresses the U-TA which is responsible for blocking event generation exactly up to time $t_k > t_j + \epsilon$. Let t_k be the earliest point in time, where c_i is incremented from zero to one, and where U-TA i is responsible for the blocking of events up to time t_k as pointed out above. –We assume that $\forall c_j \in C : c_j(t_k) \neq 0$ for $j \neq i$ holds, i.e. all other counters are greater than zero. This is justified since otherwise i must simply be chosen differently, namely as the TA i which is the last one to increment its counter (cf. Eq. 5).–
- $\tau = t_0$: Let t_0 be the last point in time, with respect to t_j, where an event was generated and $c_i = N_i^u$ held, which enforced a re-synchronization of the U-TA's local clock, due to the executed reset of x_i. This also gives that $(t_0 - t_k) \bmod \delta_i^u = 0$ must hold.

First it will be show, that any stream of events to be derived from a network of fully synchronized U-TA is bounded by the respective event curve α^u.

Lemma 1. *Any trace tr as producible by a network of fully synchronizing and parameterized U-TA is bounded by a wide-sense concave event function α^u.*

According to the above discussion the parameters N_i^u and δ_i^u of a U-TA $\mathfrak{t}a_i$ correspond to the parameters of the respective step-function α_i^u and for

the network of U-TA Eq. 4 applies. In the following we need now to show that the number of events producible by the network of synchronizing U-TA in an interval Δ is bounded by the minimum on the α_i^u-functions.

Proof. Let us assume that the n U-TAs produce a trace tr which violates α^u. The above *upper synchronization constraint* gives, that for any $\Delta \in \mathbb{R}$ the max. number of events producible is bounded by

$$\alpha_{tr}^u(\Delta) := \min_i (N_i^u + \left\lfloor \frac{\Delta}{\delta_i^u} \right\rfloor),$$

where $\alpha_{tr}^u(\Delta)$ is the event arrival curve derived from some trace as produced by the network of U-TAs. For the trace tr actually to violate α^u it must hold that $\alpha_{tr}^u(\Delta) > \alpha^u(\Delta)$. Now consider that for $c_i(t_j + \epsilon) = 0$ the production of events is blocked. As defined by the U-TAs for $c_i(t_0) = N_i^u$ an event generation resets clock x_i. Let now t_0 be the last time where this happened, –one may note that at least at $\tau = 0$ this had happened. According to Eq 5 this gives that counter c_i blocks all event generations until $x_i = \delta_i^u$ holds once again, remember we decided to chose i in such a way, that it was U-TA i which blocks the event generation. Thus up to time $t_j + \epsilon$ at most

$$\alpha_{tr}^u(t_j + \epsilon) := N_i^u + \left\lfloor \frac{t_j + \epsilon}{\delta_i^u} \right\rfloor$$

events could have been produced. This is exactly what was defined for each α_i^u in Eq. 2. Thus the number of events appearing on any trace as produced by the U-TAs is bounded by U-TA i once c_i is zero and $c_j > 0$ for $j > i$ holds. This shaping of the arrival curve is shifted from U-TA $i-1$ to U-TA i once c_i is still zero but c_{i-1} is already incremented, i.e. from now on U-TA i is the restricting resource. The respective point in time K_{i-1} can be computed as follows:

$$K_{i-1} := (N_{i-1} - N_i) \left(\frac{\delta_{i-1}\delta_i}{\delta_{i-1} - \delta_i} \right).$$

Thus one obtains exactly what was defined for α^u in Eq. 3, see also Fig. 4. This also gives that $[0, K_{i-1}]$ is currently the largest interval of consideration and therefore any trace tr as produced by the n U-TAs can not violate α^u, where of course the parameter δ_i^u and N_i^u for U-TA i must be chosen appropriately, i.e. as derivable from α^u and where the conditions of Eq. 4 apply. □

In the next step, we will now discuss, why the proposed network of U-TAs is capable of producing any trace bounded by α^u.

Lemma 2. *The network of U-TAs is capable of producing any trace tr as bounded by a wide-sense convex step-function α^u.*

The proof will be carried out by contradiction.

One may assume that there is a trace tr bounded by α^u, but it can not be produced by the network of U-TAs. From this it follows that there must be a pair (time-stamp, event) (t, e) where the U-TAs are blocked, but α^u would allow the arrival of an additional event, i.e.. $\exists(t, e) \in tr : (t, e) \notin tr^{TA}$, where tr^{TA} is a trace produced by the network of U-TAs with the same prefix up to the pair (t, e). Let this point in time be $t_j + \epsilon$, where actually it must hold that there $\exists c_i \in C$, s.t. $c_i(t_j + \epsilon) = 0$, otherwise the U-TAs would be capable of producing an event. Let t_j be now the earliest point in time for tr where $c_i(t_j) = 0$ holds and let $t_0 < t_j$ be the last point in time where $c_i(t_0) = N_i^u$ was satisfied, which is exactly what we have illustrated in Fig. 6. The choice of i and the blocking of events implies now that for the blocking period ϵ it must hold $t_j + \epsilon < t_k$, where t_k is the next time c_i is incremented and the generation of an event would not be blocked anymore. From this it follows that for the number of events $R^{tr}(0, t_j + \epsilon)$ seen on tr in the interval $[t_0, t_j + \epsilon]$ it must hold:

$$R^{tr}(0, t_j + \epsilon) \geq N_i^u + \left\lfloor \frac{(t_j + \epsilon - t_0)}{\delta_i^u} \right\rfloor + 1,$$

otherwise c_i would not be 0 and the U-TAs not blocked. But this violates the bound imposed by α^u, since for $K_{i-1} \leq (t_k + \epsilon - t_0) < K_i$ it follows that the number of events is bounded by α_i^u which is truly smaller than $R^{tr}(0, t_j + \epsilon)$. Thus from $R^{tr}(0, t_j + \epsilon) > \alpha^u$ we can conclude that such a trace tr does not exists. Concerning the assumption that t_0 was the last point in time where $c_i(t_0) = N_i^u$ was satisfied and that for t_k: $c_i(t_k) = 0$ must hold, one may note that for the initial point in time we have $c_i(0) := N_i^u$ and that $c_i = 0$ must be zero at t_j, since otherwise the synchronization constraint would not be valid.

Finally we can formulate the following theorem:

Theorem 1. *The network of U-TAs generates all traces bounded by a wide-sense convex step-function α^u.*

The above theorem can be directly derived from the upper two lemmata, since they give that neither the U-TA- can violate the upper arrival curve α^u nor that a trace is omitted.

After the conformance between U-TAs and upper arrival curve α^u have been discussed, we will now concentrate on the set of TAs modelling the lower arrival curve.

4.2.2 Lower-TA

The L-TAs have the following characteristics:

- In isolation L-TA i has to generate an event after $|N_i^l| \cdot \delta_i^l$ time units, namely once c_i is zero, which is enforced by the urgent transition:

  $$\forall s,t \in \mathfrak{S} \ : \ \text{if} \exists c_i \in \ \mathcal{C}^l : \eta_s(c_i) = 0 \, \text{then} \exists e_j \in E \ \text{s.t.} \ s \xrightarrow{e_j} t \, \text{holds.}$$

  Thus TA i bounds the max. distance between two events to $|N_1^l| \cdot \delta_1^l$ time units, namely once $c_1(t) = N_1^l$ and all TAs are inactive until $c_i(t') = 0$ holds with $t' = t + |N_1^l| \cdot \delta_1^l$. This gives that in addition to Eq. 4 it must also hold:

  $$|N_i^l| \cdot \delta_i^l < |N_{i+1}^l| \cdot \delta_{i+1}^l \, \text{for} \, 0 \leq i < n$$

  s.t. in our setting $|N_1^l| \cdot \delta_1^l$ is the largest period of in-activity, i.e. without event generation. In the absence of the above property any U-TA j with the smallest max. distances $|N_j^l| \cdot \delta_j^l$ and a larger index $j > i$ would prevent any other TA i from enforcing event generation as its slower speed δ_i^l.

- Once $c_i(t) = 0$ and no other TA executes an event generating transition, L-TA i enforces this every δ_i^l time units, namely if c_i hits zero once again, which happens at $t + \delta_i^l$, $t + 2\delta_i^l$, etc.. In combination with other L-TAs this can be done until there $\exists c_j(t') \in C$ with $i < j$ where $c_j(t') = 0$ holds. Remember $\delta_i > \delta_j$ imposes an event generation every δ_j time units once c_j is zero.

For exemplification one may refer now to Fig. 6.B which illustrates a trace to be enforced by the network of L-TAs. At time t_e the U-TAs generate an event. Let us assume that $c_1 = N_1^l$ holds, thus x_1 is reset to zero. After $|N_1^l| \cdot \delta_1^l$ time units of being in-active c_1 equals 0 which enforces the immediate generation of an event. At most every δ_1^l time units the production of

another event must follow until c_2 also equals 0. In our example this happens after exactly one step. From now on event production is enforced after at most every δ_2^l time units. In general L-TA enforces the event generation every δ_i^l time units until c_{i+1} equals zero and event generation is taken over by L-TA $i+1$. This leads us to the following theorem:

Theorem 2. *The network of L-TAs enforces the generation of events, s.t.*

$$\forall tr \in tr^{TA} : R^t r(s,t) \geq \alpha_i^l(t-s) \, for \, 0 \leq s < t \, holds$$

where $R^t r(s,t)$ is the number of arrivals between time point s and t with respect to trace tr.

For showing that the network of L-TAs does not violate α^l, we must distinguish between the following cases:

(1) $0 \leq \Delta < K_1^l$: If one chooses the parameters of L-TA 1 s. t. $K_1 = |N_1^l| \cdot \delta_1^l$ holds, the number of events the generation of which is enforced is zero, which complies with the definition of Eq. 3.

(2) $K_{i-1}^l \leq \Delta < K_i^l$: As pointed out above, L-TA $i-1$ enforces the generation of events until $c_i = 0$ holds. Thus we have the following relation between the maximal sizes of the c_i's which must be satisfied when choosing the parameters of the $L-TAs$:

$$N_{i-1}^l + \left\lfloor \frac{\Delta}{\delta_{i-1}^l} \right\rfloor = N_i^l + \left\lfloor \frac{\Delta}{\delta_i^l} \right\rfloor$$

which also gives that K_{i-1}^l can be computed as

$$K_{i-1}^l := (N_i^l - N_{i-1}^l) + \left\lfloor \frac{\delta_{i-1}^l \delta_i^l}{(\delta_i^l - \delta_{i-1}^l)} \right\rfloor$$

as already pointed out above. The last question to think of is the question if states where $c_i = 0$ holds are reachable. Given a initial configuration with $c_i = N_i^l$ and the setting of Eq. 4 and an adequate choice of the parameters of the L-TAs as pointed out above, this is evident: for $i = 1$ one simply stays inactive, s.t. at least after $|N_1^l| \cdot \delta_1^l$ $c_1 = 0$ holds. If producing events with distance δ_i^l c_{i+1} will be zero after at least K_{i+1}^l time units, as computed above.

4.2.3 Network of cooperating U-TAs and L-TAs

The discussion above gives that the U-TAs will guard the upper bound as defined by α^u, and the L-TAs enforce that the lower bound as defined by α^l is also never violated. This latter requirement is achieved by enforcing event generation by the urgent transition of L-TA i once c_i is equal to 0. This yields:

Theorem 3. *The network of TAs as illustrated in Fig. 5 models all traces bounded by a wide-sense convex, step-function α^u and a wide-sense concave step-function α^l.*

Th. 1 and 2 give that the network of TAs does not violate the upper and lower arrival curves, where for theorem 1 it was also shown, that traces not covered by the network of cooperating U-TAs are not bounded by α^u. This together proofs the above theorem.

4.3 Implementation

Since Uppaal only supports $1 : n$ with $n \in \{0, \dots N\}$ synchronization, instead of full synchronization and since it only supports urgent synchronization, rather than urgent transitions the approach had to be adapted. To do so we employed a TA for scheduling the event generating transitions, denoted in the following as manager. Furthermore we employed a broadcast channel for enforcing the synchronous execution of event generating transitions, and we made use of an urgent broadcast channel, which is necessary for enforcing the generation of events according to α^l. The implementation of the approach is illustrated in Fig. 7, where the TA on the left models arrival curve α_i^l, the one on the right the upper arrival curve α_i^u and the TA on top is the manager, for coordinating event generation. Contrary to the concept illustrated above the L-TA i enforces event generation once $c_i = |N_i^l|$ and resets its clock each time $c_i = 0$ holds.

For enforcing full synchronization between the individual TAs, the implementation makes use of the function `upperBucketsFilled`() and `updateBuckets`(). The former function simply evaluates the predicate

$$\bigwedge_{c_i \in \mathcal{C}^u} c_i > 0,$$

s.t. event generation only takes place if `upperBucketsFilled`() = *true* holds. Function `updateBuckets`() simply decrements each $c_i \in \mathcal{C}^u \cup \mathcal{C}^l$ by one as long as c_i is actually greater than 0, i.e. it computes $c_i := min(0, c_i - 1)$,

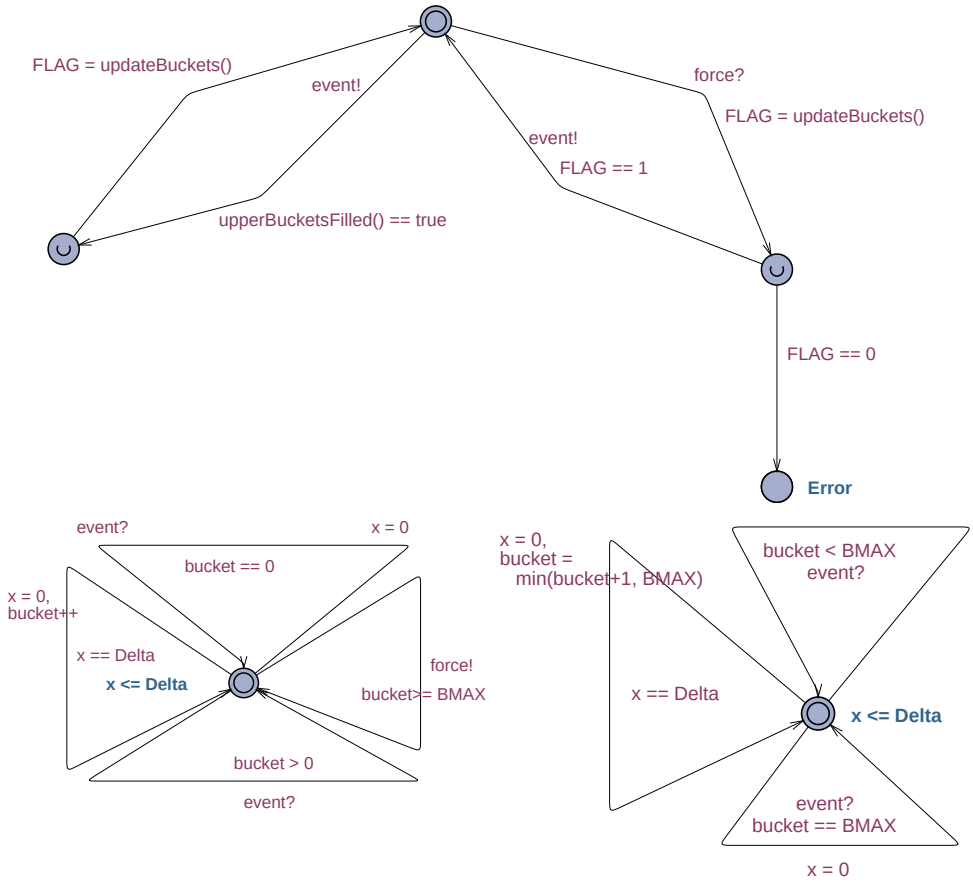


Figure 7: TA modelling arrival curve $\alpha_i^{\{u,l\}}$

rather than doing this explicitly within the individual L-TAs and U-TAs. In case `updateBuckets`() is executed, but there $\exists c_i \in \mathcal{C}^u : c_i = 0$ it returns *false* which can be used by the manager for detecting inconsistent specifications. Consequently the error location of the manager is solely meant to detect inconsistency introduced by the user then choosing wrong sets of parameters for the U-TAs and L-TAs. For enforcing a 0-time delay between testing and updating counters the manger makes use of committed locations.

5 Conclusion

5.1 Summary

Until now this work has developed a method for including RTC-based model descriptions into TA-base system models. For simplicity we limited the scenarios to be combined with TA-based models to the case of convex and concave RTC-conformant curves. Therefore this report can only be seen as first step. In particular since the other directions, i.e. the question how to derive RTC-conformant curves from TA-based system descriptions remain unanswered. However our approach significantly differs from the work of others also aiming a combination of state- and RTC-based modelling.

The authors of [12] address also the combination of RTC and TA. For including arrival and service curves of the RTC into TA-based system models one operates on an array of clocks. Each clock is associated with the number of events produced so far, as well as with a minimal and maximal number of events to be generated within the resp. time interval length. In total this gives the event generating automaton, stimulating the actual system model, i.e. its representing TA respectively. For deriving RTC-based curves from the combined model the authors suggest the usage of an observer automaton. This observer automaton records the numbers of events to be seen within a pre-defined interval of time. The usage of a priced TA [5] for each interval length allows one to derive the maximal and minimal number of events seen within the overall model and with respect to the resp. interval length. This make the limitation of this approach evident, one solely operates on a finite set of values of the RTC curves, which leaves the question when to stop with the translation of an RTC-curves into a TA and vice versus unanswered. Furthermore the number of clocks seems also to be a potential source of trouble. One either needs one observer automaton with its local clock for each interval length, or must execute a full state space exploration for each of the interval lengths. Also on the side of the event generating automaton the number of clocks may be problematic, one basically needs one clock per upper and lower bound for the number of events seen on the stream within the resp. time interval. This seems to be often not necessary, since RTC-curve may exhibit regular behavior, i.e. behavior described by piece-wise linear functions or piece-wise periodic staircase functions, where in the approach presented here only one clock for each of such regular segment is necessary.

5.2 Future work

The direction of future research is two-fold, at first we need to extend our work, so that the can lift the restrictions on the model world to be analyzed, but most importantly we need also to investigate methods for re-translating TA-based models into RTC-conformant event curves. As pointed out in the introduction to this report, a major aim of this work is the modelling of complex embedded real-time systems, where we will focus on the ComBest-project case-study [20]. In the following we will discuss these issues separately.

5.2.1 Future Research

General RTC-conformant curves: As pointed out above, the discussion in this report was limited to convex and concave upper and lower bounds. It seems straight forward to extend this work to the case of more general RTC-conformant event curves. Instead of full synchronization one simply requires the synchronization with a dedicated subset of upper or lower automata, allowing, enforcing resp. bursts after interval lengths of dedicated sizes.

Prolongation of valid behaviors: The strategy from above may have non-obvious side-effects: A valid stream can become invalid, i.e. once very few events have been seen in the past only the generation of sufficiently many events s.t. the lower bound is not violate will lead to a violation of the upper bound. Within our implementation such a situation will lead to a deadlock state, where the latter is marked as "Error"-state. How such situations can be avoided is currently investigated by our partners at Verimag [16]. However in our proposed implementation this situation can simply be resolved by excluding traces leading to such states from the model checking.

Re-transformation of TA-based behaviors into RTC-conformant curves As illustrated above the approach of [12] makes use of observer automata for doing this. This procedure is in principle applicable in our setting. However, this may lead to inefficiencies and it remains unclear when to terminate with the observations. It might be therefore useful to develop an alternate strategy, which gives one an approximated, but non-finite RTC-conformant output curve.

Benchmarking of the developed pattern Our partners at Verimag are currently investigating a back and forth translation of RTC-curves into models of the synchronous language Lustre and look into a state-dependent server model, which can not be investigate by the means of RTC.

(1) Contrary to our approach the Lustre-based methodology is based on a discrete time semantics. Where we modeled curves with networks of upper and lower TA, our partners developed a procedure based on observers. These observers allow to model input curves as well as to approximate the resp. RTC-conformant curves modelling the outgoing event streams. It seems quite useful to compare this method with our TA-based framework, not only for expressiveness, but also when it comes to the scalability of the approaches.

(2) For an analysis of the state-based server-model our partner at Verimag wish to fed RTC-conformant curves into the state-dependent server module and subsequently deriving the resp. outgoing arrival and service curves from the over all model [14]. To do the back and forth conversion of curves, the authors of [14] mainly follow the approach of [12] as illustrated above. As it turned out, the system does not nicely scale with respect to the size of the RTC-conformant input curves. To overcome this situation, our partners try to change the granularity of the model, but still being capable to compute RTC-conformant output streams at the lowest level of granularity, namely on the level of single events. In this respect, it seems highly interesting to investigate the scalability of our approach in such a setting, in particular when it comes to the modelling of long-term behaviors, which often exhibit regularity.

5.2.2 COMBEST CCS Case Study

Fig. 8 shows parts of a communication system. This system consists of a server, a Ethernet connection, a wireless network and a hub to a sensor network. The dashed lines represent service curves, the solid lines arrival curves. For a possible scenario to be analyzed, we made the following assumptions:

- The server consists of three service units: (a) a personal communication service, which handles Internet connection via wireless, (b) a database service which logs the sensor nodes data into a SQL database and (c) a watch dog emitting alarms if sensor node data is out of tolerance.

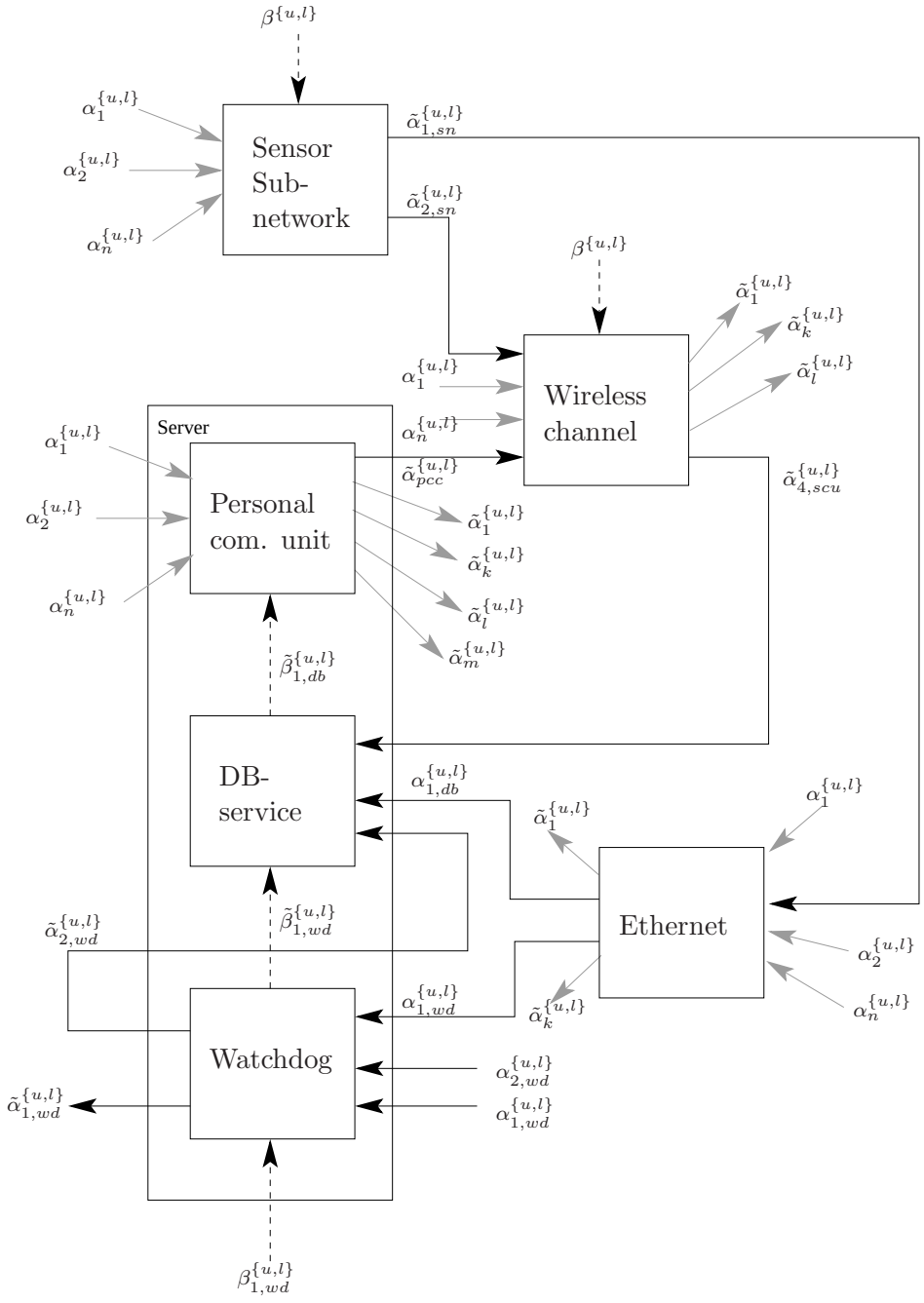


Figure 8: Heterogeneous communication system

- For the server we assume a fixed priority, where the watch dog thread has the highest priority, the database thread a medium and the thread for handling personal communication possesses the lowest priority.
- The sensor subnetwork consists of a number of nodes. However the submitted data is pre-processed in a super-ordinated hub. This hub sends data of low-priorities via the wireless network, whereas high priority data is send via Ethernet.

The complexity of the MAC-Layer protocol as induced for example by collision resolution requires now a detailed modelling. To do so we stress the usage of TA for modelling the Ethernet and wireless network units of the above model. As pointed out in this report, our approach is now capable of triggering model behavior within TA-based models, where the timed behavior of streams of stimuli are given as RTC-conformant curves. However, this is the first step only, analyzing an overall model requires to re-transform timing behavior of the TA-based submodels into curves of the RTC.

References

[1] Yasmina Abdeddaïm and Oded Maler. Preemptive job-shop scheduling using stopwatch automata. In Joost-Pieter Katoen and Perdita Stevens, editors, *Proc. of the 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *LNCS*, pages 113–126. Springer, 2002.

[2] Rajeev Alur and David L. Dill. Automata For Modeling Real-Time Systems. In Mike Paterson, editor, *Proc. of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer, 1990.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[4] Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling heterogeneous real-time components in bip. In *Pro. of the 4'th IEEE Int. Conf. on Software Engineering and Formal Methods (SEFM 2006)*, pages 3–12. IEEE Computer Society, 2006.

[5] G. Behrmann, A. Fehnker, T.S. Hune, K.G. Larsen, P. Pettersson, J.M.T. Romijn, and F.W. Vaandrager. Minimum-Cost Reachability

for Priced Timed Automata. Technical Report CSI-R0109, Computing Science Institute, Catholic University of Nijmegen, March 2001.

[6] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

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

[8] Samarjit Chakraborty, Simon Künzli, and Lothar Thiele. A General Framework for Analyzing System Properties in Platform-Based Embedded System Designs. *Design, Automation and Test in Europe Conference and Exhibition*, 1, 2003.

[9] Samarjit Chakraborty, Linh T. X. Phan, and P. S. Thiagarajan. Event count automata: A state-based model for stream processing systems. In *Proc. of the 26th IEEE International Real-Time Systems Symposium (RTSS'05)*, pages 87–98, 2005.

[10] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[11] Martijn Hendriks and Marcel Verhoef. Timed automata based analysis of embedded system architectures. In *Proc. of the 20th Int. Parallel and Distributed Processing Symposium (IPDPS 2006)*. IEEE, 2006.

[12] Pavel Krcal, Leonid Mokrushin, and Wang Yi. A tool for compositional analysis of timed systems by abstraction (extended abstract). In *Proc. of NWPT07, the 19th Nordic Workshop on Programming Theory*, October 2007. Slides of the talk with more tech. details can be found at: http://www.it.uu.se/research/docs/fm/apv/tapves/2007-02-08-abstract/leo-tapves.ppt.

[13] Simon Künzli and Lothar Thiele. Generating event traces based on arrival curves. In *Proc. of the 13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2006)*, pages 81–98. VDE Verlag, 2006.

[14] Y. Liu, K. Altisen, and M. Moy. Granularity based Interfacing between RTC and Timed Automata Performance Models. Technical Report TR-2008-xx, Verimag, Centre Équation, 38610 Gières, December 2008. http://www-verimag.imag.fr/index.php?page=techrep-list, perliminary version attached to appendix of Combest report 2008.

[15] P. Merlin and D.J. Faber. Recoverability of communication protocols. *IEEE Trans. on Communications*, 24(9), September 1976.

[16] M. Moy and K. Altisen. Connecting Real-Time Calculus to the Synchronous Programming Language Lustre. Technical Report TR-2008-xx, Verimag, Centre Équation, 38610 Gières, December 2008. http://www-verimag.imag.fr/index.php?page=techrep-list, perliminary version attached to appendix of Combest report 2008.

[17] Simon Perathoner, Ernesto Wandeler, and Lothar Thiele. Evaluation and comparison of performance analysis methods for distributed embedded systems. Technical Report 276, Computer Engineering and Networks Laboratory, ETH Zurich, 2006.

[18] Linh T. X. Phan, Samarjit Chakraborty, P. S. Thiagarajan, and Lothar Thiele. Composing functional and state-based performance models for analyzing heterogeneous real-time systems. In *Proc. of the 28th IEEE Real-Time Systems Symposium (RTSS 2007)*, pages 343–352. IEEE Computer Society, 2007.

[19] Kai Richter, Marek Jersak, and Rolf Ernst. A formal approach to mpsoc performance verification. *IEEE Computer*, 36(4):60–67, 2003.

[20] M. Sorea. Case Study on Distributed Heterogeneous Communication Systems (HCS). Technical report, November 2008. http://www.combest.eu/home/?link=Application1 (public); http://www.combest.eu/home/intranet/2008/MeetingReports/EADS-case-study-description-March-2008-V2.pdf, (confidential).

[21] Ingo Stierand, Henning Dierks, and Alexander Metzner. Combining timed automata based formal specifications and real-time scheduling analysis. Reports of SFB/TR 14 AVACS 18, SFB/TR 14 AVACS, June 2007.