A First Performance Analysis of the Admission Control in the HaRTES Ethernet Switch

Inés Álvarez*, Mladen Knezic†, Luis Almeida*, Julián Proenza‡
*Instituto de Telecomunicações, Universidade do Porto, Portugal,
ines.alvarez.91@gmail.com, lda@fe.up.pt
†Faculty of Electrical Engineering, University of Banja Luka, Bosnia and Herzegovina,
mladen.knezic@etfbl.net
‡Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain,
julian.proenza@uib.es

***Abstract*—There is a growing interest in developing embedded systems capable of being deployed in dynamic environments that may change in unpredictable manners. When such systems are Distributed Embedded Systems (DESs) they must exhibit flexibility at all levels of their architecture, including the network. On the other hand, there is a clear trend in industry towards using Ethernet-based protocols at the network level of DESs. Nevertheless, Ethernet lacks appropriate support for real-time (RT) communications, mixing different RT traffic and on-line management of the Quality of Service (QoS). Several implementations of the Flexible Time-Tiggered (FTT) protocol over Ethernet were proposed to cope with these drawbacks. FTT is a master/multi-slave protocol that is able to simultaneously convey real and non-real-time traffic and provides mechanisms for dynamically changing the QoS of the network, including Admission Control (AC). The AC is a fundamental component for on-line network management, since it guarantees that each participant gets the required QoS. This paper presents the implementation in OMNeT++ of a simulation model of the AC in the FTT HaRTES switch as well as a preliminary performance study using that model.**

I. Introduction

In recent years the industry has shown an increasing interest on deploying embedded systems in dynamic environments that may change in unpredictable manners. When these systems are Distributed Embedded Systems (DES) they must exhibit the required level of flexibility at all levels of the architecture, including the network. Ethernet has become an appealing technology in the development of DESs, due to its high bandwidth, low cost, easy scalability, high flexibility and Internet compatibility.

Nevertheless, Ethernet was not designed to provide support for real-time communications, e.g. it does not implement mechanisms to prevent packet bursts, which can lead to unbounded delays and packet losses. Several proposals were made to provide Ethernet with real-time guarantees either in industry, such as Time-Triggered Ethernet [1], as well as in academia, such as Dynamic TDMA-Ethernet [2].

Nevertheless, approaches to real-time Ethernet generally lack support for open reconfigurability and adaptivity. This led to the implementation of the *Flexible Time-Triggered* (FTT) paradigm [3] over switched Ethernet. FTT provides flexibility by supporting the transmission of hard real-time, soft real-time and non-real-time traffic over the same network with configurable bounds on each kind of traffic. Moreover, FTT implements services that allow for the dynamic management of the Quality of Service (QoS) that the network provides to the participants, including an Admission Control (AC).

Admission Control guarantees that enough resources are available in the network to transmit all the authorised information, preventing packet delay or loss due to the occurrence of bursts. Thus, AC is of great importance in order to guarantee that real-time constraints are met even in dynamic and complex environments that may change in unpredictable manners, in which anticipating the behaviour of the network participants can be hard or even impossible.

This paper describes a model of the FTT Admission Control mechanism in the OMNeT++ simulation environment. More specifically, we modelled the AC in the HaRTES implementation of FTT in which the master is embedded in the switch. Simulation is a suitable solution, since it is usually faster to carry out and facilitates the study of specific mechanisms when compared to the implementation of a real prototype. We start from a partial model of HaRTES for the OMNET++ INET framework that is currently being developed at the University of Banja Luka [4]. OMNeT++ is an extensible and modular C++-based discrete event simulation library and framework for the simulation of networks and distributed systems [5]. It provides a series of libraries and frameworks, such as INET, an open-source library containing a large number of wired, wireless and mobile network models [6]. This paper also presents, to the best of the authors' knowledge, the first performance study of the Admission Control of HaRTES switches using the referred simulation model.

II. Overview of HaRTES

FTT is a communication paradigm that supports time and event-triggered traffic in a flexible manner, by providing mechanisms for dynamically changing the communication requirements. FTT follows a master/multi-slave architecture where the master coordinates and manages the communication among the application nodes (slaves). In the HaRTES implementation of FTT, the master is embedded inside the switch.

In FTT the communication is divided in slots of fixed duration called Elementary Cycles (EC). The duration of the EC is

a parameter that depends on the dynamics of the application and must be set at the beginning of the system operation. In each EC the master schedules the synchronous messages to be sent by the slaves and triggers their transmission by means of the Trigger Message (TM). The transmission of the TM is carried out in a guarded window, isolated from the rest of the traffic. The rest of the EC is divided into two phases, the synchronous window for time-triggered data messages and the asynchronous window for event-triggered messages.

In HaRTES the confinement of the traffic in the corresponding window is forced by the switch. Therefore, each slave does not need to wait until the synchronous window expires to transmit its pending asynchronous traffic, but it can transmit it as soon as the the node sends its scheduled synchronous messages. Nevertheless, the switch will only process or forward the asynchronous messages during the asynchronous window to guarantee the timeliness of the synchronous traffic.

In FTT the communication is carried out through virtual communication channels called *message streams*. Each stream is defined by a set of attributes that vary depending on whether it is synchronous (Eq. 1) or asynchronous (Eq. 2).

$$SM_i \equiv \{C_i, D_i, T_i, O_i, Pr_i, S_i, [R_i^1..R_i^{ki}]\}, \quad (1)$$

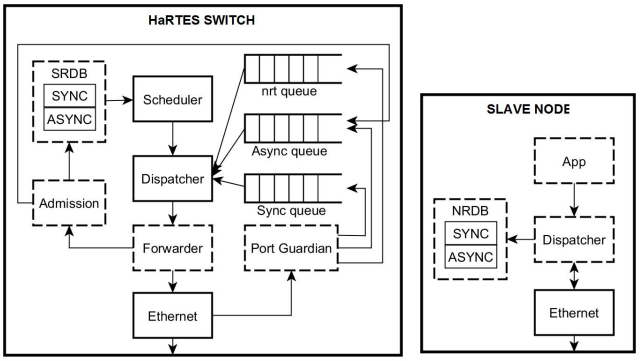
$$AM_i \equiv \{C_i, mit_i, Pr_i, S_i, [R_i^1..R_i^{ki}]\}, \quad (2)$$

The expressions show the stream attributes where C_i is the transmission time of the messages sent through the stream, D_i is the relative deadline, T_i is the period and mit_i represents the minimum inter-transmission time (for synchronous and asynchronous traffic respectively), O_i is the offset (for synchronous streams, only), Pr_i is the priority and S_i contains the publisher ID and $R_i^1..R_i^{ki}$ contain the subscribers IDs.

These attributes are used by the master to produce the EC-schedule every EC. All the information related to the streams is stored in the System Requirements Database (SRDB) in the master and in the Node Requirements Database (NRDB) in each slave. The data in these structures is also used by the AC to filter out changes on the requirements that would jeopardize the timeliness of the traffic.

Whenever an FTT slave wishes to carry out a modification in the communication requirements, be it adding, removing or updating a stream, it issues a request to the master. When receiving such a request, the master executes the schedulability analysis in order to determine if there are available resources to carry out the requested modification. In case a stream is successfully created or modified the master sends an announcement to all the slaves in the network for them to update their NRDBs. On the other hand, when the master receives a deletion request it simply removes the stream from the SRDB and notifies the slaves to update their NRDBs.

In this work we implemented two different policies for the schedulability analysis of the AC namely Rate Monotonic (RM) and Earliest Deadline First (EDF) as described in [7]. These schedulability analyses are utilisation-based, that is, the sum of the utilisation ratio of the streams per link must be under a given limit. This utilisation limit is scaled to account for the Synchronous Window and the Elementary Cycle lengths. Moreover, different analyses must be carried out in the uplink and the downlinks. Regarding the uplink, the utilisation only depends on the load generated by the node in each EC. On the other hand, the transmission of the message in the downlink also depends on the instant the frame is transmitted in the uplink. Thus, the utilisation in the downlink needs to account for the delay the message may suffer in the uplink due to the transmission of messages with different destinations. Further discussion on the analysis is out of the scope of this paper. The reader can refer to [7] for further details.



(a) Model for the HaRTES switch (b) Model of a slave node

Fig. 1. Simulation models for the HaRTES switch and the slave nodes [4].

III. Simulation Model

Our Admission Control model is based on a preliminary model of HaRTES built on top of the OMNeT++ INET framework [4]. Since this preliminary version did not implement the AC all streams were statically defined during the initialization phase and could not be modified or removed in run-time. Fig. 1a shows the model of the HaRTES switch, while Fig. 1b shows the model of a slave node. Even though the initial model already included the Admission module, this module was there as a hook for future implementation, only. Moreover, the transmission of asynchronous messages was not fully supported by the model. Thus, in this paper we report this implementation work, providing the internals of such module together with all the associated signalling messages needed for an effective Admission Control. The modules modified in this work are represented in the figures using dashed lines.

On the one hand, slave nodes are responsible for requesting the creation, modification and deletion of streams. More specifically, any request must be triggered by the application. Thus, the App module was modified to support the creation and transmission of requests. Regarding the Dispatcher module, it was adapted to handle the transmission of slave requests, as well as to process the master command messages received from the switch as a result of the AC process. Moreover, since now slaves can request the modification and removal of streams, the NRDB module was modified to support the deletion of streams. Note that the modification of a stream can be achieved by deleting the existing stream and creating

TABLE I
EXPERIMENT PARAMETERS

	Network load (ms) [a]			Concurrency
	Low	Medium	Heavy	(# nodes)
Test 1	1	50	100	1
Test 2	1	50	100	10

[a] Time required to carry out the schedulability analysis depending on the network load.

it again with the new parameters, so no specific functions were added to this module to modify existing streams. It is important to note that both NRDB in the nodes and SRDB in the switch are instances of the same module, thus taking advantage of OMNeT++ modularity.

On the other hand, in HaRTES, the master that is embedded inside the switch is responsible for processing the slave requests, carrying out the AC and informing the slaves about the result. Thus, since both slave requests and master commands are asynchronous messages the HaRTES switch model was adapted in order to support the reception and transmission of these messages. Specifically, the Port Guardian module was adapted to forward slave requests to the asynchronous queue, while the Forwarder module was modified to send slave requests to the Admission module and to broadcast master commands to the slaves. Finally, the Admission module was adapted to process slave requests, to carry out the schedulability analysis and to send master command messages.

Moreover, the definition of FTT messages was extended to include slave request and master command messages. The definition of the messages does not affect any module in the switch or the node models, but it is orthogonal to both of them.

IV. STUDY AND RESULTS

As mentioned before, the communication in FTT is divided in ECs and the duration of the EC is a parameter that depends on the dynamic of the system. Moreover, the duration of the EC limits the number of messages that can be transmitted and processed in every EC. On the other hand, the load of the network may impact in the duration of the schedulability analysis. This is because the analysis is utilisation-based, which means that the sum of the utilisation rate of all the streams that transmit through a link must be under a certain limit. Thus, in order to perform the analysis the master must read all the entries of the SRDB several times. Therefore, when the load of the network increases so do the size of the SRDB and the time required to compute the schedulability analysis.

Here we want to study the impact that the duration of the EC has on the performance of the Admission Control. To that end we measure the time taken to complete the AC from the instant a slave sends a request until that slave finishes processing the master command and updates its NRDB. Moreover, since in the normal operation of the system the load, and therefore the duration of the schedulability analysis, may change with time, we carried out several tests simulating different network loads. Specifically, we considered three loads, labeled as

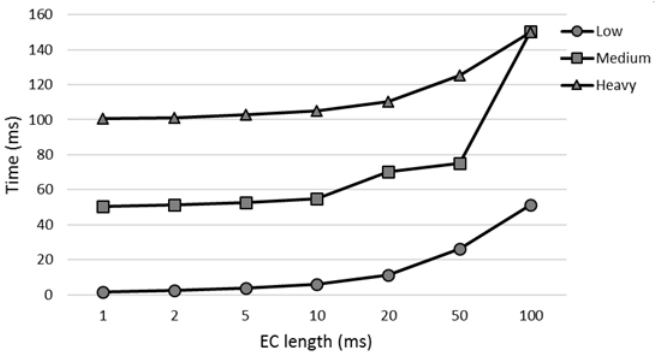


Fig. 2. Mean duration of the Admission Control for different EC lengths and network loads, without concurrent requests.

low, medium, and heavy, which correspond to schedulability analysis times of 1, 50 and 100 ms.

Note that OMNeT++ is an event-based simulator and therefore it only simulates the passing of time when events occur. Moreover, events in OMNeT++ are represented by the transmission of messages. Thus, we modified the Admission module to support the simulation of different times for the schedulability analysis. This was done using a timer that is set with the selected duration whenever there is a pending slave request and that triggers the instantaneous analysis of the request when the timer expires.

On the other hand, we also want to study the impact that concurrent slave requests may have on the duration of the AC. To this aim we considered two scenarios, one where the processing of requests does not overlap and a second one in which all the requests are issued at the same time. Table I presents an overview of the parameters used.

The network used in all the experiments is composed of one switch and ten slave nodes. Moreover, all the slaves in the network are set to request the creation of a new stream. Regarding the division of the EC, we considered the time required for the transmission of TMs to be negligible. We also considered the length of the synchronous and the asynchronous windows to be equal. Therefore, each window represents 50% of the total EC duration.

Fig. 2 shows the mean duration of the AC for different EC lengths and considering different network loads. As we can see, the mean duration of the AC grows as the length of the EC increases, for all considered network loads. In scenarios with medium and heavy loads the processing of slave requests can be delayed in the switch by the forwarding or processing of other frames. Therefore, having longer EC lengths increases the number of frames that can interfere with the requests.

In order to understand why the EC length has such an impact even in networks with low load we can look at Table II, that presents the numerical results of the experiment. We can see that the mean duration of the AC is increased by approximately the half of the EC length in most cases. In scenarios with a low load slaves have few or no synchronous frames to transmit and therefore they can transmit the slave requests shortly after the

TABLE II
MEAN DURATION OF THE AC FOR DIFFERENT EC LENGTHS NETWORK LOADS AND NO CONCURRENT REQUESTS

EC(ms)	1	2	5	10	20	50	100
Low	1.50	2.61	3.62	6.12	11.12	26.12	51.12
Medium	50.62	51.12	52.60	55.00	70.2	75.00	150.22
Heavy	100.50	101.12	102.62	105.12	110.12	125.12	150.12

EC has started, during the synchronous window, as explained in Section II. Nevertheless, the processing of requests by the master is delayed until the asynchronous window starts, to protect the timeliness of the synchronous traffic. Therefore, the size of the windows, and thus the EC, has a severe impact on the duration of the AC.

Fig. 3 shows the mean duration of the Admission Control for different EC lengths with concurrency of slave requests. If we compare these results to the ones obtained in the previous experiment, we can see that, as expected, the concurrency of requests impacts the time needed to complete the AC. Moreover, we can see that the impact is worse when the load of the network increases. This is because when the load is low and the size of the EC is large, several requests can be completed within the same EC, while when the load increases it is not possible to process more than one request per EC.

Finally, observing the results we can conclude that it might not be possible for slaves to predict the time that will take for the AC to be completed, since it will vary depending on the concurrency level that is unknown by the nodes. This is particularly critical since the AC process in HaRTES does not consider the transmission of any message to notify the slaves when a request to create or modify a stream is rejected by the master. Therefore, it may be impossible for slaves to know when a request has been rejected, lost or simply delayed by the interference of other frames. Furthermore, if slaves are set to retry the creation of streams when no response is received from the master, this could force the master to process several times a request that has been rejected, thus delaying requests of other streams. Hence, it is crucial to add the notification of rejected requests to the AC process.

V. CONCLUSIONS AND FUTURE WORK

In this paper we present the implementation of a simulation model in OMNeT++ for the Admission Control in HaRTES and a preliminary performance study carried out using that model. Our Admission Control model is based on a partial model of HaRTES built on top of the OMNeT++ INET framework that is currently being developed at the University of Banja Luka, and it includes both, a model for the HaRTES switch with the master embedded and a model for slave nodes. We studied the impact that the EC length has on the duration of the Admission Control. For the experiments we used a network with a single switch and ten slave nodes. We simulated different network loads and we considered scenarios with and without concurrent slave requests. The results obtained show that the length of the EC has an important impact on the duration of the AC, that increases with the size of the EC. We also saw that it is not easy for slaves to determine a bound for the duration of the AC, and concluded that this is specially critical since HaRTES does not yet consider any mechanisms to acknowledge slaves when their requests are rejected. We recommend this mechanism is added.

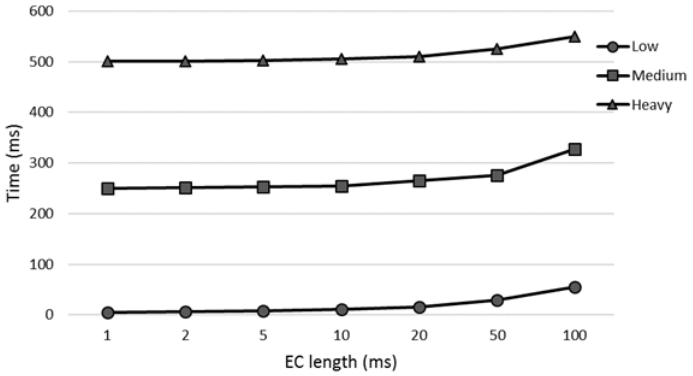


Fig. 3. Mean duration of the Admission Control for different EC lengths and network loads, with concurrent requests.

As part of the future work, we are currently working on extending the existing model to develop the model for FTT-SE, an FTT implementation over Ethernet that uses a COTS switch. This model will allow us to compare both implementations, in terms of performance and reliability. Moreover, we want to extend the comparison to include the Stream Reservation Protocol implemented in the AVB standards.

ACKNOWLEDGEMENTS

This work was supported by project TEC2015-70313-R (Spanish MINECO/FEDER), project "DES4DES: Discrete Event Simulation for Distributed Embedded Systems" (Ministry of Science and Technology of Republika Srpska) and by a grant of the Instituto de Telecomunicações, Portugal.

REFERENCES

[1] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The Time-Triggered Ethernet (TTE) Design," in *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (Seattle, Washington: TU Wien)*, May 2005, p. 2233.

[2] G. Carvajal, L. Araneda, A. Wolf, M. Figueroa, and S. Fischmeister, "Integrating Dynamic-TDMA Communication Channels into COTS Ethernet Networks," *IEEE Transactions on Industrial Informatics*, vol. PP, 2016.

[3] P. Pedreiras and L. Almeida, "The Flexible Time-Triggered (FTT) paradigm: an approach to QoS management in distributed real-time systems," in *Proc. Int. Parallel and Distributed Processing Symposium*. IEEE Computer Society, 2001.

[4] M. Knezic, A. Ballesteros, and J. Proenza, "Towards extending the OMNeT++ INET framework for simulating fault injection in Ethernet-based Flexible Time-Triggered systems," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sept 2014.

[5] A. Varga, "The OMNeT++ discrete event simulation system," in *In ESM01*, 2001.

[6] "The INET Framework—An open-source OMNeT++ model suite for wired, wireless and mobile networks." [Online]. Available: https://inet.omnetpp.org/

[7] R. Marau, L. Almeida, P. Pedreiras, K. Lakshmanan, and R. Rajkumar, "Utilization-based schedulability analysis for switched Ethernet aiming dynamic QoS management," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, Sept 2010, pp. 1–10.