Engineering and Analyzing Multi-Switch Networks with Single Point of Control

M. Behnam
Mälardalen Real-Time Research Centre
P.O. Box 883, SE-721 23 Västerås, Sweden

Z. Iqbal, P. Silva, R. Marau, L. Almeida, P. Portugal
IT / DEEC
University of Porto, Portugal

***Abstract*—Recent trends in distributed embedded systems have shown an increase in the amount and heterogeneity of the information that needs to be exchanged, together with a growing importance of supporting dynamic reconfiguration and adaptive behaviors. In this paper we focus on Ethernet technology and we address the case of middle-size networking infrastructure with a few switches. We use the FTT-SE protocol to support dynamic heterogeneous real-time transactions with temporal isolation and we propose the needed scheduling adaptations to support multi-hop network configurations. The paper also includes a companion worst-case response-time analysis that allows verifying the timeliness of the system.**

I. Introduction

The complexity of Network Embedded Systems (NES) is increasing rapidly due to growing requirements on advanced functionality, with increasing amount and heterogeneity of the information that is exchanged. This high complexity imposes major development challenges when non-functional properties must be enforced, such as real-time response and adaptability to varying operational conditions, on which this paper will focus.

One of the networking technologies that is gaining more acceptance in NES is switched Ethernet technology given its attractive high throughput, low cost, wide availability and general maturity. However, when using COTS Ethernet switches, network interface cards (NIC) and IP stacks, guaranteeing real-time behavior is still challenging due to possible uncontrolled packet arrival patterns that will lead to packet queuing inside the switches and potentially to buffer overflows and consequently to packet drops. Moreover, the queues of COTS switches are typically FIFO, which are known to exhibit a poor real-time behavior due to potentially long blocking. This problem can be slightly alleviated using queues of different priorities but the IEEE 802.1D standard allows up to 8 priority levels, only, which is insufficient for priority scheduling of traffic streams.

Therefore, several real-time Ethernet (RTE) protocols were developed that in one way or another control the traffic submitted to the network and allow enforcing real-time response, such as AFDX, TTEthernet, PROFINET-IRT, EtherCAT and EthernetPOWERLINK. However, these protocols base their real-time properties on a static definition of the traffic, which is known before hand, thus their suitability to combine real-time response with adaptability is rather low.

On the other hand, the FTT-SE protocol [8] was proposed as a solution to achieve such combination. On one hand it enforces global coordination among different streams, thus controlling the load submitted to the switch at each instant and avoiding the potential queue overflow problems. On the other hand, it provides the possibility for fast and atomic updates to the traffic streams properties at runtime, it supports virtually any streams periods, it allows for arbitrary scheduling algorithms and it handles different types of streams efficiently such as real-time synchronous, real-time asynchronous and non real-time streams, with temporal isolation using strictly defined phases for each type.

However, the scalability of FTT-SE networks was considered in a limited way in previous works, i.e., assuming a single switch network [8]. Recently, three different architectures were proposed to extend the scalability of this protocol [9]. The first solution keeps the FTT single Master that schedules the whole network. The other two approaches consider the network broken into segments, each scheduled by its own Master. A qualitative assessment of the three architectures showed that none of them was superior to the others.

In this paper, we focus on the first architecture, i.e., based on a single Master that controls the whole network, which favors adaptability under real-time behaviour rather then efficiency of using the aggregated network bandwidth, being thus suited to systems that must reconfigure in a prompt and synchronized way. We show how to engineer such a multi-switch FTT-SE network, particularly how to schedule the traffic, and we present a preliminary traffic schedulability analysis based on the streams response-times, which are the respective end-to-end delays. A simulator allows us to validate the results of the analysis in one particular test case.

The remainder of the paper is organized in the following way. The next section discusses some related work, followed by a section describing the basics of the FTT-SE protocol. Then, Section IV presents the traffic scheduling model while Section V discusses the network architecture. Section VI describes the FTT-SE scheduling algorithm and Section VII discusses how the scheduler operates online. Section VIII presents the response-times upper bounds computation and Section X concludes the paper and presents on-going work.

II. Related work

In the area of scheduling real time traffic over multi-hop switched Ethernet, some work has been done to compute the worst case response time of real-time traffic using similar COTS switches without any hardware modifications. For example, [4] presented a method to compute the worst case communication delay based on timing analysis of the switched Ethernet. The calculation of the worst case delay is based on the assumption of having all messages generated by all source nodes except the destination node, arrive to the switch attached to the destination node simultaneously.

In [3] a schedulability analysis of hard real-time traffic transmitted over multi-hop switched Ethernet was presented assuming FIFO queues. This work assumed a more general network model, e.g., switches could have different bit rate and the deadline of messages could be arbitrary, i.e., greater, equal or less than message period. Finally, an upper bound of the buffer size for each output port was derived.

In the context of Avionic Full Duplex (AFDX) switched Ethernet, [2] presented three methods based on network calculus, simulation and model checking to evaluate the end-to-end delays. However, the network calculus approach adds a high pessimism in the analysis while simulation approach may not be able to find the worst case scenario and the model checking approach is more accurate but not scalable. The pessimism in the network calculus was decreased by considering the serial transmission of messages that are sent from the same source node and also the possibility of considering the offset associated with the periodic traffic in [5]. A fourth approach based on trajectory principle was presented and optimized in [1]. The optimization in this approach is based on the serialization of messages from the same source node (similar to the optimization approach proposed for network calculus). A comparison study in [1] showed that the trajectory approach gives tighter (in average) results than the network calculus approach when calculating the worst case response time of messages.

Network calculus was also used in [10] to compute the upper delay bound of hard real-time traffic communication. Unlike all related work presented above, the network architecture considered in this work is based on using multi-master network architecture where master nodes are used as gateways to connect switches with each others. A traffic smoothing protocol is proposed to control all streams to ensure that all hard real-time messages meet their deadlines.

However, the analysis used in the previously mentioned works can not be applied directly to the network architecture considered in this paper due to the strict cycle-based traffic scheduling used by the FTT-SE protocol. Therefore, we first present the online traffic scheduler of FTT-SE and then we develop our own analysis based on response-time upper bounds.

III. FTT-SE basics

FTT-SE [8] is a research real-time communication protocol that exploits the Master/Slave paradigm to control the whole traffic in the network in a timely and flexible way. However, it relies on a specific technique known as Master/Multi-slave to reduce the typical high overhead of that paradigm while maintaining its advantages. In particular this technique takes advantage of the absence of collisions in the switches to parallelize streams within a given cycle called Elementary Cycle (EC), thus exploiting parallel forwarding. The Master node coordinates all transmissions in the network by sending a specific message in the beginning of every EC, called Trigger Message (TM), that contains the schedule for that EC.

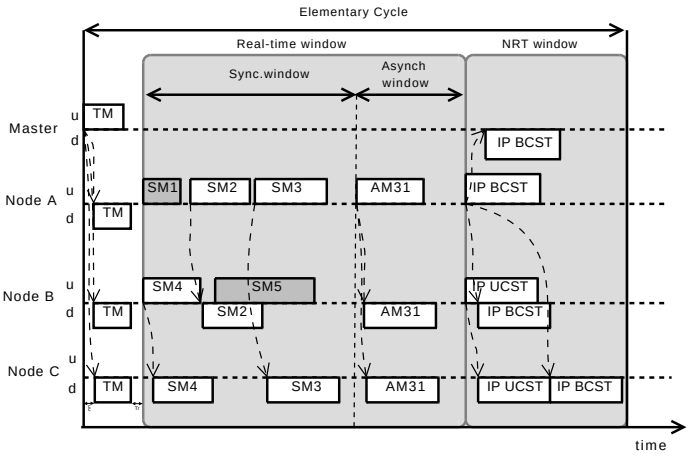

Fig. 1. The FTT-EC structure.

The traffic scheduling activity is carried out on-line and locally in the Master, invoked once per EC. The EC duration sets the timing resolution in the system. It is tunable and can be adapted to suit the application needs. Typically EC durations range from 1ms to tens of ms, depending on the application dynamics. Since the traffic scheduling is local to the Master, it is easy to enforce any kind of scheduling policy as well as to perform atomic changes in the communication requirements. This last feature facilitates on-line stream admission and removal under guaranteed timeliness as well as on-line bandwidth management.

The protocol supports real-time synchronous and asynchronous messages, as well as non real-time traffic. The synchronous traffic is time-driven, being activated autonomously by the scheduler. The asynchronous traffic and non real-time traffic are issued by the application at arbitrary time instants. These messages are locally queued and the status of the queues is periodically reported to the Master by means of a special message, called Signaling message (SIG). The Master collects the requests of the slaves and builds a global request queue which is then fed to the scheduler that schedules these and the synchronous messages in any desired integrated way.

As depicted in Figure 1, the EC is organized in two independent time windows, namely Real-time and Non-real-time. The former is further subdivided in Synchronous and Asynchronous. Each one of these windows is associated to a specific message class (real-time synchronous, real-time asynchronous and non-real-time). The windows have a user-defined maximum duration, which in enforced at run-time

by the scheduler. Consequently, no window overruns should occur, providing strict traffic isolation among traffic classes. Non real-time traffic is handled in a best-effort policy, with respect to the asynchronous traffic. Further details about the FTT-SE protocol can be found in [8] and [6].

IV. System model

In this paper, we consider a system composed by a single Master node and a set of N_{total} nodes and Ns_{total} switches, where each switch SW_i holds Np full-duplex ports. On each port, the switch-input link is referred to as uplink $ul_{j,k}$ and the switch-output is referred to as downlink ($dl_{j,k}$), where k is the switch identifier and j is the port number. The network is organized in a logical tree topology, for example by means of a spanning-tree protocol. A message transmission from one node to another, follows a single and unique route, crossing different switches that forward the message to the final destination. The Master node is connected to the root of the tree. N_{depth} defines the maximum number of branches, i.e, the maximum number of switches, in the route between any node and the Master. The switches are assumed to switch packets on store-and-forward fashion as most COTS switches do. That means a packet is only forward to a downlink upon complete arrival from the uplink. Currently, the topology and forwarding tables in the switches are assumed to be specified statically. However, a topology discovery mechanism is currently under development.

Synchronous message streams (SM) are modelled using the periodic real-time model $m_i(C_i, Mmax_i, D_i, T_i, S_i, Ds_i, R_i)$, where C_i is the total transmission time of the message and D_i and T_i are the relative deadline and period. Large messages are fragmented by the protocol into a sequence of packets with maximum $Mmax_i$ packet transmission time. Finally, R_i denotes the set of switches in the route of m_i from the source Node S_i to the destination node Ds_i. Asynchronous message streams (AM) are modelled using a sporadic real-time model which is identical to the periodic model explained above except that T_i is defined as the minimum inter arrival time, i.e., $m_i(C_i, Mmax_i, D_i, T_i, S_i, Ds_i, R_i)$.

V. Multiple switches under a single Master

Extending FTT-SE to multiple switch networks using a single Master substantially increases the complexity of the traffic scheduling due to the following reasons:

a) : Nodes may not receive the trigger message (TM) at the same time, as the TM message is delayed at each switch that it crosses. The delay of receiving the TM message, at each node, depends on the location of the node in the topology, i.e., the number of switches in the path to the Master. Moreover, at each node, the transmission of the signaling message and data traffic is synchronized with the reception of the TM. Receiving it at different times in different nodes leads to the nodes starting transmitting their traffic at different instants. For instance, the nodes that are closer to the Master will start their transmission earlier than the others and this should be taken into account when building EC schedules, i.e., when deciding which messages to transmit in each EC. Looking to the example in Figure 2, Nodes A, B and C receive the TM message before Nodes D and E since nodes A, B, C and the master are connected to the same switch while the other nodes are connected to a different switch (see Figure 3).

b) : As mentioned above, the transmission of the signaling message (from each node) is synchronized to the reception of the TM, inheriting any delay affecting it. In addition, the signaling messages are also delayed when routed on the reverse-path to the Master node. As a result, the total delay imposed on the signaling messages might be too long, possibly causing mutual interference between these messages and data messages. Thus affecting the reception of signaling messages in the Master node and the scheduling of the elementary cycle (EC).

c) : In the switches, whenever multiple messages are ready to be sent on the same downlink, the switch employs FCFS policy to sort the transmission order. The order of the message transmission has a great effect on the scheduling of messages, specially if the link serves the way to a number of message streams such as when providing connectivity between two switches. However, it is hard to predict the order of transmission at each downlink of each switch because of the possible jitter in the activation of messages (at their nodes) and also the jitter in the switch delays.

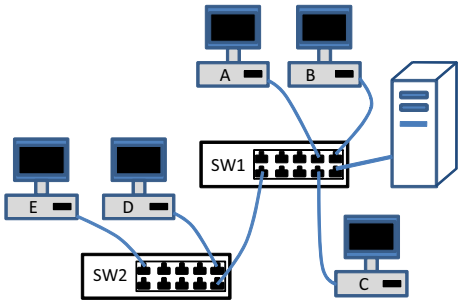

Fig. 2. An example of network with two switches and single Master

A. Message delay

The FTT-SE protocol fragments large messages into a sequence of packets that are individually scheduled. The scheduler in the Master takes into account all possible delays that packets may suffer when it builds the EC schedule to make sure that the scheduled traffic in one EC can always be fully transmitted within that EC.

A packet sent from one node to another node suffers from the following delays (see Figure 3):

1) SLD - Switch fabric Latency Delay: maximum processing time required by a switch to execute the internal forwarding functionality.
2) SFD - Store and Forward Delay: time required to save a message before forwarding it.
3) NQ - Source node queuing delay: caused by higher priority packets that are sent from the same source node.
4) SQD - Switch queuing delay: caused by all other packets that share downlinks with the packet.

5) TMD - Trigger message delay: delay of receiving the TM message at the source node.
6) IFD - Inter-frame delay: maximum delay between two consecutive frames. In this work we assume that IFD is included in the transmission time of messages.
7) TRD - Turn around delay: maximum time interval needed by a node to start its transmissions after receiving the TM message, it includes the processing time of the node to receive the TM message and to start sending the scheduled messages.

Some of these delays can be evaluated based on the specification of the network, such as SLD, SFD and IFD, while the others are dependent on the FTT-SE scheduler and transmission load and should be computed on-line. Before presenting the methods for computing these delays we will explain the FTT-SE scheduling algorithm in a multi-switch network scenario.

VI. FTT-SE SCHEDULING ALGORITHM

At every EC, the scheduler scans the Synchronous Requirements Database that holds the descriptions of the synchronous streams, looking for ready messages. All such messages are inserted in the global ready queue according to a scheduling policy (FPS, EDF, ..) see Fig. 4. Then, it picks one message at a time starting from the head of the global queue (highest priority) and it checks whether that message still fits within the scheduling window specified for that EC, which means that it can be transmitted within that EC. This operation is repeated for all messages in the ready queue in decreasing priority order until no more messages fit in the EC. Those that fit are encoded in the TM for transmission (EC-schedule) and the remaining ones stay in the ready queue for the next EC. The same procedure is applied for the Asynchronous streams.

The main goal of the scheduler is to schedule as much traffic as possible during each EC without causing overrun in the scheduling window. Therefore, a specific fitness test is used to check whether the transmission of a message, in the worst-case, fits in a given EC. The fitness function keeps track of the transmissions in each link and in each EC using bins with limited capacity that represent the scheduling windows, one for each direction, i.e., 2 bins for each switch port, one for the uplink and another for the downlink (Figure. 4). Returning to the example shown in Figure. 2, suppose that the scheduler tries to process a request to send a message from Node A to Node E, it adds the message in all bins associated to the links that the message will cross i.e., UA,DSW2,USW1,DE where UA is the uplink connected to Node A and DE is the downlink connected to Node E. Note that when the scheduler adds a message to a bin associated with a downlink in a switch, it takes into account the additional delay imposed by the switch on the message. If the message under analysis fits in all the bins along its path considering all the other messages that also fitted before in their respective bins, then that message is added to the EC-schedule of the next EC to be later encoded in the TM.

Another problem that must be considered during the scheduling is the difficulty of evaluating the message order in the queues of the switches (downlinks). We assume the worst case scenario in which the message under consideration will be forwarded after all other scheduled messages, independent of the priority and the transmission time of the messages. Based on this assumption, whenever analyzing the fitness of a new message m_i in an EC, the scheduler assumes that the message is forwarded as the last message in each downlink that it traverses until reaching its destination. Also the scheduler considers the interfering effect of m_i over all other scheduled messages ($\{m_z\}$) that share the same downlinks with m_i on the route. A similar test is repeated on all messages in $\{m_z\}$ to make sure that these messages do not overrun their scheduling windows. The online scheduling algorithm is summarized in Algorithm VI.1.

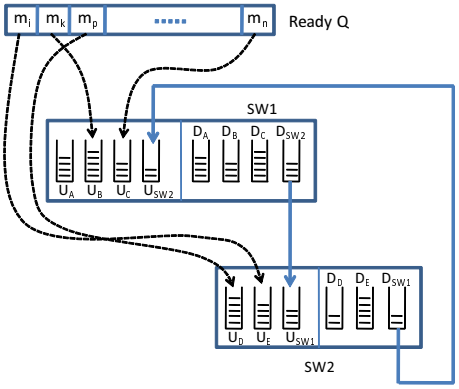

Fig. 4. Master scheduling.

Algorithm VI.1: ONLINE SCHEDULING ALGORITHM()

1 RQ //global ready queue
2 SRQ //sorted ready queue
3 $\{m_{TM}\}$ //scheduled messages
4 interf(m_i, k) //set of messages that share links with m_i in SW_k and are $\in \{m_{TM}\}$
5 $SRQ =$ SortRQ(RQ) //sort messages according to the scheduling policy
6 for all $m_i \in SRQ$ //from the highest to the lowest priority messages
7 $\{testMsg\} = m_i$
8 for all $SW_k \in R_i$
9 $\{testMsg\} = \{testMsg\} \bigcup interf(m_i, k)$
10 end for
11 add$(m_i, \{m_{TM}\})$ //add m_i in $\{m_{TM}\}$
12 for all $m_j \in \{testMsg\}$
13 isScheduled=checkSchedule(m_j) //check the schedulability of m_j based on Eq. 5 and 6
14 if isSchedule==false
15 remove$(m_i, \{m_{TM}\})$
16 end if
17 end for
18 end for
19 return $\{m_{TM}\}$

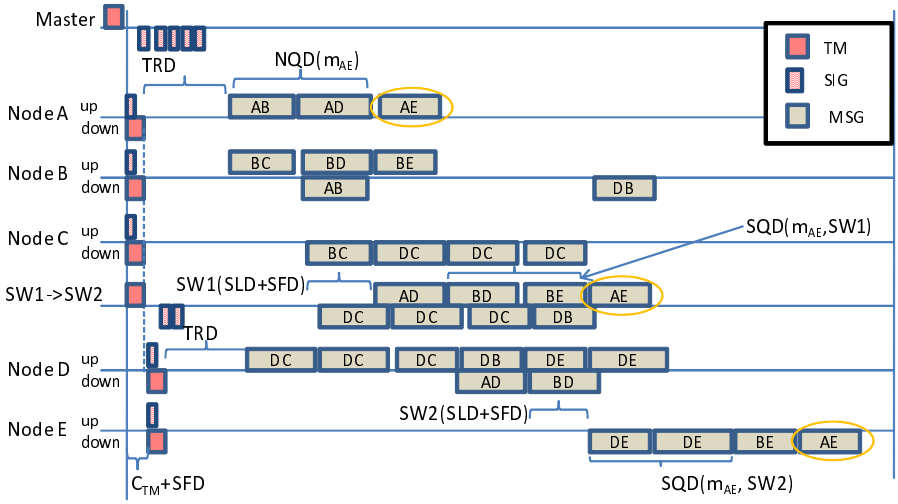

Fig. 3. A scenario for packets transmission of the network shown in Figure 2

VII. Online Scheduling Test

To determine the fitness of the ready messages, the scheduler must compute the following parameters for each message and for each EC:

- TRD All packets (including the TM and signaling messages) that cross switches suffer from a delay called switching delay (SD) due to both the SLD and SFD delays. These delays may cause interference between signaling messages and data messages and may affect the FTT-SE protocol. To avoid such interference, TRD in each node is selected to be high enough to guarantee that all signaling messages will be received by the master node before any node starts to send data packets. Since the transmission of the signaling message is synchronized with the reception of the TM, if a node knows its relative position to the switch where the Master is located, then it can compute the time instant when the Master sends the TM and it can synchronize the transmission of the signaling message with that time. However, we do not want to provide such information to nodes and we assume that all nodes, except the Master, have no knowledge about the network topology and connection. Let us assume C_{SIG} and C_{TM} are the transmission times of the signaling message and the TM message respectively. Then, to avoid any interference between the signaling messages and the data messages TRD stands as follows:

$$TRD > (N_{dep}-1)\times(SD_{TM}) + N_{total}\times C_{SIG} + N_{dep}\times(SD_{SIG}), \quad (1)$$

where $(N_{dep}-1)\times(SD_{TM})$ is the longest time delay of receiving the TM message by a node relative to the time that nodes connected to the master switch receive the TM message. Note that, when transmitting the TM message, the store and forward delay is equal to the transmission time of the TM message and the switch delay becomes $SD_{TM} = C_{TM}+SLD$. Also $N_{total}\times C_{SIG}$ is the longest time required by a signaling message to be received at the master node assuming that all signaling messages arrive simultaneously to the master node. This assumption might be pessimistic because the simultaneous arrival of all signaling messages to the master may never happen, for example the signaling messages from the nodes connected to the master switch, will always arrive earlier than the signaling messages that are sent by all other nodes. Finally, $N_{dep}\times(SD_{SIG})$ is the longest time delay for the signaling messages to reach the master node crossing N_{dep} switches. Similar to the trigger message, the store and forward delay is equal to the transmission time of the signaling message and the switch delay is computed as $SD_{SIG} = C_{SIG} + SLD$.

The computed value of TRD evaluated using Eq. 1 will be used by all nodes, which guarantees that all signaling messages will be received by the Master before any node starts to send its data traffic independent on the location of nodes.

- SFD since we assume the store and forward switch type, each message that crosses a switch will be delayed and the delay depends on the transmission time of the message itself and all other messages that cross the same downlink. Let us assume that a message m_i is crossing switch SW_k through its downlink $dl_{j,k}$, the upper bound of the store and forward latency SFD(mi,k) that m_i may suffer from is computed as follows

$$SFD(m_i,k) = \max_{\forall m_o\in\{dl_{j,k}\}\&(m_o\in\{m_{TM}\}|m_o=m_i)}(C_o), \quad (2)$$

where $\{dl_{j,k}\}$ is the set of all messages that cross the downlink $dl_{j,k}$ and $\{m_{TM}\}$ is the set of messages that are already selected to be transmitted (in the EC-schedule).

- NQD for a message m_i, the $NQD(m_i)$ delay is computed by summing up the transmission time of all messages with priority higher than that of m_i and these messages have the same source node as m_i and are selected to be transmitted at the current elementary cycle, i.e.,

$$NQD(m_i) = \sum_{m_j \in interf(m_i)} (C_j), \quad (3)$$

where $interf(m_i) = \forall m_o \in \{ul_{j,k}\} \& m_o \in \{m_{TM}\}$ and $ul_{j,k}$ is the uplink that connects the source node that sends m_i. Note that the scheduler in the Master, as explained earlier, starts first by processing the highest priority messages and continues to the lower priority ones, then all messages currently in $\{m_{TM}\}$ have a priority higher than that of m_i.

- SQD for a message m_i crossing a switch SW_k, this delay is caused by the packets that share the same downlink with m_i at SW_k. In this case and since we cannot estimate correctly the order of the messages transmission inside the queues of switches, we assume that all messages that share the same downlink $dl_{j,k}$ with m_i will be transmitted before m_i. Let us redefine the function $interf(m_i, k)$ such that it returns the set of all messages that may interfere with m_i at SW_k, excluding the messages that were considered in the previous switches, i.e., $interf(m_i, k) = \forall m_j \in \{dl_{i,k}\} \& m_j \in \{m_{TM}\} \& m_j \notin interf(m_i, l) | \forall SW_l \in SW_{(s-1)}, , SW_{(k-1)}$, where $interf(m_i, s-1) = interf(m_i)$ and $SW_s, , SW_{(k-1)}$ is the set of switches that m_i should pass to reach SW_k from the switch SW_s that source node is connected to, then SQD is evaluated as follows,

$$SQD(m_i, k) = \sum_{m_j \in interf(m_i, k)} (C_j). \quad (4)$$

A. Scheduling test

One important aspect for the scheduling test is to determine the exact boundaries of the synchronous window in each EC, in order to decide whether a given message fits in a certain EC or not. Let us assume that LSW is the length of the synchronous window. The starting time of this window is after TRD + the time of receiving the TM message, i.e., when nodes are ready to transmit after receiving the TM message. While the finishing time of the synchronous window FSW is equal to the starting time + LSW, i.e.,

$$FSW^k = N^k_{dep} \times (C_{TM} + SLD) + C_{TM} + TRD + LSW. \quad (5)$$

Note that FSW^k is the finishing time relative to the beginning of the EC and N^k_{dep} is the number of switches between SW_k and the master switch and $N^k_{dep} \times (C_{TM} + SLD) + C_{TM}$ is the time needed to receive the TM message by all nodes that are connected to SW_k.

Then, to make sure that a synchronous message will not overrun the scheduling window when it crosses a downlink in SW_k, the message finishing time $f(mi, k)$ should be no later than FSW^k, i.e., $f(mi, k) \leq FSW^k$ for all SW_k that m_i crosses from the source to the destination node, i.e,$SW_k \in R_i$.

To compute the finishing time of a synchronous message m_i at SW_k, the maximum delay that the message may suffer should be computed starting from the source node and going through all switches up to SW_k.

$$\begin{aligned} f(m_i, k) \quad = \quad & \sum_{\forall SW_q \in R_i} \big(SQD(m_i, k) + SFD(m_i, k) + SLD\big) \\ & + C_i + NQD(m_i) + TRD + C_{TM} \\ & + N^s_{dep} \times (C_{TM} + SLD). \end{aligned} \quad (6)$$

The summation in Eq. 6 represents the switching delay that m_i will suffer from when it crosses all switches in its route and $TRD + C_{TM} + N^s_{dep} \times (C_{TM} + SLD)$ is the maximum delay at the source node including the delay of receiving the TM message by the source node.

Finally, the scheduling of the asynchronous traffic follows a similar procedure. Once the signaling messages arrive at the master informing of the pending asynchronous requests, the messages are handled by some appropriate mechanism, e.g., servers, and then inserted in a specific global ready queue. After this instant, they are handled using the same fitness function as described before for the synchronous traffic in order to decide whether they can be inserted in the EC-schedule of the current EC. The only difference is that they are handled after the synchronous traffic and they can be scheduled until the end of the EC using the time left free by the synchronous traffic. Therefore, the asynchronous window at each switch SW_k finishes at the end of the EC and starts at the finishing time of the synchronous window. Then, the fitness function considers the previously synchronous messages already in the EC-schedule and allows adding the asynchronous messages whose finishing time $fa(m_i, k)$ is no later than the end of the EC, i.e., $fa(m_i, k) \leq LEC$.

VIII. Response-time calculation

In this section we present an analysis method to evaluate the response time of messages scheduled using the algorithm presented in the previous section with fixed priority scheduling.

Based on the proposed scheduling algorithm, the messages that share links with the message under consideration m_i have a grate effect on the response time of the message m_i. That means, the interference from messages that share the same links should be taken into account in the analysis of m_i which makes the analysis more complex. The following example shows the complexity of this problem.

Let us consider the example shown in Figure 5, where 6 nodes are connected through 3 switches and Node-A and Node-F send messages to Node-C and Node-B and Node-D send messages to Node-E. Let us assume that the message send from Node-A to Node-C m_{AC} has the lowest priority and m_{FC} requires one EC to be transmitted and $m_{BE} + m_{DE}$ require also one EC to be transmitted. Based on the proposed algorithm and depending on the activation times of messages, we can distinguish 4 transmission scenarios for message m_{AC}:

- m_{AC} and m_{FC} are activated simultaneously, since m_{FC} has higher priority than m_{AC}, it will be transmitted first EC and then m_{AC} will be transmitted at the second EC.
- m_{AC}, m_{BE} and m_{DE} are activated simultaneously, since m_{AC} has the lowest priority than the other two messages,

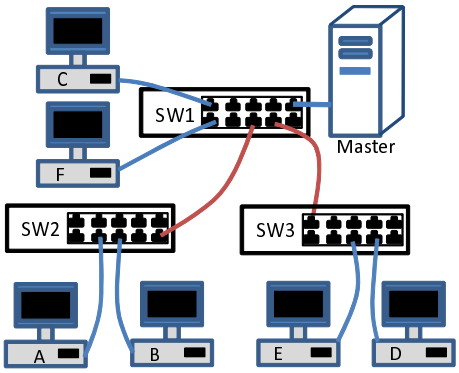

Fig. 5. Multi-hop Switched Ethernet example.

it will be transmitted at the second EC. Even though m_{AC} has different destination than the other two messages, it should be transmitted later than them because transmitting m_{AC} at the first EC may delay m_{BE} and making it overrun its scheduling window. If m_{DE} was not activated then it would be possible to schedule m_{AC} at the first EC as the size of m_{AC} is assumed to be small.

- All messages are activated simultaneously, in this case m_{FC}, m_{BE} and m_{DE} are scheduled to be transmitted at the first EC and m_{AC} will be transmitted at the second EC.
- m_{AC} and m_{FC} are activated simultaneously and m_{BE} and m_{DE} are activated one EC later, at the first EC m_{FC} will be scheduled, then at the second EC m_{BE} and m_{DE} will be scheduled and later m_{AC} will be scheduled at the third EC.

From this example, we can conclude that the simultaneous activation of all messages may not yield the worst case response time (critical instant). In addition, messages that do not share links with other messages can still affect their response time (m_{DE} affect the response time of m_{AC} even when they do not share any link) we call this effect as *indirect effect*.

To solve the problem of finding the critical instant, we can assume that all messages that share links with the message under consideration will be forwarded to the same destination, i.e., all such messages can be added in the response time analysis. On other hand, the indirect effect is resulting from the messages that do not share links with the message under consideration but delay the messages that share links (m_{DE} delays m_{BE} that shares a link with m_{AC}). Then adding all such messages in the calculation of the response time will solve the problem of the indirect effect. Note that to be general, indirect effect should include the all messages that cause NQD, SQD, SFD on the messages that share links with the message under consideration in addition to the SLD.

Now to find the response time analysis for a message m_i, all interfering messages should be defined, and the interfering messages can be classified into the following categories:

- Source node interference message set: which is the set of messages that cause NQD on m_i. This set is similar to $interf(m_i)$ defined in the previous section with one difference which is using all higher priority message instead of the set $\{m_{TM}\}$, i.e., the source node interference messages set is defines as $S\_interf(m_i) = \forall m_j \in \{ul_{i,k}\} \& m_j \in hp(m_i)$, where $hp(m_i)$ is the set of messages with priority higher than the priority of m_i.

- Direct interference message set: which are the messages that cause SQD for m_i. This set is also similar to $interf(m_i, k)$ with one difference which is using all higher priority message instead of the set $\{m_{TM}\}$, i.e., the direct interference messages set is evaluated as follows, $D\_interf(m_i) = \bigcup_{\forall SW_k \in R_i} D\_interf(m_i, k)$ which is the combination of interference messages in all switches that m_i crosses from the source to destination nodes and $D\_interf(m_i, k) = \forall m_j \in \{dl_{i,k}\} \& m_j \in hp(m_i) \& m_j \notin D\_interf(m_i, l) | \forall SW_l \in SW_{(s-1)}, , SW_{(k-1)}$

- Indirect interference message set: which is the set of messages that delay the messages that cause the direct interference on m_i. For each element in $m_j \in D\_interf(m_i)$, the messages that cause delay on m_j are the same that cause direct and source interference on m_j which are $D\_interf(m_j)$ and $Sinterf(m_j)$. Then the indirect interference message set for m_i is $I\_interf(m_i) = \bigcup_{\forall m_j \in D\_interf(m_i)} \left(D\_interf(m_j) \bigcup S\_interf(m_j)\right)$ and excluding any redundant message that has been considered in $S\_interf(m_i)$ and $D\_interf(m_i)$.

Note that the network bandwidth is not available for message transmission all the time, message are allowed to be transmitted within a scheduling window every EC, then this factor should be taken into account when performing the response time analysis. This problem has been considered previously in [7] and the proposed solution was to inflating the transmission times of all message by the percentage of the bandwidth resource availability, i.e., the inflation factor is $\alpha = (LSW - I)/EC$, where I is the idle time that is used to prevent scheduling window overrun and it is equal to the maximum packet size of messages that have priorities equal or higher than m_i. Then all transmission times will be divided by α, i.e., $C'_j = C_j/\alpha$ and the response time of m_i can be evaluated as follow:

$$
\begin{aligned}
x^{\ell} \;=\; & C'_i + \sum_{SW_k \in R_i} \left(\frac{SFD(m_i,k) + SLD}{\alpha} \right) + \\
& \sum_{m_j \in S\_interf(m_i)} \left\lceil \frac{x^{\ell-1}}{P_j} \right\rceil (C'_j) + \\
& \sum_{m_t \in D\_interf(m_i)} \left\lceil \frac{x^{\ell-1}}{P_t} \right\rceil \left(C'_t + \right. \\
& \left. \sum_{SW_k \in R_t} \left(\frac{SFD(m_t,k) + SLD}{\alpha} \right) \right) + \\
& \sum_{m_q \in I\_interf(m_i)} \left\lceil \frac{x^{\ell-1}}{P_q} \right\rceil (C'_q), \qquad (7)
\end{aligned}
$$

Where $SFD(m_i, k)$ is similar to the one defined in the previous section with one difference which is using $hp(m_i)$ instead of $\{m_{TM}\}$ and $\ell > 0$ and $x^0 = C'_i$ and the response time of m_i is equal to $RT(m_i) = x^\ell$ when $x^\ell = x^{\ell-1}$.

IX. Evaluation

We have carried out a few preliminary experiments to validate the scheduler and analysis, and assess the level of pessimism embodied in our analysis compared with the actual implementation of the proposed on-line scheduling algorithm. The results were similar with respect to the level of pessismism of the analysis. As an example, in one experiment we considered the network shown in Figure 5 having 10 synchronous messages scheduled with fixed priorities, with parameters shown in table I and deadline equal to period. The other network parameters are $TRD = 200\mu S$, $C_{SIG} = 2\mu S$, $C_{TM} = 5\mu S$, $SLD = 17\mu S$, and $EC = 2mS$. The response time of each message is measured starting from the time when the message is ready to be sent, up to the EC in which the Master includes the message in the TM, measured in number of EC (i.e., the maximum number of ECs needed by the Master to schedule the message). Selecting $LSW = 745\mu S$ the response times of each message in the network measured from the implementation and calculated using Eq. 7 are shown under "RT impl." and "RT analy." respectively in table I. For the analysis calculation, $LSW = 745\mu S$ is the minimum value that makes all messages meet their deadlines, while for the implementation experiment, the minimum value is $LSE = 571\mu S$. The results already show a small level of pessimism of the analysis proposed in this paper. We expect this level to increase as the message set dimension increases and thus we are working on adapting other more efficient analysis to this scenario.

MSG	Priority	T_i	C_i (μS)	RT impl.	RT analy.
m_{ED}	1	1 EC	18	1 EC	1 EC
m_{DE}	2	1 EC	90	1 EC	1 EC
m_{CD}	3	1 EC	49	1 EC	1 EC
m_{CA}	4	1 EC	49	1 EC	1 EC
m_{BC}	5	1 EC	18	1 EC	1 EC
m_{AB}	6	1 EC	18	1 EC	1 EC
m_{EB}	7	3 EC	90	1 EC	2 EC
m_{DA}	8	3 EC	90	1 EC	3 EC
m_{EC}	9	4 EC	49	2 EC	3 EC
m_{DC}	10	4 EC	90	2 EC	3 EC

TABLE I
Messages parameters.

X. Conclusion and future work

Distributed embedded systems have evolved towards complex network infrastructures, frequently with multi-hop configuration, to support growing requirements for more information exchange and more heterogeneity of such information, together with dynamic reconfigurations and adaptability. Focusing on the specific case of Ethernet, this paper proposed using the FTT-SE protocol to support heterogeneous real-time transactions in a dynamic way in a medium-scale multi-hop network infrastructure. For this purpose, the on-line traffic scheduler had to be adapted and a new analysis based on worst-case response-time upper bounds was proposed. This work is a first step in the referred direction towards using FTT-SE in multi-switch Ethernet architectures. On-going work aims at further verifying the proposed scheduler and analysis as well as characterizing their overhead and efficiency using a real application. These latter aspects are essential to assess the price to pay for enforcing a synchronous view on a multi-hop system as a means to provide prompt dynamic reconfigurability.

References

[1] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach. *IEEE Trans. Industrial Informatics*, 6(4):521–533, 2010.

[2] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an afdx network. In *Proc. of the 18th EUROMICRO Conference on Real-Time Systems (ECRTS'06)*, pages 10 pp. –202, 0-0 2006.

[3] Xing Fan, Magnus Jonsson, and Jan Jonsson. Guaranteed real-time communication in packet-switched networks with fcfs queuing. *Computer Networks*, 53:400–417, February 2009.

[4] Kyung Chang Lee, Suk Lee, and Man Hyung Lee. Worst case communication delay of real-time industrial switched ethernet with multiple levels. *IEEE Transactions on Industrial Electronics*, 53(5):1669–1676, oct. 2006.

[5] Xiaoting Li, J.-L. Scharbarg, and C. Fraboul. Improving end-to-end delay upper bounds on an afdx network by integrating offsets in worst-case analysis. In *Proc. of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'10)*, pages 1 –8, sept. 2010.

[6] R. Marau, P. Pedreiras, and L. Almeida. Signaling asynchronous traffic over a Master-Slave Switched Ethernet protocol. In *Proc. on the 6th Int. Workshop on Real Time Networks (RTN'07)*, Pisa, Italy, 2 July 2007.

[7] Ricardo Marau, Luís Almeida, Karthik Lakshmanan, Raj Rajkumar, and Paulo Pedreiras. Utilization-based Schedulability Analysis for Switched Ethernet aiming Dynamic QoS Management. In *Proc of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'10)*, 2010.

[8] Ricardo Marau, Luís Almeida, and Paulo Pedreiras. Enhancing real-time communication over COTS Ethernet switches. In *Proc. of the IEEE Int. Workshop on Factory Communication Systems (WFCS'06)*, pages 295–302, June 2006.

[9] Farahnaz Yekeh, Mostafa Pordel, Luis Almeida, and Moris Behnam. Scaling ftt-se to large networks. In *Proceedings of the Work-In-Progress (WIP) session of 6th IEEE International Symposium on Industrial Embedded Systems (SIES11)*, June 2011.

[10] Minghu Zhang, Jian Shi, Ting Zhang, and Yong Hu. Hard real-time communication over multi-hop switched ethernet. In *Proceedings of the 2008 International Conference on Networking, Architecture, and Storage*, pages 121–128, 2008.