Real-Time Syst (2017) 53:254–287
DOI 10.1007/s11241-016-9265-0

Timing analysis of rate-constrained traffic in TTEthernet using network calculus

Luxi Zhao[1] · Paul Pop[2] · Qiao Li[1] · Junyan Chen[1] · Huagang Xiong[1]

Published online: 7 January 2017

Abstract TTEthernet is a deterministic, synchronized and congestion-free network protocol based on the Ethernet standard and compliant with the ARINC 664p7 standard network. It supports safety-critical real-time applications by offering different traffic classes: static time-triggered (TT) traffic, rate-constrained (RC) traffic with bounded end-to-end latencies and best-effort traffic, for which no guarantees are provided. TTEthernet uses three integration policies for sharing the network among the traffic classes: shuffling, preemption and timely block. In this paper, we propose an analysis based on network calculus (NC) to determine the worst-case end-to-end delays of RC traffic in TTEthernet. The main contribution of this paper is capturing the effects of all the integration policies on the latency bounds of RC traffic using NC, and the consideration of relative frame offsets of TT traffic to reduce the pessimism of the RC analysis. The proposed analysis is evaluated on several test cases, including realistic applications (e.g., Orion Crew Exploration Vehicle), and compared to related works.

✉ Luxi Zhao
zhaoluxi@buaa.edu.cn

Paul Pop
paupo@dtu.dk

Qiao Li
avionics@buaa.edu.cn

Junyan Chen
chenjy83@hotmail.com

Huagang Xiong
hgxiong@buaa.edu.cn

1 School of Electronics and Information Engineering, Beihang University, Beijing, China

2 DTU Compute, Technical University of Denmark, Kongens Lyngby, Copenhagen, Denmark

Keywords TTEthernet · Timing analysis · Network calculus · Upper bound · Delay

1 Introduction

Ethernet (IEEE 802.3 2012), although it is low cost and has high speeds (e.g., up to 10 Gbps), is known to be unsuitable for real-time and safety-critical real-time applications (Decotignie 2005; Lee et al. 2005). For example, in half-duplex implementations, frame collision is unavoidable, leading to unbounded transmission times. Decotignie (2005) presents the requirements for a real-time network and how Ethernet can be improved to comply with these requirements. Several real-time communication solutions based on Ethernet have been proposed, such as FTT-Ethernet (Pedreiras et al. 2005), ARINC 664 Specification Part 7 (ARINC 664p7, for short) (ARINC 2009), TTEthernet (SAE 2011), EtherCAT (ETG 2013) and IEEE Audio Video Bridging (AVB) (IEEE 802.1BA 2011). Schneele and Geyer (2012), Cummings et al. (2012) and Danielis et al. (2014) describe and compare several of the proposed Ethernet-based real-time communication protocols. FTT-Ethernet is a flexible time-triggered protocol integrating event-triggered (ET) and time-triggered (TT) traffic. The main advantage of FTT-Ethernet is that it provides dynamic quality-of-service (QoS) management at some time with guaranteed timeline for the TT traffic. Its disadvantage is that it is an academic protocol, not yet available on the market. ARINC 664p7 extends Ethernet for the avionics area providing timing guarantees for ET traffic and enforcing the separation of mixed-criticality flows through the concept of virtual links. However it does not support TT traffic. EtherCAT is specialized for industrial automation based on a master/slave approach. It provides high-performance for real-time applications, however, its scalability is limited due to the logical ring topology. AVB does not support TT traffic with strict timing requirements but it is intended to provide QoS guarantees for lower latency communication, especially for multimedia flows. In this paper, we are interested in the TTEthernet protocol (SAE 2011).

TTEthernet (SAE 2011; Kopetz and Grunsteidl 2005) is a deterministic, synchronized and collision-free network protocol based on the IEEE 802.3 Ethernet standard and compliant with ARINC 664p7. TTEthernet is suitable for several safety-critical application areas, including automotive (Steinbach et al. 2012), avionics (Suen et al. 2013), space (Fletcher 2009) and industrial (Steiner and Dutertre 2010). Although current work on Time Sensitive Networking (TSN) (IEEE P802.1Qbv 2015) is intended to extend Ethernet for use in several application areas, existing Ethernet variants will still remain in certain niche areas. TTEthernet will continue to be used, for example, in the aerospace market since it provides fault-tolerance services not available in TSN, which are very relevant for avionics applications (Steiner 2016). ARINC 664p7 (ARINC 2009) is a full-duplex Ethernet network, which provides predictable event-triggered communication through the concept of rate-constrained (RC) traffic. In addition to the functionality offered by Ethernet (best-effort or BE traffic) and ARINC 664p7 (with RC traffic), TTEthernet supports time-triggered (TT) communication, especially suitable for applications with highest criticality requirements.

Thus, TTEthernet classifies flows into the three traffic classes (Steiner et al. 2009): time-triggered (TT) traffic, rate-constrained (RC) traffic and best-effort (BE) traffic.

TT traffic uses time-triggered communication implemented using static schedules relying on a synchronized time base by arranging TT frames in predetermined time slots along the transmission path. RC traffic has a lower priority than TT, uses event-triggered communication and has bounded end-to-end latencies. Both TT and RC traffic can be used for hard real-time applications with strict timing constraints, and the choice depends on the particularities of the application, legacy constraints and the experience of the system engineer (Gavrilut et al. 2015). BE traffic has the lowest priority and it is used for applications that do not require any timing guarantees.

The schedulability of the TT traffic is determined by the TT schedules, which have to be synthesized such that all frames meet their deadlines. For the RC traffic, which is event-triggered, a RC flow is schedulable if its worst-case end-to-end delay (WCD) is smaller than its deadline. Although latency analysis methods have been successfully applied to RC traffic in ARINC 664p7 networks (Frances et al. 2006; Bauer et al. 2010; Li et al. 2010; Scharbarg et al. 2009; Adnan et al. 2010; Mauclair and Durrieu 2013), they cannot be directly applied for the timing analysis of RC traffic in TTEthernet due to the static TT schedules. The earliest analysis approach for RC traffic (Steiner 2011) in TTEthernet assumes pessimistically that all RC frames in an outgoing port of a switch will delay the current frame under analysis, and considers that the TT schedules contain periodically alternating phases for TT traffic and for RC traffic. However, realistic schedules do not necessarily contain such periodic phases. Later, we (Zhao et al. 2014) proposed a network calculus (NC)-based analysis to compute the WCD of RC frames by considering a variable size of TT frames and focusing on the shuffling integration policy. But the protocol uses fixed size frames and also has preemption and timely block integration policies, which we have not considered. A recent analysis for RC flows in TTEthernet has been proposed in Tamas-Selicean et al. (2015b). The authors use a response-time analysis based on the concept of "busy period" and show that they are able to significantly reduce the pessimism compared to previous approaches. However, their analysis computes the latencies for each time instance in the TT schedule, which is time-consuming. As we will see in the experimental results (Sect. 8), their method does not scale for large problem sites. In addition, another potential drawback of the work in Tamas-Selicean et al. (2015b) is that it is based on a response-time analysis instead of network calculus (Cruz 1991). This is a drawback because there has been a strong investment by the industry in NC, which is a well-established theory, supported by academic and industrial tools (Le Boudec and Thiran 2001; Mabille et al. 2013; Wanderler and Thiele 2006). For example, NC has been used for ARINC 667p7 for the design and certification of the Airbus A380 (Frances et al. 2006; Boyer and Fraboul 2008). No such tools and certification approaches exist for the type of analysis employed in Tamas-Selicean et al. (2015b).

The contribution of this paper is the modeling of TT traffic flows in NC to reduce the pessimism of the RC analysis. By capturing the effects of different integration policies on the latency bounds of RC flows, we extend the existing NC approaches to analyze the RC traffic for all the three integration policies of TTEthernet. In addition, this paper evaluates the proposed method on several test cases, including large realistic applications (e.g., Orion Crew Exploration Vehicle) and compares the latency upper bounds and runtimes with the results from the previous work, Zhao et al. (2014), Steiner (2011) and Tamas-Selicean et al. (2015b). In this paper, we do not address the BE traffic since it has no timing requirements.

The reminder of this paper is organized as follows. Section 2 presents the architecture and application models. Section 3 introduces the TTEthernet protocol in detail. Section 4 briefly describes the network calculus theory. Section 5 presents the problem formulation and overall analysis strategy. Section 6 describes the worst-case latency analysis of RC traffic in the shuffling integration policy. In Sect. 7, the influence of timely block integration on RC traffic is given. Section 8 evaluates the proposed analysis and Sect. 9 concludes this paper.

2 System model

2.1 Architecture model

A TTEthernet network is composed of a set of clusters. Each cluster consists of end systems (ESes) interconnected by links and switches (SWs). Each ES has a buffer for the output port and is connected to exactly one input port of SW. Each SW has no input buffers on input ports and a buffer for each output port. The output port of SW is connected to one ES or another switch input port. The links are full duplex, allowing thus communication in both directions, and the networks can be multi-hop. An example cluster is presented in Fig. 1, where we have 4 ESes, ES_1 to ES_4, and 3 SWs, SW_1 to SW_3. In the following we use “node” to represent an ES or a SW, and use “node output port” to represent an output port of an ES or a SW.

We model a TTEthernet cluster as an undirected graph $G(V, E)$, where $V = ES \cup SW$ is the set of end systems (ES) and switches (SW), and E is the set of physical links. For Fig. 1, $V = ES \cup SW = \{ES_1, ES_2, ES_3, ES_4\} \cup \{SW_1, SW_2, SW_3\}$, and the physical links E are depicted with thick, black, double arrows.

A dataflow link $dl_i = [v_j, v_k] \in L$, where L is the set of dataflow links in a cluster, is a directed communication connection from v_j to v_k, where v_j and $v_k \in V$ can be ESes or SWs. A dataflow path $dp_i \in DP$ is an ordered sequence of dataflow links

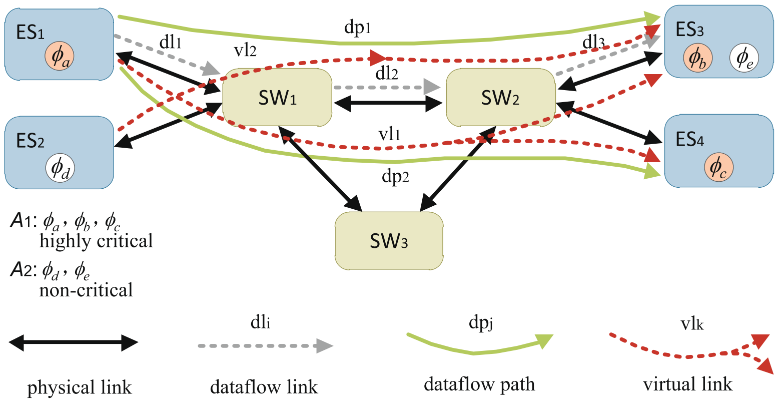


Fig. 1 TTEthernet cluster example

connecting one sender to one receiver. For example, in Fig. 1, dp_1 connects the source end system ES_1 to the destination end system ES_3, while dp_2 connects ES_1 to ES_4 (the dataflow paths are depicted with green arrows). Moreover, dp_1 in Fig. 1 can be denoted as $[[ES_1, SW_1], [SW_1, SW_2], [SW_2, ES_3]]$.

The space partitioning between messages of different criticality transmitted over physical links and network switches is achieved through the concept of virtual link. Virtual links are defined by ARINC 664p7 (ARINC 2009), which is implemented by the TTEthernet protocol, as a "logical unidirectional connection from one source end system to one or more destination end systems". We denote the set of virtual links in a cluster with $\mathcal{VL}$. A virtual link $vl_i \in \mathcal{VL}$ is a directed tree, with the sender as the root and receivers as leafs. For example, vl_1, depicted in Fig. 1 using dot-dash red arrows, is a tree with the root ES_1 and leafs ES_3 and ES_4. Each virtual link is composed of a set of dataflow paths, one such dataflow path for each root-leaf connection. More formally, we denote with $\mathcal{R}_{VL}(vl_i) = \{\forall dp_j \in \mathcal{DP} \big| dp_i \in vl_i\}$ the routing of virtual link vl_i. For example, in Fig. 1, $\mathcal{R}_{VL}(vl_1) = \{dp_1, dp_2\}$.

Let us assume that in Fig. 1 we have two applications, $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1$ is a high criticality application consisting of tasks ϕ_a to ϕ_c mapped on ES_1, ES_3 and ES_4, respectively. $\mathcal{A}_2$ is a non-critical application, with tasks ϕ_d and ϕ_e mapped on ES_2 and ES_3, respectively. ϕ_a sends message m_1 to ϕ_b and ϕ_c. Task ϕ_d sends message m_2 to ϕ_e. With TTEthernet, a message has a single sender and may have multiple receivers. The flow of these messages will intersect in the physical links and switches. Virtual links are used to separate the highly critical message m_1 from the non-critical message m_2. Thus, m_1 is transmitted over virtual link vl_1, which is isolated from virtual link vl_2, on which m_2 is sent, through protocol-level temporal and spatial mechanisms, which are presented in detail in Tamas-Selicean et al. (2012b).

2.2 Application model

TTEthernet transmits data using flows (in this paper we use the term "flow" to denote a "frame" in the TTEthernet protocol). The TTEthernet frame format fully complies with the ARINC 664p7 (ARINC 2009). Messages are transmitted in the payload of frames.

The size $m_i.size$ for each message $m_i \in \omega$ is given, where ω is the set of all messages. As mentioned, TTEthernet supports three traffic classes: TT, RC and BE. We assume that the designer has decided the traffic classes for each message. We define the sets ω_{TT}, ω_{RC} and ω_{BE}, respectively, with $\omega = \omega_{TT} \cup \omega_{RC} \cup \omega_{BE}$. Similarly, we define τ_{TT}, τ_{RC} and τ_{BE}, with $\tau = \tau_{TT} \cup \tau_{RC} \cup \tau_{BE}$ is the set of all the flows in cluster. Knowing the size $m_i.size$ for each message m_i, we can compute the frame size l_i of the flow τ_i packing m_i. In addition, for the TT and RC flows we know their periods/rate and deadlines, p_{TTi} and r_{RC_i}, and $\tau_i.deadline$, respectively. The routing of virtual links and the assignment of flows to virtual links are given.

TT flows τ_{TT} are transmitted according to offline computed schedules. Note that due to the implementation of TTEthernet schedules, the sending time offset of a TT flow τ_{TT_i} is identical in all periods. The complete set of local schedules in a cluster are denoted by $\mathcal{S}$, and the TT schedule for a dataflow link dl_j is denoted by $\mathcal{S}_j \in \mathcal{S}$.

Several approaches (Steiner 2010a; Steiner and Dutertre 2010; Steiner 2011; Tamas-Selicean et al. 2012b; Tamas-Selicean et al. 2015a) to the synthesis of schedules $\mathcal{S}$ for a cluster have been proposed.

RC flows τ_{RC} are not necessarily periodic, but have a minimum inter-arrival time. For each virtual link vl_i carrying a RC flow τ_{RC_i} the designer decides the Bandwidth Allocation Gap (BAG). A BAG_{RC_i} is minimum time interval between two consecutive frames of a RC flow τ_{RC_i} and may take fixed values of 2^i ms, $i = 0, \ldots, 7$. The BAG is set in such a way to guarantee that there is enough bandwidth allocated for the transmission of a flow on a virtual link, with $BAG_{RCi} \leq 1/r_{RC_i}$. The BAG is enforced by the sending ES. Thus, an ES will ensure that each BAG_{RCi} interval will contain at most one frame of τ_{RC_i}. The maximum bandwidth used by a virtual link vl_i transmitting a RC flow τ_{RC_i} is $\rho_{RC_i} = l_{RC_i}/BAG_{RC_i}$, where l_{RC_i} is the frame size of τ_{RC_i}. The BAG for each RC flow is computed offline, based on the requirements of the messages it packs.

3 TTEthernet protocol

3.1 Time triggered transmission

In this section we present how TT flows are transmitted by TTEthernet, using the example in Fig. 2, where the TT message m_2 is sent from task ϕ_2 on ES_1, to task ϕ_4 on ES_2. We mark each step of the TT transmission in Fig. 2 with a letter from (a) to (n) on a blue background.

Thus, in the first step denoted with (a), task ϕ_2 packs m_2 into flow τ_{TT_2} and in the second step (b), τ_{TT_2} is placed into buffer $B_{1,Tx}$ for transmission. Conceptually, there is one such buffer for every TT frame sent from ES_1. As mentioned, the TT communication is done according to static communication schedules determined offline and stored into the ESes and SWs. Thus, the sending schedule S_S contains the sending times for all the TT flows transmitted during an application cycle T_{cycle}. A periodic flow τ_{TT_i} may contain several frames within T_{cycle}. We denote the xth frame of τ_{TT_i} with $f^x_{TT_i}$. In step (d), the TT scheduler task TT_S will send τ_{TT_2} to SW_1 at the time specified in the sending schedule S_S stored in ES_1 (c). Often, TT tasks are used in conjunction with TT frames, and the task and TT schedules are synchronized such that the task is scheduled to finish before the frame is scheduled for transmission

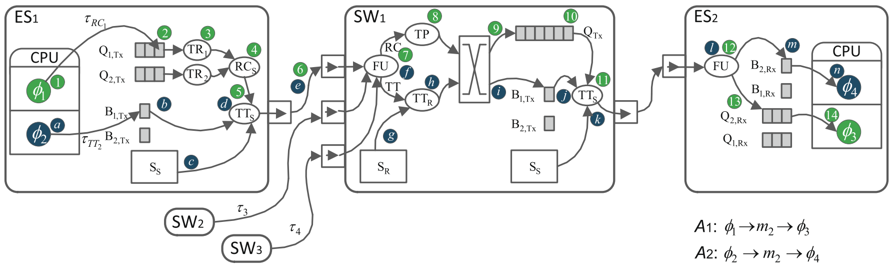


Fig. 2 TT and RC traffic transmission example

(Obermaisser 2011). Moreover, the TT scheduler task TT_S provides the separation mechanisms implemented by TTEthernet to isolate mixed-criticality flows (such as τ_{RC_1} and τ_{TT_2} in our example) and implements fault-tolerance services to protect the network e.g., by only transmitting frames as specified in the schedule table S_S, even if a task such as ϕ_2 becomes faulty and sends more frames than scheduled.

Next, τ_{TT_2} is sent on a dataflow link to SW_1 (e). The Filtering Unit (FU) task is invoked every time a frame is received by an SW to check the integrity and validity of the flow, e.g., for τ_{TT_2} see step (f). After τ_{TT_2} is checked (f), it is forwarded to the TT receiver task TT_R (h). TT_R relies on a receiving schedule S_R (g) stored in the switch to check if a TT flow has arrived within a specified receiving window and drop the faulty frames arriving outside the receiving window. This window is determined based on the sending times in the sending schedules (schedule S_S on ES_1 for the case of flow τ_{TT_2}), the precision of the clock synchronization mechanism and the integration policy used for integrating the TT traffic with the RC and BE traffic (see next subsection for details). Then, the switching fabric forwards (i) the TT flow τ_{TT_2} into the sending buffer $B_{1,Tx}$ for later transmission (j). Next, τ_{TT_2} is sent by the TT scheduler task TT_S in SW_1 to ES_2 at the time specified in the TT sending schedule S_S in SW_1.

When τ_{TT_2} arrives at ES_2 (l), the FU task will store the frame into a dedicated receive buffer $B_{2,Rx}$ (m). Finally, when task ϕ_4 is activated, it will read τ_{TT_2} from the buffer (n).

3.2 Rate constrained transmission

This section presents how RC traffic is transmitted using the example of flow τ_{RC_1} in Fig. 2 sent from ϕ_1 on ES_1 to ϕ_3 on ES_2. Similarly to the discussion of TT traffic, we mark each step in Fig. 2 using numbers from (1) to (14), on a green background.

Thus, ϕ_1 packs message m_1 into flow τ_{RC_1} (1) and inserts it into a queue $Q_{1,Tx}$ (2). Conceptually, there is one such queue for each RC virtual link. RC traffic consists of event-triggered messages. The separation of RC traffic is enforced through BAG, which is the minimum time interval between two consecutive frames of a RC flow τ_{RC_i}. The BAG is enforced by the traffic regulator (TR) task. For example, TR_1 in ES_1 in Fig. 2 will ensure that each BAG_1 interval will contain at most one frame of τ_1 (3). Therefore, even if a flow is sent in bursts by a task, it will leave the TR task within a specified BAG.

Several flows will be sent from an ES. Let us first discuss how RC flows are multiplexed, and then we will discuss the integration with the TT traffic. In an ES, the RC scheduler task RC_S (such as the one in ES_1) will multiplex several RC flows (4) coming from the traffic regulator tasks, TR_i, such as TR_1 and TR_2 in ES_1. Figure 3 depicts how this multiplexing is performed for the flow τ_{RC_x} and τ_{RC_y} with the sizes and BAGs in a output port h as specified in Fig. 3a and b, respectively. Figure 3c shows one of the possible output scenarios, in the case two flows attempt to transmit frames at the same time. How they will be sent on the outgoing dataflow link $[ES_1, SW_1]$ is non-deterministic and selected by the RC_S task. The jitter for the frame $f^{h,0}_{RC_y}$ equals to the transmission duration of τ_{RC_x}. RC traffic also has to be integrated with TT traffic, which has higher priority. Thus, RC flows are transmitted

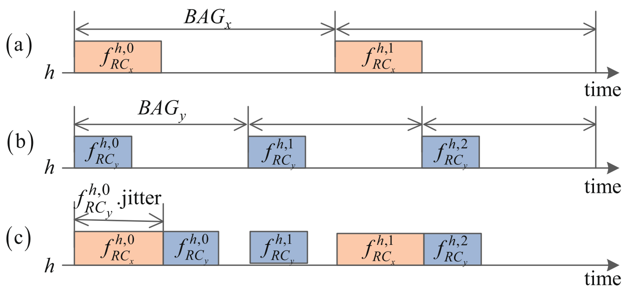


Fig. 3 Multiplexing two RC flows

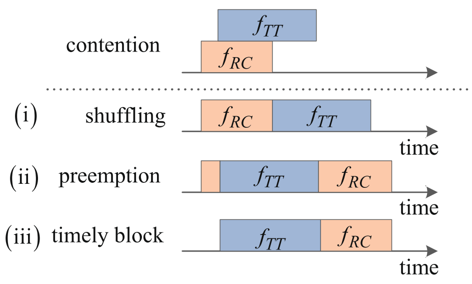


Fig. 4 Integration policies of TT and RC traffic

only when there is no TT traffic on the dataflow link. Hence, for our example on ES_1, the TT_S task on ES_1 will transmit flow τ_{RC_1} (5) to NS_1 on the dataflow link $[ES_1, SW_1]$ only when there is no TT traffic (6). There are three integration policies in TTEthernet (Steiner et al. 2009; SAE 2011): (i) shuffling, (ii) preemption and (iii) timely block, as shown in Fig. 4. (i) With shuffling, the higher priority TT frame is delayed until the RC frame finishes the transmission. In the case (ii) of preemption, the RC frame is preempted, and its transmission is restarted from the beginning after the TT frame finished transmitting. In the case (iii) of timely block, the RC frame is blocked (postponed) from transmission on a dataflow link if a TT frame is scheduled to be sent before the RC frame would complete its transmission. The timely block integration policy is used when the TT frames have very strict timing requirements, and no delays can be accepted compared to the scheduled transmission times. However, this comes at the expense of wasted bandwidth (no other frames can be transmitted in the blocked interval) and increased delays for RC frames. In case TT frames can tolerate slight variations in their transmission times, the shuffling is a good option since it is able to make use of the previously blocked intervals, which increases bandwidth usage and reduces RC delays.

When the RC flow τ_{RC_1} arrives at NS_1, the filtering unit (FU) task (7) will check its validity and integrity. Fault-containment at the level of RC virtual links is provided by the traffic policing (TP) task, see NS_1 in Fig. 2. TP implements an algorithm known as leaky bucket (ARINC 2009; SAE 2011), which checks the time interval between two consecutive frames on the same virtual link. After passing the checks of the TP task (8), τ_{RC_1} is forwarded by the switching fabric to the outgoing queue Q_{Tx} (9). In this paper we assume that all the RC flows have the same priority, thus the TT_S (11)

will send RC frames in Q_{Tx} in a FIFO order, but only when there is no scheduled TT traffic. At the receiving ES, after passing the FU (12) checks, τ_{RC_1} is copied in the receiving $Q_{2,Rx}$ queue (13). Finally, when ϕ_3 is activated, it will take τ_{RC_1} (14) from this queue.

4 Network calculus background

Network calculus (Cruz 1991; Le Boudec and Thiran 2001; Chang 2000) is a mature theory proposed for deterministic performance analysis, such as the computation of the worst-case latency of a flow transmitted over a network. The approach in network calculus is to construct appropriate arrival and service curve models for the investigated flows and network nodes, such as an ES or a SW. The arrival and service curves are defined by means of the min-plus convolution (Le Boudec and Thiran 2001).

An arrival curve $\alpha(t)$ is a model constraining the arrival process $R(t)$ of a flow, in which $R(t)$ represents the input cumulative function counting the total data bits of the flow that has arrived in the network node up to time t. We say that $R(t)$ is constrained by $\alpha(t)$ if

$$R(t) \leq \underset{0 \leq s \leq t}{inf} \{R(s) + \alpha(t-s)\} = (R \otimes \alpha)(t), \tag{1}$$

where inf means infimum (greatest lower bound) and $\otimes$ is the notation of min-plus convolution. A typical example of an arrival curve is the “leaky bucket” (Cruz 1991) model and given by,

$$\alpha_{\sigma,\rho}(t) = \begin{cases} \rho t + \sigma, & t \geq 0 \\ 0, & t < 0, \end{cases}$$

where σ represents the maximum burst tolerance of the flow and ρ is the upper bound of the long-term average rate of the flow.

A service curve $\beta(t)$ models the processing capability of the available resource. Assume that $R^*(t)$ is the departure process, which is the output cumulative function that counts the total data bits of the flow departure from the network node up to time t. We say that the network node offers the service curve $\beta(t)$ for the flow if

$$R^*(t) \geq \underset{0 \leq s \leq t}{inf} \{R(s) + \beta(t-s)\} = (R \otimes \beta)(t). \tag{2}$$

A typical example of the service curve is the rate-latency service curve and given by,

$$\beta_{R,T}(t) = R[t-T]^+,$$

where R represents the service rate, T represents the service latency and the notation $[x]^+$ is equal to x if $x \geq 0$ and 0 otherwise.

Let us assume that the flow constrained by the arrival curve $\alpha(t)$ traverses the network node offering the service curve $\beta(t)$. Then, the latency experienced by the flow in the network node is bounded by the maximum horizontal deviation between the graphs of two curves $\alpha(t)$ and $\beta(t)$,

$$h(\alpha, \beta) = \underset{s \geq 0}{sup} \{inf\{\tau \geq 0 \mid \alpha(s) \leq \beta(s+\tau)\}\}, \tag{3}$$

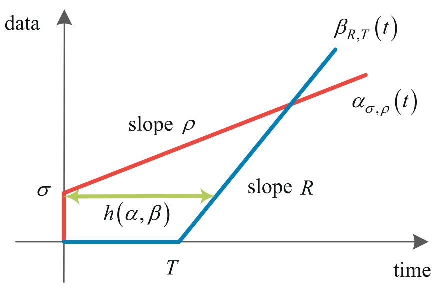


Fig. 5 A simple analysis example by using network calculus

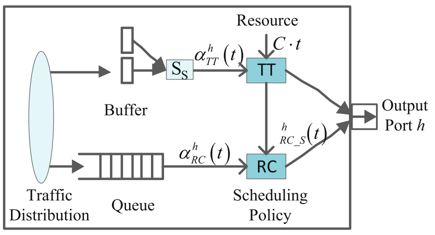
where sup means supremum (least upper bound). Consider a flow constrained by the leaky bucket $\alpha_{\sigma,\rho}(t)$ and served in a node with the rate-latency service curve $\beta_{R,T}(t)$. The worst-case latency is shown using the green line labelled $h(\alpha, \beta)$ in Fig. 5. With these definitions, the worst-case end-to-end delay of the flow is the sum of latency bounds in network nodes along its virtual link.

5 Problem formulation and overall analysis strategy

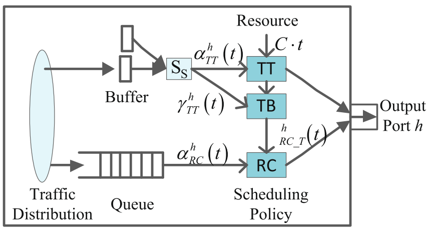
In this paper we are interested in determining the worst-case end-to-end latency D_{RC_i} of RC flow τ_{RC_i} along each path of the virtual link in a TTEthernet cluster, considering all three traffic integration policies, (i) shuffling, (ii) preemption and (iii) timely block.

Considering a RC flow, the source of delays are as follows: (1) queueing latency in the output ports of each node (both ESes and SWs) along its path from its source to its destinations, (2) the technological latency due to switches and (3) the transmission latencies on the physical links.

As in previous work, we assume an upper bound d_{tech} on the technological latency (2), which captures the latencies due to the FU and TP tasks and the switching fabric in a SW, see Sect. 3. The transmission latency (3) is determined by the frames size l_{RC_i} and the physical link rate C. The focus of the paper is on using network calculus to determine (1) the worst-case latencies at the output port h of a node. To compute the worst-case latency in a node port h, it is necessary to obtain the aggregate arrival curve $\alpha^h_{RC}(t)$ and remaining service curve $\beta^h_{RC}(t)$ of the intersecting RC flows. Figure 6 shows an overview of the network calculus model used to determine the worst-case latencies at the output port h, for two cases. Figure 6a for the shuffling traffic integration



(a) Shuffling integration policy



(b) Timely block integration policy

Fig. 6 Analysis model of a node output port

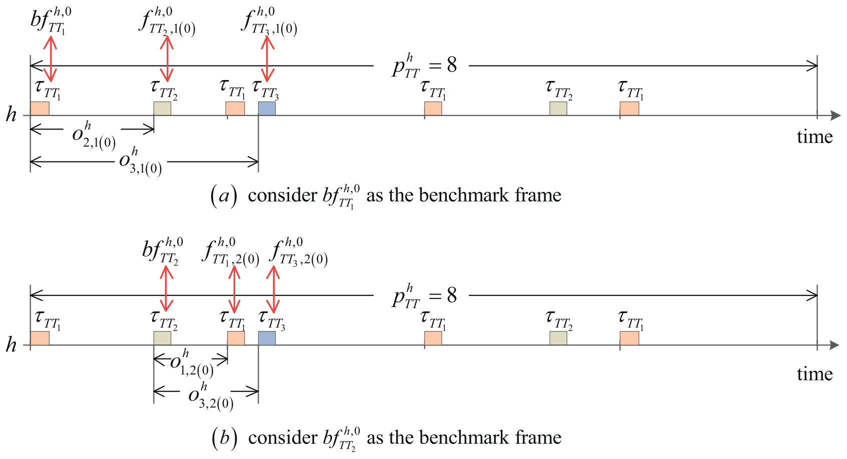


Fig. 7 An example of a TT schedule in a node port

policy, and Fig. 6b for timely block. From the definition of integration policies in Sect. 3, we can see that the worst-case latency with the preemption is the same with the timely block. This is because in the current TTEthernet protocol implementation, a RC frame that is preempted will restart its transmission from the beginning, and for the timely block all the intersecting RC flows are in FIFO order thus the service during blocked time cannot be used for other RC flows. These two cases of shuffling and timely block are presented separately in Sects. 6 and 7, respectively.

For the aggregate arrival curve $\alpha_{RC}^{h}(t)$ of the RC traffic, we use the model from the related work on ARINC 664p7 (Frances et al. 2006). This will be briefly described in Sect. 6.3. However, for the remaining service curve $\beta_{RC}^{h}(t)$ of RC traffic, we cannot use the related work on strict priority policy, since the TT traffic with higher priority is scheduled in fixed time slots, which is particular to TTEthernet. Therefore, with our approach, the remaining service curve for RC traffic is calculated by removing the service required by all TT flows going through h. The definition for $\beta_{RC}^{h}(t)$ is presented in Sect. 6.2 for the shuffling integration policy, $\beta_{RC\_S}^{h}(t)$, and in Sect. 7.2 for the timely block integration policy, $\beta_{RC\_T}^{h}(t)$.

Hence, we first have to capture the aggregate arrival curve $\alpha_{TT}^{h}(t)$ of intersecting TT flows going through an output port h. First, we model the arrival curve $\alpha_{TT_i}^{h}(t)$ for the single TT flow τ_{TT_i}. Then we put these together into an the aggregate arrival curve $\alpha_{TT}^{h}(t)$ for all intersecting TT flows in h by considering the relative offsets, based on its sending schedule S_S for the output port h. The re-ordering capability of the TTEthernet switches will not effect the worst-case timing analysis for RC traffic since the TT frames are re-ordered according to sending schedules. We present our approach to determine $\alpha_{TT}^{h}(t)$ for shuffling in Sect. 6.1. Section 7.1 presents how we consider the effect of additional waiting time (named as timely block curve $\gamma_{TT}^{h}(t)$) during which the node does not forward a RC frame when a TT frame is scheduled to be sent before the RC frame would complete its transmission, where we also need determine $\alpha_{TT}^{h}(t)$ (Sect. 6.1) for timely block.

6 Worst-case latency analysis of RC traffic with shuffling integration policy

6.1 Impact of TT traffic on RC traffic ($\alpha^h_{TT}(t)$)

In this section we discuss the aggregate arrival curve of TT traffic flows multiplexing in a node output port h to show its impact on the service of RC traffic. Recall that TT flows are transmitted according to sending static schedules S_S, determined offline as discussed in Sect. 3.1. For example, in Fig. 7a, there are three TT traffic flows on an output port h, τ_{TT_1}, τ_{TT_2} and τ_{TT_3}. They are respectively with periods $p_{TT_1} = 2\ ms$, $p_{TT_2} = 4\ ms$ and $p_{TT_3} = 8\ ms$. In our analysis, the sending times in the S_S schedule for a given output port h are captured using the concept of "relative offsets", see the later discussion in this section. For example, considering τ_{TT_1} as a reference, the sending time of τ_{TT_2} is captured by $o^h_{2,1(0)}$.

First, we discuss the arrival curve for a single TT flow τ_{TT_i}. Its arrival process on each node along its virtual link can be described as periodic transmission with the period p_{TT_i}, see Fig. 8 for an example.

Theorem 1 *The arrival curve of the single TT flow τ_{TT_i} in an node port h along its transmission is given by the staircase function*

$$\alpha^h_{TT_i}(t) = \begin{cases} l_{TT_i} \cdot \left\lceil \dfrac{t}{p_{TT_i}} \right\rceil, & t > 0 \\ 0, & t \le 0, \end{cases} \tag{4}$$

where l_{TT_i} is the size of the TT frame from τ_{TT_i} and p_{TT_i} is its period, see Fig. 8, for an illustration.

Proof For an arbitrary interval $(s, t]$, there are at most $n^h_{TT_i}(s,t)$ frames of τ_{TT_i} traversing an arbitrary node port h along its transmission path,

$$n^h_{TT_i}(s,t) = \left\lceil \frac{t-s}{p_{TT_i}} \right\rceil. \tag{5}$$

Assume that $R^h_{TT_i}(t)$ be the arrival process of the TT flow τ_{TT_i} in the node port h, we have

$$R^h_{TT_i}(t) - R^h_{TT_i}(s) \le l_{TT_i} \cdot n^h_{TT_i}(s,t) = l_{TT_i} \cdot \left\lceil \frac{t-s}{p_{TT_i}} \right\rceil = \alpha^h_{TT_i}(t-s). \tag{6}$$

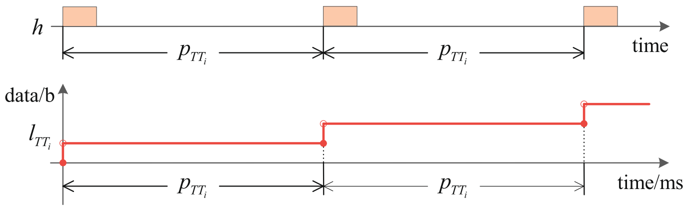


Fig. 8 Arrival curve of a single TT flow

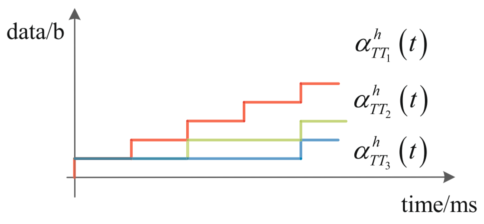


Fig. 9 Example of single arrival curves of three TT flows

Thus, the arrival curve for a single TT flow τ_{TT_i} in the node port h is given by $\alpha^h_{TT_i}(t) = l_{TT_i} \cdot \left\lceil \frac{t}{p_{TT_i}} \right\rceil$ for $t > 0$ and 0 otherwise. $\square$

Figure 9 depicts the arrival curves of each single TT flow τ_{TT_1}, τ_{TT_2} and τ_{TT_3} for the example in Fig. 7a.

However, there may be more than one TT flow, i.e., $\tau_{TT_i} = \{\tau_{TT_1}, \tau_{TT_2}, \ldots, \tau_{TT_n}\}$, traversing the same output port h of a node. And, different from flows in asynchronous networks, these intersecting TT flows cannot arrive at the same time. Their arrival times are determined by the schedule tables, which are created such that there is no contention between TT flows. Hence, it would be very pessimistic to simply sum arrival curves of each single TT flow τ_{TT_i}, $(i = 1, \ldots, n)$ intersecting. In the following, we propose how to construct the aggregate arrival curve $\alpha^h_{TT}(t)$ of these intersecting TT flows passing through h.

Even though the transmission of a single TT flow satisfies the property of strict periodicity, several TT flows passing through the same node port h, will not satisfy such a periodicity property. Since the schedule of the TT traffic is known a priori, we will know the schedule of the TT frames in an interval (s,t] relative to a given TT frame as reference. Conversely, any interval $(s, t]$ can be traversed if we consider all TT frames as a reference. This will lead to different possible aggregate arrival curves of intersecting TT flows in the node port h. Moreover, the number of benchmark frames is limited by N^h_{TT}, which is given by the following Eq. (8). Now, let us define the LCM cycle p^h_{TT} as the least common multiple (LCM) of periods of the intersecting TT flows sharing the output port h of a node,

$$p^h_{TT} = \mathrm{LCM}\left(p_{TT_1}, p_{TT_2}, \ldots, p_{TT_n}\right). \tag{7}$$

Then, we can then say that the arrangement of intersecting TT flows passing through the node port h is repeated in p^h_{TT}. In an interval $[0, p^h_{TT})$ of LCM cycle length, there are

$$N^h_{TT} = \sum_{i=1}^{n} N^h_{TT_i} = \sum_{i=1}^{n} p^h_{TT} / p_{TT_i} \tag{8}$$

TT frames from these intersecting TT flows, where $N^h_{TT_i}$ is the number of frames of τ_{TT_i} in such LCM cycle. By considering one of the TT frames as a reference, there will be N^h_{TT} possible aggregate arrival curves. Here, let us define such a reference TT frame as the "benchmark frame" $bf^{h,k}_{TT_i}$, in which the subscript TT_i represents the TT flow that such benchmark frame belongs to and the superscript k ($k = 0, \dots, N^h_{TT_i} - 1 = p^h_{TT} / p_{TT_i} - 1$) is an index of frames of τ_{TT_i} in the LCM cycle traversing through the node port h. In addition, we define $f^{h,g}_{TT_j,i(k)}$ as the frame g ($g = 0, 1, \cdots, N^h_{TT_j}$) of τ_{TT_j} arriving after the benchmark frame $bf^{h,k}_{TT_i}$, and g is the relative index with respect to k. This means that specifically $f^{h,0}_{TT_j,i(k)}$ represents the benchmark frame $bf^{h,k}_{TT_i}$ if $j = i$. Moreover, we use the concept of "relative offset" $o^h_{j,i(k)}$ to define the arrival interval between the benchmark frame $bf^{h,k}_{TT_i}$ of τ_{TT_i} and the first frames $f^{h,0}_{TT_j,i(k)}$ of TT flows τ_{TT_j} ($j \in [1, n]$) coming after $bf^{h,k}_{TT_i}$. Note that $o^h_{j,i(k)}$ equals to 0 if $j = i$.

For example, when the frame $bf^{h,0}_{TT_1}$ of τ_{TT_1} is considered as the benchmark frame in Fig. 7a, the relative offset $o^h_{2,1(0)}$ (resp. $o^h_{3,1(0)}$) is as shown in Fig. 7a. However, if $bf^{h,0}_{TT_2}$ is considered as the benchmark frame as depicted in Fig. 7b, then the relative offset $o^h_{1,2(0)}$ (resp. $o^h_{3,2(0)}$) is the interval between $bf^{h,0}_{TT_2}$ and $f^{h,0}_{TT_1,2(0)}$ (resp. $f^{h,0}_{TT_3,2(0)}$) arrived immediately after $bf^{h,0}_{TT_2}$.

Theorem 2 *The possible aggregate arrival curve of intersecting TT flows* $\{\tau_{TT_1}, \dots, \tau_{TT_n}\}$ *in an arbitrary node port h considering the benchmark frame* $bf^{h,k}_{TT_i}$ *is given by*

$$\alpha^h_{TT,i(k)}(t) = \sum_{j=1}^{n} \alpha^h_{TT_j}\left(t - o^h_{j,i(k)}\right), \tag{9}$$

where $\alpha^h_{TT_j}(t)$ *is the arrival curve of the single TT flow* τ_{TT_j} *and* $o^h_{j,i(k)}$ *is the relative offset between the benchmark frame* $bf^{h,k}_{TT_i}$ *and the adjacent frame* $f^{h,0}_{TT_j,i(k)}$ *of* τ_{TT_j}. *Note that* $o^h_{i,i(k)} = 0$.

Proof Since $bf^{h,k}_{TT_i}$ is considered as the benchmark frame, for any conditional arbitrary interval $(s, t]$ that $bf^{h,k}_{TT_i}$ is the first encountered frame, there are at most $n^h_{TT_j,i(k)}(s, t)$ frames of τ_{TT_j} ($j = 1, \dots, n$) arriving in node port h,

$$n^h_{TT_j,i(k)}(s,t) = \begin{cases} \left\lceil \dfrac{t - s - o^h_{j,i(k)}}{p_{TT_j}} \right\rceil, & t - s > o^h_{j,i(k)} \\ 0, & t - s \le o^h_{j,i(k)}. \end{cases} \tag{10}$$

Let $R^h_{TT,i(k)}(t)$ be the arrival process of intersecting TT flows in the node port h, then we have

$$\begin{aligned} R^h_{TT,i(k)}(t) - R^h_{TT,i(k)}(s) &\le \sum_{j=1}^{n} l_{TT_j} \cdot n^h_{TT_j,i(k)}(s,t) = \sum_{j=1}^{n} l_{TT_j} \cdot \left\lceil \frac{t - s - o^h_{j,i(k)}}{p_{TT_j}} \right\rceil \\ &= \sum_{j=1}^{n} \alpha^h_{TT_j}\left(t - s - o^h_{j,i(k)}\right). \end{aligned} \tag{11}$$

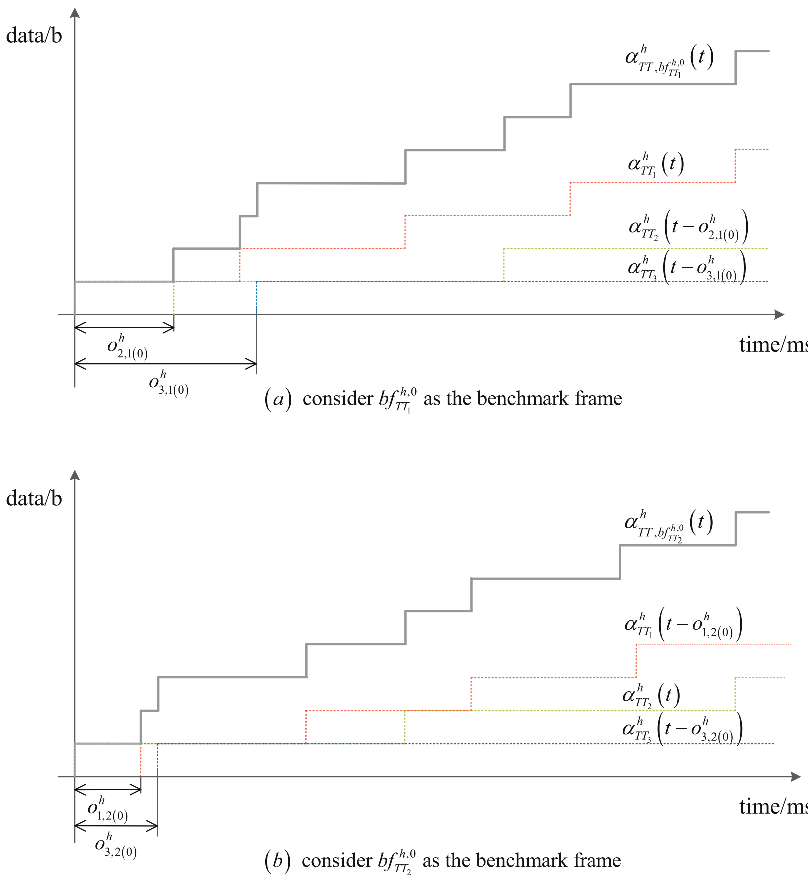


Fig. 10 Example of possible overall arrival curves of TT flows

Thus, the possible aggregate arrival curve based on $bf^{h,k}_{TT_i}$ is given by $\alpha^h_{TT,i(k)}(t) = \sum_{j=1}^{n} \alpha^h_{TT_j}\left(t - o^h_{j,i(k)}\right)$. □

Considering the TT flows from Figs. 7a and 10a shows one possible aggregate arrival curve if we consider $bf^{h,0}_{TT_1}$ of τ_{TT_1} as the benchmark frame. The red dotted line represents the arrival curve of single flow τ_{TT_1}, and yellow and blue dotted lines, respectively, represent the arrival curves of single flow τ_{TT_2} and τ_{TT_3} shifted to the right by relative offsets $o^h_{2,1(0)}$ and $o^h_{3,1(0)}$ according to the schedule considered in Fig. 7a. Similarly, Fig. 10b is another possible aggregate arrival curve considering $bf^{h,0}_{TT_2}$ of τ_{TT_2} as the benchmark frame. For our example, there are at most $\sum_{i=1}^{3} N^h_{TT_i} = 7$ possible aggregate arrival curves in Fig. 11.

Then the worst-case aggregate arrival curve $\alpha^h_{TT}(t)$ for the intersecting TT flows is the supremum or upper envelope of all these possible aggregate arrival curves, given

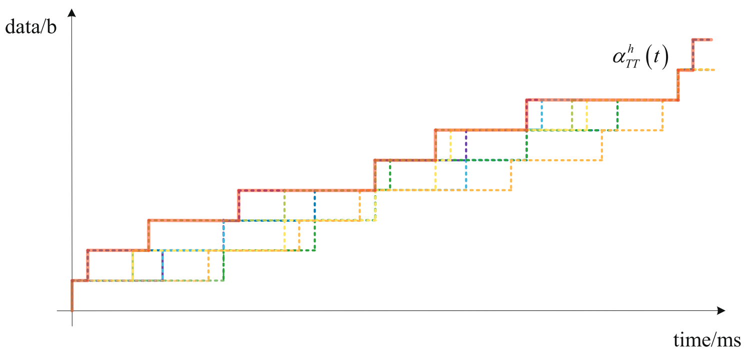


Fig. 11 Example of the overall arrival curve of TT flows

by

$$\alpha^h_{TT}(t) = \max_{i\in[1,n]} \left\{ \max_{k\in[0,N^h_{TT_i}-1]} \left\{ \alpha^h_{TT,i(k)}(t) \right\} \right\}, \tag{12}$$

where $\alpha^h_{TT,i(k)}(t)$ is given by Theorem 2. An example is shown in Fig. 11 in the red staircase curve. As mentioned, the number of possible aggregate arrival curves is limited by N^h_{TT}, thus it is feasible to enumerate.

6.2 Service curve $\beta^h_{RC\_S}(t)$ for aggregate RC flows with shuffling integration policy

With the shuffling integration, since a RC frame is non-preemptive, the service for the aggregate RC frames in a node port h is affected only by the time slots occupied by TT frames.

Theorem 3 *Assume that $\alpha^h_{TT}(t)$ is the aggregate arrival curve of TT flows, as determined in Sect. 6.1, in the node port h. Then, the service curve $\beta^h_{RC\_S}(t)$ for the aggregate RC flows in h with shuffling is given by*

$$\beta^h_{RC\_S}(t) = \left[\sup_{0\le s\le t} \left\{ C\cdot s - \alpha^h_{TT}(s) \right\} \right]^+ \tag{13}$$

where C is the physical link rate for the node output port h.

Proof Assume that $R^h_{TT}(t)$ and $R^{h*}_{TT}(t)$ (resp. $R_{RC}(t)$ and $R^{h*}_{RC}(t)$) are the arrival and departure process of n TT flows $\tau_{TT_1}, \ldots, \tau_{TT_n}$ (resp. m RC flows $\tau_{RC_1}, \ldots, \tau_{RC_m}$) crossing through the node port h. Let s be the start of the busy period[1] for TT traffic

[1] A busy period for level L is defined by an interval $[s, t)$ such that s and t are both idle times for level L and there is no idle time for level L in (s, t). An idle time s for level L is a time such that all frames with a priority greater than or equal to L generated before s have been processed at time s.

(Steven and Minet 2006). The amount of service for RC flows is lower bounded by the total output service minus the service for TT frames during $[s, t)$ in h,

$$R_{RC}^{h*}(t) - R_{RC}^{h*}(s) \geq C \cdot (t - s) - \left(R_{TT}^{h*}(t) - R_{TT}^{h*}(s)\right), \tag{14}$$

Since s is the beginning of the busy period, we have $R_{TT}^{h*}(s) = R_{TT}^{h}(s)$ and $R_{RC}^{h*}(s) = R_{RC}^{h}(s)$. Then

$$\begin{aligned} R_{RC}^{h*}(t) - R_{RC}^{h*}(s) &\geq C \cdot (t - s) - \left(R_{TT}^{h*}(t) - R_{TT}^{h*}(s)\right) \\ &= C \cdot (t - s) - \left(R_{TT}^{h*}(t) - R_{TT}^{h}(s)\right), \end{aligned} \tag{15}$$

and obviously due to $R_{TT}^{h*}(t) \leq R_{TT}^{h}(t)$,

$$R_{RC}^{h*}(t) - R_{RC}^{h*}(s) \geq C \cdot (t - s) - \left(R_{TT}^{h}(t) - R_{TT}^{h}(s)\right) \geq C \cdot (t - s) - \alpha_{TT}^{h}(t - s), \tag{16}$$

where $\alpha_{TT}^{h}(t)$ represents the arrival constraint of aggregate TT flow up to time t in the node port h. Since $R_{RC}^{h*}(t)$ is a wide-sense increasing function, we have

$$\begin{aligned} R_{RC}^{h*}(t) &\geq R_{RC}^{h*}(s) + \left[\sup_{0 \leq s \leq t} \left\{C \cdot (t - s) - \alpha_{TT}^{h}(t - s)\right\}\right]^{+} \\ &= R_{RC}^{h}(s) + \beta_{RC\_S}^{h}(t - s) \\ &\geq \inf_{0 \leq s \leq t} \left\{R_{RC}^{h}(s) + \beta_{RC\_S}^{h}(t - s)\right\} \\ &= \left(R_{RC}^{h} \otimes \beta_{RC\_S}^{h}\right)(t). \end{aligned} \tag{17}$$

□

Considering the aggregate arrival curve of TT flows from Fig. 11, the remaining service curve for RC flows is given by the blue solid line labeled $\beta_{RC\_S}^{h}(t)$ in Fig. 12. In addition, the dotted line represents the function $C \cdot t - \alpha_{TT}^{h}(t)$.

6.3 Worst-case end-to-end latency of RC flows with shuffling

Since RC traffic is compatible with ARINC 664p7, the arrival curve of each τ_{RC_i} in the source ES node port is constrained by the leaky bucket (Bauer et al. 2010),

$$\alpha_{RC_i}^{h_0} = \sigma_{RC_i}^{h_0} + \rho_{RC_i}^{h_0} \cdot t \tag{18}$$

where h_0 represents the output port of the first node (source ES) along the path, $\sigma_{RC_i}^{h_0} = l_{RC_i}$ and $\rho_{RC_i}^{h_0} = l_{RC_i} \big/ BAG_{RC_i}$. In addition, since RC flows are asynchronous, they

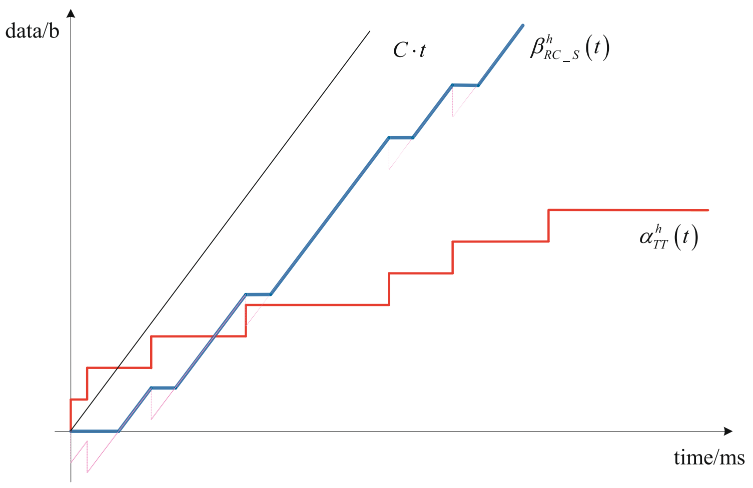


Fig. 12 Example of the service curve for RC flows in the shuffling integration policy

may simultaneously arrive and queue up in a node port. The aggregate arrival curve of competing RC flows is the sum of their respective arrival curves,

$$\alpha^h_{RC}(t) = \sum_{RC_i \in h} \alpha^h_{RC_i}(t) = \sum_{RC_i \in h} \sigma^h_{RC_i} + \sum_{RC_i \in h} \rho^h_{RC_i} \cdot t, \tag{19}$$

where $\alpha^h_{RC_i}(t)$ is the input arrival curve (Le Boudec and Thiran 2001) of τ_{RC_i} in the node port h. If $h \neq h_0$, $\alpha^h_{RC_i}(t)$ is given by

$$\alpha^h_{RC_i}(t) = \alpha^{h'}_{RC_i}\left(t + D^{h'}_{RC_i}\right) \tag{20}$$

where h' is the previous node port along the virtual link of τ_{RC_i}, and $D^{h'}_{RC_i}$ is the worst-case queuing latency of τ_{RC_i} in the node port h', which is given by Eq. (21).

According to the network calculus theory, the upper bound latency of a RC flow τ_{RC_i} in the node port h is given by the maximum horizontal deviation between the arrival curve $\alpha^h_{RC}(t)$ and the service curve $\beta^h_{RC\_S}(t)$ of the RC flows intersecting the node port h,

$$D^h_{RC_i\_S} = h\left(\alpha^h_{RC}, \beta^h_{RC\_S}\right) \tag{21}$$

An example of such horizontal deciation is shown in Fig. 13.

By disseminating the computation of latency bounds along a path dp_i of the virtual link vl_i of the flow τ_{RC_i}, the worst-case end-to-end latency with the shuffling integration policy is obtained by the sum of delays from its source ES to its destination ES,

$$D_{RC_i\_S} = \sum_h D^h_{RC_i\_S} + (h-1) \cdot d_{tech} \tag{22}$$

where $h \in dp_i$ and d_{tech} is the technological latency in a switch.

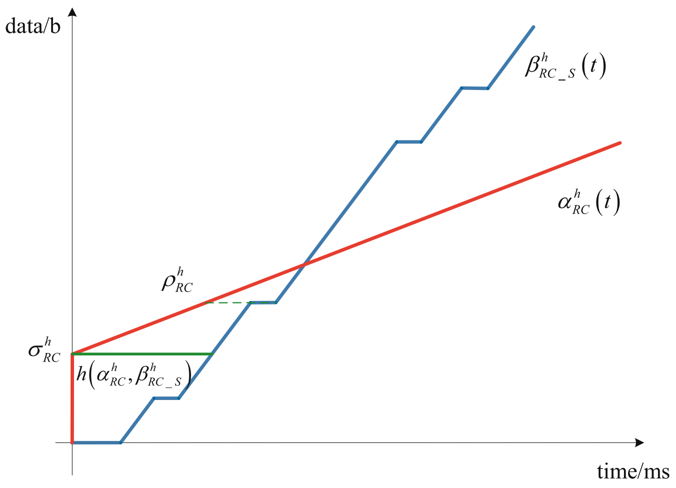


Fig. 13 Upper bound latency of RC flows in a node port in the shuffling integration

7 Worst-case latency analysis of RC traffic with timely block integration policy

7.1 Impact of timely block integration on RC traffic ($\gamma^h_{TT}(t)$)

For the timely block integration policy (see Sect. 3.2 for the definition of timely block), we should not only consider the arrival impact of TT traffic on RC traffic as we have done in Sect. 6.1, but also the impact of the waiting time before each TT frames. We define the blocking interval BI^h_{TT} to represent the worst-case waiting time before a TT frame due to the timely block integration. It is related to the maximum frame size $l^h_{RCmax} = \max_{1\leq i\leq m}\{l_{RC_i}\}$ of RC flows traversing the output port h and the interval I^h_{TT} which is the distance between such TT frame and its previous adjacent TT frame. If I^h_{TT} is larger than or equal to the maximum transmission time of RC frames, i.e., $I^h_{TT} \geq l^h_{RCmax}/C$, the blocking interval BI^h_{TT} equals to l^h_{RCmax}/C, otherwise I^h_{TT}. Let us consider the example in Fig. 14, where for simplicity, we assume that traversing RC flows and TT flows have the same frame size. Since the schedule table is repeated with the LCM cycle p^h_{TT}, the adjacent frame before $f^{h,0}_{TT_1}$ is $f^{h,3}_{TT_1}$. Then the interval $I^{h,0}_{TT_1}$ between the first TT frame $f^{h,0}_{TT_1}$ of τ_{TT_1} and its previous adjacent frame $f^{h,3}_{TT_1}$ is larger than the transmission time of a RC frame f^h_{RC}, and the worst-case blocking interval before $f^{h,0}_{TT_1}$ is $BI^{h,0}_{TT_1} = l^h_{RCmax}/C$. However, the worst-case blocking interval before the second TT frame $f^{h,1}_{TT_1}$ of τ_{TT_1} equals to $BI^{h,1}_{TT_1} = I^{h,1}_{TT_1}$ since $I^{h,1}_{TT_1} < l^h_{RCmax}/C$.

Different from the discussion of arrival curve of a single TT flow in Sect. 6.1, the impact of a timely block before each TT frame of a single TT flow is not periodic in a LCM period, due to the fact that the blocking intervals BI^h_{TT} may not equal to the same length (as mentioned in Fig. 14). Therefore, in order to traverse all the situations and easily discuss, we will consider different TT frames, the number of which is bounded

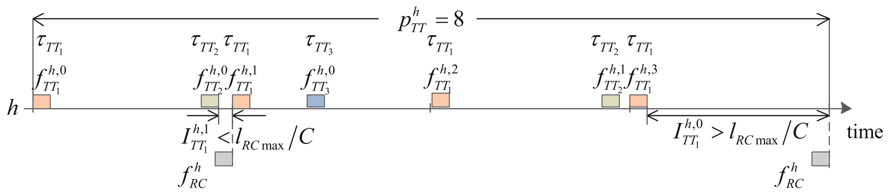

$p_{TT}^h = 8$
τ_{TT_1}
$f_{TT_1}^{h,0}$
τ_{TT_2} τ_{TT_1}
$f_{TT_2}^{h,0}$ $f_{TT_1}^{h,1}$
τ_{TT_3}
$f_{TT_3}^{h,0}$
τ_{TT_1}
$f_{TT_1}^{h,2}$
τ_{TT_2} τ_{TT_1}
$f_{TT_2}^{h,1}$ $f_{TT_1}^{h,3}$
h
$I_{TT_1}^{h,1} < l_{RC\max}/C$
f_{RC}^h
$I_{TT_1}^{h,0} > l_{RC\max}/C$
f_{RC}^h
time

Fig. 14 Time interval between two adjacent TT frames

in a LCM cycle, as a benchmark frame $bf_{TT_i}^{h,k}$ in the following. Now, we will give two Lemmas first.

Lemma 1 *The interval $I_{TT_j,i(k)}^{h,g}$ between any TT frame $f_{TT_j,i(k)}^{h,g}$ $(g = 0, \ldots, N_{TT_j}^h - 1,\ j = 1, \ldots, n)$ and its previous adjacent TT frame by considering $bf_{TT_i}^{h,k}$ $(k = 0, \ldots, N_{TT_i}^h - 1,\ i = 1, \ldots, n)$ as the benchmark frame in the node port h is,*

$$\begin{aligned} I_{TT_j,i(k)}^{h,g} = g \cdot p_{TT_j} + p_{TT}^h + o_{j,i(k)}^h \\ - \max_{1\le m\le n} \left\{ \left(\left\lceil \frac{g \cdot p_{TT_j} + p_{TT}^h + o_{j,i(k)}^h - o_{m,i(k)}^h}{p_{TT_m}} \right\rceil - 1 \right) \right. \\ \left. \cdot p_{TT_m} + o_{m,i(k)}^h + l_{TT_m} \right\}. \end{aligned} \tag{23}$$

Proof Without loss of generality, let us assume that benchmark frame $bf_{TT_i}^{h,k}$ arrives at time zero. Therefore, the first frame of τ_{TT_j} arrives at $o_{j,i(k)}^h$ after the benchmark frame. Then, the gth frame of τ_{TT_j} arrives at $o_{j,i(k)}^h + g \cdot p_{TT_j}$.

The adjacent TT frame arriving before $f_{TT_j,i(k)}^{h,g}$ may be from any TT flows τ_{TT_m} $(m = 1, \ldots, n)$ traversing the node port h. In order to ensure that there exists at least one frame from τ_{TT_m} before $f_{TT_j,i(k)}^{h,g}$ after the time zero, we consider that the frame of τ_{TT_j} arrives at $o_{j,i(k)}^h + g \cdot p_{TT_j} + p_{TT}^h$. And this is the same with the frame $f_{TT_j,i(k)}^{h,g}$ arrives at $o_{j,i(k)}^h + g \cdot p_{TT_j}$ since the TT schedule for the node port h is repeated by the LCM cycle p_{TT}^h. Then, there will be $N_{TT_m,i(k)}^{h,j(g)} = \left\lceil \left(g \cdot p_{TT_j} + p_{TT}^h + o_{j,i(k)}^h - o_{m,i(k)}^h \right) / p_{TT_m} \right\rceil$ TT frames of τ_{TT_m} before $f_{TT_j,i(k)}^{h,g}$. Thus, the nearest TT frame of τ_{TT_m} finishes its transmission at $\left(N_{TT_m,i(k)}^{h,j(g)} - 1 \right) \cdot p_{TT_m} + o_{m,i(k)}^h + l_{TT_m}$. By considering all the nearest TT frames from different TT flows, the adjacent TT frame before $f_{TT_j,i(k)}^{h,g}$ finishes transmission at $\max_{1\le m\le n} \left\{ \left(N_{TT_m,i(k)}^{h,j(g)} - 1 \right) \cdot p_{TT_m} + o_{m,i(k)}^h + l_{TT_m} \right\}$. □

Lemma 2 *The worst-case blocking interval $BI_{TT_j,i(k)}^{h,g}$ before a TT frame $f_{TT_j,i(k)}^{h,g}$ $(g = 0, \ldots, N_{TT_j}^h - 1)$ of the TT flow τ_{TT_j} $(j = 1, \ldots, n)$ by considering $bf_{TT_i}^{h,k}$ as the benchmark frame in the node port h is*

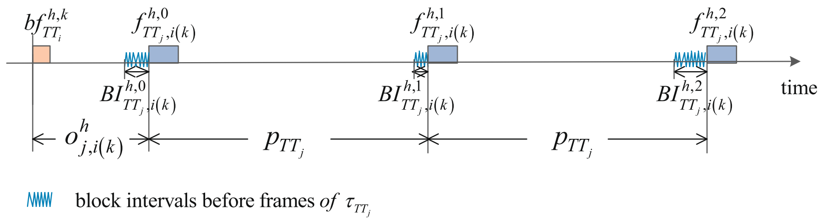


Fig. 15 Example of blocking intervals based on $bf_{TT_i}^{h,k}$

$$BI_{TT_j,i(k)}^{h,g} = min\left(\frac{l_{RCmax}^{h}}{C}, I_{TT_j,i(k)}^{h,g}\right), \tag{24}$$

where l_{RCmax} is the maximum frame size of RC flows sharing the output port h and $I_{TT_j,i(k)}^{h,g}$ is given by Lemma 1.

Theorem 4 *The possible overall timely block curve of all TT flows $\{\tau_{TT_1}, \ldots, \tau_{TT_n}\}$ in a node port h by considering the benchmark frame $bf_{TT_i}^{h,k}$ is*

$$\gamma_{TT,i(k)}^{h}(t) = \sum_{j=1}^{n} \sum_{g=0}^{N_{TT_j}^{h}-1} C \cdot BI_{TT_j,i(k)}^{h,g} \cdot n_{TB_j,i(k)}^{h,g}(t), \tag{25}$$

where

$$n_{TB_j,i(k)}^{h,g}(t) = \begin{cases} \left\lceil \dfrac{t - o_{j,i(k)}^{h} - g \cdot p_{TTj} + BI_{TT_j,i(k)}^{h,g}}{p_{TT}^{h}} \right\rceil, & t > \max\{0, o_{j,i(k)}^{h} + g \cdot p_{TTj} - BI_{TT_j,i(k)}^{h,g}\} \\ 0, & t \leq \max\{0, o_{j,i(k)}^{h} + g \cdot p_{TTj} - BI_{TT_j,i(k)}^{h,g}\}. \end{cases} \tag{26}$$

Proof For any conditional arbitrary interval $(s, t]$ that taking $bf_{TT_i}^{h,k}$ as the benchmark frame, the worst-case effect of timely block is related to the encountered TT frames during $(s, t]$. As mentioned before, since the blocking intervals before frames $f_{TT_j,i(k)}^{h,g}$ ($g = 0, \ldots, N_{TT_j}^{h}-1$) belonging to the same TT flow τ_{TT_j} may different within a LCM cycle, as shown in Fig. 15, we will discuss each of them separately. In the worst-case, there are at most $n_{TB_j,i(k)}^{h,g}(s,t) = \left\lceil (t - s - o_{j,i(k)}^{h} - g \cdot p_{TTj} + BI_{TT_j,i(k)}^{h,g})/p_{TT}^{h} \right\rceil$ blocking intervals with the maximum interval length $BI_{TT_j,i(k)}^{h,g}$ before $f_{TT_j,i(k)}^{h,g}$ in the node port h if $t - s > \max\{0, o_{j,i(k)}^{h} + g \cdot p_{TTj} - BI_{TT_j,i(k)}^{h,g}\}$, and 0 otherwise. Let $R_{TB,i(k)}^{h}(t)$ be the blocked process due to intersecting TT flows in the node port h, by

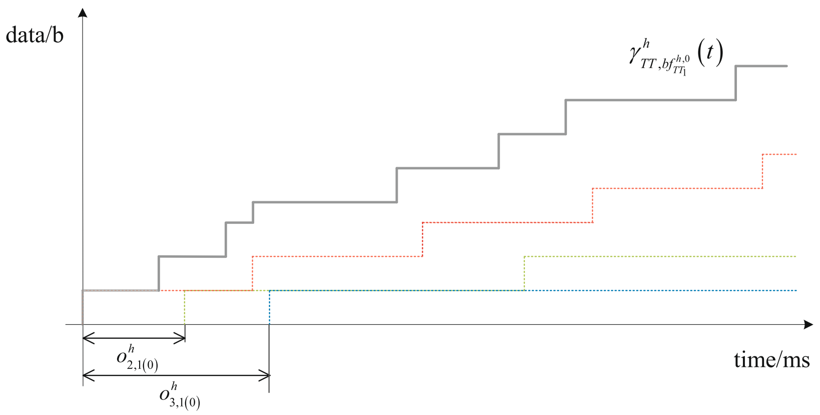


Fig. 16 Example of a possible overall timely block curve based on $bf^{h,0}_{TT_1}$

considering the maximum number of blocking intervals within $(s, t]$ from each TT frame of τ_{TT_j} and all the intersecting TT flows, we have

$$R^h_{TB,i(k)}(t) - R^h_{TB,i(k)}(s) \le \sum_{j=1}^{n} \sum_{g=0}^{N^h_{TT_j}-1} C \cdot BI^{h,g}_{TT_j,i(k)} \cdot n^{h,g}_{TB_j,i(k)}(s,t) \tag{27}$$

Thus, the possible overall timely block curve based on $bf^{h,k}_{TT_i}$ is given by

$$\gamma^h_{TT,i(k)}(t) = \sum_{j=1}^{n} \sum_{g=0}^{N^h_{TT_j}-1} C \cdot BI^{h,g}_{TT_j,i(k)} \cdot n^{h,g}_{TB_j,i(k)}(t). \tag{28}$$

□

The grey staircase curve in Fig. 16 shows a possible timely block curve $\gamma^h_{TT,1(0)}(t)$ for the benchmark frame $bf^{h,0}_{TT_1}$ from Fig. 7.

The worst-case overall timely block curve $\gamma^h_{TT}(t)$ in the node port h is the upper envelope of all the possible timely block curves $\gamma^h_{TT,i(k)}(t)$, an example is shown in Fig. 17 with a solid line,

$$\gamma^h_{TT}(t) = \max_{i\in[1,n]} \left\{ \max_{k\in[0,N^h_{TT_i}-1]} \left\{ \gamma^h_{TT,i(k)}(t) \right\} \right\}. \tag{29}$$

7.2 Worst-case end-to-end latency of RC flows with timely block

Theorem 7 gives the service curve $\beta^h_{RC\_T}(t)$ of aggregate RC flows in a node port h with the timely block integration policy.

Theorem 7 *The service curve $\beta^h_{RC\_T}(t)$ for the aggregate RC flows in h in the timely block integration policy is*

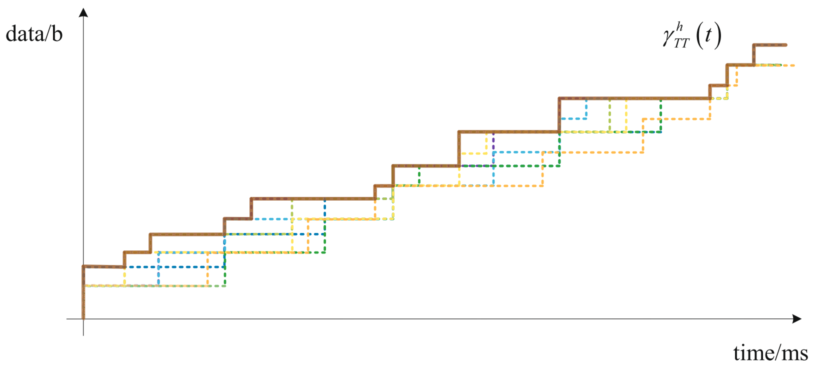


Fig. 17 Example of the overall timely block curve

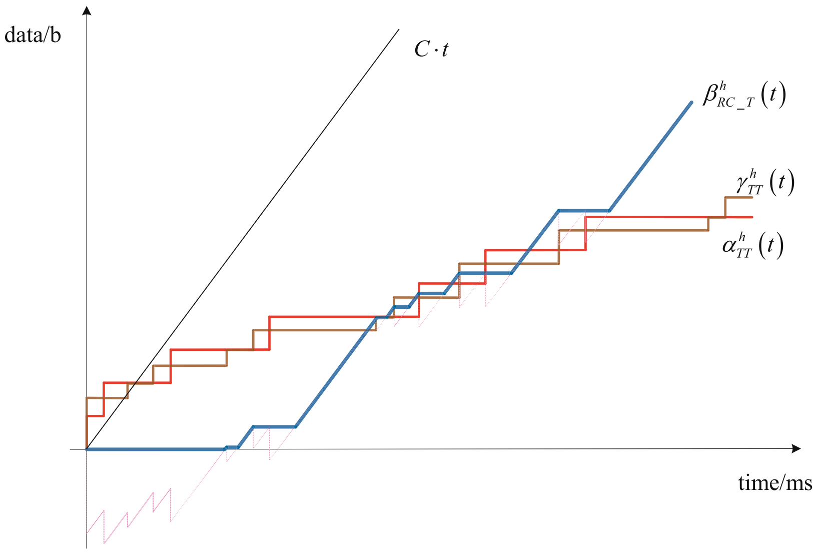


Fig. 18 Example of the service curve for RC flows in the timely block integration

$$\beta_{RC\_T}^{h}(t) = \left[\sup_{0 \le s \le t} \left\{ C \cdot s - \alpha_{TT}^{h}(s) - \gamma_{TT}^{h}(s) \right\} \right]^{+}, \tag{30}$$

where $\alpha_{TT}^{h}(t)$ *is the aggregate arrival curve of TT flows and* $\gamma_{TT}^{h}(t)$ *is the timely block curve in the node port* h. An example of the remaining service curve for RC flows is given by the blue solid line in Fig. 18, by considering the aggregate arrival curve of TT flows from Fig. 11 and the timely block curve from Fig. 17. Moreover, the dotted line represents the function $C \cdot t - \alpha_{TT}^{h}(t) - \gamma_{TT}^{h}(t)$.

Similarly, the worst-case latency of a RC flow τ_{RC_i} in a node port h is given as shown in Fig. 19 by

$$D_{RC_i\_T}^{h} = h\left(\alpha_{RC}^{h}, \beta_{RC\_T}^{h}\right). \tag{31}$$

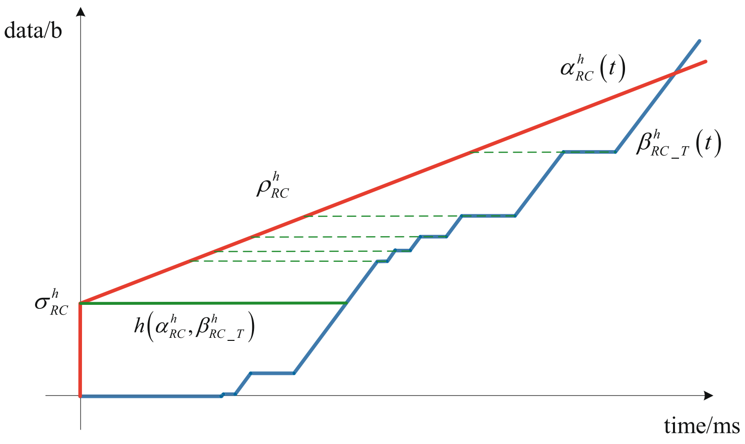


Fig. 19 Upper bound latency of RC flows in a node port

8 Experimental evaluation

In this section, for the evaluation of our approach, we used three synthetic test cases, $TC1$ to $TC3$, and one realistic test case, Orion Crew Exploration Vehicle (CEV), named here $TC4$ adapted from Tamas-Selicean (2014a), Paulitsch et al. (2011). Our proposed analysis was implemented in C++ combined with the dynamic link library of RTC toolbox (Wanderler and Thiele 2006), running on a computer with Intel Core i7-3520M CPU at 2.90 GHz and 4 GB of RAM.

The comparison of configurations of four test cases is given by Table 1. As we can see, $TC1$ is from Tamas-Selicean et al. (2015b) and has a topology of 12 ESes, 4 SWs,

Table 1 Configurations of four test cases

Test case	Number of ESes	Number of SWs	Number of VLs	Number of dataflow paths	Number of TT flows	Number of RC flows
TC1 (Tamas-Selicean et al. 2015b)	12	4	19	36	20	26
TC2 (Tamas-Selicean et al. 2015b)	10	5	55	99	58	51
TC3 (Tamas-Selicean et al. 2015b)	35	8	91	141	91	81
TC4 Orion CEV (Paulitsch et al. 2011)	31	15	100	190	100	87

Table 2 Parameters of traffic in $TC1$

Periods and frame sizes of TT traffic

Flow	Size (B)	Period (ms)	Flow	Size (B)	Period (ms)
TT1	1097	25	TT11	756	25
TT2	1167	10	TT12	86	2.5
TT3	1194	10	TT13	901	2.5
TT4	393	2	TT14	1339	5
TT5	43	2.5	TT15	1324	2.5
TT6	499	3.125	TT16	898	2
TT7	869	25	TT17	1420	6.25
TT8	1098	2.5	TT18	629	25
TT9	1077	25	TT19	195	10
TT10	1229	3.125	TT20	143	5

BAGs and frame sizes of RC traffic

Flow	Size (B)	BAG (ms)	Flow	Size (B)	BAG (ms)
RC1	1021	4	RC14	1360	16
RC2	1395	16	RC15	1332	8
RC3	134	4	RC16	728	16
RC4	1078	2	RC17	1127	16
RC5	590	8	RC18	156	4
RC6	946	2	RC19	378	8
RC7	784	16	RC20	1443	2
RC8	1120	2	RC21	1367	2
RC9	1361	8	RC22	519	16
RC10	20	4	RC23	522	2
RC11	1262	8	RC24	309	16
RC12	926	4	RC25	411	2
RC13	879	4	RC26	406	16

19 VLs and 36 dataflow paths (due to VL multicast reason), connected by dataflow link transmitting at 100 Mbps, running 20 TT flows and 26 RC flows. The details of the TT and RC flows are presented in Table 2. We use the same schedule tables from Tamas-Selicean's work (2015b) which were generated using the method from Tamas-Selicean et al. (2015a). In Table 3 we compare our method using network calculus for TTEthernet (called NC/TTE) with three other methods from related work. For each RC flow in $TC1$, Table 3 shows the WCDs obtained with each method, considering the timely block integration policy. The most recent WCD analysis method for RC traffic is the trajectory approach from Tamas-Selicean et al. (2015b), which we denote with Tam15. The method from Steiner (2011) is denoted with Ste11. None of these methods use network calculus. As we have argued in the introduction, using NC has the advantage of a well-established theory supported by certified tools used in the

Table 3 Comparison of different approaches for TC1

Flow	Ste11 (ms)	NC/SP (ms)	NC/TTE (ms)	Tam15 (ms)
RC1	4.44	3.59	0.82	0.77
RC2	19.94	6.53	2.30	1.81
RC3	20.68	4.33	1.32	1.10
RC4	10.16	4.93	1.72	1.53
RC5	13.04	5.24	1.70	1.35
RC6	14.62	6.17	2.14	1.68
RC7	3.12	3.13	0.76	0.79
RC8	14.09	4.43	1.68	1.22
RC9	8.43	5.79	1.84	1.38
RC10	17.81	5.87	2.19	1.48
RC11	11.30	4.43	1.50	1.34
RC12	15.30	5.06	1.59	1.17
RC13	12.86	6.32	2.25	1.43
RC14	16.69	6.76	2.48	1.80
RC15	14.62	6.37	2.09	1.60
RC16	13.67	6.23	1.80	1.61
RC17	18.52	6.07	2.01	1.70
RC18	5.57	3.56	1.19	0.86
RC19	20.73	4.33	1.32	1.08
RC20	20.07	6.44	1.80	1.75
RC21	20.52	6.71	2.55	1.85
RC22	13.24	5.37	1.51	1.32
RC23	19.74	5.64	1.99	1.33
RC24	11.15	5.37	1.44	1.23
RC25	11.11	2.99	0.65	0.65
RC26	7.47	5.58	1.66	1.35

safety-critical area. Without using our NC method NC/TTE, the only option with NC is to model the TT flows as high-priority periodic RC traffic that inherit their periods from TT schedule tables. We call such a method NC with strict priority, or NC/SP, and we have implemented NC/SP by using the NC approach for RC from Frances et al. (2006) extended to consider TT flows as high priority RC traffic using the method from Schmitt et al. (2003). Such a NC/SP method is the best that can be achieved with NC without considering the particularities of TTEthernet, as we do in this paper. In addition, in order to compare clearly, we also show the results of four methods in Fig. 20. We use NC/TTE as the baseline (the red "+" symbols forming a vertical line), i.e., the values on the x-axis show the percentage deviation of WCDs from the WCD obtained with our method NC/TTE, which is normalized to 100. The upper bounds calculated by NC/SP, Tam15 and Ste11 are normalized by

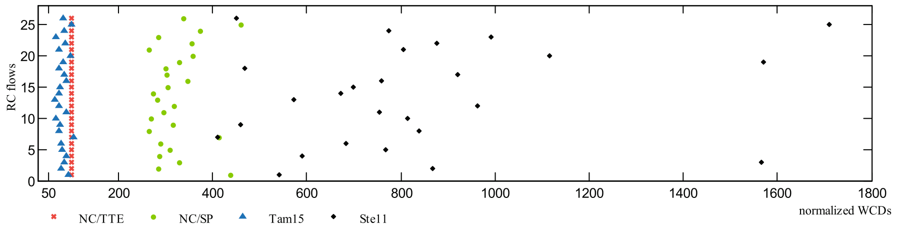


Fig. 20 Comparison of upper bounds by NC/TTE, NC/SP and Tam15

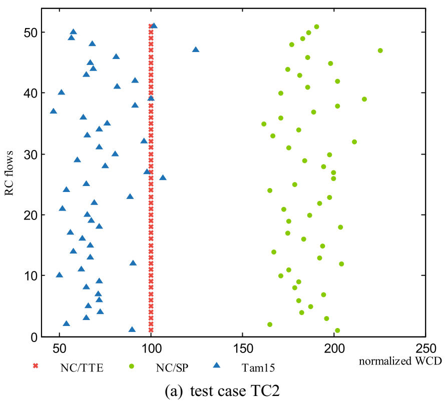
$$Nor^x = 100 - \left(\frac{D^{NC/TTE}_{RC_i\_T} - D^x_{RC_i\_T}}{D^{NC/TTE}_{RC_i\_T}} \times 100 \right). \tag{32}$$

where x represents one of the other methods, NC/SP, Tam15 and Ste11.

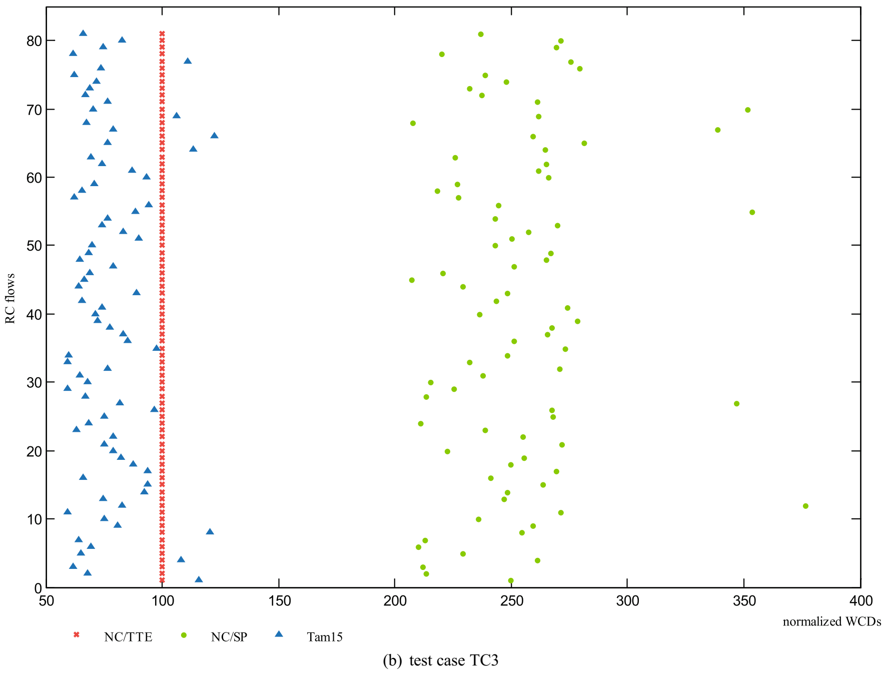
As we can see from Table 3 and Fig. 20, Ste11 obtains the largest WCDs, which is expected, since it pessimistically assumes that all RC frames in an output port will delay the RC frame under analysis. Tam15 obtains the best results, and it can improve the WCDs obtained by our NC/TTE method with 19% on average for $TC1$ (36% maximum). However, this improvement costs computation time: Tam15 needs 6 minutes for $TC1$ whereas our NC/TTE method completes in 7.94 s. As we will see for the larger test cases, Tam15 does not scale well, and is unable to obtain results for the largest test case we have used. This is because the time necessary of Tam15 to compute the WCDs depends on both the number of RC flows and on the size of the TT schedules. Also, Tam15 is not based on NC, which makes it impossible to integrate into the existing certified tool-chains. The results obtained with NC without using our method, i.e., NC/SP, are more pessimistic (larger WCDs) than our results. As we can see, our proposed analysis reduces the pessimism compared to NC/SP and Ste11 on average by 68 and 86%, respectively (78 and 94% maximum). The computation time of $TC1$ for NC/SP is 1.98 s. Note that in Table 3 we have considered timely block in order to perform a fair comparison, since this is the only integration policy handled by Ste11 and Tam15.

To show the scalability of our method, we apply it to larger test cases $TC2$ and $TC3$, which are having increasing complexity, as shown in Table 1. The compared worst-case end-to-end delays computed by NC/TTE, NC/SP and Tam15 on these two test cases are respectively presented in Fig. 21a and b. In Table 3 we have presented the WCD values for the different methods. Since these test cases are larger, and due to a lack of space, we present the WCDs visually in the figures. On the y-axis we have the RC flows, from RC1 to RC51 in Fig. 21a and from RC1 to RC81 in Fig. 21b. Each data-point in the figure corresponds to a WCD value. The results from Fig. 21 ($TC2$ and $TC3$) confirm the results from $TC1$. We can see that our network calculus approach NC/TTE, significantly reduces the pessimism compared with NC/SP. Our approach takes 17.26 and 31.62 s on $TC2$ and $TC3$, respectively, compared to Tam15 and NC/SP, which need 164 minutes and 446 minutes, and 2.02 s and 2.58 s, respectively.

In the last experiment, we use the real-life case study Orion CEV ($TC4$), derived from Tamas-Selicean (2014a), Paulitsch et al. (2011). The topology for the case study is shown in Fig. 22. The Orion CEV case has 31 ESes, 15 SWs, 100 VLs and 190



(a) test case TC2



(b) test case TC3

Fig. 21 Comparison of upper bounds by NC/TTE, NC/SP and Tam15

dataflow paths, connected by dataflow link transmitting at 100 Mbps, and running 100 TT flows and 87 RC flows, with parameters presented in Paulitsch et al. (2011). In the last set of experiments, we were interested to determine how our method handles the two traffic integration policies, namely, shuffling and timely block. We have run NC/TTE on $TC4$ and obtained WCDs for both cases. The compared results are shown in Fig. 23, where the upper bounds with timely block integration are normalized to 100. Our analysis has successfully computed on this large test case, for all traffic integration policies. We were unable to run Tam15 on TC4, since it reported that it runs out of memory, which, together with the large runtimes on $TC1$ and $TC3$, shows

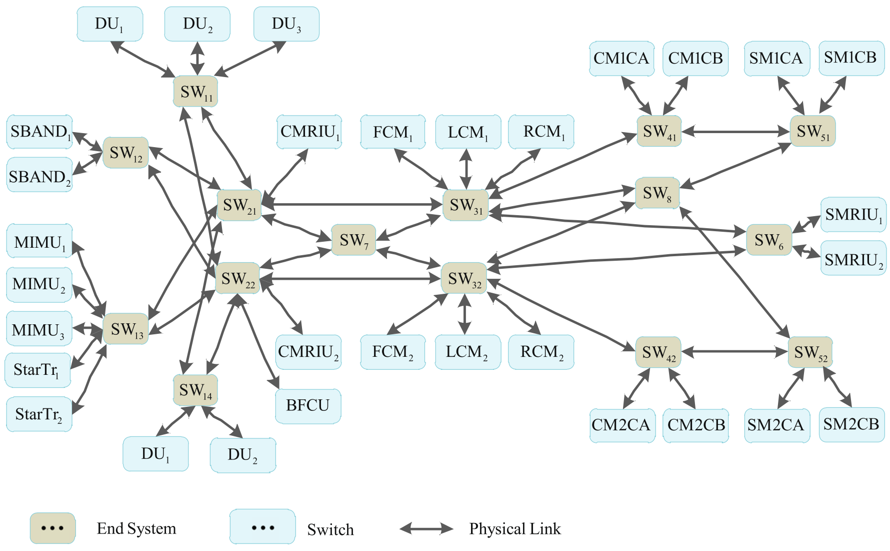
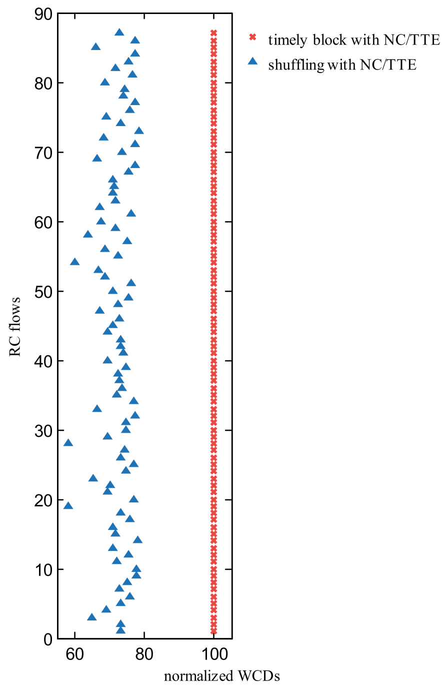


Fig. 22 Network topology of the Orion CEV, derived from Paulitsch et al. (2011)

Fig. 23 Comparison of upper bounds of shuffling and timely block by NC/TTE



that their method is not scalable. As expected, the latency bounds with the shuffling integration are lower than the bounds with timely block integration, since for the timely block integration, a RC frame will not be forwarded until there is an enough idle time interval for such the whole RC frame, and for the shuffling integration the higher priority TT frame is delayed until the RC frame finishes the transmission. On average, timely block leads to 40% larger WCDs for $TC4$. This number is related to the "porosity" of TT schedules. The porosity concept is introduced by Steiner (2011) and is related to the density (intensive or sparse) of TT scheduling slots. If TT frames are scheduled back-to-back without any intervals (pores) in-between, the RC frames will experience large delays. These delays can be reduced by introducing more porosity in the schedules.

9 Conclusions

This paper has proposed a network calculus approach for the TTEthernet to analyze the worst-case end-to-end latency of RC traffic. TTEthernet is suitable for mixed-critical systems as it offers three types of traffic classes, time-triggered, rate-constrained and best-effort.

The contribution of this paper is capturing the effects of all the integration policies, shuffling, preemption and timely block, on the latency bounds of RC traffic using network calculus, and the consideration of relative frame offsets of TT traffic to reduce the pessimism of the RC analysis. We construct the aggregate arrival curve for intersecting TT flows by considering the fixed offsets among TT flows, and the overall timely block curve due to the blocking interval before each TT frame. The experimental results and the comparison to the previously proposed analysis methods show that the network calculus approach is a viable approach for the analysis of TTEthernet, with the extensions proposed in the paper. Our NC/TTE method is scalable and can handle large problem sites, without a significant increase in pessimism compared to Tam15. In addition, our NC/TTE method reduces the pessimism of worst-case delay calculated by the traditional network calculus approach NC/SP for strict priority without a significant increase in computation time.

Acknowledgements This study is co-supported by the National Natural Science Foundation of China (61073012 and 61301086) and the Fundamental Research Funds for the Central Universities of China (YWF-14-DZXY-023 and YWF-15-GJSYS-055).

References

Adnan M, Scharbarg JL, Ermont J, Fraboul C (2010) Model for worst case delay analysis of an AFDX network using timed automata. In: IEEE conference on emerging technologies and factory automation, Bilbao, pp 1–4

ARINC (2009) ARINC 664P7: Aircraft data network, Part 7. Avionics full-duplex switched Ethernet network

Bauer H, Scharbarg JL, Fraboul C (2010) Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. IEEE Trans Ind Inform 6(4):521–533

Boyer M, Fraboul C (2008) Tightening end to end delay upper bound for AFDX network calculus with rate latency FIFO servers using network calculus. In: IEEE international workshop on factory communication systems, Dresden, pp 11–20

Chang CS (2000) Performance guarantees in communication networks, 1st edn. Springer, London

Cruz C (1991) A calculus for network delay, part I, network elements in isolation. IEEE Trans Inform Theory 37(1):114–131

Cummings R, Richter K, Ernst R, Diemer J, Ghosal A (2012) Exploring use of Ethernet for in-vehicle control applications: AFDX, TTEthernet, EtherCAT, and AVB. SAE Int J Passeng Cars Electron Electr Syst 5(1):72–88

Danielis P, Skodzik J, Altmann V et al (2014) Survey on real-time communication via Ethernet in industrial automation environments. In: Proceedings of the 2014 IEEE emerging technology and factory automation (ETFA), IEEE, pp 1–8

Decotignie JD (2005) Ethernet-based real-time and industrial communications. Proc IEEE 93(6):1102–1117

ETG (2013) ETG.1000.1 EtherCAT Specification. EtherCAT Technology Group

Frances F, Fraboul C, Grieu J (2006) Using network calculus to optimize the AFDX network. In: Proceedings of 3rd European congress embedded real-time software, Toulouse, France

Fletcher M (2009) Progression of an open architecture: from Orion to Altair and LSS. White paper S65-5000-20-0, Honeywell, International

Gavrilut VM, Tamas-Selicean D, Pop P (2015) Fault-tolerant topology selection for TTEthernet networks. In: Safety and reliability of complex engineered systems: proceedings of the 25th European safety and reliability conference. CRC Press, London, pp 4001–4010

Grieu J (2005) Analyse et valuation de techniques de com-mutation Ethernet pour l'interconnexion des systmes avioniques. Ph.D. dissertation, Laboratory Space and Aeronautical Telecommunications, Toulouse, France

IEEE (2011) IEEE 802.1BA—IEEE standard for local and metropolitan area networks—audio video bridging (AVB) systems. The Institute of Electrical and Electronics Engineers, Inc

IEEE (2012) IEEE 802.3—IEEE standard for Ethernet. The Institute of Electrical and Electronics Engineers, Inc

IEEE (2015) IEEE P802.1Qbv (Draft 3.1)—enhancements for scheduled traffic, IEEE Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/802.1bv.html

Kopetz H (1991) Event-triggered versus time-triggered real-time systems. In: Karshmer A, Nehmer J (eds) Operating systems of the 90s and beyond. Lecture notes in computer science, vol 563. Springer, Berlin, pp 86–101

Kopetz H, Grunsteidl G (2005) The time-triggered Ethernet (TTE) design. In: 8th IEEE international symposium on object-oriented real-time distributed computing, Washington DC, pp 22–33

Kopetz H (2011) Real-time systems: design principles for distributed embedded applications. Springer, Berlin

Le Boudec JY, Thiran P (2001) Network calculus: a theory of deterministic queuing systems for the internet. In: Lecture notes on computer science, 5th edn. Springer, New York

Lee YH, Rachlin E, Scandura PA (2005) Safety and certification approaches for Ethernet-Based Aviation Databuses. Technical report DOT/FAA/AR-05/52, Federal Aviation Administration

Li XT, Scharbarg J, Fraboul C (2010) Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis. In: IEEE conference on emerging technologies and factory automation, Bilbao, pp 1–8

Mabille E, Boyer M, Fejoz L et al (2013) Towards certifying network calculus. Interactive theorem proving. Springer, Berlin

Mauclair C, Durrieu G (2013) Analysis of real-time networks with monte carlo methods. EUCASS Proc Ser 6:501–514

Obermaisser R (2011) Time-triggered communication. CRC Press, London

Paulitsch M, Schmidt E, Gstottenbauer B, Scherrer C, Kantz H (2011) Time-triggered communication (industrial applications). In: Time-triggered communication. CRC Press, London, pp 121–152

Pedreiras P, Almeida L, Buttazzo G (2005) FTT-Ethernet: a flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems. IEEE Trans Ind Inform 1(3):162–172

SAE (2011) AS6802: Time-triggered Ethernet. SAE International

Scharbarg JL, Ridouard F, Fraboul C (2009) A probabilistic analysis of end-to-end delays on an AFDX avionic net-work. IEEE Trans Ind Inform 5(1):28–49

Schmitt J, Hurley P, Hollick M et al (2003) Per-flow guarantees under class-based priority queueing. In: IEEE global telecommunications conference, pp 4169–4174

Schneele S, Geyer F (2012) Comparison of IEEE AVB and AFDX. In: Proceedings of the digital avionics systems conference, pp 7A1C1–7A1C9

Steinbach T, Lim HT, Korf F, Schmidt TC, Herrscher D, Wolisz A (2012) Tomorrows in-car interconnect? A competitive evaluation of IEEE 802.1 AVB and time-triggered Ethernet (AS6802). In: IEEE vehicular technology conference. IEEE Press, New York, pp 1C5

Steiner W, Bauer G, Hall B, Paulitsch M, Varadarajan S (2009) TTEthernet dataflow concept. In: 8th IEEE international symposium on network computing and applications, Cambridge, MA, pp 319–322
Steiner W (2010a) An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In: IEEE 31st real-time systems symposium (RTSS), San Diego, CA, pp 375–384
Steiner W, Dutertre B (2010b) SMT-Based formal verification of a TTEthernet synchronization function. Formal methods for industrial critical systems. Springer, Berlin
Steiner W (2011) Synthesis of static communication schedules for mixed-criticality systems. In: 14th IEEE international symposium on object/component/service-oriented real-time distributed computing workshops, Newport Beach, CA, pp 11–18
Steiner W (2016) Deterministic Ethernet for real-time and critical applications. In: 12th IEEE world conference on factory communication systems, Aveiro, Portugal
Steven M, Minet P (2006) Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class. In: 20th international parallel and distributed processing symposium
Suen J, Kegley R, Preston J (2013) Affordable avionic networks with Gigabit Ethernet assessing the suitability of commercial components for airborne use. In: Proceedings of SoutheastCon, pp 1C6
Tamas-Selicean D, Marinescu SO, Pop P (2012a) Analysis and optimization of mixed-criticality applications on partitioned distributed architectures. In: 7th IET international conference on system safety, incorporating the cyber security conference, Edinburgh, pp 21–26
Tamas-Selicean D, Pop P, Steiner W (2012b) Synthesis of communication schedules for TTEthernet-based mixed-criticality systems. In: Proceedings of the eighth IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis, New York, pp 473–482
Tamas-Selicean D (2014a) Design of mixed-criticality applications on distributed real-time systems. PhD Thesis, Technical University of Denmark
Tamas-Selicean D, Pop P, Steiner W (2015a) Design optimization of TTEthernet-based distributed real-time systems. Real-Time Syst 51(1):1–35
Tamas-Selicean D, Pop P, Steiner W (2015b) Timing analysis of rate constrained traffic for the TTEthernet communication protocol. In: International symposium on real-time computing (ISORC)
Wanderler E, Thiele L (2006) Real-time calculus (RTC) toolbox. http://www.mpa.ethz.ch/Rtctoolbox
Zafirov A (2013) Modeling and simulation of the TTEthernet communication protocol. Masters thesis, Technical University of Denmark
Zhao LX, Xiong HG, Zheng Z, Li Q (2014) Improving worst case latency analysis for rate-constrained traffic in the Time-Triggered Ethernet Network. IEEE Commun Lett 18(11):1927–1930

Luxi Zhao received the PhD in communication and information system from the Beihang University, Beijing, China, in 2017. Since 2017, she will be a postdoc at Technical University of Denmark (DTU) for two years. Her main research interest concerns worst-case analysis and performance evaluation on real-time networks.

Paul Pop received his Ph.D. degree in computer systems from Linköping University in 2003, he has been an associate professor at DTU Compute, Technical University of Denmark, and since 2016 he is a Professor of Cyber-Physical Systems. His research is focused on developing methods and tools for the analysis and optimization of dependable embedded systems. In this area, he has published over 130 peer-reviewed papers, 3 books and 7 book chapters. He and has received the best paper award at DATE 2005, RTIS 2007, CASES 2009, MECO 2013 and DSD 2016. He has also received the EDAA Outstanding Dissertations Award (co-supervisor) in 2011. His research has been highlighted as "The Most Influential Papers of 10 Years DATE". He is the director of DTU's IoT Research Center and has coordinated the Danish national InfinIT Safety-Critical Systems Interest Group. He is the chairman of the IEEE Danish Chapter on Embedded Systems. He has served as technical program committee member on several conferences, such as DATE and ESWEEK.

Qiao Li received his Ph.D. degree in communication systems from Beihang University in 2005, and he has been an associate professor in School of Electronics and Information Engineering, Beihang University, Beijing, China. His research is focused on digital communication technology, avionics systems and real-time networks.

Junyan Chen received the Ph.D. degree in Electronic Information Engineering from Beihang University, Beijing, China, in 2014. His main research interest is real-time performance evaluation in avionics context.

Huagang Xiong received his Ph.D. degree in communication systems from Beihang University in 1998, and he has been a professor in School of Electronics and Information Engineering, Beihang University, Beijing, China. His main fields of interest are communication network theory and avionics system and synthesis.