Response Time Analysis of Multi-Hop HaRTES Ethernet Switch Networks

M. Ashjaei[1], P. Pedreiras[2], M. Behnam[1], R. J. Bril[1,4], L. Almeida[3], T. Nolte[1]
[1] MRTC/Mälardalen University, Västerås, Sweden
[2] DETI/IT/University of Aveiro, Aveiro, Portugal
[3] IT/DEEC/University of Porto, Portugal
[4] Technische Universiteit Eindhoven (TU/e), The Netherlands

Abstract

In this paper we focus on micro-segmented switched-Ethernet networks with HaRTES switches. HaRTES switches provide synchronous and asynchronous real-time traffic scheduling, dynamic Quality-of-Service adaptation and transparent integration of real-time and non-real-time nodes. Herein we investigate the challenges of connecting multiple HaRTES switches in order to build multi-hop communication and we propose a method, named Distributed Global Scheduling, to handle the traffic forwarding in such an architecture while preserving the unique properties of the single HaRTES switch case. Moreover, we develop a response time analysis for the method. We also evaluate the level of pessimism embodied in the analysis. Finally, we show the applicability of the proposed method in an industrial setting by applying it in an automotive case study.

1. Introduction

Over the last decades, the communication requirements of networked real-time embedded systems became overly complex, exceeding the capabilities of conventional communication protocols. The complexity arises from advances in embedded equipments, increments in their functionalities along with a high amount of information to be exchanged within the embedded systems. On the other hand, many challenges are imposed by new requirements in the mentioned complex system when timeliness constraints must be enforced. These new requirements include incorporating the traffic with diverse activation patterns (event- and time-triggered) and on-the-fly reconfiguration support without service disruption.

Ethernet was introduced as a promising approach for the communication among embedded systems due to properties such as cost, availability and expandability. However the non-determinism of COTS Ethernet impaired its use in time-critical applications. Thus, many Real-Time Ethernet (RTE) protocols were introduced to overcome the limitations, such as EtherCAT, TTEthernet, PROFINET IRT, to name a few. These protocols profit from features of traditional Ethernet technology like high throughput, affordability and availability, yet also providing deterministic communications.

However, most of the RTE protocols found in the literature have severe limitations dealing with dynamic real-time applications. These applications are characterized by having evolving requirements (e.g. message streams may be added, removed and updated) which, despite being volatile, are subject to strict timeliness requirements. RTE protocols that provide strict determinism adopt static scheduling, thus impeding any sort of effective online adaptation to the communication requirements.

As a result, the Hard Real-Time Ethernet Switching architecture (HaRTES) [1] [2] has been developed in order to provide a more dynamic, adaptive and resource-efficient protocol in distributed embedded systems. HaRTES supports all types of traffic including real-time periodic, real-time sporadic and non-real-time traffic. The former is classified as synchronous traffic and the two latter types are identified as asynchronous traffic. The non-real-time traffic is scheduled in the background. Moreover, HaRTES allows reserving bandwidth for each type of traffic (synchronous and asynchronous) in order to create temporal isolation among them.

Distributed embedded systems comprising several tens of nodes are becoming a commonplace. Handling such a high number of nodes is far beyond the capacity of a single switch. The single switch HaRTES architecture has been investigated in depth, whereas the extension to multi-hop communication is still an open issue. The effort to build a multi-switch HaRTES architecture without jeopardizing the unique dynamic, real-time and traffic isolation capabilities of the HaRTES switch has been initiated in [3]. Herein, we complement one of the proposed solutions and describe it in more detail. The main contributions of this paper are:

1. We propose a method, namely Distributed Global Scheduling (DGS), to handle the traffic in a multi-hop HaRTES architecture, preserving the basic properties of the HaRTES architecture.

2. We develop a response time analysis for synchronous traffic in a single-switch HaRTES architecture and we extend it for the multi-hop architecture with the DGS method. The response time analysis for asyn-

chronous traffic has been presented in [1] and it is out of the scope of this paper.

3. Using the simulation tool presented in [4], we check the validity of the response time analysis for a specific example and we study the potential pessimism embodied in the analysis. The simulation tool was extended to support the HaRTES framework.

4. Finally, we study the applicability of the method using an automotive case study.

The rest of the paper is organized as follows. The next section describes the related work. Then, Section 3 presents the HaRTES architecture. Section 4 presents the multi-hop HaRTES architecture. Section 5 presents the system model, while Section 6 describes the response time analysis. Section 7 shows the evaluation of the proposed method and finally Section 8 concludes the paper and presents future work.

2. Related Work

The literature on switched Ethernet is vast and there have been many works addressing its adequacy to real-time communication. There are relatively old research proposals such as EtheReal [5] and the EDF Scheduled Switch [6], both based on channel reservations supported on enhanced switches.

In the meanwhile, many solutions to real-time Ethernet actually made it to the market, such as TTEthernet [7] and PROFINET IRT [8], both optimized for time-triggered operation, and EtherCAT [9], optimized for quick forwarding with on-the-fly update of the Ethernet frames while traversing the nodes.

Also, AFDX [10] is a network communication specification with enhanced forwarding and rate filters, which has been used mainly in avionics. More recently, Audio Video Bridging (AVB), which is the common name for a set of technical standards developed by IEEE, is gaining a momentum, mainly in the automotive industry. This protocol supports clock synchronization, bandwidth reservation and traffic shaping services. However its market support is still residual and it has some intrinsic limitations, like the low number of priorities (8 max.), lack of explicit support of isochronous traffic and a rigid reservation mechanism, that has limited capabilities in terms of specification of the stream properties.

These solutions, using enhanced switches, present improved performance but result in high cost and lower availability than current COTS Ethernet switches (IEEE 802.1D). 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 [11] and the FTT-SE [12] protocol, both using master/slave techniques.

Although the FTT-SE protocol provides a bandwidth-efficient solution, it presents some structural limitations due to the use of COTS switches. In fact, the protocol requires all nodes to be FTT-compliant. This leads to have a specific network device driver in the operating system. HaRTES overcomes this problem by inserting the master module inside the switch in order to add traffic confinement capabilities to the switch. We will focus on the HaRTES switch in this paper.

Despite the obvious similarities between FTT-SE and HaRTES, there are also subtle but important differences, which have a strong impact in the operation and performance of both protocols. In particular, in HaRTES it is possible to have differentiated bandwidth allocations, i.e., different links may have different reservations, which are computed individually, according to the traffic that actually crosses them. Therefore, the direct application of the results previously developed for multi-hop communication in FTT-SE networks (e.g. [13]) would result in sub-optimal performance. This, in fact, is the main reason that motivated this work.

Concerning the timing analysis of multi-switch Ethernet networks, several methods are also available. For example, Network Calculus is used in [14] to analyse the end-to-end delays in FTT-SE using a single master multi-switch topology and in [15] for networks of standard Ethernet switches. The work in [16] 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 higher pessimism. A tighter timing analysis for AFDX networks can be achieved using the trajectory approach as reported in [17]. However, it has been shown in [18] that for some specific cases, which have not existed in AFDX configuration, the trajectory approach presents some optimism.

Network calculus is also used in [19] to derive end-to-end traffic delays for Ethernet AVB, showing a case study based on an automotive infotainment system. The work in [20] presents a worst-case delay 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 [21] and [22] 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 the 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 [23] utilized the mentioned method to investigate an admission control in sink-tree networks.

In this paper we present a response time analysis applied to the multi-hop HaRTES architecture, as the Network Calculus showed higher pessimism in similar cases.

3. HaRTES Architecture

We define the HaRTES architecture as a micro-segmented network using exclusively HaRTES switches. The switch is responsible to schedule the traffic, both synchronous and asynchronous types, on-line according to any desired scheduling policy, e.g., Fixed Priority Scheduling Policy. The switch organizes the communication in fixed-duration time-slots, designated Elementary Cycle (EC). Each EC is composed by two windows, one for scheduling the synchronous traffic and the other one for asynchronous traffic (Figure 1).

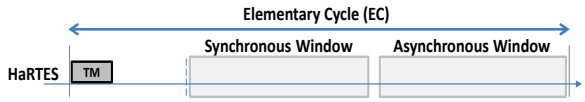

Figure 1. The EC Partitioning in HaRTES

The HaRTES switch implements a central repository which contains all information related to the traffic management, namely the message attributes. Such attributes include deadline, minimum inter-arrival time/period, message length and priority.

Synchronous messages are activated and scheduled by the switch. In each EC the switch computes the new activations, updates the ready queue and determines which messages that fit in the EC's synchronous window. The scheduled messages are encoded into a Trigger Message (TM) which is then transmitted to the slave nodes at the beginning of the EC. In contrast to the synchronous messages, asynchronous traffic is transmitted autonomously by the slave nodes and forwarded by the switch through a hierarchy of servers [1]. This leads to enforced traffic isolation that effectively handles the distribution of shared resources and guarantees the minimum QoS levels.

The HaRTES switch has two kinds of latency, known as *store-and-forward delay* and *hardware fabric latency*. The former delay corresponds to the time required to receive the message before forwarding it to the output link, thus it depends on the message size and bit rate, while the latter delay is due to the message processing inside the switch. The summation of the two latencies is called *switching delay*.

The main aim of the scheduler inside the switch is to schedule the messages on-line without causing overrun in the EC, i.e., all scheduled messages should be received by the end of the EC. The HaRTES switch is full duplex which means that transmission of messages in uplinks (nodes to switch) and downlinks (switch to nodes) are distinguished. In order to keep track of the utilization of the allocated windows in each link, the scheduler considers two bins per link and per window, representing the uplink and the downlink. The size of the bins is equal to the length of the windows allocated in the EC. The scheduler starts from the higher priority message in the ready queue and fills the bins associated to the links in the route of the message. Also, the scheduler takes into account the switching delay of the message when filling the associated bins.

Assume a network example depicted in Figure 2 and consider that two messages, m_1 and m_2, are ready to be transmitted from node S1 to node S2, and from node S3 to node S2, respectively.

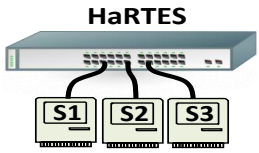

Figure 2. HaRTES Network Example

As it is shown in Figure 3, the scheduler considers 3 bins to schedule messages m_1 and m_2. These 3 bins are the uplinks from node S1 and S3, and the downlink to node S2. Assume that m_2 is the higher priority message. Therefore, the scheduler first fills the uplink bin S3 and the downlink bin S2 with m_2 considering its switching delay. Continuously, the scheduler picks m_1, as the next ready message in the ready queue, and fills the uplink bin S1 and the downlink bin S2. Note that messages m_1 and m_2 are sent to the switch concurrently, thus the switching delay does not add up. In the worst case it suffices to consider the longest delay, which is the reason why in Figure 3 only the m_2 switching delay is considered in the downlink S2. Also note that, in the example the size of m_2, hence the size of its switching delay, is greater than the size of m_1. In general, in each EC the scheduler picks the maximum switching delay among the messages that fit in that EC. This mechanism will be taken into account in the response time analysis.

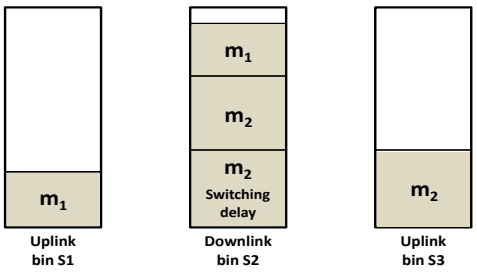

Figure 3. The Bins for Scheduling

In the above outlined example, both m_1 and m_2 fit in the respective bins, thus they can be transmitted during the next EC.

4. Multi-Hop HaRTES Architecture

In this section, we describe a topology of the multi-hop HaRTES architecture as well as the DGS method for the message forwarding in such an architecture.

4.1 Multi-Hop HaRTES Topology

In order to build an architecture with multiple HaRTES switches, we propose to connect the switches in a tree topology as illustrated in Figure 4. This topology is commonly found in distributed embedded systems, presenting a good compromise between cabling length and routing complexity.

In this architecture we define two types of messages. The messages that are transmitted between nodes connected to the same switch are called *local* messages, whereas the messages that are transmitted between two nodes connected to different switches are called *global* messages. Moreover, we define the connections between nodes and a switch as *local-links*, while the connections among the switches are defined as *inter-links*. Finally, the HaRTES switch on the top of the tree topology is called the *root switch*.

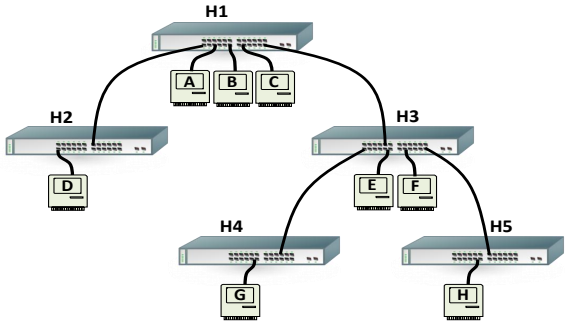

Figure 4. The Multi-Hop HaRTES Architecture

4.2 DGS Method

In this section, we propose a method, named Distributed Global Scheduling (DGS), to handle the traffic forwarding through multiple switches.

In this method, the scheduling of global traffic is carried out in a distributed fashion by all involved switches. Basically, each switch schedules its hop without distinction of global or local messages. Firstly, the switch to which the source node is directly connected to, schedules the message for transmission and stores it in its own memory, thus behaving as a consumer of the message. Then, in a posterior EC, the next switch in the route schedules the message for transmission, which is sent by the first switch and stored in the local memory of the second switch. The step-wise scheduling continues until the last switch, for which the destination node is connected to. The last switch schedules the message to be received and transmitted to the destination node, as the consumer of the message is attached to that switch. Thus, the message is not buffered in the last switch and it is immediately forwarded to the destination node in one EC.

In order to keep the time-triggered model for synchronous global traffic, the messages are activated periodically in the switches considering a *phase* defined for each message. The phase for a message is defined differently in each switch in the route of the message, and it determines a time delay between the activation time in the switch where the message is being scheduled and the activation time in the source node. The phase is specified in number of ECs and is essential to guarantee that, in each hop, messages are always forwarded after being received from the previous switch.

To illustrate the operation of the message forwarding process, consider the network depicted in Figure 4 and the Gantt chart represented in Figure 5. Message m_1 shall be transmitted from node D, connected to switch H2, to node E, connected to switch H3. Firstly, H2 schedules m_1 and stores it in its own local memory (EC_k). In the following EC the message is scheduled by switch H1, being transferred from the internal memory of H2 to the internal memory of H1 (EC_{k+1}). Finally, the message is scheduled by H3, being transmitted from the internal memory of H1 to node E (EC_{k+2}). In the first two ECs there is no impact of the switching delay on the message response time since the message is buffered in the switch, while in the last EC the switching delay needs to be accounted for as the message is forwarded to the destination node without being buffered.

Note that this is a very simple case. In reality messages may be delayed several ECs in each switch if several higher priority messages, exceeding the capacity of the EC, are waiting to be transmitted in the interlink that connects the switches. Thus, the phase for a message, to be taken into account in each hop, is the worst-case response time of the message from the source node to that hop, i.e, the maximum delay accumulated in the ancestor links.

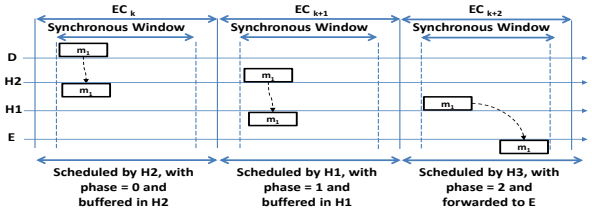

Figure 5. The Operation of the DGS Method

The definition of consistent phases in the transmission of synchronous messages across multiple switches requires global synchronization. The HaRTES architecture guarantees a contention-free transmission of the TM, which is broadcasted to all the nodes connected to a switch, including other switches down the tree topology. Therefore, the TM can be used as a precise time mark. In this proposal, all switches synchronize with their parent switch whenever receiving a TM, hence providing synchronized ECs. The exception is the root switch, which behaves as a time master.

The asynchronous traffic is forwarded by the switches, within the asynchronous windows through a hierarchy of servers. During the synchronous windows, or when the capacity of associated servers is exhausted, such traffic is suspended and queued.

5. System Model

In this paper, we use the real-time periodic model to represent synchronous messages. The message set, composed by N messages, is defined as follows:

$$\Gamma = \{m_i(C_i, D_i, T_i, P_i, S_i, Ds_i, \mathcal{L}_i), i = 1..N\} \quad (1)$$

In this model, C_i is the transmission time, D_i is the relative deadline and T_i is the period of m_i. The period and deadline for the messages are stated as an integer number of ECs. The priority of m_i is denoted by P_i and the messages may share a priority level. Moreover, S_i is the source node and Ds_i is the destination node of the message. Currently we restrict our analysis to unicast streams, hence only one destination port per message is considered. $\mathcal{L}_i$ is the set of links that m_i passes through. Each element in $\mathcal{L}_i$ presents a tuple $l =< x_1, x_2 >$ which shows a link l between node/switch x_1 to node/switch x_2. Note that, as the switch is full duplex, the sequence inside the tuple shows the direction of the message transmission in that link.

In addition we consider a fixed-priority scheduling policy and that the priority of messages is assigned according to the Rate-Monotonic algorithm.

The switching delay of messages consists of the store-and-forward delay, which is denoted by SFD_i, and the fabric latency, specified by Δ. Note that, the store-and-forward delay equals to the transmission time (C_i) of the message.

In the HaRTES architecture, all messages scheduled to be transmitted in one EC, should be received by the end of the EC. In order to prevent overruns, scheduling of messages that cannot be fully transmitted within the transmission window is delayed for the next EC, e.g., m_3 in Figure 6.

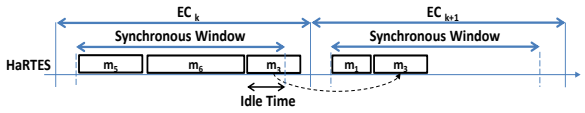

Figure 6. Inserted Idle Time

This property introduces an idle time in each transmission window. The idle time is denoted by I and should be taken into account in the response time analysis.

6. Response Time Analysis

In this section, we present the response time analysis for the single-switch HaRTES architecture. Then, we extend the presented analysis for the DGS method.

6.1 Single-Switch Response Time Analysis

The scheduling policy in the HaRTES switch is based on bandwidth reservation in each link which presents a resemblance with the periodic model in hierarchical scheduling [24]. Thus, in this paper we have used the associated analysis based on a *request bound function (rbf)* and a *supply bound function (sbf)* as a suitable method for evaluating the response time of the messages. The $rbf(t)$ represents the maximum load generated by a message and the $sbf(t)$ is the minimum effective communication capacity that a link in the network provides.

As the period and the deadline for messages are stated in number of ECs, the response time of m_i is also computed in number of ECs as shown in (2), where $\theta_i^{l_s,l_d} = min(t > 0) : sbf_i^{l_s,l_d}(t) \geq rbf_i^{l_s,l_d}(t)$. The response time is calculated in the route of the message that contains the source node local-link (l_s) and the destination local-link (l_d).

$$RT_i^{l_s,l_d} = \left\lceil \frac{\theta_i^{l_s,l_d}}{EC} \right\rceil \quad (2)$$

$rbf_i^{l_s,l_d}(t)$, as shown in (3), is computed by summing the transmission time of m_i, the interference from other messages and the switching delay of the interfering messages as well as the switching delay of m_i itself.

$$rbf_i^{l_s,l_d}(t) = C_i + W_i^{l_s,l_d}(t) + Is_i^{l_s,l_d}(t) \quad (3)$$

The interference from other messages is caused by the messages with a same or higher priority than m_i which share link l_s or l_d with m_i and it is denoted by $W_i^{l_s,l_d}(t)$. This type of interference is called *Shared Link Delay (SLD)*.

In order to calculate the SLD, all higher or same priority messages ($hep(m_i)$) that share link l_s or l_d with m_i should be accounted for. The SLD computation is shown in (4).

$$W_i^{l_s,l_d}(t) = \sum_{\substack{\forall j \in [1,N] \\ \wedge\ m_j \in hep(m_i) \\ \wedge (l_s \in \mathcal{L}_j \vee l_d \in \mathcal{L}_j)}} \left\lceil \frac{t}{T_j} \right\rceil C_j \quad (4)$$

As described in Section 3, the switching delay of interfering messages is combined with the switching delay of message m_i, in order to bound its impact in the transmission window. The switching delay interference is denoted by $Is_i^{l_s,l_d}(t)$.

To determine the switching delay interference $Is_i^{l_s,l_d}(t)$, we define a multi-set $G_i^{l_s,l_d}(t)$ which contains the switching delays of all messages that contribute to the SLD and are activated within interval $[0, t)$ as well as the switching delay of m_i. Such a method was introduced in [25] to improve the response time analysis of a synchronization protocol in a hierarchical scheduling framework. A message may be activated several times within an interval. Therefore, all activation instants of a message need to be added to the multi-set. The multi-set is formulated in (5), where y shows the number of instances that the message m_j is activated, hence y

number of switching delays for that message should be added to the multi-set.

$$G_i^{l_s,l_d}(t) = \{ \bigcup_{y=1..\lceil \frac{t}{T_j} \rceil} (SFD_j + \Delta) : m_j \in hep(m_i) \wedge (l_s \in \mathcal{L}_j \vee l_d \in \mathcal{L}_j)\} \cup (SFD_i + \Delta) \quad (5)$$

Tracking time t, in the analysis, one element per each EC will be taken from $G_i^{l_s,l_d}(t)$ and to consider the worst-case scenario, the largest element will be selected. Therefore, we define a sequence $\left(G_i^{l_s,l_d}(t)\right)^{sort}$ that includes the switching delay values from $G_i^{l_s,l_d}(t)$ in a descending order. The computation of the switching delay interference is shown in (6), where $z(t)$ denotes the number of ECs that has been passed, hence the number of elements from $\left(G_i^{l_s,l_d}(t)\right)^{sort}$.

$$Is_i^{l_s,l_d}(t) = \sum_{k=1}^{z(t)} \left(G_i^{l_s,l_d}(t)\right)^{sort}[k] \quad (6)$$

$$z(t) = \left\lceil \frac{t}{EC} \right\rceil \quad (7)$$

$sbf_i^{l_s,l_d}(t)$ is the communication capacity that links l_s and l_d provide in the time interval $[0, t)$. Note that each EC comprises a window dedicated to each type of traffic and therefore a different length can be set for each link (i.e., LW^{l_s} for the source local-link and LW^{l_d} for the destination local-link). The minimum guaranteed window size, which may actually be used to carry messages, is smaller than the nominal value due to the need of preventing window overruns, being given by $LW^{l_s} - I_i^{l_s}$ for the source local-link, as an example. The idle time is upper bounded by the maximum packet size of m_i and all messages with a higher priority than m_i sharing link l_s or l_d with m_i. Therefore, the supply bound function is computed in (8), where the idle time for link l is computed in (9). Note that, the window length provided in the source local-link may be different from the destination local-link. In order to consider the worst-case scenario we take the minimum between them in (8).

$$sbf_i^{l_s,l_d}(t) = \frac{min(LW^{l_s} - I_i^{l_s}, LW^{l_d} - I_i^{l_d})}{EC} \times t \quad (8)$$

$$I_i^l = \max_{\forall m_p \in hep(m_i) \wedge\ l \in \mathcal{L}_p} (C_p) \quad (9)$$

Note that the $sbf_i^{l_s,l_d}(t)$ presented in (8) is the approximation of the supply bound function. However, as we calculate the response time in number of ECs, the approximation does not introduce additional pessimism. Figure 7 shows the exact $sbf_i^{l_s,l_d}(t)$, the approximation of that and the $rbf_i^{l_s,l_d}(t)$ for m_i.

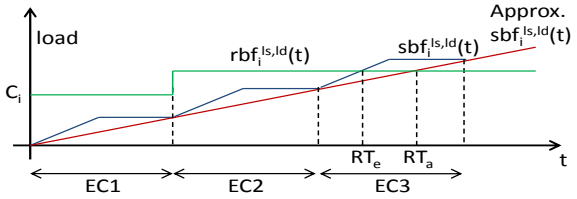

Figure 7. The Approximation of the Supply Bound Function

The $rbf_i^{l_s,l_d}(t)$ meets both exact and approximate $sbf_i^{l_s,l_d}(t)$ in the third EC. Thus, the response time using both $sbf_i^{l_s,l_d}(t)$ and the approximation is 3 EC. Therefore, we can use the approximation of the supply bound function in this particular network protocol to calculate the exact response time.

Determining $\theta_i^{l_s,l_d}$ in (2) requires checking the inequality at all instants where $rbf_i^{l_s,l_d}(t)$ changes due to the interference of other messages. Such set of checkpoints is given in (10).

$$CP_{rbf_i^{l_s,l_d}} = \{\bigcup cp_{m_a} : m_a \in hp(m_i) \wedge (l_s \in \mathcal{L}_a \vee l_d \in \mathcal{L}_a)\} \cup T_i \quad (10)$$

$$where, cp_{m_a} = T_a, 2T_a, ..., \alpha T_a, \alpha = \left\lfloor \frac{T_i}{T_a} \right\rfloor$$

The response time analysis for asynchronous streams for a single-switch HaRTES architecture is presented in [1], hence it is out of the scope of this paper.

6.2 Multi-Hop Architecture Response Time Analysis

In this section, we extend the presented analysis for the context of a multi-hop HaRTES architecture.

As described in Section 4, in the DGS method, the global messages are buffered in each switch before being scheduled by the next switch. Therefore, the response time of the global messages is the sum of response times in each switch, which is calculated separately. Moreover, local messages are transmitted through one switch within the same window as global messages (i.e., synchornous window). Thus, the global and local messages that share links may interfere with each other.

Global messages are buffered in each switch, except in the last switch where they are forwarded to the destination node in the same EC, i.e., the last switch acts as a single-switch case. Therefore, response time analysis in the last switch is computed similarly to the analysis presented in Section 6.1. However, in the other switches, the messages are scheduled for the input links, only. Thus, the response time analysis should be performed for each one of the input links of the switches.

The response time for a global message m_i is calculated in (11), where l is the set of links in the route of the message except the last two links (i.e., the inter-link to the last switch denoted by l_x and the local-link to the destination node denoted by l_d). The response time in the last

switch $RT_i^{l_x,l_d}$ is computed similarly to the single-switch case as presented in Section 6.1.

$$RT_i = \sum_{\forall l \in \mathcal{L}_i \wedge\ l \neq l_x \neq l_d} RT_i^l + RT_i^{l_x,l_d} \quad (11)$$

The response time in link l is computed in number of ECs as shown in (12), where $\Theta_i^l = min(t > 0) : sbf_i^l(t) \geq rbf_i^l(t)$.

$$RT_i^l = \left\lceil \frac{\Theta_i^l}{EC} \right\rceil \quad (12)$$

$rbf_i^l(t)$ is calculated similarly to (3), except considering l instead of two links (l_s, l_d). Moreover, thanks to the buffering feature of the switch, the switching delay is not applicable as we are not forwarding the messages in the same EC, i.e., $Is_i^l(t) = 0$. Therefore, $rbf_i^l(t)$ is computed in (13).

$$rbf_i^l(t) = C_i + W_i^l(t) \quad (13)$$

The SLD ($W_i^l(t)$) is the interference from the same or higher priority messages than m_i which share link l with m_i. This delay is computed in (14).

$$W_i^l(t) = \sum_{\substack{\forall j \in [1,N] \\ \wedge\ m_j \in hep(m_i) \wedge \\ l \in \mathcal{L}_j}} \left\lceil \frac{t}{T_j} \right\rceil C_j \quad (14)$$

$sbf_i^l(t)$ is also computed similarly to (8), except there is one link l available for message transmission. Therefore, the supply bound function is calculated in (15), where I_i^l is computed in (9).

$$sbf_i^l(t) = \frac{LW^l - I_i^l}{EC} \times t \quad (15)$$

In order to determine the checkpoints in which $rbf_i^l(t)$ changes due to the interference of other messages, we verify the checkpoints similarly to (10), except considering one link l instead of l_s and l_d.

Note that, the response time analysis for local messages is performed similarly to the single-switch case.

7. Evaluation

In this section, we present two different evaluations. First, we check the validity of the presented response time analysis by comparing the results observed from simulation with the ones calculated by the analysis, using a high loaded example. Then, we investigate the applicability of the DGS method in an industrial setting by applying this method in an automotive case study.

7.1 Analysis Evaluation

In this evaluation, we assess the level of pessimism embodied in the analysis by loading up the links in a network example. In order to simulate the example we have used the simulation tool presented in [4]. However, the simulation tool was initially developed to evaluate the FTT-SE protocol architectures. Therefore, for this paper, we have modified the tool to support the proposed method.

The simulation tool [4] has been implemented using Simulink. The kernel of the tool has been designed in a cycle-based way such that it is easy to accommodate any FTT-based protocols. To modify the tool for this paper we have changed the scheduler module in the kernel.

In order to evaluate the analysis, we considered a network comprising 4 switches along with 10 nodes, as depicted in Figure 8.

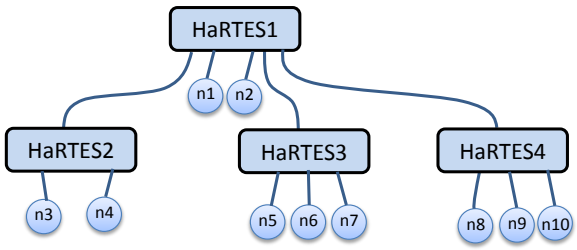

Figure 8. A Network Example

We defined 28 messages that include both local and global traffic. The parameters of the defined messages are shown in Table 1. From the defined messages we tagged two of them, one local and one global, and we generated an unfavorable situation for them. The tagged message m_1, as a global message, and m_2, as a local message. In order to increase the interference for the tagged messages, we assigned the lowest priority to them. Also, for the global tagged message, we selected the longest route in the network to increase the total interference that it is subject to, which accumulates over each link that the message crosses. The other messages (m_3 to m_{28}) are defined to interfere with the two tagged messages such that they generate a long delay for them. The periods of the interfering messages are selected within $[4, 12]EC$ and the priorities of them are selected based on the Rate Monotonic algorithm within $[1, 10]$. Moreover, the transmission time of the interfering messages are chosen within $[100, 123]\mu s$, where $123\mu s$ equals to 1542KB as the maximum Ethernet frame size. Note that, a higher value for P_i equals to the lower priority (i.e., 1 presents the highest priority).

In this example, the network capacity is set to $100Mbps$ and the switch latency is $\Delta = 3\mu s$. Also, the EC is considered to be $1ms$. Moreover, we assigned $600\mu s$ to the synchronous window within the EC.

We simulated the above described example for a hyper period (least common multiple of the periods), which is 27720 ECs. Figure 9 illustrates the maximum response time of the messages measured from the simulation and the ones computed using the proposed analysis. In this figure, the x-axis represents a message id and the y-axis shows the message response time in number of ECs.

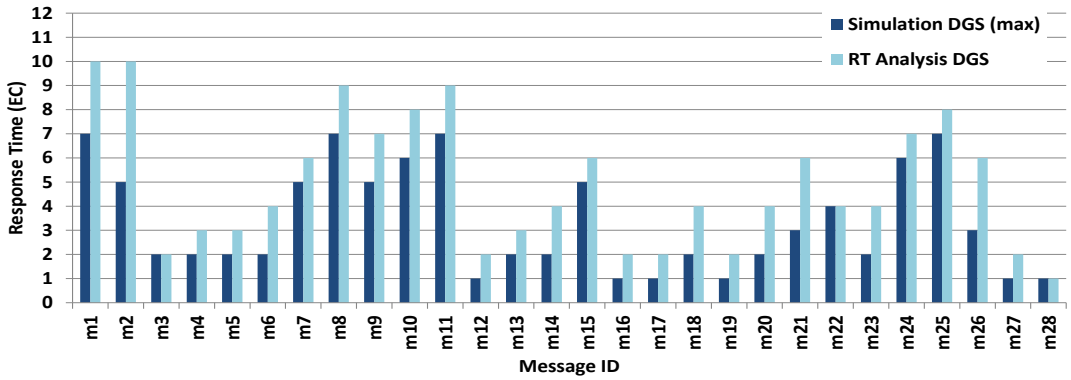

Figure 9. Response Time of the Messages using the DGS Method

ID	T=D (EC)	P	C(μs)	S	Ds	G/L
m_1	12	10	123	n3	n8	G
m_2	12	10	123	n5	n6	L
m_3	5	1	123	n3	n1	G
m_4	6	2	123	n3	n2	G
m_5	7	3	100	n4	n1	G
m_6	8	4	123	n4	n2	G
m_7	9	5	100	n3	n8	G
m_8	9	6	123	n4	n9	G
m_9	10	7	123	n5	n8	G
m_{10}	11	8	123	n6	n9	G
m_{11}	11	9	100	n7	n10	G
m_{12}	12	10	110	n9	n8	L
m_{13}	4	1	123	n1	n9	G
m_{14}	6	2	123	n1	n10	G
m_{15}	9	4	100	n4	n10	G
m_{16}	5	1	100	n7	n6	L
m_{17}	6	2	123	n7	n6	L
m_{18}	7	3	123	n5	n6	L
m_{19}	7	6	123	n5	n7	L
m_{20}	10	7	123	n5	n6	L
m_{21}	10	8	110	n5	n6	L
m_{22}	12	10	123	n7	n6	L
m_{23}	4	1	123	n1	n6	G
m_{24}	6	2	123	n8	n6	G
m_{25}	7	3	110	n10	n6	G
m_{26}	10	8	123	n5	n6	L
m_{27}	4	1	123	n7	n6	L
m_{28}	6	2	123	n5	n7	L

Table 1. The Tagged Message Parameters

As we can see from the results, the analysis introduces a pessimism for both local and global messages. The response time measured in the simulation for m_1 (as the global tagged message) is $7EC$, while the analysis calculates $10EC$. Moreover, the local tagged message (m_2) has $10EC$ computed response time, while it is measured $5EC$ in the simulation.

We believe that the pessimism in the analysis stems from two different sources.

The first source of the pessimism is the phase of messages, described in the DGS method, which is not considered in the response time analysis. As we observed in the simulation, neglecting the phase in the analysis is the main reason of the pessimism. However, adding that to the analysis is not straightforward and it is one of the directions for future work.

In order to describe the pessimism cause, assume a simple network example depicted in Figure 10. Also, consider two messages, m_1 transmitted from n1 to n2 and m_2 transmitted from n3 to n2. We assume m_2 is the lower priority message and that each message needs one EC for transmission. The defined messages share the n2 downlink. Therefore, when computing the response time for m_2, the interference from m_1 is added.

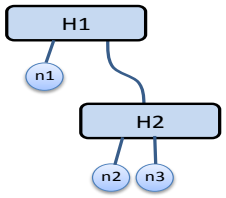

Figure 10. A Network Example to Show the Pessimism

The scheduling window is depicted in Figure 11. m_1 is transmitted to switch H1 in EC1, then it is forwarded to n2 in EC2. m_2 is transmitted in the first EC, where m_1 does not interfere in that EC. Therefore, m_1 does not delay m_2 as it is transmitted with a 1EC phase. Considering this example, the response time for m_2 is 1EC, while by adding the interference of m_1 in the analysis, it is computed 2EC.

The second source of the pessimism is the calculation of the switching delays for the interfering messages.

For the global messages we consider the switching delay in the last switch of the route of the message, only. However, it can produce a pessimism when we define several interfering messages sharing links with the message under analysis in the last switch. The reason is that the scheduler takes the maximum switching delay of the messages scheduled for each EC, while in the analysis, as we do not have such an information, we consider the max-

imum switching delay among all messages that are activated during an interval of time.

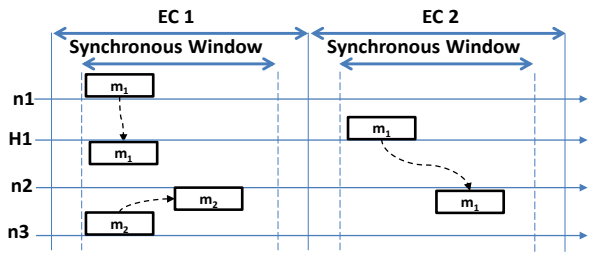

Figure 11. Scheduling Window for Pessimism Example

7.2 Automotive Case Study

Currently the interest of using high bandwidth network technologies in automotive industries is growing rapidly. This demand is inherent in the use of innovative applications as well as entertainment devices in a vehicle. The work presented in [26] and [27] investigate the performance of in-car switched Ethernet using real traffic sets in a vehicle. The former work studies the use of traffic without prioritization in a switched Ethernet network, whereas the latter work investigate the average end-to-end delay of the same traffic set with prioritization and scheduling it using a weighted fair queuing. In this case study, we use the same set of traffic as in [26] and we define a network using HaRTES switches on which we apply the DGS method.

The network comprises 12 nodes, including the Head Unit, cameras, a multimedia node and control nodes. The network topology for this case study is depicted in Figure 12, where the nodes are connected according to the geometrical arrangement of nodes in a vehicle (HaRTES2 is in the front and HaRTES3 is in the rear of the vehicle).

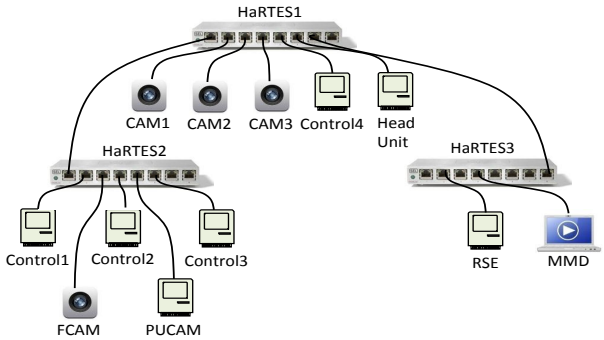

Figure 12. Network Topology for Automotive Case Study

There are three different types of traffic including control data, camera streaming, video and audio streaming. The parameters of the traffic are given in Table 2.

In this example, all traffic types are synchronous. We set the EC length to $2ms$, where we, within the EC, assigned $1700\mu s$ to the synchronous window.

The maximum end-to-end delay of the messages as they are measured from the simulation and their corresponding deadlines in number of ECs are shown in Table 3. The more critical messages in this set is the control messages (m_1 to m_4) with very short and firm deadlines. The other messages have larger deadlines as they are audio and video streams. As it can be seen from the results, all deadlines are met by observing the response time of the messages using the simulation.

ID	D(EC)	RT(EC)	ID	D(EC)	RT(EC)
m_1	5	2	m_6	22	2
m_2	5	2	m_7	22	2
m_3	5	2	m_8	22	1
m_4	5	1	m_9	75	1
m_5	22	2	m_{10}	75	1

Table 3. Response Time of the Automotive Case Study

8. Conclusion and Future Work

In this paper we investigated the challenges of multi-hop communication using HaRTES switches. We proposed a method to handle traffic forwarding when connecting multiple HaRTES switches in a tree topology. The proposed method is based on buffering the messages in each hop and scheduling them for the following ECs. Also, we developed a response time analysis for the single-switch HaRTES architecture and we extended it for the multi-switch case. We evaluated the response time analysis by comparing the results obtained from the analysis with the ones measured in the simulation using an example. Also, we showed the applicability of the proposed method using an industrially inspired case study.

Our future work aims at: (i) improving the response time analysis by further removing the sources of pessimism in the analysis (e.g., adding the phase of the messages into the analysis), and (ii) implementing the proposed method in the HaRTES switches and experimentally validate the methodology.

Acknowledgment

This work is supported by the Swedish Foundation for Strategic Research via the PRESS project. Also, it is partially supported by the Portuguese Government through FCT grants Serv-CPS PTDC/EEA-AUT/122362/2010 and CodeStream PTDC/EEITEL/3006/2012.

References

[1] R. Santos, M. Behnam, T. Nolte, P. Pedreiras, and L. Almeida, "Multi-level Hierarchical Scheduling in Ethernet Switches", in *Proceedings of the Int. Conference on Embedded Software (EMSOFT)*, October 2011.

ID	Type	Packet Size (bytes)	T (ms)	D (ms)	P	S	Ds
m_1 to m_4	Control	20	10	10	1	Control 1 to 4	Head Unit
m_5 to m_7	Camera	6288	4	45	2	CAM 1 to 3	Head Unit
m_8	Fcam	6288	4	45	2	FCAM	PU CAM
m_9	Audio	1472	10	150	3	MMD	RSE
m_{10}	Video	14720	10	150	3	MMD	RSE

Table 2. Traffic Parameters for Automotive Case Study

[2] "SERV-CPS Project, available at http://serv-cps.av.it.pt/".

[3] M. Ashjaei, P. Pedreiras, M. Behnam, L. Almeida, and T. Nolte, "Supporting Multi-Hop Communications with HaRTES Ethernet Switches", in *IEEE Real-Time Systems Symposium (RTSS), Work-in-Progress (WiP) session*, December 2013.

[4] 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)*, July 2012.

[5] S. Varadarajan and T. Chiueh, "EtheReal: a host-transparent real-time Fast Ethernet switch", in *6th Int. Conference on Network Protocols*, 1998.

[6] H. Hoang and M. Jonsson, "Switched real-time Ethernet in industrial applications - deadline partitioning", in *9th Asia-Pacific Conference on Communications (APCC)*, 2003.

[7] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan, "TTEthernet Dataflow Concept", in *8th IEEE International Symposium on Network Computing and Applications*, 2009.

[8] Z. Hanzalek, P. Burget, and P. Sucha, "Profinet IO IRT Message Scheduling", in *21st Euromicro Conference on Real-Time Systems (ECRTS)*, 2009.

[9] "EtherCAT, available at http://www.ethercat.org/".

[10] I. Land and J. Elliott, *Architecting ARNIC 664, Part 7 (AFDX) Solutions*, 2011.

[11] "POWERLINK, available at http://www.ethernet-powerlink.org/".

[12] 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)*, June 2006.

[13] M. Ashjaei, M. Behnam, L. Almeida, and T. Nolte, "Performance Analysis of Master-Slave Multi-Hop Switched Ethernet Networks", in *8th IEEE Int. Symposium on Industrial Embedded Systems (SIES)*, June 2013.

[14] A. Mifdaoui, F. Frances, and C. Fraboul, "Performance Analysis of a Master/Slave Switched Ethernet for Military Embedded Applications", *IEEE Transactions on Industrial Informatics*, 2010.

[15] M. Zhang, J. Shi, T. Zhang, and Y. Hu, "Hard Real-Time Communication over Multi-hop Switched Ethernet", in *The IEEE Int. Conference on Networking, Architecture, and Storage (NAS)*, 2008.

[16] 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*, 2006.

[17] 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*, July 2010.

[18] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard, "Optimistic problems in the trajectory approach in FIFO context", in *18th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, September 2013.

[19] R. Queck, "Analysis of Ethernet AVB for automotive networks using Network Calculus", in *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, July 2012.

[20] 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)*, October 2011.

[21] 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.

[22] 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.

[23] 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 ACM international conference on Performance evaluation methodolgies and tools*, October 2006.

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

[25] 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)*, December 2010.

[26] H.-T. Lim, K. Weckemann, and D. Herrscher, "performance study of an in-car switched ethernet network without prioritization", in *Proceedings of the Third international conference on Communication technologies for vehicles*, 2011. Springer-Verlag.

[27] H.-T. Lim, L. Volker, and D. Herrscher, "Challenges in a future IP/Ethernet-based in-car network for real-time applications", in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, 2011.