Performance Analysis of Master-Slave Multi-Hop Switched Ethernet Networks

Mohammad Ashjaei, Moris Behnam, Thomas Nolte
Mälardalen University, Västerås, Sweden
mohammad.ashjaei@mdh.se

Luis Almeida
IT / DEEC, University of Porto, Portugal
lda@fe.up.pt

***Abstract*—There is an increasing trend towards using switched Ethernet in real-time distributed systems due to features like absence of collisions and high throughput. Nevertheless, a few problems persist, in particular related to priority inversion and limited length in queues. In this paper we focus on a protocol which uses a master-slave technique over standard switched Ethernet in order to overcome such problems, namely FTT-SE protocol. We present an improved response time analysis for such a network and we compare, analytically and with simulations, the results achieved with Network Calculus on a worst-case scenario. We show that our proposed response time analysis gives tighter bounds compared to Network Calculus. Moreover, we compare the performance of different solutions to scale the FTT-SE protocol with respect to the bandwidth utilization. Finally, we propose a new architecture to improve the average performance of master-slave switched Ethernet networks.**

I. Introduction

The complexity of the communications within distributed embedded systems is rapidly increasing due to the growing number of nodes and larger amount of exchanged data, which impose challenges when timeliness must be enforced.

Recently, the interest on using Switched Ethernet technology in distributed embedded systems has increased because of its wide availability, low cost, absence of collisions and traffic segregation. Nevertheless, using COTS (Commercial Off-The-Shelf) switches in time critical applications may affect the possibility of guaranteeing real-time behavior. Basically, the queues in the switch ports may overflow due to uncontrolled arrivals of packets, a situation that, in a worst-case, leads to drop packets. Moreover, COTS switches typically have FIFO queues that can generate long blocking times for urgent packets. This latter problem can be partially mitigated prioritizing the traffic and using separate queues for different priorities. Nevertheless, the IEEE802.1D standard considers up to 8 priority levels, which is too few to support effective priority scheduling.

A more effective solution to such problem is controlling the traffic submitted to the switch avoiding queue build up and using adequate traffic scheduling policies. This can be achieved with a master-slave approach. In this paper we use the FTT-SE protocol [1], which is a bandwidth-efficient master-slave protocol based on the Flexible Time-Triggered (FTT) paradigm. In addition, the FTT-SE protocol handles all types of message streams including real-time periodic, real-time sporadic and non-real-time traffic, by defining and using specific reserved bandwidth for each type of message streams and thus providing temporal isolation between them.

However, the FTT-SE protocol was initially developed for a network containing a single switch with a set of directly connected nodes. Recently, two different architectures were presented to extend the protocol to networks consisting of multiple switches connected in a tree topology. In one architecture, the network is controlled by a single master connected to the top of the network hierarchy [2], whereas in the second one each switch has an attached master that controls the respective traffic [3]. For the second case, the work in [4] presents a method based on Network Calculus to compute the traffic end-to-end delay.

In this paper, we comparatively assess the two referred architectures, i.e., single-master and multi-master, and we propose a new hybrid one that is better suited to large networks. Therefore, the main contributions of this paper are:

1) We complement the multi-master approach with an improved response time analysis and we compare the upper bounds obtained by this method with those obtained with Network Calculus and those observed in experiments with a simulation tool [5] on a worst-case scenario. We show that the proposed response time analysis gives tighter upper bounds compared to Network Calculus.
2) In addition, we compare the performance of single-master and multi-master architectures with respect to bandwidth utilization and scheduler overhead.
3) Finally, we propose a new hybrid architecture suited to large scale networks, based on a combination of the two previous ones. Furthermore, we also present a response time-based worst-case end-to-end delay analysis for the new architecture.

The rest of the paper is organized in the following way. The next section discusses some related work. Section III describes the basics of the FTT-SE protocol. Then, Section IV illustrates the system model while Section V presents the traffic delay analysis and shows the comparison between response time and Network Calculus on a worst-scenario experiment. Moreover, Section VI shows the comparison evaluation of two different architectures. Section VII presents a new architecture including the evaluation and end-to-end delay analysis, and Finally Section VIII concludes the paper and presents the future work.

II. Related Work

The literature on switched Ethernet is vast and there have been many works addressing its adequacy to real-time communication. We can find from relatively old research proposals

such as EtheReal [6] and the EDF Scheduled Switch [7], both based on channel reservations supported on enhanced switches, to recent ones such as the enhanced switches proposed in [8] that combine synchronous and asynchronous traffic with rate protection.

In the meanwhile, many solutions to real-time Ethernet actually made it to the market, such as TTEthernet and PROFINET IRT, both optimized for time-triggered operation, and EtherCAT, optimized for quick forwarding with on-the-fly update of the Ethernet frames while traversing the nodes. AFDX is also in this category, with enhanced forwarding and rate control based on static forwarding tables and rate filters, and has been used mainly in avionics. More recently, a new standard has been created, named Ethernet AVB (Audio-Video Bridges), which supports channel reservations with bounded end-to-end latency, but with practically no market support, yet.

These solutions using enhanced switches present improved performance but result in high cost and lower availability than current COTS Ethernet switches (IEEE802.1D). Thus, several solutions were also researched and eventually marketed, based on overlay protocols that control the traffic submitted to COTS switches. This is the case of Ethernet POWERLINK and FTT-SE, both master/slave approaches, the latter of which will be used in this paper due to its superior bandwidth efficiency and traffic scheduling flexibility.

Concerning the timing analysis of multi-switch Ethernet networks, several methods are also available. For example, Network Calculus is used in [9] to analyse the end-to-end delays in FTT-SE using a single master multi-switch topology and in [10] for networks of standard Ethernet switches. The work in [11] presents three methods to derive the end-to-end traffic delays in a multi-switch AFDX network, namely using Network Calculus, network simulation and model checking, among which Network Calculus exhibited a little higher pessimism. A tighter timing analysis for AFDX networks can be achieved using the trajectory approach as reported in [12].

Network calculus is also used in [13] to derive end-to-end traffic delays for Ethernet AVB, showing a case study based on an automotive infotainment system. The work in [14] presents a worst-case delays verification of in-vehicle Ethernet networks using the same analytical framework to generate upper bounds and checking them against experiments in worst-case scenarios.

A different approach is followed in [15] and [16] that derive end-to-end delay bounds for a single flow in FIFO multiplexed sink-tree networks using a modified Network Calculus framework. These works use partitioning of a network topology into a set of logically separated sink-trees having egress nodes at the root and ingress nodes at the leaves. The traffic is aggregated in nodes by introducing a FIFO policy called aggregated scheduling. A class of service curves is introduced to determine the service that is received in an aggregate scheduling network. Furthermore, the work in [17] utilized the mentioned method to investigate an admission control in sink-tree networks.

As can be seen from the brief survey above, worst-case response-time analysis of networks of standard Ethernet switches is unexplored to a large extent. In this paper we further explore this kind of analysis applied to a master-slave network of switches, running the FTT-SE protocol with one master controlling the traffic at each switch, and we show that we can obtain tighter results than with Network Calculus.

III. FTT-SE PROTOCOL

FTT-SE (Flexible Time-Triggered Switched Ethernet) is a bandwidth efficient master-slave protocol that was originally developed for small networks using a single Ethernet switch [1]. Its extension to larger networks with multiple switches was recently addressed with two main approaches presented in [2] and [3]. In this section we sketch the second approach which uses multiple-master nodes to control the traffic transmission through the network.

The FTT-SE network consists of multiple switches connected together in a tree topology, each with its own master as illustrated in Figure 1. The switch in the top of the hierarchy is called the root switch. Moreover, we define a *sub-network* as the ensemble of each switch along with all nodes that are directly connected to it, e.g., SW1, M1, S1 and S2 in Figure 1. Each sub-network is a parent for the lower level sub-networks that are attached to it. The group of sub-networks that have the same parent sub-network is called a *cluster*, e.g., cluster2 in Figure 1. The only exception is the root sub-network that is considered included in its children cluster since it cannot be accounted as one cluster itself.

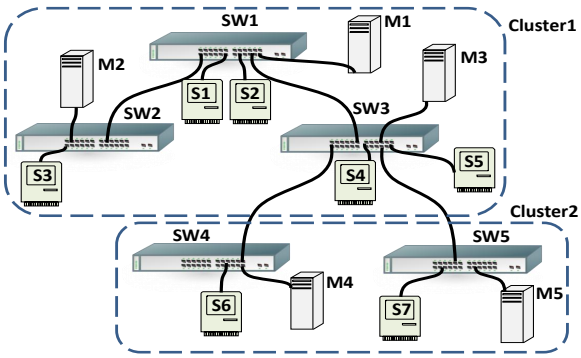

Fig. 1. The FTT-SE Network

In addition, we have defined two categories of traffic. A message that is transmitted within a sub-network is called *local*, otherwise it is called *global*.

As in any FTT protocol, the master schedules the messages on-line according to any desired policy, e.g. Fixed Priority Scheduling. The scheduler is invoked periodically, exactly once every interval of time called Elementary Cycle (EC). Each EC, which is configured off-line, is mainly divided in two windows, one to manage the protocol called *Initialization Time* and another one to transmit the data messages which is called *Data Transmission Window* (Figure 2). In the multi-master architecture, the data transmission window is further partitioned among all traffic types, i.e. local/global and synchronous/asynchronous. The global asynchronous window is

still divided among the clusters, e.g. Cluster1 and Cluster2 in Figure 2.

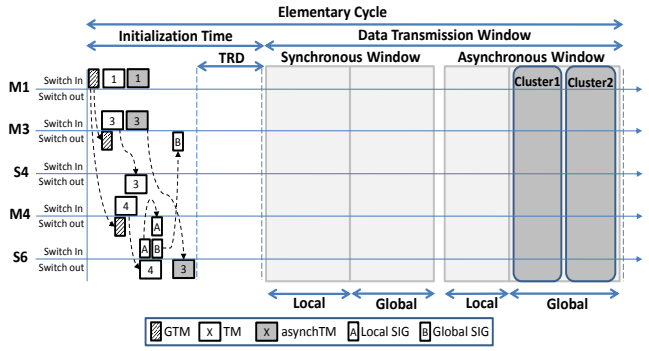

Fig. 2. The FTT-SE Elementary Cycle

This architecture requires that the ECs of all masters are synchronized. The solution we currently use, for simplicity, is to have a particular message, called Global Trigger Message (GTM), being broadcast by the root master and propagated down the entire network. The remaining masters synchronize upon receiving the GTM and start their local ECs. In the event that a GTM is lost, the receiving master times out, triggers its own local EC and generates a GTM itself, for the clusters below it in the topology tree. Certainly, other solutions are possible, e.g., using clock synchronization among masters. The selection of the best global synchronization approach is kept as future work.

Algorithm 1 describes the operation of a generic master node. The only different master is the root master, which follows a similar algorithm except that its cycle is timed, not synchronizing with the GTM.

The main loop of the master nodes starts with broadcasting the GTM for the lower level masters (lines 7). Then the local EC is triggered, broadcasting the Trigger Message (TM) to all the nodes in its sub-network (line 8) and the asynchronous TM (asynchTM) to its children sub-networks (line 9). The TM conveys the IDs of the scheduled local/global synchronous and local asynchronous messages (e.g., TM4 from M4 to S6 in Figure 2). The global asynchronous messages, however, are scheduled by the parent master of each cluster based on the dedicated window allocated for that cluster (e.g., asynchTM3 from M3 to S6 in Figure 2).

Unlike synchronous messages, activation of asynchronous ones is unknown in advance and can occur at any time. A signaling mechanism [18] allows the slaves to notify the master of pending requests using a Signaling Message (SIG) that is transmitted in the beginning of the EC (e.g. A for local asynchronous and B for global asynchronous messages in Figure 2). Therefore, each master collects the local SIG messages of its sub-network and the global ones from its cluster (cycle starting in line 11) and updates the ready queues for all types of traffic (lines 18 and next). Then, the master node starts to pick the messages from the head of the queues until no more messages fit in the respective window and computes the TM and asynchTM for the next EC (lines 21 and next).

Algorithm 1 Generic Master Node

```
1:  InitializeMaster()
2:  TM = initialTM
3:  asynchTM = initialAsynchTM
4:  receive(GTM) //sync with GTM
5:  timeOut = currentTime + EC + waitTime
6:  loop
7:      send(GTM)
8:      send(TM)
9:      send(asynchTM)
10:     //wait for all SIG messages
11:     waitSIG = currentTime + SIGwindow
12:     while currentTime < waitSIG do
13:     end while
14:     while inputBuffer(SIG*) do
15:         //read all received SIG messages
16:         receive(SIG*)
17:     end while
18:     update(LocSyncQ, LocAsyncQ)
19:     update(GlobSyncQ, GlobAsyncQ)
20:     //prepare TM and asynchTM for next EC
21:     TM = schedule(LocSyncQ, LocAsyncQ, GlobSyncQ)
22:     asynchTM = schedule(GlobAsyncQ)
23:     // sync with GTM for next cycle
24:     receive(GTM, timeOut)
25:     timeOut = timeOut + EC
26: end loop
```

At the end of the cycle, the master synchronizes again with the GTM to trigger the next cycle (lines after 23). In case the GTM does not arrive in time, a time out occurs and triggers the cycle anyway. The timeout is initialized with an offset *waitTime* in order to keep track of GTM receiving timeout.

Note that the protocol entails several broadcast messages, namely the GTM, TM and asynchTM. Their dissemination, however, is confined by appropriate configuration of Virtual LANs (VLANs) so that they do not create undesired interference outside their domain. For example, each sub-network has an associated VLAN that confines the broadcast of the associated TM. The masters in a cluster and the parent master are also integrated in another VLAN within which the cluster parent master broadcasts the GTM and the asynchTM.

The algorithm of the slave nodes is significantly simpler as shown in Algorithm 2. The main loop is synchronized on the reception of the TM and asynchTM (lines 3 and next). Concurrently (despite the sequential representation for convenience), the slaves send their SIG messages to their master (line 5). Then, they decode the TM and asynchTM to determine which data messages (ID) they must transmit in the current EC (line 6). This process takes a time that depends on the processing speed of the slave node called turn around time (TRD) (Figure 2). Afterwards, the slaves initiate the transmission of the scheduled data messages (lines

8 and next) and the reception of the data messages sent by others (lines 12 and next). These two activities also happen concurrently, despite the sequential representation.

Algorithm 2 Slave Node

```
1:  InitializeSlave()
2:  loop
3:      receive(TM) //wait on TM
4:      receive(asynchTM)
5:      send(SIG)
6:      (ID[1..m],m) = decode(TM,asynchTM)
7:      //send all DATA messages inTM and asynchTM
8:      for i = 1 → m do
9:          send(ID[i])
10:     end for
11:     //read all received DATA messages
12:     while inputBuffer(DATA*) do
13:         receive(DATA*)
14:     end while
15: end loop
```

Note that the windows inside the EC (Figure 2) are considered for traffic scheduling purposes, only, particularly to bound the amount of traffic of each type that can be transmitted every EC. At the time of transmitting the data messages scheduled for a given EC, they are all sent as soon as possible, as shown in the *send* cycle in Algorithm 2, line 8 and following.

A. Scheduling Algorithm

The main aim of the master nodes is to schedule traffic online without causing overrun in the EC, i.e., the scheduled traffic must be received before the end of the EC. This is important to enforce the master traffic scheduling policy without interference of the management policy used by the switch in its queues. The scheduler in a master node picks the messages from the ready queues of all traffic types and checks whether they fit in their dedicated window in that EC. This procedure continues until the last message in the ready queues. The unscheduled messages are kept in the ready queues for the next ECs. Note that the master node considers an idle time in each transmission window to prevent overrun of messages in the allocated window.

To keep track of the utilization of the windows in each link connecting to the switch (assuming common full duplex switches), the master considers two bins per link and per window, one bin representing the uplink (node to switch) and the other bin the downlink (switch to node). Then, the scheduler starts from the higher priority messages and fills-up the bins associated to the links in the message path while considering the delays imposed by the switching itself, namely store-and-forward and switch latency delays, and the interference caused by other traffic.

Considering again the example in Figure 1, assume that two synchronous messages, m_1 and m_2, are ready to be transmitted from S1 to S4 and S2 to S4, respectively. These two messages cross two different links and share two other links. Figure 3 shows the bin corresponding to the downlink of S4.

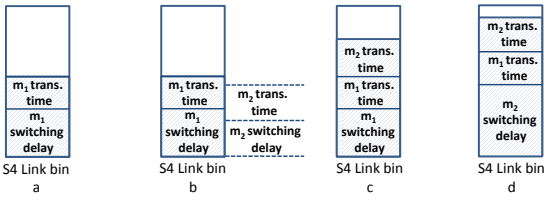

Fig. 3. The bin representing the downlink SW3 to S4

Assuming m_1 has higher priority, the scheduler fills-up first the bin with m_1 transmission time plus its switching delay resulting from crossing SW1 and SW3 (Figure 3.a). Then, the scheduler checks m_2, realizing that its switching delay is shorter than that of m_1 (Figure 3.b). Note that the switching delays affect messages transmitted in sequence only once, in fact, the longest such delay, only. Thus, there is no need to add the switching delay of m_2 in the bin (Figure 3.c). If the switching delay of m_2 was larger than the one for m_1, the scheduler would account m_2 switching delay instead (Figure 3.d). In general, in each EC the scheduler picks the maximum switching delay among the messages that fit in that EC. This mechanism will be used to develop the response-time analysis shown further on in this paper.

IV. System Model

In this paper, we use the real-time periodic model to present synchronous and asynchronous messages, which is characterized by the following tuplet:

$$m_i = m_i(C_i, D_i, T_i, O_i, S_i, Ds_i, R_i, sn_i) \quad (1)$$

In this tuplet, C_i is the transmission time of the message, D_i is the deadline and T_i is the period of the message, both presented as integer number of ECs. We assume the constrained deadline model, i.e. $D_i \leq T_i$. Moreover, S_i and Ds_i are the source node and destination node of the message, respectively (we currently restrict our analysis to unicast streams). Also, O_i denotes the offset of the message, R_i is the set of switches that the message crosses during its transmission and sn_i is the number of switches in the R_i set. The FTT-SE protocol supports both synchronous and asynchronous traffic. Thus, we model both with the same tuplet in which for asynchronous messages T_i is the minimum inter-arrival time and O_i is unspecified. Moreover, the messages scheduling is based on fixed priorities and these can be implicitly set according to Rate-Monotonic, Deadline-Monotonic or any other criteria reflected in the indexes.

In addition, the switches are assumed to be Commercial Off-The-Shelf (COTS) having several parallel FIFO queues with different priority levels for each output port. The switches are considered as store-and-forward and non-blocking, i.e., they do not build up queues in the input ports. We assume two different components for switching delays, namely the store-and-forward (SFD) delay and the switch relaying latency (Δ).

V. Traffic Delay Analysis

In this section we compute the end-to-end delay of messages in the multi-master FTT-SE architecture. The FTT-SE scheduling is based on reserving a bandwidth for each type of messages that is provided periodically every EC. Within each reservation a scheduling policy is used to select the messages to be transmitted. This model is equivalent to the hierarchical scheduling model presented in [19] and the associated analysis based on a *request bound function* (rbf) and a *supply bound function* (sbf) is more suitable for such a network. Moreover, in this section we validate the proposed analysis and compare with Network Calculus based analysis presented in [4].

A. Worst-Case Delay Analysis

The $\mathbf{rbf_i(t)}$ represents the maximum load generated by m_i and all the higher priority messages during the time interval $[0,t]$. It is computed by summing the transmission time of m_i with all delays that m_i may suffer during its transmission as shown in (2), where Wl_i is the maximum time that m_i is blocked by other messages in each link in its route from source to destination node that we call Shared Link Delay. Also, Wr_i denotes the time which m_i is blocked due to the messages which do not share links with m_i, yet might interfere with m_i as will be explained later and we call that Remote Link Delay.

$$rbf_i(t) = C_i + sn_i \times (SFD_i + \Delta) + Wl_i(t) + Wr_i(t) \qquad (2)$$

Shared Link Delay. Message m_i, under analysis, may be delayed at each link in its route while crossing switches due to the interference from all higher priority messages that share those links with m_i. To avoid accounting for the same message multiple times, all messages that caused this delay in the previous links are excluded in the current link. Moreover, the store-and-forward and switch latency delays of the messages generating the Shared Link Delay are collaborating to block the message under analysis. The latter is due to the scheduling algorithm in which the scheduler fills up higher priority messages in the respective bin considering their store-and-forward delays, thus the message under analysis will be scheduled over those delays in that bin. Therefore, Wl_i is calculated as shown in (3), where $hp(m_i)$ is the set of messages with priority higher than that of m_i and $WT(m_i)$ is the set of messages with a type similar to that of m_i, i.e., periodic/aperiodic or synchronous/asynchronous traffic as m_i.

$$Wl_i(t) = \sum_{\substack{\forall j \in [1,n], j \neq i \\ \wedge\, R_j \cap R_i \neq 0 \\ \wedge\, m_j \in hp(m_i) \\ \wedge\, m_j \in WT(m_i)}} \lceil \frac{t}{T_j} \rceil (C_j + sn_j \times (SFD_j + \Delta)) \qquad (3)$$

Remote Link Delay. The message that do not share links with m_i may still delay m_i indirectly through other messages. To show this effect, let us consider the following example, referring to the network shown in Figure 1.

We define four global messages, m_1 is transmitted from S1 to S4, m_2 is sent from S2 to S5, m_3 is transmitted from S7 to S5 and m_4 is sent from S6 to S4. We assume that m_1 has the lowest priority among the messages and $m_2 + m_3$ can be scheduled in one EC. Let us consider that m_1, m_2 and m_3 are activated simultaneously in the first EC and m_4 is activated in the second EC. Also, m_3 is started to transmit with an offset in the source node due to other higher priority messages in the source node of m_3. The scheduling window for the first and the second ECs is depicted in Figure 4.

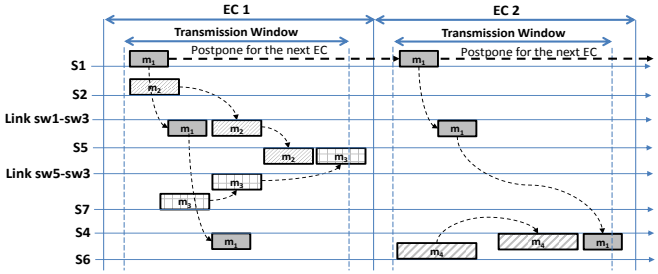

Fig. 4. Window Scheduling for Remote Link Delay Example

In this example, m_1 and m_2 have a share link between SW1 and SW3 and due to FIFO queue inside the switch it may possible that m_1 is sent earlier in the FIFO even though it has lower priority (Link SW1-SW3 in Figure 4, EC1). Therefore, scheduling of m_1 can push m_3 through m_2 makes m_3 overrun of the window, although they do not share links. Thus, the scheduler postpones m_1 for the next EC to prevent any possible overrun.

In the second EC, m_4 is activated and m_1 was suspended from the first EC. However, as m_4 has share destination with m_1 and due to have higher priority, m_4 is scheduled to be transmitted in the second EC, hence m_1 is again postponed for the third EC (considering enough size for m_4 to fill S4 link). If all messages have been released simultaneously then m_1 would be transmitted in the second EC as m_4 could be scheduled in the first EC. Therefore, we can conclude that simultaneously activation of messages does not yield the worst-case response time (critical instant).

In order to formulate the above outlined effect, in the worst-case response time analysis, all messages that share links with those contributed in the Shared Link Delay are taken into account. Thus, the Remote Link Delay is computed as shown in (4). Note that, the messages which are already considered in the Shared Link Delay are excluded.

$$Wr_i(t) = \sum_{\substack{\forall k, j \in [1,n], k \neq j \neq i \\ \wedge\, R_k \cap R_j \neq 0 \wedge\, R_k \cap R_i = 0 \wedge\, R_j \cap R_i \neq 0 \\ \wedge\, m_k \in hp(m_j) \\ \wedge\, m_k \in WT(m_j)}} \lceil \frac{t}{T_k} \rceil (C_k) \qquad (4)$$

The $\mathbf{sbf(t)}$ is the minimum effective communication capacity that the network supplies in the time interval $[0,t]$. Note that in each EC, the utilization bandwidth provided for transmitting each type of message is $\frac{BW-I}{EC}$, where BW is the specific width of the respective transmission window and I is the idle time in that window, which can be upper bounded by the maximum

message size allocated in BW. Thus, $sbf(t)$ can be computed in (5).

$$sbf(t) = (\frac{BW - I}{EC}) \times t \tag{5}$$

The response time of m_i is computed based on $t^* = min(t > 0) : sbf(t) \geq rbf_i(t)$. Determining t^* requires checking the inequality at all instants in which $rbf_i(t)$ changes due to interference of other messages. Such set of check points is given by $CP_{rbf_i} = [\cup cp_{m_q}, \forall_{m_q \in hp(m_i)}] \cup T_i$, where $cp_{m_q} = \{T_q, 2T_q, ..., n_q T_q\}$, $n_q = \lfloor \frac{T_i}{T_q} \rfloor$.

The scheduling occurs every EC but in general we do not know exactly when a scheduled message will be transmitted within the EC. Therefore, we compute the messages response times in number of ECs and the response time for m_i ($RT(m_i)$) is given by (6).

$$RT(m_i) = \lceil \frac{t^*}{EC} \rceil \tag{6}$$

The analysis for the asynchronous traffic is essentially similar except for an extra delay caused by the signaling mechanism. In fact, an asynchronous request may have to wait approximately 1 EC before its node signals it in the next SIG message and the master then takes another EC to insert the respective asynchronous message in the ready queue. This extra delay can be simply added to the $RT(m_i)$ for all asynchronous messages.

For local traffic, messages may suffer from both Shared Link Delay and Remote Link Delay which are evaluated using (3) and (4) respectively. However, local messages are transmitted through one switch, only, thus $sn_j = 1$. Moreover, $rbf_i(t)$ for local messages is computed as in (2) except that $sn_i = 1$.

B. Improved Response Time Analysis

One of the sources of pessimism in the presented response time analysis is the accumulation of the switching delay from all messages that contribute in Shared Link Delay term in (3). According to the scheduling algorithm presented in Section III-A, during the scheduling process in each EC, the scheduler considers the maximum switching delay among all messages that are fit in each link. In contrast, in the analysis all switching delays are accumulated due to the lack of scheduling information at each specific time. This makes the analysis very pessimistic especially when the number of messages is large. For instance, assume that there are 30 messages that can be transmitted in 3 ECs. Therefore, 3 switching delays out of 30 are considered in the scheduling. However, in the analysis all 30 switching delays are added, since the number of ECs that has been passed at time t is not evaluated.

In order to improve the analysis we modify (3) by separating the switching delay effect from the transmission time of messages in (7).

$$Wl_i(t) = \sum_{\substack{\forall j \in [1,n], j \neq i \\ \wedge R_j \cap R_i \neq 0 \\ \wedge m_j \in hp(m_i) \\ \wedge m_j \in WT(m_i)}} \lceil \frac{t}{T_j} \rceil (C_j) + Is_i(t) \tag{7}$$

In (7) the activation of all higher priority messages that can contribute to this equation can be extracted by knowing the current time t. We define a multi-set $G_i(t)$ [20] that contains the upper bound number of switching delays from each message that contribute in Shared Link Delay at time t. This upper bound is equal to the maximum number of possible activation of messages within time interval $[0,t]$. Depending on the time t, a number of elements will be taken from $G_i(t)$ and to consider the worst-case scenario, the selected elements should be the highest ones. Therefore, we define multi-set $G_i^{sort}(t)$ which contains the switching delay values from $G_i(t)$ sorted in a descending order.

According to the scheduler, one switching delay per each EC (the largest) is taken into account for scheduling. The number of EC that has been passed in an interval $[0,t]$ is given by (8). Therefore, in the response time analysis we select the first $z(t)$ number of elements from $G_i^{sort}(t)$ in order to consider the worst-case scenario for switching delay effect in our analysis as shown in (9).

$$z(t) = \lceil \frac{t}{EC} \rceil \tag{8}$$

$$Is_i(t) = \sum_{l=1}^{z(t)} G_i^{sort}(t)[l] \tag{9}$$

Note that, if $z(t)$ is larger than the number of elements in the multi-set $G_i^{sort}(t)$, then the value of the extra elements are equal to zero.

In addition, the presented improvement in the analysis is applied for the case of single-master FTT-SE architecture because the scheduling algorithm used by the master node is in similar way as multi-master approach.

C. Evaluation of the Analysis

In this section we evaluate the presented analysis method and compare it with the delay analysis proposed in [4]. Moreover, we asses the level of pessimism embodied in both analysis methods compared with the simulation measurements in one particular example.

We considered a network containing 10 switches along with 30 nodes directly connected to the switches as shown in Figure 5. The network parameters are $EC = 10ms$, $TM = 12\mu s$, $SIG = 7\mu s$, $\Delta = 17\mu s$ and the capacity of the Ethernet network is considered as $100Mbps$. Moreover, the transmission windows during each EC is divided as follows. The synchronous local and global windows are selected to have values equal to $1.5ms$ and $2ms$ respectively, asynchronous local transmission window is $1.5ms$ and finally the asynchronous global transmission window is $4.4ms$. In this particular example, the network composed of 4 clusters which the asynchronous global transmission window is further splitted equally among them, i.e. $1.1ms$.

We generated 90 messages that include all four types of traffic such that one specific message per each type is tagged to have the worst-case scenario among all other messages. In order to maximize the response time for global synchronous

Message id	Message Type	T(EC)	C(μs)
m_1	Global Synchronous	20	100
m_2	Global Asynchronous	18	150
m_3	Local Synchronous	20	150
m_4	Local Asynchronous	19	150

TABLE I
TAGGED MESSAGES PARAMETERS

and asynchronous messages, the priorities of the tagged messages are selected to be lowest and the route of the messages to be longest. Moreover, we delay the tagged messages with several higher priority messages. Furthermore, to generate the worst-case scenario for local synchronous and asynchronous tagged messages we set their priorities to the lowest and blocked them with several higher priority messages in source and destination links. The tagged messages parameters are presented in Table I.

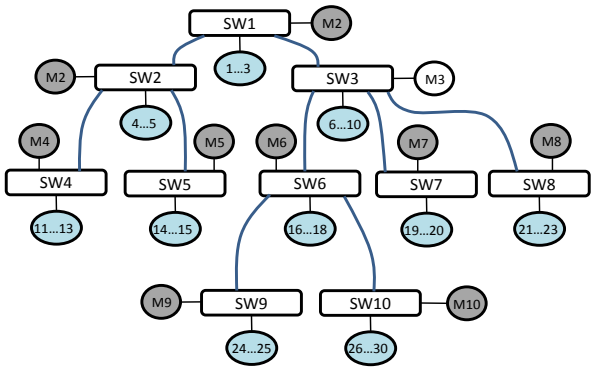

Fig. 5. A Network Example (1...3 means node 1, 2 and 3)

In order to simulate the outlined example we used a tool which was presented in [5]. The messages which are generated to block the tagged messages are considered to have different activation times in the simulation. We then measured the maximum end-to-end delays of the tagged messages and compared with the results obtained from the proposed delay analysis. This example does not guarantee to produce the worst-case situation since it is rather complex to determine with accuracy, however it is assuredly a very unfavorable case.

Figure 6 illustrates the maximum response time of the tagged messages measured from 10,000 ECs simulation time, the response time computed using the proposed delay analysis and the analysis based on Network Calculus [4]. In the figure, the x-axis represents a message id and the y-axis shows the message delay in number of ECs.

Comparing the results of simulation, proposed analysis and Network Calculus analysis, the maximum measured delay from simulation is always less or equal to the outcomes of both analysis methods. We observed that the response time analysis gives tighter results due to the improvement that has been proposed in this paper compared with Network Calculus analysis. An extreme level of pessimism provided by Network Calculus for global synchronous tagged message (m_1) is due to the large number of interfering messages in the route of m_1 in which Network Calculus accumulates all their switching delay in the analysis.

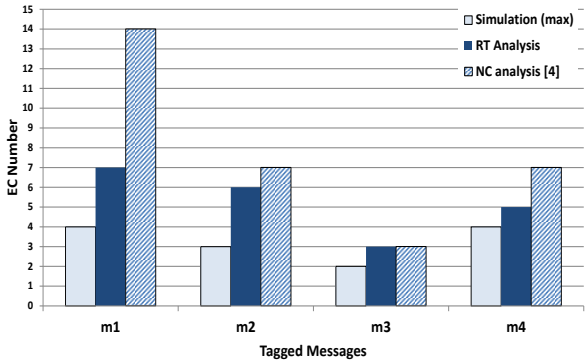

Fig. 6. Tagged Messages Response Time

Although the presented analysis based on response time performs better, it still illustrates a level of pessimism varying between 50% for local synchronous tagged message (m_3) and 100% for the global asynchronous tagged message (m_2). The revealed level of pessimism for local and global asynchronous tagged messages is due to considering one extra EC in both analysis methods to capture the scenario in which the request for the asynchronous message needs to wait for next EC. Also, we believe that another source of the pessimism is the accuracy of an actual worst-case scenario in simulation and the effect of Remote Link Delay that might not be computed by the simulator.

VI. COMPARATIVE EVALUATION

In this section we compare the two architectures quantitatively with respect to initialization time requirement, data transmission window and scheduler overhead.

A. Initialization Time

Ideally, the initialization time should be a small fraction of the EC but it is lower bounded by the time needed to transmit and process the TM and SIG messages. Here we compare the initialization time required by both architectures under similar scenarios.

1) Single-Master Architecture: Beyond their transmission times, the TM and SIG messages are further delayed due to interference between different SIGs and to switch delays ($SFD+\Delta$) crossed in their path. Therefore, the data transmission window should start late enough to guarantee that TM and SIG messages have arrived before. The initialization time for this architecture is given by (10) based on the number of nodes in the network (N_{node}) and the turn around time (TRD).

$$\begin{aligned} init\_time = N_{dep} \times (C_{TM} + \Delta) + max\{TRD, \\ N_{node} \times C_{SIG} + N_{dep} \times (C_{SIG} + \Delta)\} \end{aligned} \tag{10}$$

C_{TM} and C_{SIG} are the transmission times of TM and SIG messages, respectively. N_{dep} represents the depth in the network topology, e.g., $N_{dep} = 3$ in Figure 1. Moreover, $N_{dep} \times (C_{TM} + \Delta)$ is the TM reception instant in the farthest slave nodes. The transmission of SIGs overlaps with the TRD and thus we consider the maximum between the TRD and the time to transmit all the SIGs in the network, given by $N_{node} \times C_{SIG} + N_{dep} \times (C_{SIG} + \Delta)$.

2) Multi-Master Architecture: In the multi-master approach, extra messages are used to control the protocol operation, including the GTM and the asynchTM. Moreover, the TM is synchronized with the GTM reception while the SIG is synchronized with the TM reception.

The initialization time for this architecture is given by (11), where N_{max} is the maximum number of nodes in a sub-network. Furthermore, $N_{dep} \times (C_{GTM} + \Delta)$ is the longest time to receive the GTM by all masters. Also, $C_{TM} + C_{asynchTM} + \Delta$ required to be added to consider the transmission time of the TM and asynchTM to the respective master node. Sending the SIGs for local and global aperiodic messages overlaps with the TRD ,Thus the maximum between them must be account for.

$$init\_time = N_{dep} \times (C_{GTM} + \Delta) + C_{TM} + C_{asynchTM} + \Delta + max\{TRD, N_{max} \times (C_{SIG} + C_{asynchSIG})\} \quad (11)$$

To compare the initialization time required in both architectures, we apply (10) and (11) on a network with the following setting: capacity of 100Mbps, TM and asynchTM messages assumed 24μs long, SIG messages for both local and global aperiodic traffic are 10μs long, the GTM transmission time is 8μs, the Δ is 17μs and the TRD is set to 100μs.

We considered a network with 100 nodes and varied the depth in the topology (number of levels) while limiting 30 nodes for each sub-network. We observed that the initialization time for both approaches increases with an increasing depth as shown in Figure 7. However, in the multi-master architecture the initialization time is much less than in the single-master architecture.

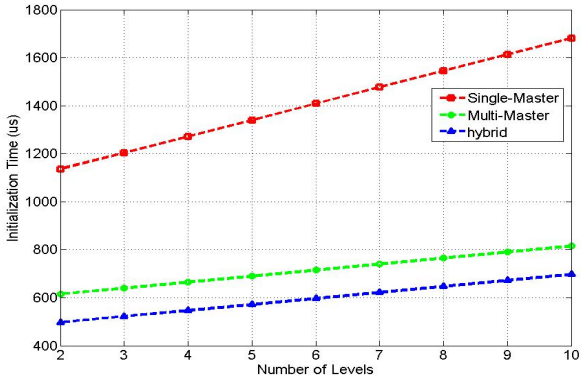

Fig. 7. Initialization Time (constant nodes)

As a result, the multi-master approach is more suited to networks with many nodes, given its shorter initialization time. In fact, in a single master architecture all SIGs must be received by the single master connected to the top switch in the network, and sending the TM to the deepest levels also incurs in a significant delay, both contributing to increase the initialization time. Conversely, in the multi-master architecture, SIGs impact is confined to clusters and the TM transmissions to the sub-networks, thus occurring in parallel in larger parts of the network, leading to smaller initialization time.

B. Data Transmission Window

Concerning the data transmission window, the goal would be to require the shortest window BW that allows meeting the network requirements of a given application. Thus, we compute $rbf_i(t)$ assuming a given constant EC using Equation (2), and for all $t \in CP_{rbf}$ the BW is derived such that $rbf_i(t) = sbf(t)$. The minimum BW among those computed for all check points is the window required for that message to meet its deadline. Then, BW is computed for all messages and the maximum value is the minimum window that makes all messages schedulable.

To show the difference in the data transmission window between the two architectures, we apply the analysis of each architecture to two different scenarios. In the first one, the architecture is the same as in Figure 1 with 20 nodes distributed in the network. In the second scenario, we use the same topology but with 70 nodes distributed in the network. In both cases, we generate 100 instances for two different sets, one with 100 and another with 500 messages, i.e., we generate 100 sets randomely for two different cases, a set of 100 messages and a set of 500 messages. The average window computed for the 100 instances of each set is presented for the two network scenarios in Figure 8. Moreover, to include the effect of initialization time in the examples, we sum it with the synchronous and asynchronous windows in Figure 8.

The vertical lines in Figure 8 shows the deviation of minimum and maximum calculated synchronous and asynchronous windows in all 100 instances. The deviation states that the average values of different architectures are representative for the evaluation.

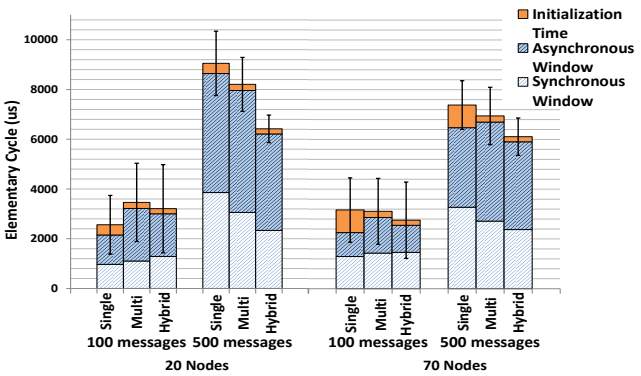

Fig. 8. Data Transmission Window Requirement

The results show that, for less amount of messages in both 20-nodes and 70-nodes examples, the single-master case needs shorter windows for synchronous and asynchronous traffic compared with the multi-master approach. The reason is that each window in multi-master architecture is divided into different types, i.e., global and local which cause significant increase of idle times assumed in each sub-windows.

However, initialization time for the single-master architecture increases when the number of nodes increases, in agreement with Figure 7. Therefore, the time needed within

EC to transmit the traffic is increased in case of 70-nodes example, even though the data transmission window is less than the multi-master approach.

Nevertheless, by increasing the number of messages to 500, in both 20-nodes and 70-nodes examples, the synchronous window in multi-master architecture is much less than single-master approach. The reason is that in the single-master the switching delays of local traffic are accumulated to global traffic which cause to allocate larger window. However, the asynchronous window in multi-master is larger due to have more idle time, i.e. more partitioned windows for clusters. Finally, considering initialization time the multi-master approach requires less EC to transmit the traffic, in particular for a network with more number of nodes.

In addition, increasing the number of nodes in the example decreases the window requirement since the messages are distributed among more nodes decreasing the interference in each link.

C. Scheduler Overhead

The master nodes in both architectures schedule the ready messages in parallel with the transmission of data messages. The time needed to schedule the messages depends on the processing capacity of the master node and on properties of the message set. However, the scheduling process should be finished before the start of the next EC to avoid missing cycles. The time which is allocated for the master node to schedule the ready messages is $(EC - InitializationTime)$. Therefore, according to Figure 7 and considering a constant EC, the time available for the master node to schedule the messages is decreased when increasing the number of levels in the network topology. This scheduling time is tighter for the single-master architecture. Moreover, in the single-master approach, one master node is responsible for scheduling all the traffic in the network, while in the multi-master architecture this duty is divided among several master nodes for local messages. Therefore, this situation is more demanding in the single-master approach with less available time for scheduling.

VII. Hybrid Approach

The previous section showed that neither architecture dominates the other one. Single-master architecture performs better for small networks, while multi-master architecture is more fit for bigger networks having high amount of nodes and messages. Thus, combining the two approaches has the potential to give better results. This can be done by connecting sets of single-master networks in *clusters*. An example of such an architecture is depicted in Figure 1, excluding M2, M4 and M5. In this approach, the traffic in each cluster is coordinated by a master node that is connected to the parent switch of the cluster (e.g. M1 for CL1 and M3 for CL2 in Figure 1).

Based on this new architecture, we classified the traffic in two categories according to the source and destination nodes of each message. If the source and destination of the message belong to the same cluster, it is called *internal*, otherwise the message is called *external*. The window for data transmission is divided into internal and external sub-windows within synchronous and asynchronous windows as illustrated in Figure 9. Each master node schedules the internal messages considering the internal sub-window, while all master nodes schedule all external traffic in the network within the external sub-window in parallel. Thus, each master node ensures that enough external sub-window is available for its external messages.

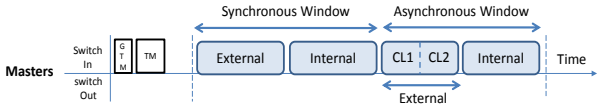

Fig. 9. EC Structure of Hybrid Architecture

Again, the synchronization of the ECs can be achieved by different solutions but we follow the same GTM mechanism as we did in the multi-master architecture.

A. Worst-Case Delay Analysis

A message in the network may suffer from delays and interference during the transmission. The interference is similar to the previous approaches except that the bandwidth allocation between the traffic types is different.

The Shared Link Delay for message m_i occurs when any other higher priority message shares any links from the source to the destination node in the route of m_i. This delay is derived using (3) considering that the messages should have the same window allocation as m_i in the transmission time $(WT(m_i))$.

The Remote Link Delay is applicable as well, in which all the messages that have share link with the messages account for Shared Link Delay should be taken into account. The Remote Link Delay is computed in (4) considering the window allocation for the messages (WT).

Finally, the $rbf_i(t)$ and $sbf(t)$ are calculated in (2) and (5) in order to derive the response time $(RT(m_i))$.

B. Evaluation

In order to evaluate the proposed architecture, we use the same experiment as for the previous architectures.

We determine the initialization time according to the signaling in the hybrid architecture using (12). In the proposed architecture, one TM is used for both synchronous and asynchronous messages compared with the multi-master approach which uses both the TM and asynchTM. Also, one SIG is used to inform the master node about aperiodic messages, instead of using two signals in the multi-master approach. However, all TM and SIG messages are transmitted through two switches which increases the delay due to the store-and-forward feature of the switches.

$$\begin{aligned} init\_time &= N_{dep} \times (C_{GTM} + \Delta) + 3 \times C_{TM} \\ &+ max\{TRD, N_{CL} \times C_{SIG}\} \end{aligned} \tag{12}$$

N_{CL} is the maximum number of nodes in one cluster and $N_{dep} \times (C_{GTM} + \Delta)$ is the latest time that GTM is received by the farthest slave node in the network topology. Moreover,

$3 \times C_{TM}$ is the TM reception time considering crossing two switches, i.e. from parent master of cluster to the slave node. Also, SIGs are transmitted in parallel with the *TRD*, hence the maximum of them should be considered.

Using the same example with the same transmission times which was used in Section VI, we computed the initialization time for this architecture. The result compared with the other architectures is depicted in Figure 7. The results show that increasing the number of levels in the network topology and assuming 100 slave nodes, the initialization time for the hybrid architecture is less than for the multi-master approach.

Furthermore, we computed the minimum window required for each type of traffic such that the traffic is schedulable. The results are also included in Figure 8 and show that the window needed for transmitting the data is much less than in the multi-master and single-master approaches specially in a network with high amount of nodes and messages. Also, including the initialization time as in Figure 8, the time needed for data transmission plus the initialization time in the hybrid approach for large networks is much less than for the other two architectures.

In addition, in the hybrid architecture, the number of masters used to coordinate the traffic is noticeably lower than in the multi-master approach while still having the benefits of that architecture. For instance, the system in Figure 1 requires two master nodes in the hybrid approach and five in the multi-master approach which makes the former more cost effective as well.

VIII. Conclusion and Future Work

In this paper, we presented an improved response time analysis for the multi-master architecture of master-slave multi-hop switched Ethernet networks, particularly applied to the FTT-SE protocol. We also compared the results obtained using three different approaches, the proposed analysis, Network Calculus analysis and simulation, on a worst-case scenario experiment. We concluded that the proposed response time analysis gives tighter upper bounds rather than Network Calculus.

In addition, we evaluate the performance of the architectures with respect to data transmission window requirement, initialization time and scheduler overhead. In general, the results show that a multi-master architecture has better bandwidth utilization for large networks compared with a single-master approach. Moreover, we presented a new hybrid architecture that provides the benefits of the previous two approaches. Finally we presented an evaluation of the proposed hybrid architecture in which the overall performance is shown improved compared with previous solutions. On-going work aims at studying solutions for time synchronization between the master nodes in the network to find the optimum way with less effect on the performance.

IX. Acknowledgments

This work is supported by the Swedish Foundation for Strategic Research, via Mälardalen Real-time Research Center (M-RTC) at Mälardalen University. Also, it is partially supported by FEDER through the COMPETE program, and by the Portuguese Government through FCT grant Serv-CPS PTDC/EEA-AUT/122362/2010".

References

[1] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over cots ethernet switches," in *6th IEEE International Workshop on Factory Communication Systems (WFCS'06)*, June 2006.

[2] R. Marau, M. Behnam, Z. Iqbal, P. Silva, L. Almeida, and P. Portugal, "Controlling multi-switch networks for prompt reconfiguration," in *Proc. of 9th Int. Workshop on Factory Communication Systems (WFCS12)*, May 2012.

[3] M. Ashjaei, M. Behnam, T. Nolte, L. Almeida, and R. Marau, "A compact approach to clustered master-slave ethernet networks," *9th IEEE Int. Workshop on Factory Communication Systems (WFCS'12)*, May 2012.

[4] M. Ashjaei, M. Liu, M. Behnam, A. Mifdaoui, L. Almeida, and T. Nolte, "Worst-case delay analysis of master-slave switched ethernet networks," in *2nd International Workshop on Worst-Case Traversal Time (WCTT'12)*. ACM, December 2012.

[5] M. Ashjaei, M. Behnam, and T. Nolte, "The design and implementation of a simulator for switched ethernet networks," in *3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS'12)*, July 2012.

[6] S. Varadarajan and T. Chiueh, "Ethereal: a host-transparent real-time fast ethernet switch," in *6th Int. Conference on Network Protocols*, oct 1998.

[7] H. Hoang and M. Jonsson, "Switched real-time ethernet in industrial applications - deadline partitioning," in *9th Asia-Pacific Conference on Communications (APCC'03)*, vol. 1, sept. 2003.

[8] R. Santos, P. Pedreiras, F. Yekeh, T. Nolte, and L. Almeida, "On hierarchical server-based communication with switched ethernet," in *15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'10)*, sept. 2010.

[9] A. Mifdaoui, F. Frances, and C. Fraboul, "Performance analysis of a master/slave switched ethernet for military embedded applications," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, nov. 2010.

[10] M. Zhang, J. Shi, T. Zhang, and Y. Hu, "Hard real-time communication over multi-hop switched ethernet," in *The 2008 IEEE Int. Conference on Networking, Architecture, and Storage (NAS'08)*, june 2008.

[11] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an afdx network," in *18th Euromicro Conference on Real-Time Systems*, 0-0 2006.

[12] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach," *IEEE Trans. on Industrial Informatics*, nov. 2010.

[13] R. Queck, "Analysis of ethernet avb for automotive networks using network calculus," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES'12)*, July 2012.

[14] M. Manderscheid and F. Langer, "Network calculus for the validation of automotive ethernet in-vehicle network configurations," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC'11)*, October 2011.

[15] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks," *Elsevier Performance Evaluation*, vol. 63, October 2006.

[16] J. Schmitt, F. Zdarsky, and M. Fidler, "Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch..." in *The 27th IEEE Conference on Computer Communications*, April 2008.

[17] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "A novel approach to scalable cac for real-time traffic in sink-tree networks with aggregate scheduling," in *the 1st international conference on Performance evaluation methodolgies and tools*. ACM, October 2006.

[18] R. Marau, P. Pedreiras, and L. Almeida, "Asynchronous traffic signaling over master-slave switched ethernet protocols," in *6th International Workshop on Real Time Networks (RTN'07)*, July 2007.

[19] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *24th IEEE International Real-Time Systems Symposium (RTSS'03)*, 2003.

[20] M. Behnam, T. Nolte, and R. J. Bril, "Bounding the number of self-blocking occurrences of sirap," in *31th IEEE Real-Time Systems Symposium (RTSS'10)*, December 2010.