Verifying Weakly-Hard Real-Time Properties of Traffic Streams in Switched Networks

L. Ahrendts[1], S. Quinton[2], T. Boroske[3], and R. Ernst[4]

1 **Institute for Network and Computer Engineering, TU Braunschweig, Braunschweig, Germany**
ahrendts@ida.ing.tu-bs.de
2 **Inria Grenoble Rhône-Alpes, Montbonnot, France**
sophie.quinton@inria.fr
3 **Institute for Network and Computer Engineering, TU Braunschweig, Braunschweig, Germany**
thomasb@ida.ing.tu-bs.de
4 **Institute for Network and Computer Engineering, TU Braunschweig, Braunschweig, Germany**
ernst@ida.ing.tu-bs.de

Abstract

In this paper, we introduce the first verification method which is able to provide weakly-hard real-time guarantees for tasks and task chains in systems with multiple resources under partitioned scheduling with fixed priorities. Existing weakly-hard real-time verification techniques are restricted today to systems with a single resource. A weakly-hard real-time guarantee specifies an upper bound on the maximum number m of deadline misses of a task in a sequence of k consecutive executions. Such a guarantee is useful if a task can experience a bounded number of deadline misses without impacting the system mission. We present our verification method in the context of switched networks with traffic streams between nodes, and demonstrate its practical applicability in an automotive case study.

1998 ACM Subject Classification C.3 Computer Systems Organization – Special-purpose and application-based systems – Real-time and embedded systems, D.2.4 Software – Software Engineering – Software/Program Verification – Formal methods

Keywords and phrases Embedded and cyber-physical systems, weakly-hard real-time systems and networks, timing analysis

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Modern embedded systems often have a distributed hardware platform, where the individual processing resources are linked by data buses or switched networks. A software application, which is mapped to such a platform, consists of a set of communicating tasks and has often to provide results within a limited response time. Timely communication between sender and receiver tasks is therefore a critical aspect in design and verification. In this paper, we concentrate on the timing behavior of traffic streams in switched networks like Switched Ethernet. By traffic stream we understand an infinite sequence of data transmissions between a sender and a receiver node of the network.

If the classical hard real-time paradigm is applied to a traffic stream, then the duration of a data transmission over the network must not violate a given end-to-end deadline. However, with increasing functionality and growing bandwidth demand of data transmission in modern embedded systems in the automotive or industrial domain, it becomes more and more difficult to fulfill the end-to-end deadlines of all traffic streams in unfavorable scheduling scenarios. A promising option is the shift to the weakly-hard real-time paradigm [1] which relaxes these timing requirements. Here a traffic stream is feasible from a timing perspective, if it does not exceed a certain budget of end-to-end deadline misses. For instance, a traffic stream may not miss more than m end-to-end deadlines in any k consecutive transmissions. The traffic stream is said to be (m, k)-constrained.

The practical justification of weakly-hard real-time paradigm in the context of communication builds on the observed robustness of many real-time software systems. In the field of image processing, a late transmission may result in a skipped frame. Given that the number and distribution of frame skips is appropriately bounded, it will not be noticeable to the human eye. In the field of control, an end-to-end deadline miss may cause the calculation of the control law to fail at time instant k so that no new control input is sent to the actuator at this instant. Several works could show that under given (m, k)-constraints the required control performance could be maintained [15] [9] [8] . Blind et al. [2] could show stability in the classical sense of Lyapunov for a networked control system, where the network is unreliable in the (m,k)-sense.

So far, verification techniques have been developed which allow to derive (m, k)-guarantees for tasks which are executed on a system with a *single* service-providing resource. A switched network, however, comprises *several* service-providing resources as detailed in Section 2. In this paper, we therefore provide a compositional verification method which is able to provide (m,k)-guarantees for multi-resource problems. The main challenge in extending an existing (m,k)-verification method to the multi-resource setting is to deal with inter-resource dependencies. Our approach builds on both

1. *Compositional Performance Analysis (CPA).* CPA [11] is a compositional framework to verify classical hard real-time properties, e.g., worst case response times. It deals with inter-resource dependencies by the formulation of a fixed-point problem.
2. *Typical Worst Case Analysis (TWCA)* TWCA [21] is one of the existing (m,k)-verification techniques for single resource systems.

We adapt and extend CPA and TWCA, calling the resulting procedure *TypicalCPA*. The paper is structured as follows. We begin by defining our system model, and then introduce the CPA approach. We continue by explaining the basic principle of TWCA, and reason how CPA and TWCA can be coupled. Finally, we perform and discuss experiments. An overview of related work is given before the conclusion.

2 Network Model

The system model represents a real-time network setting with unicast, multicast and broadcast streams and is depicted in Figure 1. The scope of the model includes, for instance, Switched Ethernet but is not limited to it. The main components of the network model are switches and nodes. A pair of nodes may communicate by sending frames over the network which are forwarded by the switches using appropriate output ports. The service of output ports for frame transmission is scarce and has to be arbitrated according to a static priority non-preemptive (SPNP) scheduling policy. The output ports therefore represent the service-providing resources R_k in the system [6].

An infinite sequence of frames between a source node and 1 [a subset of resp. all] destination node(s) is called a unicast [multicast resp. broadcast] stream. A unicast [multicast resp. broadcast] stream s_i is modeled as a linear [forked] chain of N tasks, where each task represents a hop in the route and is mapped to the output port of the respective switch. We call the set of N tasks contained in the stream s_i $\mathcal{T}_{s_i} = \{\tau_{i,1}, \tau_{i,2}, \dots \tau_{i,N}\}$ and define the respective precedence constraints, e.g. for a unicast stream as $\tau_{i,1} \prec \tau_{i,2} \prec \dots \prec \tau_{i,N}$. The first task in the stream s_i, is activated by an external event source. All successor tasks are activated by the termination events of their respective predecessor task in the chain. Each task $\tau_{i,j}$ in stream s_i has a non-unique priority p_i. The best case execution time (BCET) resp. worst case execution time (WCET) of task $\tau_{i,j}$, denoted as $C^-_{i,j}$ resp. $C^+_{i,j}$, represents the minimum resp. maximum frame delay in the switch plus the constant wire transmission time, and is independent of other traffic in the network. Dynamic delays resulting from contention at the switch output ports are considered in the response time computation of tasks. The maximum response time of a task $\tau_{i,j}$ is constrained by the relative deadline $d_{i,j}$, while the maximum network traversal time w.r.t. a stream s_i should not exceed the end-to-end deadline $D_i = \sum_j d_{i,j}$.

We describe the occurrence of activation events over time w.r.t. a task $\tau_{i,j}$ by the concept of event flows as well as by minimum and maximum event models.

▶ **Definition 1** (Event flow). An event flow $e_{i,j}(t)$ is a function which returns the number of events which activate task $\tau_{i,j}$ within the time interval $[0, t)$ in a given execution run.

▶ **Definition 2** (Event model). The minimum and maximum event models $\eta^-_{i,j}(\Delta t)$ and $\eta^+_{i,j}(\Delta t)$ indicate a lower and upper bound, respectively, on the number of activation events for task $\tau_{i,j}$ in any time interval $[t, t + \Delta t)$. Any event flow $e_{i,j}(t)$ of task $\tau_{i,j}$ is therefore constrained by

$$\forall t_1, t_2 : t_1 \leq t_2 : \eta^-_{i,j}(t_2 - t_1) \leq e_{i,j}(t_2) - e_{i,j}(t_1) \leq \eta^+_{i,j}(t_2 - t_1).$$

If convenient, we also use the pseudo-inverses of event models, i.e., the event distance functions. The event distance function $\delta^-_{i,j}(n)$ $[\delta^+_{i,j}(n)]$ is the pseudo-inverse of event model $\eta^+_{i,j}(\Delta t)$ $[\eta^-_{i,j}(\Delta t)]$.

▶ **Definition 3** (Event distance functions). The minimum and maximum distance $\delta^-_{i,j}(n)$ and $\delta^+_{i,j}(n)$ indicate a lower and upper bound, respectively, on the temporal distance between the first and the last event of a sequence of n activation events for task $\tau_{i,j}$. For the special case $n \in \{0, 1\}$, the definition $\delta^-_{i,j}(n) = \delta^+_{i,j}(n) = 0$ applies.

3 Compositional Performance Analysis

CPA [11] is a verification framework which derives lower and upper bounds on the timing properties of distributed real-time software systems with partitioned scheduling. Computed timing properties include in particular the best case response times (BCRTs) and worst case response times (WCRTs) of tasks. CPA is implemented in Python as pyCPA [4], the basic libraries of pyCPA are available on-line [5]. The CPA method breaks the verification problem down into a set of local, i.e. resource-related, analysis problems. A subsequent analysis step then relates the local verification problems such that inter-resource dependencies are taken into account and a global fixed point problem is formulated.

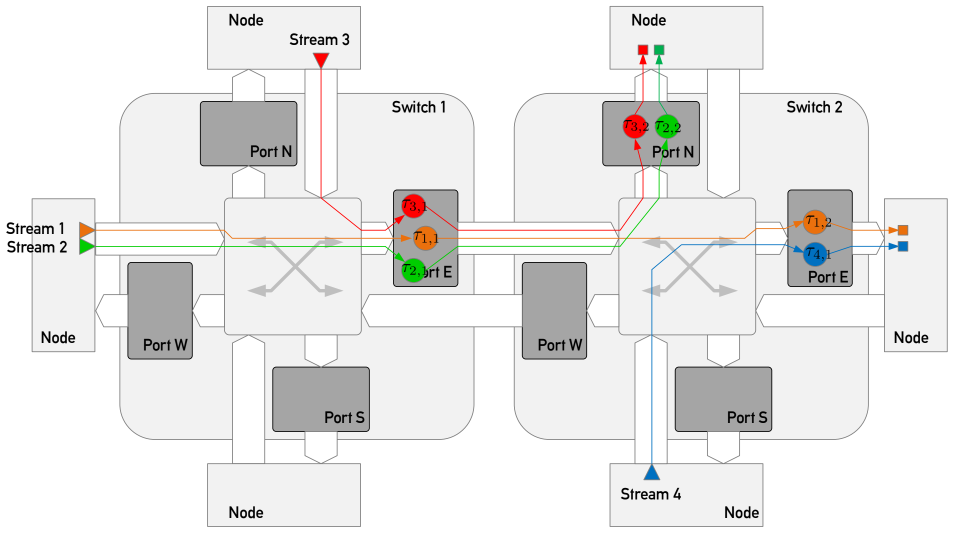


Figure 1 Network model. *The figure illustrates a network with six nodes and two switches. The output ports of a switch are named after the points of the compass. Four exemplary unicast streams are represented.*

▶ **Definition 4** (Attributes local & global). The attribute «local» refers to parameters, properties etc. of a specific resource R_k and the associated (mapped) task set $\mathcal{T}_{R_k}$.
The attribute «global» refers, on the contrary, to parameters, properties etc. of the processing platform $\mathcal{P} = \bigcup_{\forall k} R_k$ and the entire task set $\mathcal{T} = \bigcup_{\forall k} \mathcal{T}_{R_k}$.

3.1 Local Analysis

The local analysis focuses on the isolated resource R_k and derives the timing properties of the associated task set $\mathcal{T}_k$. The analysis objective is in particular to compute (a) the BCRT and WCRT for each task $\tau_{i,j} \in \mathcal{T}_k$, and (b) the output event model of each task $\tau_{i,j} \in \mathcal{T}_k$.

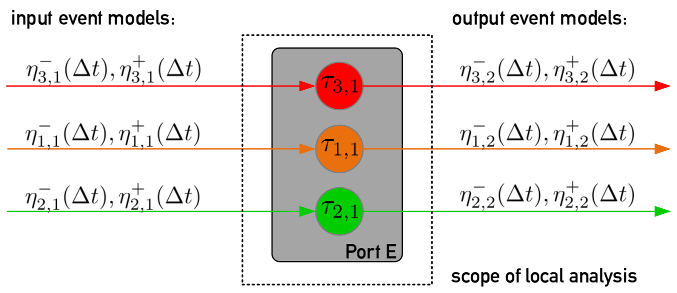

input event models:
output event models:
$\eta^-_{3,1}(\Delta t), \eta^+_{3,1}(\Delta t)$
$\eta^-_{1,1}(\Delta t), \eta^+_{1,1}(\Delta t)$
$\eta^-_{2,1}(\Delta t), \eta^+_{2,1}(\Delta t)$
$\tau_{3,1}$
$\tau_{1,1}$
$\tau_{2,1}$
Port E
$\eta^-_{3,2}(\Delta t), \eta^+_{3,2}(\Delta t)$
$\eta^-_{1,2}(\Delta t), \eta^+_{1,2}(\Delta t)$
$\eta^-_{2,2}(\Delta t), \eta^+_{2,2}(\Delta t)$
scope of local analysis

Figure 2 Scope and interface of the local CPA. *The figure shows as an example the output port E of switch 1 with mapped tasks.*

3.1.1 Computation of Response Times

In the following, we very briefly sketch the response time analysis for a task $\tau_{i,j}$ which is mapped to an SPNP-scheduled resource R_k. For a detailed presentation, please refer to [7]. To find the WCRT of task $\tau_{i,j}$, a scheduling scenario has to be known which induces the longest response time of task $\tau_{i,j}$. This worst case scenario is often called the *maximum level-$\tau_{i,j}$ busy period*. It is known to start if $\tau_{i,j} \cup hsp(\tau_{i,j})$[1]are activated synchronously and a task in $lp(\tau_{i,j})$, which has just been activated before, causes the maximum blocking delay [3]. It closes as soon as the resource becomes idle w.r.t. $\tau_{i,j}$ and $hsp(\tau_{i,j})$-tasks. The processing behavior of task $\tau_{i,j}$ within the maximum level-$\tau_{i,j}$ busy period can be described by the so called *multiple event busy times* $B^+_{i,j}(q)$.

▶ **Definition 5.** The maximum q-event busy time $B^+_{i,j}(q)$ indicates the processing time of q consecutive activation events of task $\tau_{i,j}$ within the maximum level-$\tau_{i,j}$ busy period. $B^+_{i,j}(q)$ always starts with the beginning of maximum level-$\tau_{i,j}$ busy period [17].

The busy times $B^+_{i,j}(q)$ depend on the input event models and WCETs of the tasks $\mathcal{T}_{R_k}$. It has been shown that the WCRT $R^+_{i,j}$ of task $\tau_{i,j}$ is among the response times of the maximum level-$\tau_{i,j}$ busy period, such that we can write

$$R^+_{i,j} = \max_{1 \le q \le K_{i,j}} \left\{ B^+_{i,j}(q) - \delta^-_{i,j}(q) \right\} \tag{1}$$

where $K_{i,j}$ is the maximum number of jobs of task $\tau_{i,j}$ contained in the maximum level-$\tau_{i,j}$ busy period. The BCRT of task $\tau_{i,j}$ can be approximated by its BCET $R^-_{i,j} = C^-_{i,j}$.

3.1.2 Computation of Output Event Distance Functions and Output Event Models

The local analysis problems are linked because precedence relations extend over tasks on different resources as illustrated in Figure 1. According to the synchronous task chain semantics, a termination event of a task $\tau_{i,j}$ is interpreted as an activation event by the successor task $\tau_{i,j+1}$. This interaction between tasks $\tau_{i,j}$ and $\tau_{i,j+1}$ can be quantified by the distance functions $\delta^+_{i,j+1}(n)$ resp. $\delta^-_{i,j+1}(n)$ indicating the maximum resp. minimum number of distance between any n consecutive termination events of task $\tau_{i,j}$ or, equivalently, activation events of task $\tau_{i,j+1}$. Firstly, let us present safe, easy-to-interpret bounds for the distance functions with $n \ge 2$ using the *jitter method* [16]

$$\delta^-_{i,j+1}(n) \ge \max\left\{ (n-1) \cdot C^-_{i,j}, \delta^-_{i,j}(n) - J^+_{i,j} \right\} \tag{2}$$

$$\delta^+_{i,j+1}(n) \le \delta^+_{i,j}(n) + J^+_{i,j} \tag{3}$$

Eq. 2 expresses that, in the worst case, n termination events at the output of task $\tau_{i,j}$ are closer by the maximum response time jitter $J^+_{i,j} = R^+_{i,j} - R^-_{i,j}$ than n activation events at the input of the same task. Also, the density of activation events increases with every stage of the task chain due to the accumulation of response jitter. Eq. 3 describes that, in the best case, the distance of n termination events grows with every stage of a task chain by the jitter $J^+_{i,j}$. Secondly, we introduce more accurate but less intuitive bounds which have been derived in [18] (*busy window method*)

$$\delta^-_{i,j+1}(n) \ge \max\{B^-_{i,j}(n-1), \min_{1 \le q \le q^+_{i,j}} \left\{ \delta^-_{i,j}(n+q-1) - B^+_{i,j}(q) \right\} + B^-_{i,j}(1)\} \tag{4}$$

$$\delta^+_{i,j+1}(n) \le \max_{1 \le q \le q^+_{i}} \left\{ \delta^+_{i,j}(n-q+1) + B^+_{i,j}(q) \right\} - B^-_{i,j}(1)\}. \tag{5}$$

[1] We use $hsp(\tau_{i,j})$ to denote the set of tasks which have *higher or same priority* than task $\tau_{i,j} \in \mathcal{T}_k$ and are mapped to the same resource R_k. Likewise we write $lp(\tau_{i,j})$ to denote the set of tasks which have *lower priority* than task $\tau_{i,j} \in \mathcal{T}_k$ and are mapped to the same resource R_k.

According to the rules of network calculus [12], the event distance function $\delta^-_{i,j+1}(n)$ resp. $\delta^+_{i,j+1}(n)$ can even be more improved in accuracy if replaced by its superadditive closure $\bar{\delta}^-_{i,j+1}(n)$ resp. subadditive closure $\bar{\delta}^+_{i,j+1}(n)$. In the following, we continue to write $\delta^-_{i,j+1}(n)$ resp. $\delta^+_{i,j+1}(n)$ (without bar) stating explicitly when we make use of the superadditivity property ($\delta^-_{i,j+1}(m+n) \geq \delta^-_{i,j+1}(m) + \delta^-_{i,j+1}(n)$) or subadditivity property ($\delta^+_{i,j+1}(m+n) \leq \delta^+_{i,j+1}(m) + \delta^+_{i,j+1}(n)$). Note again that the output event models $\eta^+_{i,j+1}(\Delta t)$, $\eta^-_{i,j+1}(\Delta t)$ can be obtained from the output event distance functions by pseudo-inversion.

It is desirable for efficiency reasons to have a finite representation of event distance functions, meaning that it is possible to construct the event distance functions for every n on the basis of a limited number of l known points. This can be achieved by approximating $\delta^-_{i,j+1}(n)$, $\delta^+_{i,j+1}(n)$ by bounds with a repetitive behavior. The approximation is very acceptable with regard to accuracy, if the repetition period is chosen large enough. In the particular context of this paper, repetitive bounds restrict the value range that needs to be processed by the algorithm given in Theorem 23. We concentrate in the following on $\delta^-(n)$ and its pseudo-inverse $\eta^+(\Delta)$, but analogous rules can be applied to $\delta^+(n)$ and $\eta^-(\Delta)$.

▶ **Lemma 6** (Repetitive extension of an event distance function). *Given the superadditive event distance function $\delta^-(n)$ for $1 \leq n \leq l$, an l-repetitive extension $\hat{\delta}^-(n)$ is defined by*

$$\hat{\delta}^-(n) = \begin{cases} 0 & \text{for } 0 \leq n \leq 1 \\ \lfloor \frac{n-2}{l} \rfloor \cdot \delta^-(l) + \delta^-(n - \lfloor \frac{n-2}{l} \rfloor \cdot l) & \text{for } n \geq 2. \end{cases}.$$

The l-repetitive extension $\hat{\delta}^-(n)$ is a lower bound for $\delta^-(n)$, s.t. $\forall n : \hat{\delta}^-(n) \leq \delta^-(n)$.

Proof. We have $\hat{\delta}^-(n) = \delta^-(n) = 0$ for $n \in \{0, 1\}$, and $\hat{\delta}^-(n) = \delta^-(n)$ for $2 \leq n \leq l+2$. For $n > l+2$, we make use of the superadditivity property $\delta^-(n_1) + \delta^-(n_2) \leq \delta^-(n_1+n_2)$ and set $x = \lfloor \frac{n-2}{l} \rfloor$: $\hat{\delta}^-(n) = x \cdot \delta^-(l) + \delta^-(n - x \cdot l) \leq \delta^-(x \cdot l) + \delta^-(n - x \cdot l) \leq \delta^-(n)$. ◀

▶ **Lemma 7** (Repetitive extension of an event model). *Given the subadditive event model function $\eta^+(\Delta t)$, a T-repetitive extension $\hat{\eta}^+_l(\Delta t)$ is defined by*

$$\hat{\eta}^+_l(\Delta t) = \left\lfloor \frac{\Delta t}{T} \right\rfloor \cdot \eta^+(T) + \eta^+(\Delta t - \left\lfloor \frac{\Delta t}{T} \right\rfloor \cdot T).$$

If $\hat{\delta}^-(n)$ is l-repetitive, then its pseudo-inverse $\hat{\eta}^+_l(\Delta t)$ must be $T = \hat{\delta}^-(l)$-repetitive.

Proof. This results from the symmetry of function inversion. ◀

3.2 Global Analysis

The global analysis now couples the local analysis problems according to the following iterative procedure, which is also depicted in Figure 3 (box entitled "original CPA"). Firstly, each header task of a stream $\tau_{i,1}$ has a known activation behavior bounded by $\eta^-_{i,1}(\Delta t), \eta^+_{i,1}(\Delta)$ and imposed by external event sources. Since initially no event models are available for successor tasks in the stream, i.e. for $\tau_{i,j}$ with $j > 1$, they are initialized with the event model assigned to the header task $\tau_{i,1}$. The local analysis is then performed for each resource, such that response time bounds and output event models are obtained. The computed output event models are then *propagated* to the direct successor tasks, where they are interpreted as input event models. The local analysis is then repeated with the updated event models. If all propagated event models are identical to the event models used in the previous analysis run, a global fixed point is reached and the analysis terminates.

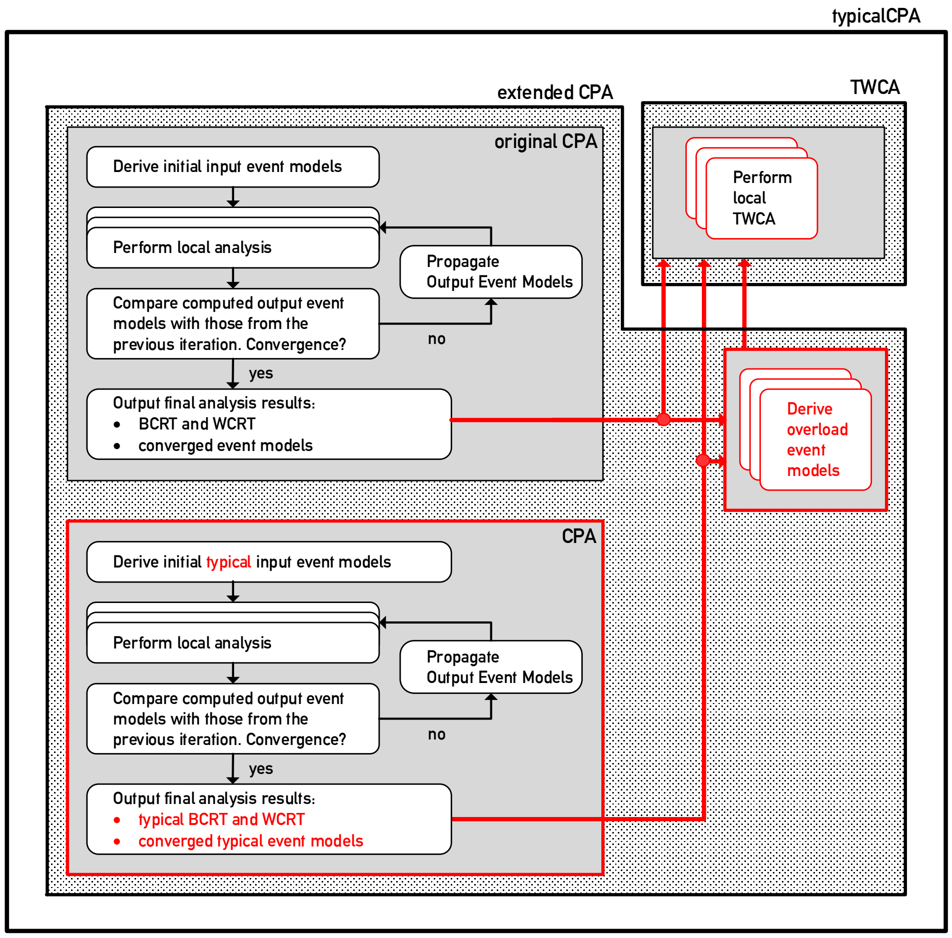

typicalCPA
extended CPA
original CPA
Derive initial input event models
Perform local analysis
Propagate Output Event Models
Compare computed output event models with those from the previous iteration. Convergence?
no
yes
Output final analysis results:
• BCRT and WCRT
• converged event models
TWCA
Perform local TWCA
Derive overload event models
CPA
Derive initial typical input event models
Perform local analysis
Propagate Output Event Models
Compare computed output event models with those from the previous iteration. Convergence?
no
yes
Output final analysis results:
• typical BCRT and WCRT
• converged typical event models

Figure 3 TypicalCPA. *The extended CPA also derives typical and overload event models as detailed in Section 5, which are then processed by a TWCA for each component.* New or adapted elements of CPA and TWCA are marked in red.

4 Typical Worst Case Analysis

Typical Worst Case Analysis (TWCA) models and analyzes systems with a single service-providing resource R_k under transient overload conditions. It provides weakly-hard real-time guarantees for tasks $\mathcal{T}_k$. In this section, we firstly present which extensions to the CPA system model presented in Section 2 are necessary to apply TWCA. Then the TWCA procedure is introduced together with a needed generalization of a schedulability criterion.

4.1 Extended System Model

The system model of CPA presented in Section 2 is a subset of the TWCA system model. The important extension of the CPA model by TWCA is that each task $\tau_{i,j}$ may be activated by events of two distinct classes, namely by *typical and overload events*. The idea is that in the exclusive presence of typical events, the task set $\mathcal{T}_k$ is schedulable. In contrast, the supplementary overload events are a potential cause for transient overload.

▶ **Definition 8** (Local typical worst case). If every task $\tau_{i,j} \in \mathcal{T}_{R_k}$ is only activated by typical

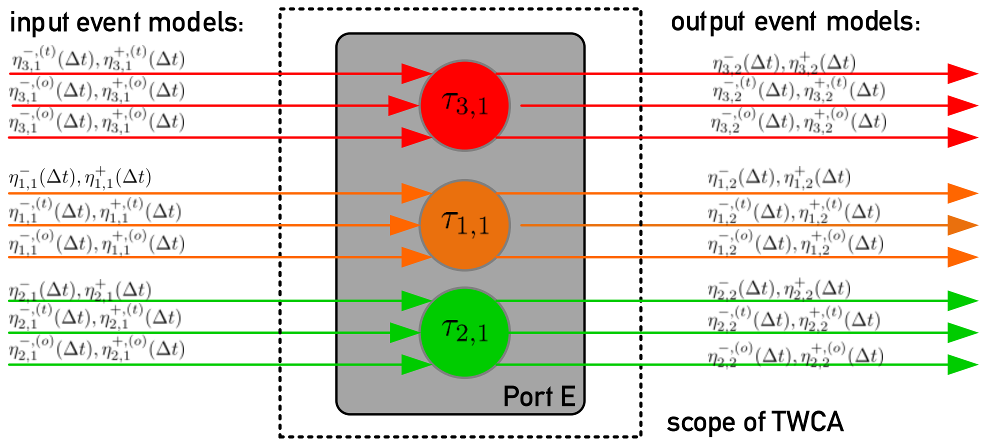

input event models:
$\eta_{3,1}^{-,(t)}(\Delta t), \eta_{3,1}^{+,(t)}(\Delta t)$
$\eta_{3,1}^{-,(o)}(\Delta t), \eta_{3,1}^{+,(o)}(\Delta t)$
$\eta_{3,1}^{-,(o)}(\Delta t), \eta_{3,1}^{+,(o)}(\Delta t)$
$\eta_{1,1}^{-}(\Delta t), \eta_{1,1}^{+}(\Delta t)$
$\eta_{1,1}^{-,(t)}(\Delta t), \eta_{1,1}^{+,(t)}(\Delta t)$
$\eta_{1,1}^{-,(o)}(\Delta t), \eta_{1,1}^{+,(o)}(\Delta t)$
$\eta_{2,1}^{-}(\Delta t), \eta_{2,1}^{+}(\Delta t)$
$\eta_{2,1}^{-,(t)}(\Delta t), \eta_{2,1}^{+,(t)}(\Delta t)$
$\eta_{2,1}^{-,(o)}(\Delta t), \eta_{2,1}^{+,(o)}(\Delta t)$
$\tau_{3,1}$
$\tau_{1,1}$
$\tau_{2,1}$
Port E
output event models:
$\eta_{3,2}^{-}(\Delta t), \eta_{3,2}^{+}(\Delta t)$
$\eta_{3,2}^{-,(t)}(\Delta t), \eta_{3,2}^{+,(t)}(\Delta t)$
$\eta_{3,2}^{-,(o)}(\Delta t), \eta_{3,2}^{+,(o)}(\Delta t)$
$\eta_{1,2}^{-}(\Delta t), \eta_{1,2}^{+}(\Delta t)$
$\eta_{1,2}^{-,(t)}(\Delta t), \eta_{1,2}^{+,(t)}(\Delta t)$
$\eta_{1,2}^{-,(o)}(\Delta t), \eta_{1,2}^{+,(o)}(\Delta t)$
$\eta_{2,2}^{-}(\Delta t), \eta_{2,2}^{+}(\Delta t)$
$\eta_{2,2}^{-,(t)}(\Delta t), \eta_{2,2}^{+,(t)}(\Delta t)$
$\eta_{2,2}^{-,(o)}(\Delta t), \eta_{2,2}^{+,(o)}(\Delta t)$
scope of TWCA

Figure 4 Scope and interface of TWCA.

events, then the task set $\mathcal{T}_{R_k}$ is schedulable even in the most unfavorable scheduling scenario (*local typical worst case*).

▶ **Definition 9** (Local worst case). If every task $\tau_{i,j} \in \mathcal{T}_{R_k}$ is activated by both typical and overload events, then in the most unfavorable scheduling scenario (*local worst case*) the task set $\mathcal{T}_{R_k}$ is possibly unschedulable.

The occurrence of typical or overload activation events over time w.r.t. a task $\tau_{i,j}$ is also modeled by the concept of event flows, while the minimum and maximum frequency of typical and overload event arrival is described by event models. The corresponding definitions are given below, while Figure 4 shows the extended system model with the additional event models.

▶ **Definition 10** (Typical and overload event flows). A typical event flow $e_{i,j}^{(t)}(t)$, resp. overload event flow $e_{i,j}^{(o)}(t)$, is a function which returns the number of typical, resp. overload, events which activate task $\tau_{i,j}$ within the time interval $[0, t)$ in a given execution run.

▶ **Definition 11** (Typical and overload event models). The event models $\eta_{i,j}^{-,(t)}(\Delta t), \eta_{i,j}^{+,(t)}(\Delta t)$, resp. $\eta_{i,j}^{-,(o)}(\Delta t), \eta_{i,j}^{+,(o)}(\Delta t)$, indicate a lower and an upper bound on the number of typical, resp. overload, events which activate task $\tau_{i,j}$ within Δt.

▶ **Definition 12** (Decomposition). Any observed event flow of task $\tau_{i,j}$ which satisfies the lower and upper bounds $\eta_{i,j}^{-}(\Delta t), \eta_{i,j}^{+}(\Delta t)$ can be partitioned in

(1) an event flow of typical termination events satisfying $\eta_{i,j}^{-,(t)}(\Delta t), \eta_{i,j}^{+,(t)}(\Delta t)$ and

(2) an event flow of overload events satisfying $\eta_{i,j}^{-,(o)}(\Delta t), \eta_{i,j}^{+,(o)}(\Delta t)$.

This implies that the maximum event model $\eta_{i,j}^{+}(\Delta t)$ is *decomposable*, s.t. $\eta_{i,j}^{+}(\Delta t) \leq \eta_{i,j}^{+,(t)}(\Delta t) + \eta_{i,j}^{+,(o)}(\Delta t)$. If $\eta_{i,j}^{+}(\Delta t) = \eta_{i,j}^{+,(t)}(\Delta t) + \eta_{i,j}^{+,(o)}(\Delta t)$ holds, then the maximum event model is said to be *exactly decomposable*. Please refer for illustration to Figure 5c.

The intuition related to the system model is that a computing platform may be designed to provide sufficient processing service for a typical workload. For instance, if all tasks have a periodic (= typical) activation pattern, then the task set is schedulable. If, however, some tasks experience additional sporadic (= overload) activations, then the task set may become unschedulable in unfavorable scheduling scenarios.

4.2 Basic Procedure

The objective of TWCA is to determine weakly-hard real-time guarantees for all tasks in the task set. More precisely, a deadline miss model (DMM) is obtained for every task $\tau_{i,j} \in \mathcal{T}_k$.

▶ **Definition 13** (Deadline miss model)**.** A deadline miss model for a task $\tau_{i,j}$ is a function $dmm_{i,j} : \mathbb{N} \to \mathbb{N}$ with the property that out of any k consecutive jobs of task $\tau_{i,j}$, at most $dmm_{i,j}(k)$ might miss their deadline $d_{i,j}$.

To compute $dmm_{i,j}(k)$ under SPNP scheduling, TWCA quantifies the impact of overload activations. We summarize the procedure in the following steps.

1. Firstly TWCA derives the maximum impact which a single overload activation of a task $\tau_{m,n} \in hsp(\tau_{i,j})$ can have on the task $\tau_{i,j}$. The impact is counted by the maximum number jobs of task $\tau_{i,j}$ which can miss their deadline due to this overload activation, and is denoted as $N_{i,j}$.
2. It is computed how many overload activations of task $\tau_{m,n}$ can at most influence the k-sequence of task $\tau_{i,j}$. This number is given by $\eta^{+,(o)}_{m,n}(\Delta T^{i,j}_k)$, where $\Delta T^{i,j}_k$ describes the maximum time interval during which a k-sequence of task $\tau_{i,j}$ is sensitive to overload events.
3. The overall impact of task $\tau_{m,n}$ is then derived as the product $N_{i,j} \cdot \eta^{+,(o)}_{m,n}(\Delta T^{i,j}_k)$.
4. Finally, the impact of all $\tau_{m,n}$ tasks which may interfere with task $\tau_{i,j}$ is summed. Interfering tasks have higher or same priority (hsp) than task $\tau_{i,j}$.

Thus we have

$$dmm_{i,j}(k) = \sum_{\tau_{m,n} \in hsp(\tau_{i,j})} N_{i,j} \cdot \eta^{+,(o)}_{m,n}(\Delta T^{i,j}_k) \tag{6}$$

where

$$N_{i,j} = \# \left\{ q \in \mathbb{N}^+ | 1 \leq q \leq K_{i,j} \wedge d_{i,j} < R^+_{i,j}(q) \right\} \tag{7}$$

$$\Delta T^{i,j}_k \leq B^+_{i,j}(K_{i,j}) + \delta^+_{i,j}(k) + (R^+_{i,j} - C^+_{i,j}) \tag{8}$$

Please refer for a detailed explanation to [10].

4.3 Improved Procedure

The presented basic TWCA assumes that *every* isolated overload activation of a task $\tau_{m,n}$ which interferes with task $\tau_{i,j}$ causes at most $N_{i,j}$ deadline misses. The approach presented in [21] improves over the basic TWCA by considering that often actually the *combined* effect of overload from several interferer tasks is required to cause a deadline miss of task $\tau_{i,j}$. We introduce therefore the following definitions.

▶ **Definition 14** (Combination)**.** A local combination $C \subseteq \mathcal{T}_{R_k}$ is a set of tasks which may experience both typical as well as overload activation events, whereas the tasks of the complementary set, $\mathcal{T}_{R_k} \setminus C$, experience only typical activation events.

▶ **Definition 15** (Unschedulable combinations)**.** $R^{+,C}_{i,j}$ denotes the longest response time of task $\tau_{i,j} \in \mathcal{T}_R$, assuming that only tasks in C experience overload activations. A combination C is said to be schedulable w.r.t. to task $\tau_{i,j}$, if $R^{+,C}_{i,j} \leq d_{i,j}$, otherwise it is unschedulable. The set of unschedulable combinations w.r.t. to task $\tau_{i,j}$ is called $\mathcal{U}_{i,j}$.

Note that special local combinations are $C = \emptyset$ and $C = \mathcal{T}_{R_k}$. In this context, $R_{i,j}^{+,\mathcal{T}_{R_k}}$ is the usual worst case response time and $R_{i,j}^{+,\emptyset}$ is called typical worst case response time.

The improved TWCA [21] is now based on the fact that the sensitivity interval $\Delta T_k^{i,j}$ of the k-sequence of task $\tau_{i,j}$ can be divided into a sequence of busy periods [13]. The timing behavior of busy periods is mutually independent, because of the idle times which separate them. Within in any such busy period, an unschedulable combination is necessary to cause at most $N_{i,j}$ deadline misses of task $\tau_{i,j}$ within this interval. A single task $\tau_{m,n}$ can be part of unschedulable combinations at most $\Omega_{m,n} = \eta_{m,n}^{+,(o)}(\Delta T_k^{i,j})$ times, which corresponds to the maximum number of overload activations in $\Delta T_k^{i,j}$.

Let $x_C \in \mathbb{N}$ count the number of busy periods in $\Delta T_k^{i,j}$, which suffer from an unschedulable combination $C \in \mathcal{U}_{i,j}$. Then the DMM can be obtained by solving the following optimization problem

$$dmm_{i,j}(k) = \max\ N_{i,j} \sum_{C\colon C \in \mathcal{U}_{i,j}} x_C \tag{9}$$

$$\text{s.t.} \sum_{C,(m,n)} x_C \leq \Omega_{m,n} \tag{10}$$

$$\text{with } C,(m,n) : (\tau_{m,n} \in hsp(\tau_{i,j}) \cup \tau_{i,j}) \wedge (\tau_{m,n} \in C) \wedge (C \in \mathcal{U}_{i,j})$$

To determine whether a combination C is schedulable or not, a fast schedulability criterion is required. We rely on the criterion presented in [21], but generalize it for (1) non-unique priorities, and (2) the general relation where the maximum event models are not exactly decomposable. The generalization is presented in Theorem 16; notation and explanations of the theorem contents are given in the corresponding proof and Figure 5.

▶ **Theorem 16** (Generalized schedulability criterion). *Equation 11 formulates a schedulability criterion for task $\tau_{i,j}$ under a given combination C.*

$$\forall l \in K_{i,j} : \sum_{\forall \tau_{m,n} : \tau_{m,n} \in hsp(\tau_{i,j}) \cup \tau_{i,j} \wedge \tau_{m,n} \notin C} wl_{over}^{(m,n),l} \geq \Lambda_{i,j}^l - \Gamma_{i,j}^l. \tag{11}$$

The following abbreviations are used

$$\Lambda_{i,j}^l = B_{i,j}^+(l) - \delta_{i,j}^-(l) - d_{i,j}$$

$$\Gamma_{i,j}^l = \sum_{\tau_{m,n} \in hp(\tau_{i,j})} C_{m,n}^+ \cdot [\eta_{m,n}^+(B_{i,j}^+(l) - C_{i,j}^+) - \eta_{m,n}^+(\Delta t_{i,j}^l)]$$

$$\Delta t_{i,j}^l = \delta_{i,j}^-(l) + d_{i,j} - C_{i,j}^+$$

$$wl_{over}^{(m,n),l} = \begin{cases} C_{m,n}^+ \cdot \left(\eta_{m,n}^+(\Delta t_{i,j}^l) - \eta_{m,n}^{+,(t)}(\Delta t_{i,j}^l)\right) & \textit{for } \tau_{m,n} \in hp(\tau_{i,j}) \\ C_{m,n}^+ \cdot \left(\eta_{m,n}^{+]}(\delta_{i,j}^-(l)) - \eta_{m,n}^{+],(t)}(\delta_{i,j}^-(l))\right) & \textit{for } \tau_{m,n} \in sp(\tau_{i,j}) \cup \tau_{i,j} \end{cases}$$

Proof. Let us verify the schedulability of task $\tau_{i,j}$ under a given combination C, i.e. we verify whether $R_{i,j}^{+,C} \leq d_{i,j}$ is true. We start from the unschedulable local worst case with $C' = \mathcal{T}_R$, which is represented by the maximum level-$\tau_{i,j}$ busy period which contains $K_{i,j}$ jobs of task $\tau_{i,j}$ (cf. Figure 5a). If the task $\tau_{i,j}$ is schedulable in the local worst case, then it schedulable for every combination and the problem is solved. If, however, task $\tau_{i,j}$ is unschedulable in the local worst case, then some of the $K_{i,j}$ jobs of task $\tau_{i,j}$ miss their deadline. The lth job of $\tau_{i,j}$ exceeds its deadline in the local worst case by (cf. also Figure 5)

$$\Lambda_{i,j}^l = R_{i,j}^+(l) - d_{i,j} = B_{i,j}^+(l) - \delta_{i,j}^-(l) - d_{i,j}.$$

If its deadline is enforced by removing overload, an amount of workload $\Gamma^l_{i,j}$ will disappear automatically. Namely the workload from interfering activations which occur after the deadline but before the non-preemptive execution of the lth job. Jobs of tasks with the same priority (sp) as $\tau_{i,j}$ do not contribute to $\Gamma^l_{i,j}$, because they influence the response time of the lth job only if they have arrived earlier than or simultaneously with this job.

$$\Gamma^l_{i,j} = \sum_{\tau_{m,n} \in hp(\tau_{i,j})} C^+_{m,n} \cdot [\eta^+_{m,n}(B^+_{i,j}(l) - C^+_{i,j}) - \eta^+_{m,n}(\delta^-_{i,j}(l) + d_{i,j} - C^+_{i,j})]$$

The RHS of inequality 11 describes the smallest amount of overload of interfering tasks that needs removed for sufficient schedulability of the lth job of $\tau_{i,j}$ in the maximum busy period. The LHS of Eq. 11 describes how much overload is removed compared to the local worst case, if we assume combination C (cf. Figure 5b for $C = \emptyset$). Under combination C, all tasks $\tau_{m,n} \notin C$ experience only typical activations and their overload is not present. In other words, the tasks $\tau_{m,n} \notin C$ follow their event model $\eta^{+,(t)}_{m,n}(\Delta t)$. In particular, an amount of overload per task $\tau_{m,n}$

$$wl^{(m,n),l}_{over} = \begin{cases} C^+_{i,j} \cdot \left(\eta^+_{m,n}(\Delta t^l_{i,j}) - \eta^{+,(t)}_{m,n}(\Delta t^l_{i,j})\right) & \text{for } \tau_{m,n} \in hp(\tau_{i,j}) \\ C^+_{i,j} \cdot \left(\eta^{+]}_{m,n}(\delta^-_i(l)) - \eta^{+],(t)}_{m,n}(\delta^-_i(l))\right) & \text{for } \tau_{m,n} \in sp(\tau_{i,j}) \cup \tau_{i,j} \end{cases}$$

is removed which impacts the response time of the lth job of task $\tau_{i,j}$. Namely, the interfering overload of $hp(\tau_{i,j})$-tasks until the timely nonpreemptive execution of job $\tau_{i,j}(l)$ is absent. Likewise, the overload of all $sp(\tau_{i,j})$-jobs and overload jobs of $\tau_{i,j}$ are absent, which interfere if they arrive before or simultaneously with job $\tau_{i,j}(l)$.[2] ◀

5 Typical Compositional Performance Analysis

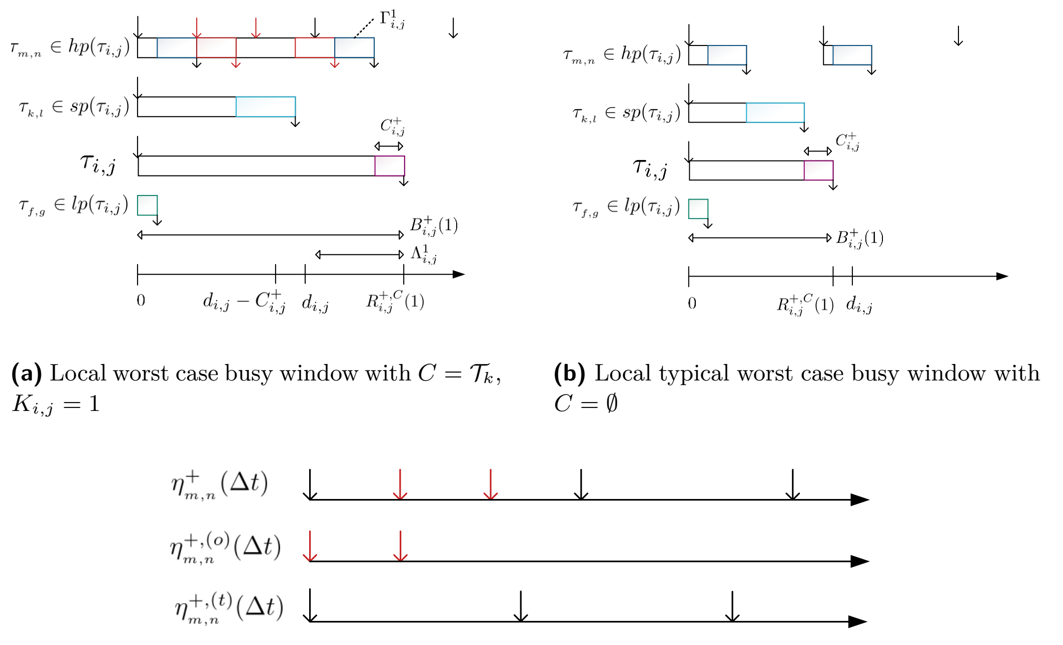
The new framework TypicalCPA, which we develop in this paper, combines CPA and TWCA such that weakly-hard real-time guarantees can be given for tasks in a multi-resource system. More concretely, the local analysis method TWCA will performed for each component after an extended CPA has terminated. This is illustrated in Figure 3. To apply TWCA as a local analysis method, for each task minimum and maximum event models together with the corresponding minimum and maximum typical and overload event models have to be provided. The state-of-the-art CPA, however, computes as a result, besides BCRT and WCRT, so far only the converged minimum and maximum event models of each task (not their typical and overload variants) and thus has to be extended.

In the following we assume that the complete set of event models – $(\eta^-_{i,1}(\Delta t), \eta^+_{i,1}(\Delta t))$, $(\eta^{-,(t)}_{i,1}(\Delta t), \eta^{+,(t)}_{i,1}(\Delta t))$ and $(\eta^{-,(o)}_{i,1}(\Delta t), \eta^{+,(o)}_{i,1}(\Delta t))$ – is given for the header tasks $\tau_{i,1}$, since they are activated by external event sources. The problem to be addressed is how to derive these event models for all successor tasks in the context of CPA such that they can be used for the subsequent TWCA.

5.1 Basic Definitions

We begin by introducing the concept of a *global combination* describing the activation behavior of each task $\tau_{i,j}$ contained in the global task set $\mathcal{T}$. Due to the existing precedence constraints

[2] The notation $\eta^{+]}(\Delta t)$ expresses that the maximum event model refers to the *closed* time interval $[0, t]$.



$\tau_{m,n} \in hp(\tau_{i,j})$
$\tau_{k,l} \in sp(\tau_{i,j})$
$\tau_{i,j}$
$\tau_{f,g} \in lp(\tau_{i,j})$
$\Gamma^1_{i,j}$
$C^+_{i,j}$
$B^+_{i,j}(1)$
$\Lambda^1_{i,j}$
0
$d_{i,j} - C^+_{i,j}$
$d_{i,j}$
$R^{+,C}_{i,j}(1)$
$\eta^+_{m,n}(\Delta t)$
$\eta^{+,(o)}_{m,n}(\Delta t)$
$\eta^{+,(t)}_{m,n}(\Delta t)$

(a) Local worst case busy window with $C = \mathcal{T}_k$, $K_{i,j} = 1$

(b) Local typical worst case busy window with $C = \emptyset$

(c) Exemplary decomposition of the maximum event model $\eta^+_{m,n}(\Delta t)$ in the maximum overload event model $\eta^{+,(o)}_{m,n}(\Delta t)$ and the maximum typical event model $\eta^{+,(t)}_{m,n}(\Delta t)$ for task $\tau_{m,n}$

Figure 5 Theorem 16: Generalized schedulability criterion

in a stream s_i, the activation behavior of any task $\tau_{i,j}$ with $j > 1$ is fully determined by the respective predecessor task and therefore in the end by the header task $\tau_{i,1}$. It is thus sufficient to include the activation behavior of the header tasks in the definition of a global combination.

▶ **Definition 17** (Global combination). A global combination $C_g \subseteq \{\tau_{i,1} | \forall i : \tau_{i,1} \in \mathcal{T}\}$ is a set of header tasks which may experience both typical as well as overload activations. All other header tasks follow their typical event model.

Special global combinations are the *global typical combination* with $C_g = \emptyset$, and the *global worst case combination* with $C_g = \{\tau_{i,1} | \forall i : \tau_{i,1} \in \mathcal{T}\}$.

▶ **Definition 18** (Schedulability of a global combination). We say a global combination C_g is schedulable if and only if under all possible scheduling scenarios (1) all streams can satisfy their end-to-end deadlines $D_{i,j}$ and (2) every task meets its local deadline $d_{i,j}$.

We require that the given event models of the header tasks are such that the following schedulability constraints are respected.

▶ **Definition 19** (Global typical worst case). If the system behaves according to the global typical combination, then the task set $\mathcal{T}$ is schedulable even in the most unfavorable scenario (global typical worst case).

▶ **Definition 20** (Global worst case). If the system behaves according to the global worst case combination, the task set $\mathcal{T}$ is possibly unschedulable in the most unfavorable scenario (global worst case).

We would like to mention that for computing weakly-hard real-time guarantees, naturally only systems which *are* unschedulable in the global worst case are of interest.

5.2 Computation of Minimum and Maximum Event Models

While for the header tasks the minimum and maximum event model $\eta^-_{i,j}(\Delta t)$, $\eta^+_{i,j}(\Delta t)$ is given by the system specification, it has to be derived for successor tasks $\tau_{i,j}$ with $j > 1$. The classical CPA is capable of deriving these event models for all successor tasks from the original CPA input model as defined in Section 2. Thus CPA explores here the most favorable and the most unfavorable behavior of the global worst case combination.

5.3 Computation of Minimum and Maximum Typical Event Models

The minimum and maximum typical event model $\eta^{-,(t)}_{i,j}(\Delta t)$, $\eta^{+,(t)}_{i,j}(\Delta t)$ have also to be computed for the successor tasks $\tau_{i,j}$ with $j > 1$. Our claim is that CPA can also be used for this purpose, given that in the input model the worst case bounds $\eta^-_{i,1}(\Delta t)$, $\eta^+_{i,1}(\Delta t)$ are replaced by the typical event models $\eta^{-,(t)}_{i,1}(\Delta t)$, $\eta^{+,(t)}_{i,1}(\Delta t)$. In other words, CPA is now applied for the best case and worst case scenario where all header tasks see only typical events (global typical combination). CPA, which is agnostic of event types, computes the converged minimum and maximum event models for all stream tasks. We assume in this paper that all typical events that are injected at the head of a stream keep their typical nature while propagating through the system. Knowing that only typical events have served for stream activation, we can interpret the CPA-derived event models as typical and have thus $\eta^{-,(t)}_{i,j}(\Delta t)$ and $\eta^{+,(t)}_{i,j}(\Delta t)$ for all stream tasks.

5.4 Computation of Minimum and Maximum Overload Event Models

Finally, our intention is to obtain the minimum and maximum overload event models for each successor task $\tau_{i,j}$ with $j > 1$. We begin by describing how an arbitrary event flow $e_{i,j}(t)$ can be decomposed in a typical event flow $e^{(t)}_{i,j}(t)$ and an overload event flow $e^{(o)}_{i,j}(t)$. In this context, we use the concept of a sliding window function which returns a maximum event model for a specific event flow.

▶ **Definition 21** (Sliding window function). A sliding window function f_{slw} takes a specific event flow $e_{i,j}(t)$ of task $\tau_{i,j}$ defined on $0 \le t \le T$ as an input, and returns a maximum event model for $e_{i,j}(t)$, denoted as $\eta^+_{e_{i,j},T}(\Delta t)$ for any interval size $0 \le \Delta t \le T$. This maximum event model $\eta^+_{e_{i,j},T}(\Delta t)$ is derived by passing a window of size Δt over the event flow $e_{i,j}(t)$ of length T and noting down the maximum number events contained in any position of the window Δt such that

$$\eta^+_{e_{i,j},T}(\Delta t) = \max_{t_1,t_2\,:\,0\le t_1\le t_2\le T\wedge t_2-t_1=\Delta t} \left\{e_{i,j}(t_2) - e_{i,j}(t_1)\right\}.$$

▶ **Theorem 22** (Decomposition of an event flow). *Let $e_{i,j}(t)$ be an arbitrary event flow of length T belonging to task $\tau_{i,j}$. Known bounds for the activation frequency of task $\tau_{i,j}$ are i.a. $\eta^+_{e_{i,j},t}(\Delta t)$ for all (sub)lengths of the event flow with $0 \le t \le T$ and the maximum typical event model $\eta^{+,(t)}_{i,j}(\Delta t)$. A valid decomposition of $e_{i,j}(t)$ in a typical and overload event flow is given by*

$$e^{(o)}_{i,j}(t) = \max_{0\le\Delta t\le t} \left\{0,\ \eta^+_{e_{i,j},t}(\Delta t) - \eta^{+,(t)}_{i,j}(\Delta t)\right\} \qquad e^{(t)}_{i,j}(t) = e_{i,j}(t) - e^{(o)}_{i,j}(t). \qquad (12)$$

Proof. The event flow $e_{i,j}(t)$ cannot contain more than $\eta_{i,j}^{+;(t)}(\Delta t)$ typical events in the observed interval $[0,t)$ by Def. 11, where $\Delta t = t - 0$. All events that occur additionally to the maximum number of typical events $\eta_{i,j}^{+;(t)}(\Delta t)$ in $[0,t)$ are a potential source of overload in the system and can therefore be safely interpreted as overload events.

To determine the number overload events in $e_{i,j}(t)$, we (1) apply the sliding window function to $e_{i,j}(t)$ within $[0,t)$ which results in $\eta^{+}_{e_{i,j},t}(\Delta t)$, and then (2) compare pointwise $\eta^{+}_{e_{i,j},t}(\Delta t)$ with $\eta_{i,j}^{+;(t)}(\Delta t)$. Pointwise comparison is done chronologically by increasing continuously the size of Δt with $0 \leq \Delta t \leq t$. The largest nonnegative difference $\max\limits_{0\leq\Delta t\leq t}\left\{0,\ \eta^{+}_{e_{i,j},t}(\Delta t) - \eta_{i,j}^{+;(t)}(\Delta t)\right\}$, is the number of overload events in $e_{i,j}(t)$.

Why is it not sufficient to compute $\max\left\{0, \eta^{+}_{e_{i,j},t}(\Delta t) - \eta_{i,j}^{+;(t)}(\Delta t)\right\}$ for $\Delta t = t$? Let $\Delta t'$ be the first interval, where the maximum budget of typical events is exceeded by the event flow such that $\eta^{+}_{e_{i,j},t}(\Delta t') - \eta_{i,j}^{+;(t)}(\Delta t') > 0$. This information should not be contradicted by a later smaller value of overload events derived at $\Delta t'' > \Delta t'$. This, however, may happen due to the cumulative representation of event arrival within Δt by event models, where information on the alignment of events gets lost with increasing interval size. The alignment information is however important to distinguish overload from typical events. The formulation $e_{i,j}^{(o)}(t) = \max\limits_{0\leq\Delta t^*\leq t}\left\{0,\ \eta^{+}_{e_{i,j},t}(\Delta t^*) - \eta_{i,j}^{+;(t)}(\Delta t^*)\right\}$ preserves the information on the maximum number of overload events once gained at Δt^*. Also, $e_{i,j}^{(o)}(t)$ is a wide-sense increasing function which accumulates the number of occurred overload events over time, and therefore satisfies Def. 10 of an event flow. Furthermore, we have $e_{i,j}^{(t)}(t) = e_{i,j}(t) - e_{i,j}^{(o)}(t)$ since an event in an event flow can either be overload or typical. ◀

In the following Theorem 23, we state how to compute a maximum overload event model. We would like to note that the minimum overload event model is the zero function $\eta_{i,j}^{+;(o)}(\Delta t) = 0$ since overload events can be completely absent cf. global typical combination.

▶ **Theorem 23** (Obtaining an overload event model). *A maximum overload event model is*

$$\eta_{i,j}^{+;(o)}(\Delta t) = f_{slw}\left(\max_{0\leq\Delta t^*\leq\Delta t}\left\{\eta^{+}_{i,j}(\Delta t^*) - \eta_{i,j}^{+;(t)}(\Delta t^*)\right\}\right)$$

where f_{slw} is a sliding window function.

Proof. An upper bound for all event flow-specific maximum event models $\eta^{+}_{e_{i,j},T}(\Delta t)$ of task $\tau_{i,j}$ is the maximum event model $\eta^{+}_{i,j}(\Delta t)$ by Def. 2. Thus we have

$$\max_{0\leq\Delta t^*\leq t}\left\{0,\ \eta^{+}_{e_{i,j},t}(\Delta t^*) - \eta_{i,j}^{+;(t)}(\Delta t^*)\right\} \leq \max_{0\leq\Delta t^*\leq\Delta t}\left\{\eta^{+}_{i,j}(\Delta t^*) - \eta_{i,j}^{+;(t)}(\Delta t^*)\right\}.$$

In other words, the overload event flow $\tilde{e}_{i,j}^{(o)}(t) = \max\limits_{0\leq\Delta t^*\leq t}\left\{\eta^{+}_{i,j}(\Delta t^*) - \eta_{i,j}^{+;(t)}(\Delta t^*)\right\}$ is always larger than any other arbitrary overload event flow $e_{i,j}^{(o)}(t)$. To derive from the largest overload event flow $\tilde{e}_{i,j}^{(o)}(t)$ the corresponding maximum overload event model, we apply once again the sliding window function such that $\tilde{e}_{i,j}^{(o)}(t_2) - \tilde{e}_{i,j}^{(o)}(t_1) \leq \eta_{i,j}^{+;(o)}(t_2 - t_1) = f_{slw}\left(\tilde{e}_{i,j}^{(o)}(t_2 - t_1)\right)$. The computation of the overload event model $\eta_{i,j}^{+;(o)}(\Delta t)$ is illustrated in Figure 6. ◀

Calculating a maximum overload event model according to Theorem 23 requires a high computational effort since the sliding window approach has to be applied to the infinitely long event flow $\tilde{e}_{i,j}^{(o)}(t) = \max\limits_{0\leq\Delta t^*\leq t}\left\{\eta^{+}_{i,j}(\Delta t^*) - \eta_{i,j}^{+;(t)}(\Delta t^*)\right\}$. Fortunately most event flows

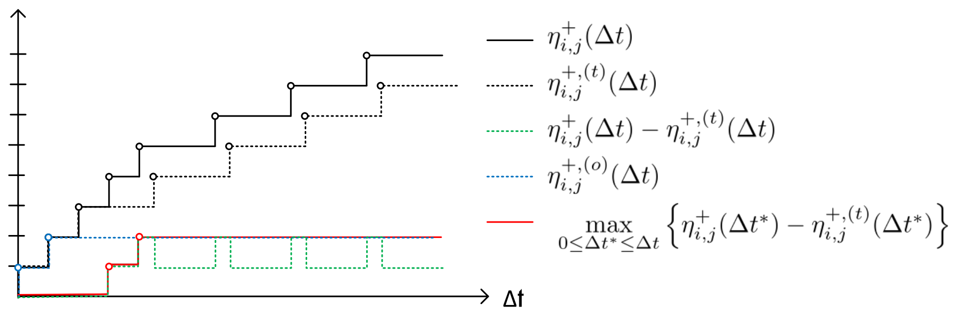


Figure 6 Computing a maximum overload event model

have a repetitive behavior or can be approximated by repetitive functions, so that the effort to derive overload event models is significantly reduced. In the following, we discuss special and practically relevant cases for the computation of overload event models.

▶ Case 1 (Zero typical event model). In this trivial but important case, the task $\tau_{i,j}$ has a zero typical event model $\eta^{+,(t)}_{i,j}(\Delta t) = 0$. Obviously, we have $\eta^{+,(o)}_{i,j}(\Delta t) = \eta^+_{i,j}(\Delta t)$. This is case is relevant for header tasks, which have the character of a sporadic interferer.

▶ Case 2 (Zero overload event model). In a second trivial but important case, the worst case and typical worst case event model of task $\tau_{i,j}$ such that $\eta^+_{i,j}(\Delta t) = \eta^{+,(t)}_{i,j}(\Delta t)$. Consequently, we have a zero overload event model $\eta^{+,(o)}_{i,j}(\Delta t) = 0$. Header tasks with a periodic activation have often this behavior.

▶ Case 3 (Repetitive overload event flow). If the overload event flow $\tilde{e}^{(o)}_{i,j}(t)$ is T-repetitive possibly with an offset (cf. Lemma 7), then applying the sliding window algorithm can be restricted to the interval $[0, 2T)$ to construct the maximum overload event model. In the following Theorem 24, we show that a T-repetitive overload event flow is obtained if the event model $\eta^+_{i,j}(\Delta t)$ and the typical event model $\eta^{+,(t)}_{i,j}(\Delta t)$ are both T-repetitive extensions (which can be achieved by appropriate output model computation described Section 3.1.2).

▶ Theorem 24 (Repetitive overload event flow). *If the event model $\eta^+(\Delta t)$ and the typical event model $\eta^{+,(t)}(\Delta t)$ are both T-repetitive extensions, then the resulting overload event flow $\tilde{e}^{(o)}_{i,j}(t)$ is likewise T-repetitive, such that*

$$\tilde{e}^{(o)}_{i,j}(t) = \max_{0\le\Delta t^*\le t}\{\left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot \left(\eta^+(T) - \eta^{+,(t)}(T)\right) + \eta^+(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor T) - \eta^{+,(t)}(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor T)\}.$$

Proof.

$$\begin{aligned}
&\max_{0\le\Delta t^*\le\Delta t}\{\eta^+(\Delta t^*) - \eta^{+,(t)}(\Delta t^*)\} \\
&= \max_{0\le\Delta t^*\le\Delta t}\{\left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot \eta^+(T) + \eta^+(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot T) - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot \eta^{+,(t)}(T) - \eta^{+,(t)}(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot T)\} \\
&= \max_{0\le\Delta t^*\le\Delta t}\{\left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot \left(\eta^+(T) - \eta^{+,(t)}(T)\right) + \eta^+(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor T) - \eta^{+,(t)}(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor T)\} \\
&= \max_{0\le\Delta t^*\le\Delta t}\{\left\lfloor\frac{\Delta t^*}{T}\right\rfloor \cdot \eta^{diff}(T) + \eta^{diff}(\Delta t^* - \left\lfloor\frac{\Delta t^*}{T}\right\rfloor T)\}
\end{aligned}$$

where $\eta^{diff}(\Delta t) = \eta^+(\Delta t) - \eta^{+,(t)}(\Delta t)$.

◀

6 Experiments

The presented experiments focus on computing end-to-end (m, k)-guarantees for traffic streams in realistic network settings, while exploring how a varying amount of overload impacts the timing behavior of the investigated system.

6.1 System Generation

The case study presented in Thiele et al. [20] provides characteristics of future automotive backbone networks by Daimler. Based on this data, we have randomly generated a set of automotive switched Ethernet networks with mapped traffic streams. Firstly, let us present the data used from the case study. Figure 7 illustrates three possible network topologies. The topologies vary in the number of switches (SWs) which interconnect 8 electronic control units (ECUs). Links operate at 100 Mbit/s, only ECU0 and ECU7 are equipped with 1Gbit/s links due to high load. Stream characteristics are described statistically by [20], they are summarized in Table 1a. There are 50 periodic control streams of highest priority and 4 periodic camera streams of lower priority. Control streams have relatively small payloads and rather long periods, while camera streams have large payloads and shorter periods. Some of the streams are unicast, others are multicast or broadcast. A periodically sent Ethernet frame is mapped to exactly one stream. Information on the frame payload as well as on periods is given by [20] only in form of minimum and maximum values, averages, and quartiles for the purpose of data anonymization. In case of camera traffic, the number of streams is too small for quantifying quartiles. IPv4/UDP is used at the network/transport layer, which adds 28 bytes of protocol overhead (not shown in Table 1a). Furthermore, the communication matrix in Table 1b is given by [20] indicating the number of control and camera streams sent between a tuple of nodes. We use a parser to translate the network described in terms of topology and streams into a CPA/TWCA system model as defined in Sections 2 and 4.1.

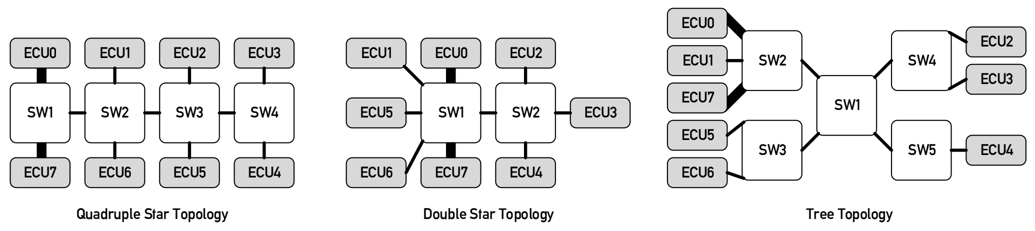


Figure 7 Network topologies. Thin lines represent links at 100 Mbit/s, while thick lines represent links at 1 Gbit/s. A maximum wire length of 10 m is assumed, which translates to a maximum wire propagation delay of 33 ns.

Secondly, we describe the random generation of systems which conform to the presented properties. The generation process is designed to produce a configurable number of systems and consists of several runs. A single generation run first creates the set of 54 streams with their respective source and destination ECUs, and then the streams are mapped to each of the three topologies. A run thus creates 3 systems at once. However, this set of 3 systems is discarded if at least one is not schedulable to enable meaningful comparisons between the different topologies.

- *Generation of control streams.* Periods and payloads of control streams are only described by statistic figures. Therefore, we used fitting to find distributions which come closest

	CONTROL	CAMERA
STREAMS		
# total	50	4
# unicast	26	3
# 2-cast	13	1
# 3-cast	4	0
# 4-cast	1	0
# broadcast	6	0
FRAME PAYLOAD IN BYTES		
[min, max]	$[1, 250]$ B	$[875, 1400]$ B
average	54 B	1231 B
quartiles	$q_{0.25} = 8$ B, $q_{0.50} = 25$ B, $q_{0.75} = 74$ B	
PERIOD		
[min, max]	$[5ms, 1s]$	$[100us, 1ms]$
average	$182ms$	$440us$
quartiles	$q_{0.25} = 10ms$, $q_{0.50} = 40ms$, $q_{0.75} = 175ms$	

(a) Stream characteristics

Src/Dst	ECU0	ECU1	ECU2	ECU3	ECU4	ECU5	ECU6	ECU7
ECU0		1			1, 1		10	2, 2
ECU1	1							1, 1
ECU2					5			
ECU3					1			
ECU4	1	2	3	3		1	1	1
ECU5							3	2
ECU6	10	6	4	3	4	3		10
ECU7	5	2	2	2	4, 1	3	8	

(b) Communication matrix indicating the number of control streams (black number, 1st entry) and camera streams (blue number, 2nd entry) between a pair of ECUs.

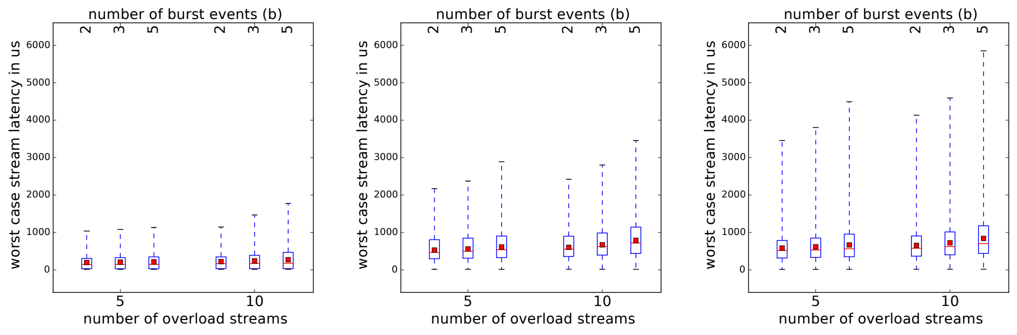
Table 1 Traffic properties as given in Thiele et al. [20]

the indicated average and quartiles. For the periods, we opted for a Weibull distribution with the parameters *shape* $= 0.54$ and *scale* $= 88.09$. For the payload, an exponential distribution with $\lambda = 0.02$ was used.

- *Generation of camera streams.* The few, i.e. 4, camera streams $s_{cam,i}$ are assigned the same payloads and periods in each system generation run: $s_{cam,0} \mapsto (100\mu s, 875B)$; $s_{cam,1} \mapsto (1ms, 1400B)$; $s_{cam,2} \mapsto (330\mu s, 1325B)$; $s_{cam,3} \mapsto (330\mu s, 1325B)$.
- *Generation of stream sources & destinations and topology mapping.* The given communication matrix defines constraints on pairs of source-destination ECUs and on the number of streams sent between them. Stream sources & destinations are generated randomly respecting these constraints. The traffic is then mapped to each of the 3 topologies, creating 3 different systems with identical streams.
- *Schedulability test.* For control streams, local deadlines are set to the stream period and the end-to-end deadline is the sum of the local deadlines. For camera streams, we choose arbitrarily an end-to-end deadline of $2ms$ ($s_{cam,0}$, $s_{cam,2}$, $s_{cam,3}$) or $4ms$ ($s_{cam,1}$), such that – without any overload in the system – worst case stream latencys (WCSLs) of camera streams are already close to their end-to-end deadlines.[3] Local camera deadlines are derived by uniform distribution of the end-to-end deadline. Based on these timing constraints, the generated systems are filtered such that they are all schedulable as mentioned above.

After a generation run, we dispose of a set of 3 systems in which no overload is present. We then add sporadic control streams to each system as transient overload. We see this as a realistic extension of the system description, representing event-triggered communication. A sporadic control stream s' is a duplicate of a randomly chosen control stream s from the original stream set but with modified activation behavior. The typical activation behavior

[3] The worst case stream latency (WCSL) for a unicast stream is computed by summing the WCRTs of tasks included in the stream. For multi- or broadcast streams, the WCSLs are computed separately for each path from the source to a destination.



(a) Double star topology **(b)** Tree topology **(c)** Quadruple star topology

Figure 8 Worst case latencies of control and camera streams under varying topologies and overload. Each single box plot is based on the streams of 50 randomly generated systems with the indicated properties (num. of bursts, num. of overload streams).

of s' is zero, while the nonzero overload activation behavior is modeled as sporadically bursty [16]: A burst of b events with a minimum distance T_{in} is repeated after an outer period T_{out} such that $\eta^{+,(o)}(\Delta t) = \left\lfloor \frac{\Delta t}{T_{out}} \right\rfloor \cdot b + \min\left\{ \left\lceil \frac{\Delta t - \lfloor \frac{\Delta t}{T_{out}} \rfloor \cdot T_{out}}{T_{in}} \right\rceil, b \right\}$. While the burst length b is used as a variable parameter in the experiments, fixed parameters are $T_{in} = 100\mu s$ and T_{out} is 10-times the period of the original stream s.

6.2 Experimental Results

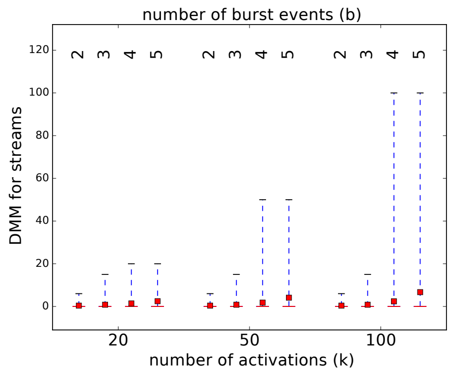
In the experiments, we investigate the impact of overload on the timing behavior of the generated systems. For each presented overload configuration, we randomly generated *50* systems of the same topology. We first present worst case stream latencys (WCSLs), and then discuss the DMMs computed for streams. The results in this section are presented in box plots as, for instance, in Figure 8a. This is done to summarize results (WCSLs or DMMs) over all streams from a set of similar systems. A single box plot indicates the average (red square) and the quartiles $q_{0.25}$, $q_{0.50}$, $q_{0.75}$ of the results. The 1st and 3rd quartiles $q_{0.25}$ and $q_{0.75}$ are the top and the bottom of the blue framed box, while the red band inside the box is the 2nd quartile (median). The whiskers indicate results outside the quartiles.

Worst Case Stream Latencies. The WCSLs depend both on the system characteristics as well as on the amount of introduced overload. Figure 8 shows that the double star topology has the shortest WCSLs, compared to to the tree topology with intermediate WCSLs and the quadruple star topology with even higher WCSLs. This behavior is due to the varying number and extent of contention points in the different topologies. Moreover, Figure 8 confirms the intuition that WCSLs increase with the amount of overload in the system, which is controlled by the number of overload streams in the system and the number of burst events b of each overload stream.

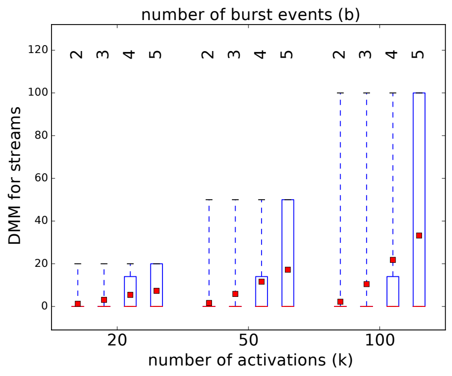
Deadline Miss Models of Streams. While the control streams satisfy their end-to-end deadlines even in the presence of overload, camera streams suffer from occasional deadline misses in particular in case of the quadruple star topology. A deadline miss in the context of a camera stream can be interpreted as a frame loss which impacts then video quality. We therefore focus on the DMMs of the camera streams. Figure 9 illustrates the DMMs for all camera streams of generated systems with quadruple star topology. Overload is varied by the number of overload streams and the burst length. We compute the DMM of a unicast stream as the sum of the task DMMs included in the stream. In the case that one or more local deadlines are violated but the global deadline is satisfied, the stream DMM is set to zero. Multicast and broadcast streams are decomposed into unicast streams in order to compute the DMMs according to the above rule. Figure 9a indicates DMMs for camera streams in the presence of 5 sporadic overload streams, while Figure 9b shows DMMs for an increased number of 10 sporadic overload streams. Table 2 lists the nonzero DMMs results for $k = 100$ to get a more detailed impression of the individual weakly-hard real-time guarantees. The number of deadline misses grows as expected with the number of overload streams. Furthermore, the m-k-ratio is improving for growing k.

- For 5 overload streams many camera streams are schedulable for any burst length. Few systems have camera streams that are not schedulable. Among these systems with late camera streams, most of them have a very acceptable (m, k) behavior – in particular for $b \in \{2, 3\}$.
- For 10 overload streams more camera streams experience occasional deadline misses. For $b \in \{2, 3\}$, the maximum number of deadline misses m in k executions is acceptable for many camera streams depending on system requirements. For $b \geq 4$ many of the investigated systems are clearly overloaded.

A note on run times: On a PC with an Intel i5-4210M processor at 2.6 GHz and 8GB RAM, the analysis of a single system is in the order of 15-30 seconds.



(a) 5 overload streams



(b) 10 overload streams

Figure 9 DMMs for camera streams under varying overload for the quadruple star topology. *Results evaluate camera streams in 50 systems. For a multicast camera stream with n destinations, there n end-to-end DMMs computed.*

7 Related Work

The seminal paper by Bernat et al. [1] has presented the principles of weakly-hard real-time systems. It summarizes existing work in a similar direction, introduces (m,k)-constraints, and derives (m,k)-guarantees for periodic task sets with known offsets under fixed priority scheduling. More powerful verification techniques for weakly-hard real-time systems have been subsequently developed. In particular, Quinton et al. [14] has introduced a method called TWCA, which can handle more comprehensive system models covering, e.g., arbitrary activation event models. The initial work [14] has been extended and refined in a sequence of publications; the latest analysis version is presented in [21]. A new and recent development

bursts	nonzero $\mathbf{dmm(100)}$ results with number of occurrence n in brackets (n)
5 overload streams	
$b = 2$	**2**(6), **3**(5), **4**(15), **6**(2)
$b = 3$	**2**(6), **3**(5), **4**(22), **5**(1), **6**(1), **7**(6), **9**(1), **13**(1), **15**(1)
$b = 4$	**2**(2), **3**(3), **4**(19), **5**(4), **6**(1), **7**(1), **8**(2), **9**(1), **10**(3), **11**(2), **12**(1), **13**(1), **14**(1), **16**(1), **18**(1), **30**(1), **100**(3)
10 overload streams	
$b = 2$	**2**(2), **3**(2), **4**(31), **5**(1), **6**(6), **11**(2), **12**(1), **14**(1), **16**(1), **19**(1), **100**(3)
$b = 3$	**4**(15), **5**(5), **7**(2), **8**(1), **9**(1), **10**(2), **12**(5), **14**(2), **15**(4), **16**(2), **100**(23)
$b = 4$	**4**(2), **5**(1), **8**(1), **9**(3), **10**(1), **13**(2), **14**(10), **16**(1), **20**(2), **21**(2), **30**(1), **100**(51)

Table 2 Details on nonzero DMM results for camera streams for $k = 100$

is the verification technique for weakly-hard real-time systems presented by Sun et al. [19]. The work by Sun et al. [19] has only a limited focus on systems with fully periodic tasks with unknown offsets under fixed priority scheduling, but it has a higher accuracy than TWCA since it provides exact results. However, all of the verification techniques are restricted to systems with a single service-providing resource. In this paper, we lift this restriction by integrating TWCA as local analysis technique in the context of the CPA framework [11]. CPA is an established compositional analysis framework, which uses for each component a dedicated scheduling analysis and specifies the coupling of the component-based results. The advantage of using a compositional analysis framework is that large and heterogeneous systems can be analyzed. The choice of the combination (TWCA, CPA) is due to the similarities in the system models and interface definitions, which reduces the number of compatibility issues.

8 Conclusion

In this paper, we presented TypicalCPA which is the first verification method for weakly-hard real-time systems with *multiple* resources and we evaluated it in a network context with traffic streams. Previous verification techniques providing weakly-hard real-time guarantees have aimed at systems with only a single service-providing resource. The method builds on (1) CPA, a compositional performance verification framework for hard real-time guarantees, and (2) TWCA, an analysis method which derives weakly-hard real-time guarantees for systems with a single resource. CPA allows to use different local scheduling analysis techniques for each component in the investigated system, and defines a coupling mechanism between the results provided by each component analysis. We have interpreted TWCA as such a local scheduling analysis technique, but we had to extend (1) elements of TWCA as well as (2) the existing coupling mechanism to achieve compatibility of both CPA and TWCA. In particular, the computation and propagation of typical and overload event models between tasks on different resources has been introduced. In an industrial case study, focusing on automotive switched Ethernet networks, we demonstrated the applicability of TypicalCPA to realistic problems. In the future, we intend to work on improved accuracy of the provided weakly-hard real-time guarantees.

References

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

2 Rainer Blind and Frank Allgöwer. Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 7510–7515. IEEE, 2015.

3 Robert I Davis, Alan Burns, Reinder J Bril, and Johan J Lukkien. Controller area network (can) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

4 Jonas Diemer, Philip Axer, and Rolf Ernst. Compositional performance analysis in python with pycpa. *Proc. of WATERS*, 2012.

5 Jonas Diemer, Philip Axer, Daniel Thiele, and Johannes Schlatow. pyCPA. URL: `https://pycpa.readthedocs.io/en/latest/index.html#`.

6 Jonas Diemer, Jonas Rox, and Rolf Ernst. Modeling of ethernet avb networks for worst-case timing analysis. *IFAC Proceedings Volumes*, 45(2):848–853, 2012.

7 Jonas Diemer, Daniel Thiele, and Rolf Ernst. Formal worst-case timing analysis of ethernet topologies with strict-priority and avb switching. In *Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on*, pages 1–10. IEEE, 2012.

8 Goran Frehse, Arne Hamann, Sophie Quinton, and Matthias Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *Real-Time Systems Symposium (RTSS), 2014 IEEE*, pages 53–62. IEEE, 2014.

9 Mongi Ben Gaid, Daniel Simon, and Olivier Sename. A design methodology for weakly-hard real-time control. *IFAC Proceedings Volumes*, 41(2):10258–10264, 2008.

10 Zain AH Hammadeh, Sophie Quinton, and Rolf Ernst. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *Proceedings of the 14th International Conference on Embedded Software*, page 10. ACM, 2014.

11 Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis–the symta/s approach. *IEE Proceedings-Computers and Digital Techniques*, 152(2):148–166, 2005.

12 Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.

13 John P Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Real-Time Systems Symposium, 1990. Proceedings., 11th*, pages 201–209. IEEE, 1990.

14 Sophie Quinton, Matthias Hanke, and Rolf Ernst. Formal analysis of sporadic overload in real-time systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 515–520. EDA Consortium, 2012.

15 Parameswaran Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):549–559, 1999.

16 Kai Richter. *Compositional Scheduling Analysis Using Standard Event Models*. PhD thesis, TU Braunschweig, IDA, 2005.

17 Simon Schliecker, Jonas Rox, Matthias Ivers, and Rolf Ernst. Providing accurate event models for the analysis of heterogeneous multiprocessor systems. In *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*, pages 185–190. ACM, 2008.

18 Simon Schliecker, Jonas Rox, Matthias Ivers, and Rolf Ernst. Providing accurate event models for the analysis of heterogeneous multiprocessor systems. In *Intern. Conference on HW/SW Codesign and System Synthesis. Proceedings*, pages 185–190, New York, 2008. ACM.

19 Youcheng Sun and Marco Di Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):171, 2017.

20 Daniel Thiele, Philip Axer, Rolf Ernst, and Jan R Seyler. Improving formal timing analysis of switched ethernet by exploiting traffic stream correlations. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, page 15. ACM, 2014.

21 Wenbo Xu, Zain AH Hammadeh, Alexander Kröller, Rolf Ernst, and Sophie Quinton. Improved deadline miss models for real-time systems using typical worst-case analysis. In *Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on*, pages 247–256. IEEE, 2015.