Modeling of Ethernet AVB Networks for Worst-Case Timing Analysis

Jonas Diemer * **Jonas Rox** * **Rolf Ernst** *

* *Institute of Computer and Network Engineering, Technische Universität Braunschweig, Germany (e-mail: {diemer|rox|ernst}@ida.ing.tu-bs.de).*

Abstract: Ethernet is currently explored as the upcoming network standard for distributed control applications in many different industries such as automotive, avionics and industrial automation. It offers higher performance and flexibility over traditional control bus systems such as CAN and ProfiBus. For distributed control applications, predictable communication timing is highly important which can be problematic using standard Ethernet. The new Ethernet AVB standard aims to improve this by a new scheduling algorithm based on traffic shaping. However, the current AVB standard lacks a formal timing guarantee which is important for safety-critical control applications. As a solution to this, we present a model for Ethernet AVB networks and a transformation into a timing analysis model. Based on the timing model, we apply a compositional performance analysis approach known from the analysis of distributed real-time systems to derive worst-case timing properties and hence timing guarantees of the original Ethernet AVB network. For this, we provide the required formalism for the analysis of the scheduling of Ethernet AVB.

Keywords: Communication Networks, Ethernet, Performance analysis, Timing analysis, Formal verification, System models, Mathematical models, Industry automation, Automotive control

1. INTRODUCTION

The use of Ethernet for distributed control applications is currently investigated in many industries, such as automotive, avionics and industrial automation. A major expected benefit is the flexibility of Ethernet as an open standard with no tight bounds to a specific supplier. Having the same network infrastructure in automation and offices, Ethernet also simplifies installation and maintenance and integration into corporate IT infrastructures. At the same time, Ethernet offers high data rates at an extremely low price point compared to domain-specific protocols like CAN, FlexRay etc.

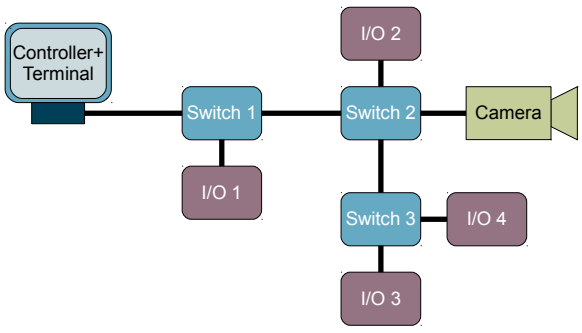

Fig. 1. Example for a small Ethernet Network in Industrial Automation

Figure 1 shows an example of a simple industrial automation network. Multiple nodes with sensors (e.g. light barrier, pressure, temperature) and actuators (e.g. conveyor belt motors, soldering robots) are connected via Ethernet switches to a central controller. The sensors periodically send their measurement data to the controller which processes the data and then sends control messages to the actuators. The controller also implements a terminal station for the human-machine-interface, which includes observation data received from a camera connected via Ethernet.

A major challenge when using Ethernet in the industrial domains is the predictable timing of data transfers. Many control applications in the automation industry require periodic transferal of sensor and control messages with low latency and jitter in order to guarantee correct functionality of the control loop. Different solutions exist that tackle this problem on the network protocol layer such as PROFINET (http://www.profinet.com/), EtherCAT (http://www.ethercat.org), TT-Ethernet (SAE (2011)), or AFDX (ARINC (2009)). However, these solutions require either special hardware media access controllers (MACs) in the nodes, special switches or both, which is problematic in terms of cost and availability. Here, the use of the Ethernet IEEE 802.1 (2011) standard for audio-video bridging (AVB) is a promising alternative which was originally conceived to facilitate transfer of real-time audio and video streams for studio applications. Being an official Ethernet standard, it is likely that AVB-capable switch integrated circuits (ICs) will be available at very high quantities and hence very low price compared to the industry-specific solutions mentioned above. This motivates the evaluation of the usability of Ethernet AVB for other domains such as industrial automation.

Ethernet AVB augments classical Ethernet with capabilities of reserving bandwidth to streams of certain classes. These are enforced using a credit-based shaping algorithm (CBSA), which is based on priorities and adds a traffic shaper to every output port for the corresponding classes. With this, higher

priorities can no longer starve lower priorities and the allocated bandwidth can be guaranteed (per class). However, the current Ethernet AVB standard does not provide formulas for worst-case latencies that cover all corner cases. To provide accurate upper bounds on latency and hence formal guarantees, a formal compositional performance analysis (CPA) approach can be used which is based on the schedulability and timing analysis of distributed computing systems as described in e.g. Henia et al. (2005). To this end, this paper presents a model transformation to transform an Ethernet AVB model into a timing analysis model. For this model, we provide a formal timing analysis approach to obtain timing guarantees for Ethernet transfers.

The remainder of the paper is organized as follows: Section 2 provides some background and related work about Ethernet AVB and formal timing analysis. In Section 3, we will describe the modeling of Ethernet AVB systems and how such models can be translated into timing analysis models. Section 4 will present the analysis of worst-case timing properties of the obtained analysis model in theory, whereas Section 5 evaluates this analysis approach for an example network.

2. BACKGROUND AND RELATED WORK

In this paper, we present a worst-case analysis of Ethernet AVB networks based on a model transformation into a timing-analysis model. A similar approach has been shown for regular Ethernet in Rox and Ernst (2010) and for networks-on-chip in e.g. Shi and Burns (2008) and Diemer et al. (2011). A study of the timing properties of Ethernet AVB has been presented by Imtiaz et al. (2009). In this study, however, only per-class timings were obtained instead of those for individual streams. In the following subsections, we will provide further background on Ethernet AVB and formal timing analysis.

2.1 Ethernet AVB

Ethernet AVB is described in several standards of IEEE 802.1. Among other things, the standard defines a new scheduling mechanism for output port arbitration which is based on prioritized traffic classes (like regular Ethernet QoS) but adds traffic shaping to certain traffic classes as shown in Figure 2(a). As with classical Ethernet, each traffic class uses dedicated queues so scheduling within a class follows a FIFO order. The traffic shapers limit the number of frames which are transferred per time on a specific traffic class in order to leave enough room for low-priority traffic. According to the 802.1Qav standard, there are at least (and usually only) two classes to which traffic shaping is applied, called stream reservation (SR) classes A and B. The traffic shaping is implemented using credits which are replenished at a constant rate (the so-called *idleSlope*) and consumed at the rate allowed by the port transmission rate (the *sendSlope*) when data is transferred.

The right side of Figure 2(b) shows an example of the transfer of two frames ('1' and '2') on class A with an interfering non-real-time frame. The graphs show (in descending order) the credit level and the queue occupancy of class A and the data transmission on the output port. Frames on an SR class are only sent if the corresponding credit level is zero or higher and the credit level is reset to zero as soon as there are no frames waiting on the corresponding queue. Hence, the traffic shaper enforces an idle time between consecutive frames in each class as shown in Figure 2(b) (between frames 2 and 3). When the corresponding traffic class is blocked due to a non-preemptive transfer that started earlier, credit can accumulate, resulting in a burst of frames once the output is free again as shown for frames 1 and 2 in Figure 2(b).

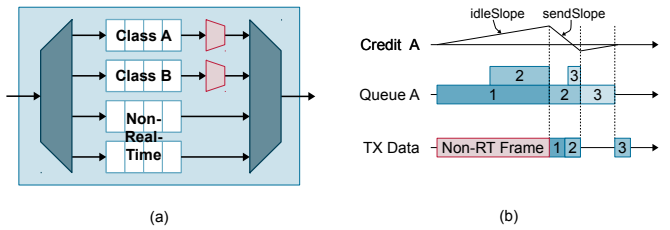

Fig. 2. Architecture (a) and operational example (b) of the Credit-Based Shaping Algorithm (CBSA).

2.2 Formal Timing Analysis

To analyze the worst-case timing behavior of Ethernet AVB, we use the Compositional Performance Analysis (CPA) approach as implemented by the tool SymTA/S (see Richter et al. (2003); Henia et al. (2005); Schliecker et al. (2009)), which is similar to Real-Time Calculus (Thiele et al. (2000)). This formal analysis assumes a set of tasks processed by communication or processing resources. For the tasks, formal and conservative characteristics are assumed, such as the lower and upper bounds on the task execution time (C^- and C^+). Tasks are activated by events which can originate from an external source, such as a timer interrupt, from other tasks via inter-task communication corresponding to their activation dependencies. Events are modeled using minimum and maximum distance functions $\delta^-(n)$ and $\delta^+(n)$, which are an lower/upper bound on the size of a time window containing n event arrivals. These functions have pseudo-inverse counterparts $\eta^+(\Delta t)$ and $\eta^-(\Delta t)$, which are called maximum and minimum arrival curves and return the maximum and minimum number of events that can arrive within any time window of size Δt. Such an event model covers all possible event arrivals of a specific event source and is not just a specific trace of events.

From these task properties and knowledge about the scheduling mechanism, one can derive upper bounds on the timing properties using a busy-window-based approach based on Lehoczky (1990); Tindell et al. (1994). It works by constructing a critical instant scenario for each task on each resource by assuming the worst-case arrival of all interfering tasks to maximally delay the processing of the task under consideration. For such a scenario, the n-event busy window (or busy period) is computed for each task, i.e. an upper bound on time required to process n activations of the task. This allows the computation of an output event model describing the minimum and maximum numbers of tasks completion in a specific interval as described in Henia et al. (2005) and Schliecker et al. (2008). The output event models are then forwarded as input event models of the dependent tasks, which are then analyzed again using the updated event models. This procedure is iterated until a fixed point (stable event models) is reached or a timing constraint (e.g. maximum path latency) is violated (see Henia et al. (2005)). To break cyclic analysis dependencies, initial event models for all tasks are derived from the external input event model of each task chain. In addition to the validation of the schedulability (all deadlines met) the analysis also yields upper bounds on the

worst-case response time (WCRT) of tasks and other timing properties such as the end-to-end latency of a chain of tasks.

3. MODELING OF ETHERNET AVB FOR TIMING

In this section, we will first describe the domain-specific model for Ethernet networks and the system model used by the compositional performance analysis before we discuss how the former is transformed into the latter.

3.1 Ethernet AVB Model

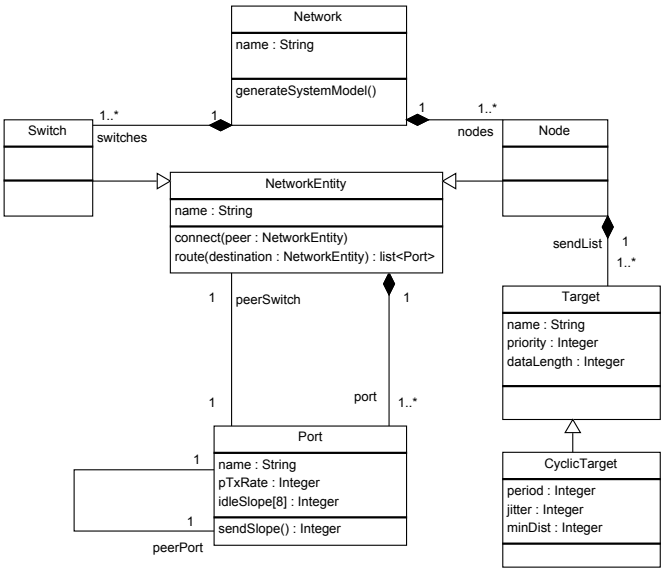

Fig. 3. Domain-specific model for Ethernet AVB

Figure 3 shows our domain-specific model in which Ethernet AVB networks can be expressed. The main components are switches and nodes, which are interconnected through ports. All components are identified by a unique name. For each port, the physical transmission rate ($pTxRate$) is specified. Additionally, each port specifies the CBSA *idleSlope* for each of the 8 priorities. Note that according to the Ethernet AVB standard, there are normally only two SR classes A and B (corresponding to priorities 3 and 2) and at most 7 SR classes but the model offers more flexibility. For convenience, there is also a function to compute the *sendSlope*, which is the difference between the $idleSlope$ and the $pTxRate$. Targets describe traffic streams by source and destination nodes. Special derived classes such as CyclicTarget capture specific frame injection patterns such as periodic with jitter and a minimum distance (minDist) between frames. Each Target also specifies its priority and the maximum packet size (dataLength). In order to compute the route from a specific source node to a target, the route()-function is used for a recursive depth-first search.

3.2 CPA System Model

Figure 4 shows the CPA system model as described in Section 2.2. For resources, a scheduling policy (e.g. static priority non-preemptive) is specified. For tasks, the execution time bounds or best-case and worst-case execution times (WCET and BCET) as well as a scheduling priority are specified. Each task has an activating event model containing the $\delta^-(n)$ and $\delta^+(n)$ functions and their pseudo-inverse counterparts $\eta^+(\Delta t)$ and $\eta^-(\Delta t)$ (which are actually derived from $\delta^-(n)$ and $\delta^+(n)$). Tasks form double-linked lists to model activation behaviors (task chains) and forward output event models. Streams can be a subset of such a task chain to define a path for the computation of an end-to-end latency including static per-hop and per-stream overheads.

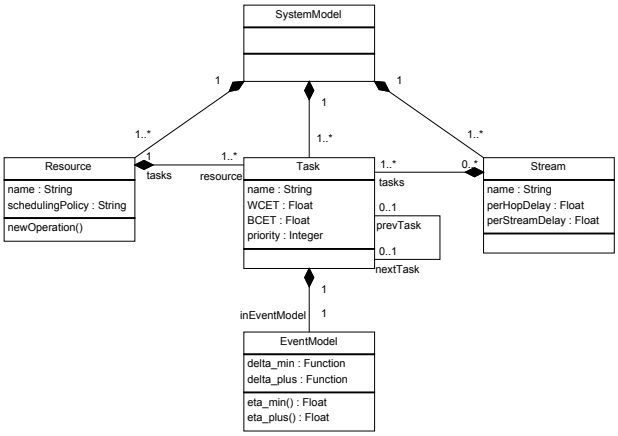

Fig. 4. Compositional Performance Analysis system model

3.3 Model Transformation

In order to apply compositional performance analysis to compute worst-case bounds on network timing, the domain-specific Ethernet AVB model must be transformed into a CPA system model. For this, we need a link between the entities in the Ethernet network and the components of the CPA system model. Since we are interested in the delay of frame transfers, it is obvious to map these to task executions in the CPA system model. To maintain compositionality, we divide the transfer of a frame through the network into a chain of individual link and switch traversal tasks. Hence the transfer latency of a frame becomes the end-to-end path latency of the corresponding task chain.

In order to derive accurate timing bounds from the transformed model, we must capture the delays encountered by packets in the network. These delays can be static and dynamic. Static delays such as wire transmission delays occur independently of other traffic in the network, while dynamic delays result from contention in the network which is resolved by arbitration or scheduling, e.g. at the switch output ports. Static delays can be trivially captured in the latency computation and are modeled as per-hop and per-stream overheads in the streams. To capture dynamic delays we create a scheduling resource for each arbitration point in the network with tasks for each stream passing through. The worst-case execution time (WCET) of these tasks is equal to the maximum time it takes for a frame to pass through the arbitration point without contention. Respecting the minimum Ethernet frame size, the frame overhead and the inter-frame gap, the WCET C_i^+ can be obtained as

$$C_i^+ = \frac{(48Byte + max(36Byte, dataLength))}{pTxRate} \qquad (1)$$

The only components where significant arbitration delays can be observed are the switch output ports. Hence, we map each

switch output port in the Ethernet AVB model to a resource. Nodes and targets (i.e. communication streams) are mapped to task chains, which are assigned to the corresponding resources according to the route taken in the Ethernet network. The targets' priorities are assigned to the corresponding task priorities. The event model describing packet injection is derived from the packet injection pattern of the target, e.g. for a an injection with period P, jitter J and minimum distance d, we use (see Schliecker et al. (2008))

$$\delta^-(n) = \max\{(n-1)\cdot d, (n-1)\cdot P - J\} \quad (2)$$

$$\delta^+(n) = (n-1)\cdot P + J \quad (3)$$

The Ethernet AVB model might not include a complete specification of low-priority legacy traffic as the characteristics of such best-effort traffic are often not known. In this case, we add low-priority interferer tasks to all output resources, which covers the worst-case of all possible low-priority traffic in the Ethernet network.

Figure 5 shows the transformed CPA system model of the example from Figure 1. For clarity, the sensor and control streams are drawn as two different models. Light boxes represent resources (transformed from switch output ports) which contain circles representing tasks for each stream passing through. Tasks are interconnected to form streams representing targets in the Ethernet model. Event models are shown in white boxes. Dark boxes represent switches and are shown to highlight the grouping of resources; they are not part of the actual CPA system model. Note that tasks/resources representing static wire delays have been omitted in Figure 5 for compactness.

The model transformation is unidirectional. However, the results obtained from the analysis of the CPA system model can be back-annotated into the Ethernet AVB model. The transformation process described here works for all possible Ethernet AVB network configurations.

4. WORST-CASE TIMING ANALYSIS OF ETHERNET AVB

With the model transformation described above, we have obtained an analyzable timing model of an Ethernet AVB network. Although we can apply the compositional performance analysis approach to the transformed system model, no existing local analysis matches the behavior of Ethernet AVB. Hence, we require formulas to compute the upper bounds on the busy window, output event model and response time for a resource under Ethernet AVB scheduling.

In principle, the AVB switch follows a static-priority non-preemptive schedule (SPNP) for which local analyses exist (e.g. Davis et al. (2007)). However, the traffic shaping applied to Class A and B traffic requires an extension of the standard SPNP analysis. Furthermore, Ethernet allows different streams with the same priority being scheduled in FIFO ordering which requires an extension similar to the analysis of earliest-deadline-first (EDF) scheduling (see Palencia and Harbour (2003)). Hence, we will now derive the required formalism for the local analysis for AVB, considering a task τ_i modeling the transfer of a frame over a single switch. Under AVB scheduling, the transfer latency of a single frame of a specific stream (and hence the execution of the task τ_i) is impacted by:

- Transfer time $t_{transfer}$: The time to transfer a frame (execute the task τ_i) is determined by the network speed (and the resulting core execution time), not including any blocking (no-load transfer time).
- Blocking time by lower-priority frame I_{LPB}: Task τ_i can be blocked by lower-priority tasks that commenced transfer just before the activation of the task.
- Blocking time by same-priority frames I_{SPB}: Task τ_i can be blocked by other tasks of the same priority which were activated before itself.
- Blocking time by higher-priority frames I_{HPB}: All higher-priority tasks may block task τ_i, limited by the traffic shaping applied to the high-priority classes. For the scope of this paper, we focus only on the analysis of the highest priority class-A traffic and thus omit the higher-priority blocking.
- Blocking time by traffic shaping I_{TSB}: Task τ_i may have to wait for shaper credits to be replenished before it may execute.

For the analysis of AVB, we extend the definition of the level-i busy-period $w_i(q)$ (cf. Tindell et al. (1994)) to account for the traffic shaper:

Definition 1. *The maximum (minimum) q-event busy time $w_i(q)$ of a task τ_i is given by the maximum (minimum) time the resource is busy processing q events, if all but the first of the q events arrive within the busy-window of their respective predecessor. The resource is considered busy if it processes a task (transmits a frame) or if the traffic shaper of task τ_i still has negative credit.*

The maximum busy-window $w_i(q)$, i.e. the longest time required to forward q frames of a stream τ_i, can be bound by maximizing and adding all of the above delays:

$$\begin{aligned} w_i(q) \leq & t_{transfer}(q) + I_{LPB} + \\ & I_{SPB}(w_i(q)) + I_{TSB}(w_i(q)) + I_{HPB}(w_i(q)) \end{aligned} \quad (4)$$

Note that $w_i(q)$ appears on both sides of the equation which results in an integer fixed point problem that can be resolved iteratively by starting with $w_i(q) = t_{transfer}(q)$. We will now discuss the upper bounds of each component of $w_i(q)$.

The **maximum transfer time** for q packets of a stream τ_i is given by

$$t_{transfer}(q) = q \cdot C_i^+ \quad (5)$$

The interference from the traffic shaper I_{TSB} is interdependent with other interference I_{LPB}, I_{SPB}, and I_{HPB}. This is because positive credit accumulates when a task is blocked which reduces traffic shaper blocking later. Hence, in order to maximize the overall interference, i.e. the sum of the terms, we assume that the traffic shaper interference occurs as early as possible, according to the following lemma.

Lemma 1. *If a task is blocked by its own traffic shaper within its busy window, all previous interference by other traffic is irrelevant.*

Proof. To verify this lemma, we assume that a task is blocked by its traffic shaper at t_{TS}. If it had been blocked by another task before at $t_S < t_{TS}$, it would have built up positive credit during this time. This would postpone the time of depletion of the shaper credits and hence increase t_{TS} by t_{credit}. t_{credit} is exactly identical to the amount of blocking the task received by the other stream. Hence any blocking before t_{TS} is compensated by

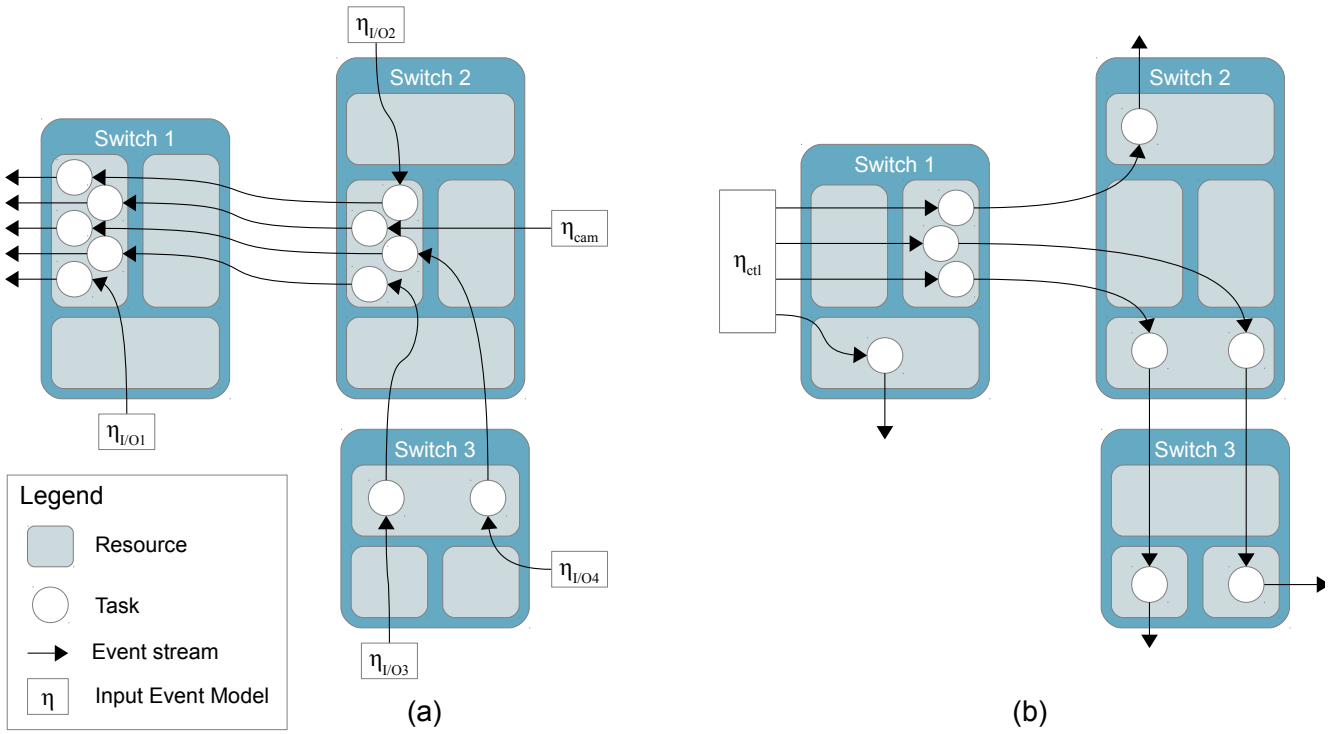

Fig. 5. Transformed analysis models of the example network for the sensor streams (a) and control streams (b)

a reduced blocking by the traffic shaper. From this, the lemma follows. □

A task τ_i can suffer **lower-priority blocking** only once by a non-preemptive lower-priority task that started executing just before τ_i was ready. In the worst case, the longest executing lower-priority task must be assumed to be the blocker:

$$I_{LPB} = \max_{j \in lp(i)} \left\{ C_j^+ \right\} \tag{6}$$

where $lp(i)$ is the set of lower priority tasks mapped to the same resource as task τ_i.

The **same-priority blocking** depends on the arrival time $a_i(q)$ of the q-th event of task τ_i due to the FIFO scheduling within the same priority. Hence, the blocking inferred by the same-priority tasks can be bounded by

$$I_{SPB}(a_i(q)) \leq \sum_{j \in sp(i)} \left(\eta_j^+(a_i(q)) \cdot C_j^+ \right) \tag{7}$$

with $sp(i)$ being the set of same-priority tasks mapped to the same resource as task τ_i and η_j^+ being the worst-case arrival function of task j. Thus $\eta_j^+(a_i(q))$ is the maximum number of activations of task j before the arrival of the q-th activation of task τ_i. For tasks which are activated by preceding tasks, this bound can be improved by exploiting the traffic shaping performed on the preceding task's resource. The maximum load allowed by the corresponding preceding traffic shaper (derived from the *idleSlope* and *sendSlope* parameters) can be used as an upper bound for the interference from tasks from that resource. Although this improvement is implemented in the analysis, a detailed formulation is out of the scope of this paper.

The **traffic shaper blocking** depends on the allowed rate (*idleSlope*) of the corresponding class. Every task executed on the class of task τ_i (i.e. tasks in $sp(i) \cup \{\tau_i\}$) consumes credits, which may lead to a blocking by the shaper. The first frame is not blocked by the shaper according to the definition of the busy window (Definition 1). The amount of credits consumed by a transmission lasting C_{trans} time units is $K_{consumed} = -sendSlope_i \cdot C_{trans}$. Considering traffic shaper blocking observed by task τ_i, C_{trans} can be bounded by the time required to transmit own and same-priority frames, i.e. $C_{trans} \leq t_{transfer}(q) + I_{SPB}(a_i(q))$. Hence, to replenish the credits consumed by previous frame transfers, the following time is required:

$$I_{TSB}(a_i(q)) = \frac{K_{consumed}}{idleSlope_i} = \frac{-sendSlope_i}{idleSlope_i} \cdot C_{trans} \tag{8}$$

$$\leq \left[(q-1) \cdot C_i^+ + I_{SPB}(a_i(q)) \right] \cdot \frac{-sendSlope_i}{idleSlope_i} \tag{9}$$

This formula assumes that the first frame does not observe any traffic shaper blocking, hence the $q-1$.

Now that we have derived the q-event busy window for Ethernet AVB scheduling, we can bound the worst-case response time R_i^{max} of task τ_i. For this, we must find the maximum distance between the completion of q activations ($w_i(q)$) and the arrival of the q-th activation relative to the arrival candidate $a_i(q)$:

$$R_i^{max} = \max_{q=1,2,\ldots} \left\{ \max_{a_i(q)} \left\{ w_i(q) - a_i(q) - \delta_i^-(q) \right\} \right\} \tag{10}$$

The end-to-end latency of a frame can be computed as a sum of the individual per-hop delays and the additional wire and (de-)packetization delays. Furthermore, we can derive the output event model for each task which becomes the input

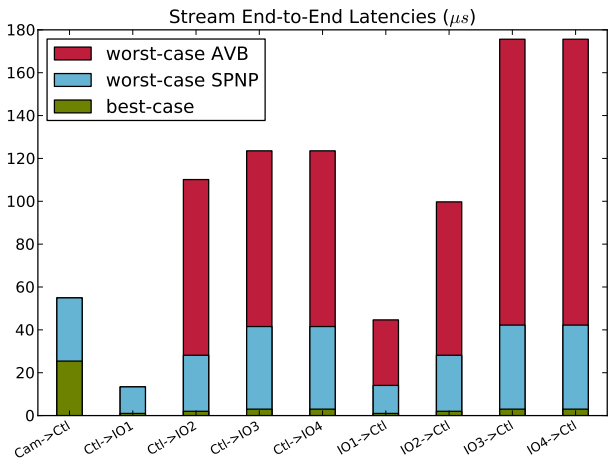

Fig. 6. Analysed worst-case latency bounds for priority- and AVB-based Ethernet.

event models of dependent tasks as proposed in Schliecker et al. (2008). To improve the output event model, we can again exploit the known blocking inferred by the shaper as done in the computation of the same-priority blocking.

5. EXPERIMENTAL EVALUATION

We have implemented the presented Ethernet and CPA models, the model transformation and the analysis in Python. For evaluation, we have modeled the simple network from the introductory example (see Figures 1 and 5). We compare the latency bounds for the system assuming (a) standard Ethernet (only priorities) and (b) Ethernet AVB with additional traffic shaping. The per hop latency for (a) is obtained using Equation 4 with $I_{TSB} = 0$.

In this example, all streams between the controller and the I/O modules are SR class-A real-time streams (priority 3), while the stream between the camera and the controller is regular traffic (priority 0). We assume that all streams are periodically activated. For the streams between the I/O blocks and the controller the period is 250 μs and 4$Bytes$ are sent per period (both directions). For the stream from the camera to the terminal, the period is 195 μs with 1500$Bytes$ sent, which corresponds to an uncompressed black&white VGA video at 25fps. For the class-A streams, the *idleSlope*s are configured to allow twice the requested data rate. This overreservation reduces the otherwise very high worst-case traffic shaper when multiple same-priority frames arrive in a burst.

Figure 6 shows the results of the analysis. For streams Cam→Ctl and Ctl→IO1, priority and AVB Ethernet achieve nearly identical worst-case latencies. For all other streams, however, priority-based Ethernet significantly outperforms AVB. This is due to the fact that the interference caused by the traffic shaper in CBSA is always negative to class-A streams and is maximized in the worst-case when a burst of class-A frames arrives. This is exaggerated due to the compositional approach, which assumes worst-case traffic shaper blocking on every hop independently. The only positive effect is on the interference of lower-priority streams (e.g. stream Cam→Ctl), which is currently not exploited in the analysis.

6. CONCLUSION

In this paper, we have presented the modeling of Ethernet AVB and how such models can be transformed into timing analysis models. We have shown how worst-case timing parameters can be computed from the transformed models. This way, formal guarantees on the timing of Ethernet AVB streams on the system level can be obtained which enables the use of such networks in real-time critical embedded systems.

REFERENCES

ARINC (2009). ARINC Report 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network. Technical report, ARINC.

Davis, R., Burns, A., Bril, R., and Lukkien, J. (2007). Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised. *Real-Time Systems*, 35(3).

Diemer, J., Rox, J., Negrean, M., Stein, S., and Ernst, R. (2011). Real-Time Communication Analysis for Networks with Two-Stage Arbitration. In *EMSOFT'11*.

Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., and Ernst, R. (2005). System Level Performance Analysis–the SymTA/S Approach. *IEE Proceedings-Computers and Digital Techniques*, 152(2).

IEEE 802.1 (2011). Bridging and management. http://standards.ieee.org/about/get/802/802.1.html.

Imtiaz, J., Jasperneite, J., and Han, L. (2009). A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*.

Lehoczky, J. (1990). Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the 11th Real-Time Systems Symposium*.

Palencia, J. and Harbour, M. (2003). Offset-based response time analysis of distributed systems scheduled under EDF. In *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*.

Richter, K., Jersak, M., and Ernst, R. (2003). A Formal Approach to MpSoC Performance Verification. *IEEE Computer*, 36(4).

Rox, J. and Ernst, R. (2010). Formal Timing Analysis of Full Duplex Switched Based Ethernet Network Architectures. In *SAE World Congress*, volume System Level Architecture Design Tools and Methods (AE318). SAE International.

SAE (2011). SAE International Aerospace Standard SAE-AS6802, Time-Triggered Ethernet. Technical report, SAE.

Schliecker, S., Rox, J., Ivers, M., and Ernst, R. (2008). Providing Accurate Event Models for the Analysis of Heterogeneous Multiprocessor Systems. In *CODES-ISSS*.

Schliecker, S., Rox, J., Negrean, M., Richter, K., Jersak, M., and Ernst, R. (2009). System Level Performance Analysis for Real-Time Automotive Multi-Core and Network Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(7).

Shi, Z. and Burns, A. (2008). Real-time communication analysis for on-chip networks with wormhole switching. In *NOCS*. IEEE Computer Society.

Thiele, L., Chakraborty, S., and Naedele, M. (2000). Real-time calculus for scheduling hard real-time systems. In *ISCAS*, volume 4.

Tindell, K., Burns, A., and Wellings, A. (1994). An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2).