Compositional Analysis of Switched Ethernet Topologies

Reinhard Schneider, Licong Zhang, Dip Goswami, Alejandro Masrur, Samarjit Chakraborty
Institute for Real-Time Computer Systems, TU Munich, Germany
(reinhard.schneider@rcs.ei.tum.de, licong.zhang@rcs.ei.tum.de,
dip.goswami@tum.de, masrur@rcs.ei.tum.de, samarjit@tum.de)

***Abstract*—In this paper we study distributed automotive control applications whose tasks are mapped onto different ECUs communicating via a switched Ethernet network. As traditional automotive communication buses like CAN, FlexRay, LIN and MOST are gradually reaching their performance limits because of the increasing complexity of automotive architectures and applications, Ethernet-based in-vehicle communication systems have attracted a lot of attention in recent times. However, currently there is very little work on systematic timing analysis for Ethernet which is important for its deployment in safety-critical scenarios like in an automotive architecture. In this work, we propose a compositional timing analysis technique that takes various features of switched Ethernet into account like network topology, frame priorities, communication delay, memory requirement on switches, performance, etc. Such an analysis technique is particularly suitable during early design phases of automotive architectures and control software deployment. We demonstrate its use in analyzing mixed-criticality traffic patterns consisting of messages from performance-oriented control loops and timing-sensitive real-time tasks. We further evaluate the tightness of the obtained analytical bounds with an OMNeT++ based network simulation environment, which involves long simulation time and does not provide formal guarantees.**

I. Introduction

The use of Ethernet as an in-vehicle communication technology has lately attracted a lot of attention within the automotive electronics domain. This is primarily because of the recent developments in unshielded twisted pair Ethernet by Broadcom (blog.broadcom.com) and other companies (www.its-jp.org), which makes Ethernet deployment in cars much more cost effective, and hence feasible, compared to other technologies that require shielded cables to prevent electromagnetic interference. However, there is very little work on timing analysis of Ethernet, whereas tight timing bounds are required to map safety-critical control tasks on a distributed automotive architecture with multiple communicating ECUs.

In this paper we propose a timing and performance analysis technique for *full-duplex switched* Ethernet networks in distributed automotive architectures (see Fig 1). The performance of control applications mapped onto such architectures depends on the delays experienced by the control-related signals, which in turn depend on the *topology* of the network, and the specific characteristics of the *switches* that are used.

We also show that the Ethernet topology, frame priorities, and switch parameters have a significant influence on control performance, and therefore need to be systematically determined. The timing analysis technique proposed in this paper can serve as an important design tool for configuring both the Ethernet network and switch parameters, and may also drive task mapping and priority assignment decisions for the applications. Our main focus in this paper is the timing analysis technique; its use for task mapping, priority assignment and other related design decisions in the context of Ethernet based architectures will be a subject for future study.

A. Related work

In spite of several advantages, the major bottleneck in applying Ethernet based networks in real-time and safety-critical domains is to provide performance guarantees such as worst-case delays, jitter, and control performance. Recently, there have been significant efforts on timing and performance analysis of Ethernet networks from both industrial [1] and academic [2]–[4] communities. Another important research question is the impact of such timing/performance properties on the feedback control loops [3], [5], [6]. While these works have made notable contribution in this area, many of them are ad-hoc solutions for a specific network and communication pattern. Moreover, the design problem becomes significantly more complex when the mixed-criticality traffic such as real-time and feedback control applications are implemented together over such networks. A big challenge here is to establish a compositional approach that scales to different network topologies, communication patterns, and provides safe yet not too pessimistic performance guarantees.

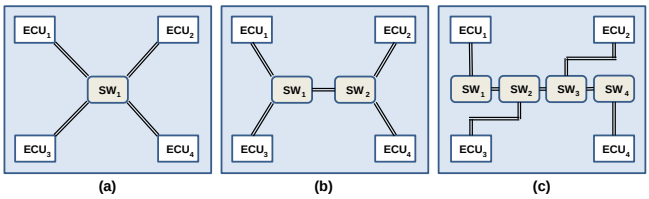

Fig. 1. Switched Ethernet topologies: (a) Star topology (b) Twin Star topology (c) Line topology.

In this context, the common system-level performance analysis tools are real-time calculus (RTC) [7], network calculus [8]–[10] and SymTA/S [2], [11]. In [9], [10], the timing analysis frameworks are presented to model an Ethernet switch. In these works, the timing bounds are derived based on the assumption of maximum frame size for each frame (i.e, each frame is assumed to have a maximum frame length). In general, such assumption introduces pessimism, which is further aggravated in the case of compositional analysis.

B. Contributions

In this work, we present a compositional analysis technique for switched Ethernet networks that allows studying the performance of automotive systems at early design phases. A model of an Ethernet switch is presented to compute the worst-case delays and the memory requirement in a switch for a given frame priority assignment. Next, we utilize this switch model to conduct performance analysis of various Ethernet topologies. We consider a traffic pattern consisting of real-time frames and control-related frames. Our analysis is based on RTC and is especially useful to compare different network topologies regarding multiple performance properties such as:

- the memory requirement on Ethernet switches,
- the performance of distributed feedback control applications closed over the network
- the communication delays of data paths (routes) across multiple switches in various topologies.

Towards evaluating the tightness of our analysis, we developed a simulation environment using OMNeT++, a discrete-event

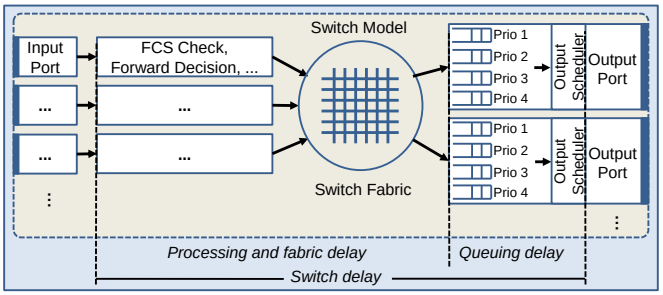

Fig. 2. Switch model under consideration.

simulation tool. The main focus in this paper lies on exploring a number of relevant design parameters and their impact on various performance measures. The proposed analysis technique is fully compositional and scalable, paving the way for optimizing design procedures to improve performance of future automotive systems based on Ethernet networks.

II. Full-duplex switched Ethernet

In this work, we are concerned with a full-duplex switched Ethernet (IEEE 802.1D[1]) featuring priority operation and 100Mbps links. In the setting under consideration, multiple ECUs are connected in a network consisting of *Fast Ethernet (100Mbps) full-duplex* links and a number of *switches*. The full-duplex operation allows the separation of incoming and outgoing traffic and thus contention on the links can be avoided. In such a network, an ECU can be either directly connected to another ECU or via one or more switches (which is a more common configuration). The network can employ different topologies, including star, twin star, line (see Fig. 1), ring and other topologies. The switches forward the Ethernet frames to their corresponding destination ECUs and mainly consist of the following components as depicted in Fig. 2:

- **Input ports:** Each switch has multiple ports, which can normally both serve as input ports and output ports. These ports are capable of receiving frame streams from the connected Ethernet link.
- **Processing and switch fabric:** The stream of frames received by input ports are then processed (e.g., FCS checks, forwarding decisions, etc.), and forwarded to the corresponding output port(s) through the switch fabric.
- **Output ports:** An output port is composed of a number of queues and an *output scheduler*. The frames received by the switch have pre-defined fixed priorities (there can be at most 8 priorities). Frames with different priorities are stored in separate queues and are transmitted on an output port according to the output scheduler.

Depending on the topology, a frame from a sender ECU passes through one or more switches to reach a receiver ECU. For example, a frame from ECU_1 to ECU_4 travels through only one switch in the star topology (Fig. 1(a)) while the same frame passes through two switches in the twin star topology (Fig. 1(b)). In this process, the communication delay experienced by a frame consists of two components:
(i) **Transmission delay:** This is the transmission time over the Ethernet links. It depends on the link's bandwidth, the frame size, and the number of links in the data path (which again depends on the topology). For example, a 100-byte-frame from ECU_1 to ECU_4 in the star topology (Fig. 1(a)) experiences a transmission delay of $8\mu s$ per link and a total transmission delay of $16\mu s$. Therefore, the transmission delay of a particular frame is constant for a given topology. Further, the transmission of two consecutive frames is separated by an inter-frame gap (IFG) of 12 bytes. In our analysis, we take this into account by extending the frame size by 12 bytes.
(ii) **Switch delay:** This refers to the delay in the switches (see Fig. 2). The switch delay is made up of two parts:

- **Processing and fabric delay:** This can be modeled as a constant delay and is usually in the range of 3 to 10 μs.
- **Queuing delay:** As described above, the frames with different priorities are stored in the corresponding output queues, and are transmitted according to an output scheduling policy. This scheduler can follow any arbitration policy (e.g. weighted round robin or fixed priority non-preemptive).

III. Mathematical Background

In this section, we present the theoretical background of RTC [7] which we use for our analysis.

Event model: Data streams are modeled using a *count-based abstraction* where an arrival pattern of a stream is modeled as a cumulative function $R(t)$ denoting the number of events that arrive during the time interval $(0, t]$. The maximum and minimum number of events that are recorded during *any* time interval of length Δ is represented by a pair of *arrival functions* $\alpha = (\alpha^u, \alpha^l)$ defined as

$$\forall \Delta \geq 0,\ \forall\, t \geq 0 :\quad \alpha^l(\Delta) \leq R(\Delta + t) - R(t) \leq \alpha^u(\Delta).$$

Arrival curves allow for an expressive characterization of event streams which are able to represent standard event models, e.g., *periodic*, *periodic with jitter* and *sporadic*, as well as arbitrary arrival patterns. Further, α can also be expressed in terms of resource units available in any time interval of length Δ according to workload transformations [12]:

$$\bar{\alpha}^u = \mathcal{W}^u_\alpha(\alpha^u), \quad \bar{\alpha}^l = \mathcal{W}^l_\alpha(\alpha^l) \tag{1}$$

where $\mathcal{W}^u_\alpha(e)$ and $W^l_\alpha(e)$ are functions denoting the maximum and minimum number of resource units (e.g., processor cycles) required to process e consecutive events.

Resource model: Similarly, resource capacities are captured by a cumulative function $C(t)$ denoting the number of events that can be processed by a resource in the time interval $(0, t]$. The maximum and minimum number of events that can be processed in *any* time interval of length Δ can be bounded by a pair of *service functions* $\beta = (\beta^u, \beta^l)$ defined as

$$\forall \Delta \geq 0,\ \forall\, t \geq 0 :\quad \beta^l(\Delta) \leq C(\Delta + t) - C(t) \leq \beta^u(\Delta).$$

Similar to resource-based arrival curves, β can also be expressed in terms of the maximum and minimum number of available resource units:

$$\beta^u = \mathcal{W}^u_\beta(\bar{\beta}^u), \quad \beta^l = \mathcal{W}^l_\beta(\bar{\beta}^l) \tag{2}$$

where $\mathcal{W}^u_\beta(r)$ and $W^l_\beta(r)$ denote the maximum and minimum number of successive events that can be processed with r resource units.

Performance analysis: The bounds on the output arrival functions $\alpha' = (\alpha^{u'}, \alpha^{l'})$, and remaining service functions

[1]It should be noted that our proposed model and result is also valid for networks with VLAN operations ($IEEE802.1Q$) and can be adapted to the AVB standards. In addition, we only consider the performance up to the Data Link Layer, i.e., we do not include protocols like TCP, UDP and IP. However, the model and result of this paper also applies to networks with such protocols implemented.

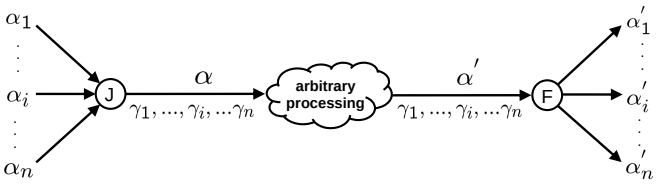

Fig. 3. *Join* (J) and *fork* (F) operations on the structured event stream α. The corresponding ECCs γ_i of sub-streams α_i remain unaffected after arbitrary processing of the structured event stream α.

$\beta' = (\beta^{u'}, \beta^{l'})$ for a greedy processing component can be computed as follows:

$$\alpha^{u'} = \min\{(\alpha^u \otimes \beta^u) \oslash \beta^l,\ \beta^u\} \quad (3)$$

$$\alpha^{l'} = \min\{(\alpha^l \oslash \beta^u) \otimes \beta^l,\ \beta^l\} \quad (4)$$

$$\beta^{u'} = (\beta^u - \alpha^l)\ \overline{\oslash}\ 0 \quad (5)$$

$$\beta^{l'} = (\beta^l - \alpha^u)\ \overline{\otimes}\ 0 \quad (6)$$

where the (min,+) convolution $\otimes$ and deconvolution $\oslash$ operators are defined as: $\forall t \in \mathbb{R}^+$

$$(f \otimes g)(t) = \inf\{f(s) + g(t-s)\ \mid\ 0 \le s \le t\},$$
$$(f \oslash g)(t) = \sup\{f(t+u) - g(u) \mid u \ge 0\}.$$

The (max,+) convolution $\overline{\otimes}$ and deconvolution $\overline{\oslash}$ operators are defined as: $\forall t \in \mathbb{R}^+$

$$(f \overline{\otimes} g)(t) = \sup\{f(s) + g(t-s)\ \mid\ 0 \le s \le t\},$$
$$(f \overline{\oslash} g)(t) = \inf\{f(t+u) - g(u)\ \mid\ u \ge 0\}.$$

The maximum memory space B that is required to buffer the input stream α, and the worst-case delay D experienced by the input stream α are given by

$$B(\alpha^u, \beta^l) = \sup_{\lambda \ge 0}\Big\{\alpha^u(\lambda) - \beta^l(\lambda)\Big\} \quad (7)$$

$$D(\alpha^u, \beta^l) = \sup_{\Delta \ge 0}\Big\{\ \inf\{\tau \ge 0 : \alpha^u(\Delta) \le \beta^l(\Delta + \tau)\}\Big\} \quad (8)$$

Join and fork of event streams: In [13], Perathoner et al. introduced a compositional method to merge and extract sub-streams based on event type information. One of their main results is the concept of Event Count Curves (ECC) which involves *join* (J) and *fork* (F) operations on event streams to compose and decompose *structured event streams*. A structured event stream composed of n single event streams with arrival curves $\alpha_1, ..., \alpha_n$ is defined as

$$\alpha(\Delta) = [\alpha^l(\Delta), \alpha^u(\Delta)] = \Big[\sum_i \alpha_i^l, \sum_i \alpha_i^u\Big], \Delta > 0 \quad (9)$$

and a set of upper and lower ECCs $\gamma_i(n) = [\gamma_i^l(n), \gamma_i^u(n)], n \ge 0$, one for each event type e_i. The lower and upper ECCs, $\gamma_i^l(n)$ and $\gamma_i^u(n)$, characterize the minimum and maximum number of events of type e_i in any sequence of $n \in \mathbb{N}$ events of the structured stream. It is worth mentioning that ECCs are not affected by stream processing, i.e., ECCs store information about the resulting structured event stream $\alpha(\Delta)$ after joining several event streams $\alpha_i(\Delta)$. After arbitrary processing of the joint stream $\alpha(\Delta)$, we again use the ECCs to decompose the structured stream into the corresponding sub-streams (see Fig. 3)

$$\alpha_i^l(\Delta) = \gamma_i^l(\alpha^l(\Delta)), \quad \alpha_i^u(\Delta) = \gamma_i^u(\alpha^u(\Delta)). \quad (10)$$

IV. Proposed Switch model

Essentially, an Ethernet frame experiences transmission delays and switch delays while being transmitted over the network. As already described, the computation of transmission delays is straight forward. However, the switch delay depends on the queuing time in the output queues. This queuing time further depends on the traffic pattern, and the scheduling policy employed by the output scheduler. In this section, we describe how to compute the delay in the switches.

We consider a switch which has m input/output ports and receives n frame streams as inputs, denoted as α_i, where $i = 1 \dots n$. Each frame stream is associated with a priority assignment π_i, and a destination port ϵ_i (obtainable from the destination address of the frame and the address table of the switch). As output, we obtain output arrival functions $\alpha_i^{'}$ for each single frame stream, the maximum delay D_i (switch delay + output transmission delay) and total switch memory requirement B. Further, the switch has a constant processing and fabric delay of D_{fab}, and each output port is associated with an Ethernet service function β_{eth}. Alg. 1 shows the proposed procedure for the performance analysis of a switch, which is also illustrated in Fig. 4. Fig. 5 visualizes the data paths in the switch model with five input frame streams.

Algorithm 1 Generic Ethernet switch model

Input: $\{\alpha_i\}$, $\{\pi_i\}$, $\{\epsilon_i\}$, π_{max}
Output: $\{\alpha_i^{'}\}$, $\{D_i\}$, B, where $i = 1 \dots n$ and for all j
Parameters: D_{fab}, m, β_{eth}
Indexing: $i = 1 \dots n$, $j = 1 \dots \pi_{max}$, $p = 1 \dots m$
Initialization: $D_i = 0$, $B = 0$

1: $out_p = \mathbf{GS}(\alpha_i, \epsilon_i)$
2: $D_i = \mathbf{FAB}(D_i, D_{fab})$
3: $\{out_p.prio_j\} = \mathbf{SGS}(out_p)$
4: $\bar{\beta}_{eth} = \mathbf{WLT}(\beta_{eth})$, $\bar{\alpha}_i = \mathbf{WLT}(\alpha_i)$
5: $[\bar{\alpha}_{out_p.prio_j}, \{\gamma_i\}] = \mathbf{JOIN}(out_p.prio_j)$, where $\alpha_i \in \{out_p.prio_j\}$
6: $[\bar{\alpha}_{out_p.prio_j}^{'}, \{D_{p,j}\}, B_{p,j}] = \mathbf{FPNP}(\bar{\alpha}_{out_p.prio_j}, \bar{\beta}_{eth})$
7: $\bar{\alpha}_i^{'} = \mathbf{FORK}(\bar{\alpha}_{out_p.prio_j}^{'}, \{\gamma_i\})$, where $\alpha_i \in \{out_p.prio_j\}$
8: $\alpha_i^{'} = \mathbf{IWLT}(\bar{\alpha}_i^{'})$
9: $D_i = D_i + D_{p,j}$, where $\alpha_i \in \{out_p.prio_j\}$
10: $B = \sum B_{p,j}$
11: **return** $\alpha_i^{'}$, D_i, B

- **Grouping streams:** (line 1) First, the incoming frame streams α_i are grouped into sets out_p according to their destination ports ϵ_i.
- **Passing through the switch fabric:** (line 2) The streams then pass through the switch fabric. In this process, each stream experiences a processing and fabric delay D_{fab}. This value is added to the total delay of each stream.
- **Sub-grouping streams:** (line 3) When the streams are forwarded to the output ports out_p, they are further sub-grouped according to their priorities π_i into sets $out_p.prio_j$.
- **Workload transformation:** (line 4) The arrival functions of all streams α_i are then transformed into their resource-based counterparts $\bar{\alpha}_i$ according to their frame sizes in bits. Similarly, the service function β_{eth} of the Ethernet output port is transformed into $\bar{\beta}_{eth}$ according to the

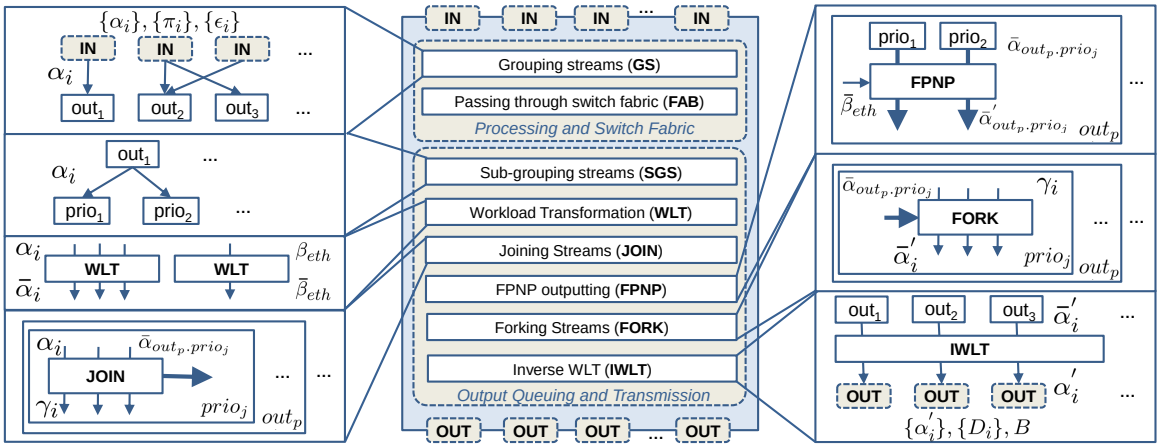

Fig. 4. Schematic of the proposed Alg. 1.

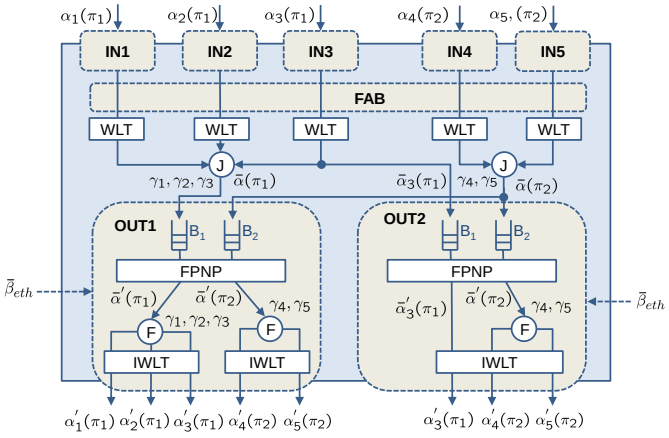

Fig. 5. This figure shows the data path of streams in a switch model.

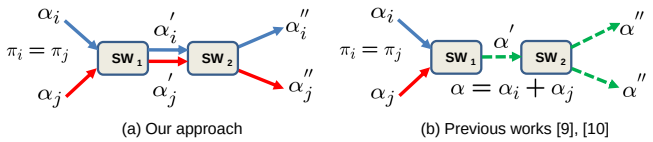

Fig. 6. Two stream of α_i and α_j with same priority and different destination (a) our approach (b) previous works.

available bandwidth. The workload transformations are performed using (1) and (2).

- **Joining streams:** (line 5) The streams belonging to the set $out_p.prio_j$ are joined to form $\bar{\alpha}_{out_p.prio_j}$ for each priority (queue) using (9). For each stream, a set of ECCs γ_i is obtained, which will be used to identify and retrieve each single stream after processing.
- **Outputting according to FPNP:** (line 6) We assume an output scheduler employing the fixed-priority non-preemptive scheme with $\bar{\alpha}_{out_p.prio_j}$ as input functions and $\bar{\beta}_{eth}$ as service functions. The output functions $\bar{\alpha}'_{out_p.prio_j}$, the delay $D_{p,j}$, and memory requirement $B_{p,j}$ for each queue in an output port can be obtained according to (7) and (8). $D_{p,j}$ consists of two parts, (i) the queuing delay, and (ii) the transmission delay on the output port.
- **Forking streams:** (line 7) After processing of each (priority) stream, the output arrival functions for each individual frame stream are retrieved from the joint output stream $\bar{\alpha}'_{out_p.prio_j}$ using the ECCs γ_i according to (10).
- **Inverse Workload transformation:** (line 8) The processed streams $\bar{\alpha}'_i$ are again transformed into the event-based functions α'_i, and serve as the switch output. The queuing and transmission delay of each set $out_p.prio_j$ is added to all the delay D_i of all streams belonging to this set. The total switch memory requirement B is the sum of all $B_{p,j}$.

The output functions α'_i may serve as input arrival functions to the next switch in the path in a compositional manner. To improve readability, we only describe uni-cast frames in the algorithm. It should be noted that the algorithm also supports multi-cast operation, in which case the incoming arrival functions are duplicated in the **SG** part and forwarded to multiple output ports. Since the proposed model is compositional and independent of the switch configuration and traffic patterns, it can be used to conduct performance analysis of Ethernet networks with different topologies and traffic patterns.

Technical distinction from previous works

In this context, network calculus has been used for modeling and analysis of similar design aspects of Ethernet switches [9], [10]. The major difference between the previous works and our analysis is illustrated in Fig. 6. When two streams, e.g., α_i and α_j, have the same priority but different data paths, the previous works essentially do not explicitly analyze individual frame streams. Rather, they consider them together as a single flow. If the frames are considered as a single stream, and not properly separated, the joint stream is required to be used for further analysis. This introduces considerable analysis pessimism than if the streams are treated independently. It becomes more prominent when a larger and more complex network topology is considered. In our work, we could retrieve the individual frame streams after they pass through the switch. This way, we could avoid the analysis pessimism that can be noticed in the works based on network calculus [9], [10].

V. Case Study

In this section, we apply the proposed analysis technique to a relatively small yet meaningful Ethernet based automotive system. In this setup, we consider 4 ECUs and a task set

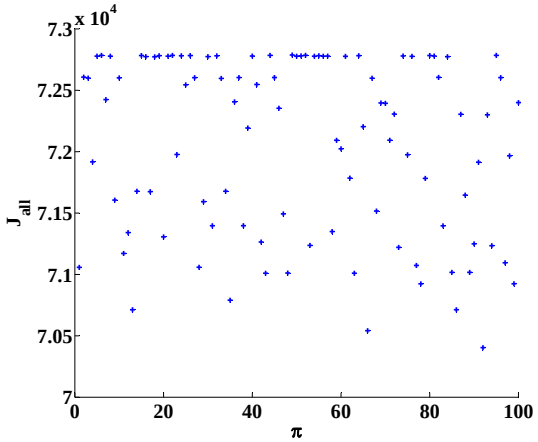

Fig. 7. Star Topology: J_{all} at 100 randomly generated priority assignments.

T_i	Sender	Receiver	$p_i[ms]$	$c_i[bytes]$	π_i
$T_{1,ct}$	ECU_1	ECU_3	1	80	π_1
$T_{2,ct}$	ECU_1	ECU_4	5	80	π_2
$T_{3,ct}$	ECU_2	ECU_4	2.5	94	π_3
$T_{4,ct}$	ECU_2	ECU_3	1	94	π_4
$T_{5,rt}$	ECU_1	ECU_3, ECU_4	10	130	π_5
$T_{6,rt}$	ECU_2	ECU_3, ECU_4	20	158	π_6
$T_{7,rt}$	ECU_3	ECU_4	5	180	π_7
$T_{8,rt}$	ECU_3	ECU_4	5	180	π_8
$T_{9,rt}$	ECU_3	ECU_4	20	180	π_9
$T_{10,rt}$	ECU_3	ECU_4	10	180	π_{10}

TABLE I
TASK SET SPECIFICATION

τ consisting of 10 tasks, as shown in Table I. All 10 tasks are mapped onto different ECUs as indicated by the sender and receiver column in Table I. Each task sends an Ethernet frame of size c_i to one or more receiver ECUs according to a period p_i. Tasks $T_{5,rt}$ and $T_{6,rt}$ are mapped onto one sender ECU each (ECU_1 and ECU_2 respectively) and send frames to two receiver ECUs (ECU_3 and ECU_4) by a *multi-cast* operation. All other tasks have only one single receiver. The tasks are further categorized as feedback control tasks $T_{i,ct} \in \tau$, and real-time tasks $T_{i,rt} \in \tau$. We assume a strictly priority-based output scheduling on the switches. Further, four global priorities are considered $\pi = \{1, 2, 3, 4\}$ (1 denoting the highest priority) whereas each Ethernet frame is associated with one priority $\pi_i \in \pi$. We study three different network topologies: star, twin star, and line topology (see Fig. 1).

Control applications and performance: We consider four feedback control tasks $T_{i,ct} \in \tau$ and use the following sampled-data model for the feedback control loops [14]:

$$x[k+1] = Ax[k] + B_0(D)u[k] + B_1(D)u[k], \quad (11)$$

where $x[k]$ is the vector of feedback states and $u[k]$ is the control input. $(A, B_0(D), B_1(D))$ are system matrices which are constant for a given *sensor-to-actuator* delay D. Often, the performance of a feedback loop is measured by a quadratic cost function,

$$J = \sum (x[k]'x[k] + u[k]'u[k]), \quad (12)$$

which depends on D, as indicated by equation (11). Intuitively, the performance of a feedback loop deteriorates with increasing delay D. In this case study, we assume that $T_{i,ct} \in \tau$ performs *measure* operation periodically and sends the sensor reading $x[k]$ on an Ethernet frame over the network. The *compute* and *actuate* operations are performed at the receiving ECU, which is a typical automotive configuration. The *actuate* operation is performed periodically with the same period as $T_{i,ct}$ in a time-triggered fashion, and with an offset D which is the worst-case *sensor-to-actuator* delay D for the corresponding control loop. Thus, for a given control task $T_{i,ct} \in \tau$, D needs to be obtained analytically but it is a constant value (i.e., the worst-case value). To this end, we consider the following overall performance as a design parameter (which should be minimized for a better design),

$$J_{all} = \sum_{T_{i,ct} \in \tau} J. \quad (13)$$

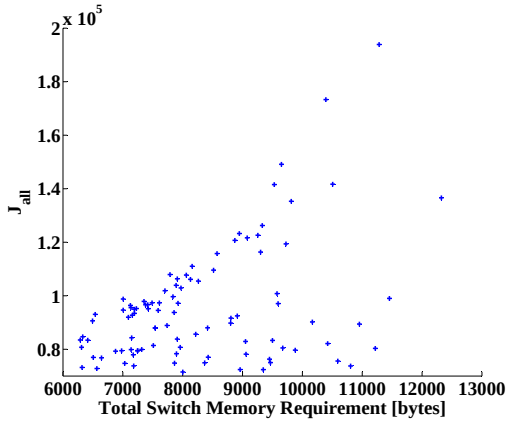

Fig. 8. Line Topology: J_{all} vs. total memory requirement.

A. Experimental Results

We implemented the proposed analysis technique in MATLAB using the RTC based model described in Section III for the three topologies in Fig. 1. We used 4 priorities $\{1, 2, 3, 4\}$ for the Ethernet frames $\pi_i \in \pi$ $(i = 1, 2...10)$. Towards demonstrating the impact of priority assignment, we generated 100 priority sets, and evaluated the system-level performance for each such priority set for the different topologies. Further, we have simulated the above network topologies using OMNeT++ framework (www.omnetpp.org) to evaluate our analysis. Table II shows the analytical bounds (A), the simulated worst-case (S), and average delay (Avg) for each frame.

Communication delay: As described above, 10 tasks send 10 periodic frames and $T_{5,rt}$, $T_{6,rt}$ are duplicated by multi-cast operation. Table II shows the communication delay for the priority set resulting in the best control performance (among the 100 generated priority sets). For each topology, we also simulated the network behavior with the same priority set corresponding to the best control performance. We conducted each simulation run with 5^6 combinations of randomly generated offsets to account for different release patterns.

Comparing the simulation with analytical results, we can observe that: (i) All the analytical delay upper bounds are larger than or equal to the worst-case communication delays in the simulation. Clearly, the analytical bounds are *safe*. (ii) For the frames with relatively shorter delays (e.g., all frames in star and twin star topology, some frames in line topology), we can see that the simulated worst-case is close to the analytical one. (iii) For the frames experiencing larger delays, the analytical worst-case delay differs from the simulated one by a higher margin. A possible explanation is that the worst-case requires a combination of factors (e.g., synchronized releases at different stages in the network) that are unlikely to happen in the simulation.

Topology		Communication delay [μs]												$\{\pi_1 \dots \pi_4\}$	Min J_{all}
		$T_{1,ct}$	$T_{2,ct}$	$T_{3,ct}$	$T_{4,ct}$	$T^3_{5,rt}$	$T^4_{5,rt}$	$T^3_{6,rt}$	$T^4_{6,rt}$	$T_{7,rt}$	$T_{8,rt}$	$T_{9,rt}$	$T_{10,rt}$		
Star	A	33.32	72.69	37.32	54.44	57.32	118.94	51.11	78.93	111.64	80.69	111.64	122.94	$\{1, 2, 1, 4\}$	7.040e+4
	S	33.32	47.34	37.32	54.24	48.53	77.51	50.82	56.03	73.41	60.05	69.88	66.92		
	Avg	19.90	19.86	22.22	22.02	27.74	27.82	32.23	32.28	35.77	35.77	35.75	35.76		
Twin Star	A	33.32	138.33	37.32	81.53	57.32	269.70	78.19	101.49	239.38	146.33	239.38	273.70	$\{1, 2, 1, 4\}$	7.041e+4
	S	33.32	68.89	37.32	64.40	48.65	80.92	69.38	56.03	78.16	78.16	77.90	90.31		
	Avg	19.90	32.36	22.22	35.52	27.74	44.22	50.83	32.28	56.13	56.13	56.11	56.12		
Line	A	57.04	280.07	66.16	81.70	88.68	612.67	78.19	212.01	434.60	256.85	434.60	585.49	$\{1, 2, 1, 4\}$	7.151+e4
	S	47.41	101.68	57.68	64.40	64.97	115.31	69.35	76.39	99.81	98.52	98.88	110.67		
	Avg	32.33	57.32	35.96	35.52	44.10	77.00	50.83	50.89	76.49	76.49	76.47	76.48		

TABLE II
A: ANALYTICALLY COMPUTED WORST-CASE DELAY, S: SIMULATED WORST-CASE DELAY, AVG: SIMULATED AVERAGE VALUE, $T^3_{5,rt}$: $T_{5,rt}$ SENT TO ECU_3, $T^4_{5,rt}$: $T_{5,rt}$ SENT TO ECU_4, $T^3_{6,rt}$: $T_{6,rt}$ SENT TO ECU_3, $T^4_{6,rt}$: $T_{6,rt}$ SENT TO ECU_4

Control performance: Depending on the priority assignment and topology, the resulting overall control performance J_{all} changes. In Fig. 7, we have plotted how the overall control performance J_{all} varies with 100 priority sets in the star topology. We can see that the best control performance is achieved with priority set $\#92$ where the control tasks are assigned relatively high priorities ($\pi_1 = 1, \pi_2 = 2, \pi_3 = 1, \pi_4 = 4$). On the other hand, the priority set $\#49$ with ($\pi_1 = 3, \pi_2 = 3, \pi_3 = 3, \pi_4 = 4$) results in the worst control performance since the sensor-to-actuator delay D increases when the control-related frame priorities are low. Moreover, we can notice from Table II that the control performance degrades from the star to the twin star topology, and similarly from the twin star to the line topology (since J_{all} increases). The control frames pass through only one switch in star topology whereas they might pass through more than one switch in the other topologies depending on the data paths. Intuitively, the control frames experience shorter delays in star topology compared to the other two. As already mentioned, the control performance does not only depend on the topology but also on the priority of the control-related frames.

In general, the control performance improves when the communication delay is less. Intuitively, assigning higher priorities to control tasks leads to a better performance. However, the priority assignment for the best overall control performance also depends on the nature of the controlled plants (e.g., their sensitivity to the sensor-to-actuator delay, control algorithms). Thus, assigning the highest priority to the all control tasks does not necessarily result in the best performance since it can possibly lead to a higher delay for some of the more delay-sensitive control plants. Therefore, the priority assignment problem for such a task set with multiple intertwining factors such as timing requirements of real-time tasks, performance requirements of control tasks is a complex problem on its own. In this work, we are rather concerned with introducing an analysis technique that allows investigating these design factors than with providing a concrete solution to the priority assignment problem.

Memory requirement: Similarly, we have obtained the total switch memory requirement in the different topologies. In the case of star topology, the total memory requirement is constant and equal to the sum of the sizes of all frames. This is because each frame passes through only one switch and requires a memory space equal to its frame size at the output ports. In the other two topologies, the memory requirement varies with the priority assignment. Here some frame streams pass through multiple switches. As a result, the arrival frame stream at a switch depends on the output stream from the previous switch, which obviously depends on the frames' priorities sharing the same output port. In Fig 8, we have plotted memory requirement for various priority assignments in the line topology. In general, a greater total delay of frames corresponds to more total memory space, since frames with greater delays require more buffer space at the switches.

VI. CONCLUSIONS

In this work, we presented a design and analysis technique for full-duplex switched Ethernet topologies that explores the relation among a number of design considerations – network topology, frame priorities, communication delays, memory requirements, performance of distributed feedback controllers. The proposed technique is a foundation for advanced design and analysis problems such as optimal priority assignment for frames, suitable topology selection, optimal positioning of switches, optimizing performance of the feedback control loops, optimizing task mapping on ECUs, etc.

REFERENCES

[1] M. Rahmani, K. Tappayuthpijarn, B. Krebs, E. Steinbach, and R. Bogenberger, "Traffic shaping for resource-efficient in-vehicle communication," in *Industrial Informatics*, 2009.

[2] J. Rox, R. Ernst, and P. Giusto, "Using timing analysis for the design of future switched based ethernet automotive networks," in *DATE*, 2012.

[3] K. C. Lee, S. Lee, and M. H. Lee, "Worst case communication delay of real-time industrial switched ethernet with multiple levels," 2006.

[4] C. W. Wu, S. Fischmeister, and G. Carvajal, "Evaluation of communication architectures for switched real-time ethernet (pre-print)," 2012.

[5] Q. Zhang and W. Zhang, "Priority scheduling in switched industrial ethernet," in *ACC*, 2005.

[6] V. Hassani, H. Talebi, M. Shafiee, and H. Taheri, "Priority scheduling in switched industrial ethernet," in *European Control Conference*, 2007.

[7] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *DATE*, 2003.

[8] J. L. Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[9] J.-P. Georges, T. Divoux, and E. Rondeau, "Network calculus: application to switched real-time networking," in *International ICST Conference on Performance Evaluation Methodologies and Tools*, 2011.

[10] A. Mifdaoui, F. Frances, and C. Fraboul, "Full duplex switched ethernet for next generation "1553b"-based applications," in *RTAS*, 2007.

[11] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," in *IEE Proceedings Computers and Digital Techniques*, 2005.

[12] W. Haid and L. Thiele, "Complex task activation schemes in system level performance analysis," in *CODES+ISSS*, 2007.

[13] S. Perathoner, T. Rein, L. Thiele, K. Lampka, and J. Rox, "Modeling structured event streams in system level performance analysis," in *LCTES*, 2010.

[14] D. Goswami, M. Lukasiewycz, R. Schneider, and S. Chakraborty, "Time-triggered implementations of mixed-criticality automotive software," in *DATE*, 2012.