Universitat de les Illes Balears

Study of the Admission Control in the Flexible Time-Triggered and the Audio Video Bridging Communication Protocols

MSc Candidate
Inés Álvarez Vadillo

A MSc thesis submitted to *Departament de Ciències Matemàtiques i Informàtica* of the University of Balearic Islands in accordance with the requirements for the degree of
Màster Universitari Enginyeria Informàtica (MINF)

Author
Inés Álvarez Vadillo

MSc Supervisor
Julián Proenza Arenas

MSc Supervisor
Luís Miguel Pinho de Almeida

MINF Director
Antonia Mas Pichaco

21/09/2016

Study of the Admission Control in the Flexible Time-Triggered and the Audio Video Bridging Communication Protocols

Inés Álvarez Vadillo
Supervisors: Julián Proenza Arenas and Luís Miguel Pinho de Almeida
Treball de fi de Màster Universitari Enginyeria Informàtica (MINF)
Universitat de les Illes Balears
07122 Palma de Mallorca
ines.alvarez.91@gmail.com

Resumen

En los últimos años la industria ha mostrado un interés creciente en usar Ethernet como protocolo para desarrollar sistemas empotrados distribuidos, incluso en campos como la automatización, la aviónica o la automoción. Esto coincide con un interés creciente en el uso de aplicaciones multimedia para el entretenimiento (*streaming* de vídeo) o la asistencia (Sistemas Avanzados de Asistencia a la Conducción). Estas aplicaciones deben coexistir con aplicaciones tradicionales de control distribuido, lo que genera una gran diversidad en el tráfico que atraviesa la red. Concretamente, el tráfico multimedia se caracteriza por tener un gran tamaño, baja intensidad (ocupación media del canal) y requisitos de tiempo real débil. Por el contrario, el tráfico de control normalmente contiene información de pequeño tamaño con una alta intensidad y requisitos de tiempo real estricto. Además, muchas aplicaciones modernas pueden ser lanzadas en cualquier momento por lo que la red debe permitir la conexión y desconexión de participantes en tiempo de ejecución. Sin embargo, Ethernet no provee servicios de temporalidad adecuados para transmitir tráfico de tiempo real, estricto o débil. Además, Ethernet no permite modificar dinámicamente la Calidad de Servicio (QoS) de la red y, por tanto, no soporta la conexión y desconexión de participantes en tiempo de ejecución. Se han propuesto varios protocolos basados en Ethernet para lidiar con estas limitaciones, como *Flexible Time-Triggered* (FTT) y, recientemente, *Audio Video Bridging* (AVB). En este trabajo estudiamos la relevancia de los mecanismos de Control de Admisión (CA) de estos protocolos para la gestión dinámica de su QoS. Además, hacemos una comparación cualitativa del CA de los protocolos mencionados. También presentamos la implementación de los mecanismos de CA sobre un modelo de simulación preliminar de la versión HaRTES de FTT que está siendo desarrollado en la Universidad de Banja Luka. Finalmente, presentamos un análisis cuantitativo del desempeño del protocolo utilizando el modelo mencionado. Los análisis presentados en este trabajo constituyen un primer paso hacia una futura comparación completa de los protocolos mencionados.

Abstract

In the last years industry has shown a growing interest in using Ethernet as the protocol for developing distributed embedded systems, even in the automation, avionics and automotive fields. This coincides with a growing interest in the use of multimedia-based applications for entertainment (e.g. video streaming) as well as for assistance (e.g. Advanced Driver Assistance Systems). These applications must coexist with traditional distributed control applications, what generates a great diversity in the traffic traversing the communication network. More specifically, multimedia traffic is characterised by having a large size, low intensity (average occupation of the channel) and soft real-time requirements. Conversely, control traffic usually conveys small amounts of information with high intensity and hard real-time requirements. Furthermore, many modern applications can be launched at any moment of the system's operation and thus, the network must support the on-line connection and disconnection of participants. Nevertheless Ethernet does not provide adequate timing services to support the transmission of hard or soft real-time traffic. Moreover, Ethernet does not allow to dynamically change the Quality of Service (QoS) of the network and thus it does not support the connection and disconnection of participants at run-time. Several Ethernet-based protocols were proposed to cope with these drawbacks, namely *Flexible Time-Triggered* (FTT) and more recently *Audio Video Bridging* (AVB), among others. In this work we study the relevance of the Admission Control (AC) mechanisms of these protocols for the dynamic management of their QoS. Moreover, we carry out a qualitative comparison of the AC mechanisms of the aforementioned protocols. We also present the implementation of the FTT's AC mechanism over a preliminary simulation model of the HaRTES implementation of FTT that is currently being developed in the University of Banja Luka. Finally, we present a quantitative analysis of the performance of the HaRTES' AC using said model. The analyses presented in this work constitute a first step towards a future complete comparison of the mentioned protocols.

Key words: Distributed Embedded Systems, Communication for Automation, Real-time Systems, Adaptive Systems, Admission Control, Flexible Time-Triggered Paradigm, HaRTES, Audio Video Bridging

1 Introduction

In recent years the industry has shown interest in using Ethernet as the technology for developing distributed embedded systems, including in the automation, avionics and automotive areas. This coincides with a growing interest in deploying novel multimedia-based applications for entertainment or assistance over said systems. On top of that, in these areas multimedia applications must coexist with traditional distributed control applications, what leads to a wide diversity of types of traffic traversing the network in terms of size, intensity (average occupancy of the channel) and timing requirements. More specifically, traffic exchanged by multimedia applications is characterised by having a large size, low intensity and soft real-time requirements [9], while traffic exchanged by control applications usually has small size, high intensity and hard real-time requirements.

Many of these novel multimedia applications, such as Advanced Driver Assistance Systems (ADAS) and infotainment applications, can be launched at any moment of the system's operation and therefore these systems require support for the on-line connection and disconnection of participants.

Nevertheless, Ethernet does not provide the required services to support real-time traffic conveyed by control and multimedia applications. For instance, Ethernet does not prevent traffic bursts from happening, which can lead to unbounded delays in the transmission of packets and packet losses due to buffer overflow. In this context, several protocols were proposed to provide Ethernet with real-time capabilities by both industry, such as Time-Triggered Ethernet [8] and academia, i.e. Dynamic-TDMA Ethernet [5], among others.

However, full support for reconfigurability and adaptivity has generally lacked. For instance, Ethernet does not support the dynamic management of the Quality of Service (QoS) of the network, which means that the network parameters (e.g. number of participants, resources dedicated to each participant, etc.) can not be modified during the normal operation of the system. This prevents nodes from entering or leaving the network at run-time. These drawbacks led to the proposal of several implementations of the *Flexible Time-Triggered* (FTT) paradigm [11] based on switched Ethernet networks. FTT is a master/multi-slave communication paradigm that provides flexibility by supporting several types of traffic over the same communication channel, namely hard real-time, soft real-time and best effort traffic. Moreover, FTT also provides the network with dynamic management of the QoS. Currently two different approaches for FTT over switched Ethernet exist: FTT Switched Ethernet (FTT-SE), with the slaves and master connected through a Commercial Off-The-Shelf (COTS) switch; and Hard Real-Time Ethernet Switching (HaRTES), with the slaves connected to a custom switch that contains the master.

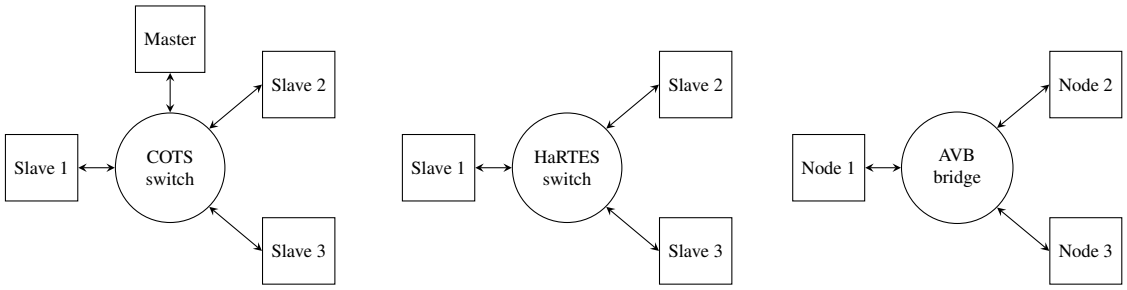
More recently the IEEE published a series of technical standards known as *Audio Video Bridging* (AVB) [2] [3] [4] that aims at providing Ethernet with soft real-time capacities and adaptivity services. To this, AVB supports the transmission of several types of traffic over the same network infrastructure and allows for on-line changing the QoS that the network provides to the participants. Each technical standard specifies different mechanisms to achieve the desired timing guarantees and QoS management. The IEEE Std 802.1AS-2011 implements synchronization services, the IEEE Std 802.1Qav-2010 describe frame forwarding policies for AVB traffic and the IEEE Std 802.1Qat-2010 implements Admission Control mechanisms for the streams in the switches. All these mechanisms together guarantee bounded maximum latency in the transport of audio and video frames in a flexible manner.

As has been said before, both FTT and AVB implement mechanisms to support the dynamic change of the QoS. One of these mechanisms is the Admission Control (AC). AC allows for the on-line reservation of resources in the network, ensuring that enough resources are available along the path that connects a transmitter with a receiver to carry out the communication, preventing packet delays beyond predefined limits and packet losses due to buffer overflow. When working in dynamic environments that may change in unpredictable manners AC is of great importance to guarantee timing constraints while providing an adequate level of flexibility.

In this work we compare the Admission Control mechanism implemented by each of the aforementioned protocols from three different points of view: 1) reliability, paying special attention to transient and permanent faults that may occur in the communication network and those permanent that may affect the nodes; 2) flexibility, studying the mechanisms implemented and the services provided by the protocols to increase the overall adaptivity of the system; and 3) performance, in terms of the overhead generated by the AC mechanisms in the communication and the nodes. These comparisons are done assuming a mono-hop architecture for all protocols, which means that the topology used has a single switch connected to all the nodes of the network through direct links.

Furthermore, we present the implementation of a simulation model of the Admission Control of the HaRTES implementation of FTT, to carry out a quantitative analysis of its performance. This model was implemented over the OMNeT++ INET simulation framework. Simulation represents a suitable option to study the AC in HaRTES, since it is usually faster to carry out and facilitates the study of specific mechanisms when compared to the implementation of a real prototype. As the starting point for this work, we counted with a partial model of HaRTES for the OMNET++ INET framework that is currently being developed at the University of Banja Luka [7].

As indicated above, the platform chosen to carry out the simulation is OMNeT++ that is an extensible and modular C++-based discrete event simulation framework for the simulation of



(a) FTT-SE network architecture (b) HaRTES network architecture (c) AVB network architecture

Figure 1: Mono-hop network architectures for FTT-SE, HaRTES and AVB.

networks and distributed systems [13]. Moreover, OMNET++ 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 [1].

The structure of the remainder of the document is as follows, Section 2 presents an overview of the protocols and the specific Admission Control mechanisms they implement, Section 3 contains the qualitative comparison of the protocols in terms of reliability, flexibility and performance, in Section 4 we describe the simulation model of the AC of HaRTES, the experiments carried out to study the performance and the results obtained, finally Section 5 presents a summary of the most relevant aspects of this work and points out to some of the future work to carry out.

2 Related Work

In this section we will provide an overview of the communication protocols and the Admission Control mechanisms that each one implements.

2.1 Overview of FTT over Ethernet

As already mentioned, FTT is a communication paradigm that supports the transmission of hard and soft real-time traffic over the same communication infrastructure and provides mechanisms for dynamically changing the QoS provided by the network. FTT follows a master/multi-slave scheme, that is, the master node manages and coordinates the communication among the application nodes (slaves).

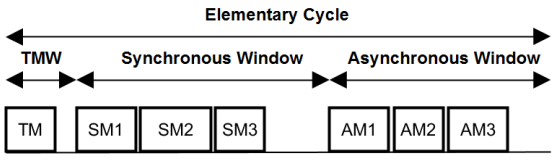


Figure 2: Structure of the Elementary Cycle.

In FTT the master organises the communication in slots of fixed duration called *Elementary Cycles* (EC). Figure 2 shows the structure of an EC. At the beginning of each EC the master transmits the so called *Trigger Message* (TM), a special message used to synchronize and trigger the communication among several slaves. The TM contains the schedule of the messages to be transmitted within that EC and its transmission is done isolated from the rest of the traffic during the TM Window (TMW). The rest of the EC is divided into two different phases, the Synchronous Window for time-triggered data messages and the Asynchronous Window for event-triggered data and control messages.

The communication is carried out through virtual communication channels called *message streams* or *streams* for short. Streams are defined by a series of parameters, depending on whether they convey synchronous or asynchronous traffic. These parameters are depicted by Equation 1 and Equation 2 respectively:

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

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

The expressions show the stream attributes. These attributes determine how messages are transmitted through the stream. Specifically, C_i is the transmission time of the messages sent through the stream, D_i is the relative deadline, which is the time available to complete the transmission of a message since it is requested by the master; T_i is the period with which a synchronous message is transmitted and mit_i represents the minimum inter-transmission time between two asynchronous messages; O_i is the offset (for synchronous streams, only) used to shift the transmission instants of a stream as shown in Figure 3; Pr_i is the priority and S_i contains the publisher ID and $R_i^1..R_i^{ki}$ contain the subscribers IDs. All information related to the streams is stored in the *System Requirements Database* (SRDB) in the master and in the *Node Requirements Database* in the slaves (NRDB).

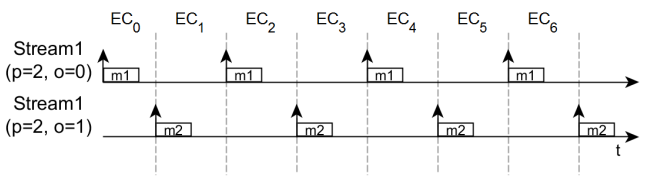


Figure 3: Example of two streams with the same period and different offset.

Whenever an FTT slave wants to carry out a modification in the communication requirements, be it adding, removing or updating a stream (e.g. reducing the period of message transmission for a synchronous stream), it issues a *slave request* to the master. When receiving a slave request, the master executes a schedulability analysis in order to determine if the resources available in the network are sufficient to carry out the requested modification. In case a stream is successfully created or modified the master sends an *master command* message to all the slaves in the network with the stream information. Slaves use the master command message to update their NRDBs. On the other hand, when the master receives a slave request to delete a stream it simply removes the stream from the SRDB and notifies the slaves to update their NRDBs by means of a master command. It is important to note that in FTT any slave can request the modification of the communication requirements of any stream, even if this node does not participate in the communication. The application can decide which nodes can request changes on which streams.

FTT was implemented over Ethernet according to two different approaches, FTT-SE that uses COTS switches and HaRTES that uses custom switches with the master node embedded. Figure 1a depicts the FTT-SE architecture. By using COTS switches FTT-SE preserves some of the Ethernet desirable features, such as low cost, high availability of components and general purpose LANs compatibility. Nevertheless, it requires all nodes to be FTT-aware, since non-FTT-compliant nodes could send their traffic at any moment of the EC, interfering with the timeliness of the protocol.

Conversely to FTT-SE, the HaRTES architecture, shown in Figure 1b, is based on a custom switch that contains the master node embedded. This way, the master has a complete vision of the communication and the traffic traversing the network, which allows implementing traffic shaping, i.e., the master confines the traffic sent by the nodes to the appropriate windows of the EC. Thus, non-FTT-enabled nodes can be attached to the network, since their traffic is only forwarded by the master during the Asynchronous Window when enough bandwidth is available after transmitting the FTT-compliant messages, preventing best-effort traffic from interfering in the transmission of real-time traffic.

2.2 Overview of Audio Video Bridging

AVB is a set of technical standards that provides synchronisation services, frame forwarding policies and Admission Control mechanisms in the scope of Ethernet. Figure 1c depicts the architecture of a one-hop AVB network. The Admission Control is performed by the *Stream Reservation Protocol* (SRP), that allows for the reservation of resources in the switches along the path between two nodes willing to communicate. SRP considers two classes of real-time traffic, A and B, with different QoS, the former providing lower latencies and higher priority.

Similarly to FTT, nodes that want to transmit real-time traffic must do it through virtual channels called *streams*. Nevertheless, the attributes used to describe a stream in AVB differ from those used in FTT. Equation 3 depicts the attributes of AVB streams.

$$Stmi \equiv \{Id_i, MAC_i, VLAN_i, Msize_i, Mint_i, Pr_i, Rk_i\}, \quad (3)$$

The values assigned to the stream attributes determine the way messages are transmitted through the stream. Concretely, Id_i is the identifier of the stream, MAC_i represents the destination MAC address of the stream, $VLAN_i$ is the Virtual LAN identifier where the messages will be transmitted through, $Msize_i$ is the maximum size of the messages to be transmitted through the stream, $Mint_i$ is the maximum number of frames transmitted by the talker during the transmission interval assigned to its class by the the traffic shaper, Pr_i represents the priority class of the stream and Rk_i indicates if the stream is an emergency stream with higher priority than the rest of streams of the same class.

A node can be attached to a stream whether as *talker*, the node that generates the traffic, or as *listener*, nodes that consume the traffic. When a node declares its intention to create a stream this information is stored in the switches, called *bridges* within AVB standards, to manage the Admission Control and the communication.

More specifically, the creation of streams is triggered by the talker by sending a *talker advertise* message. A bridge that receives a talker advertise checks for the availability of resources in all the output ports that support the transmission of messages that belong to the stream class, except for the port the talker advertise arrived to. If the required resources are available in the port, the bridge forwards the talker advertise and registers its transmission in the *talker advertise table* of the port. Otherwise the bridge forwards a *talker failed* messageto the talker and the listener connected to the port. The talker failed message contains the reason for the failure. Thus, a node only receives a talker advertise if there are resources available along the path that connects it to the talker.

When a node different from the talker receives a talker advertise message, it will respond by sending a *listener attribute* message. The listener attribute will be different depending on the intention of the node to listen to the stream. If the node wants to listen to the stream, it will respond with a *listener ready* attribute message. If the node does not want to listen or if it receives a talker failed message (no resources available) it will respond with a *listener asking failed* attribute message.

When a bridge receives a listener ready message it checks again for the availability of resources in the arriving port. If resources are still available the bridge locks them, otherwise it changes the listener attribute to a listener asking failed and sends it to the affected listener.

The slave attributes received by the bridge are used to answer to the talker. Nevertheless, bridges do not forward all the slave attributes, instead they merge all the attributes to create a unique response for the talker. This combination of attributes is done as follows:

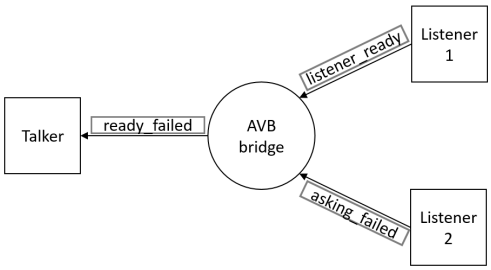


Figure 4: Transmission of a listener ready failed scenario.

- If all the attributes received by the bridge are listener ready, the bridge forwards a listener ready to the talker, indicating that the reservation of resources has been successful for all the downstream listeners.
- If all the attributes are listener asking failed messages, the bridge forwards a listener asking failed to the talker.
- Figure 4 shows an scenario in which the bridge receives different attributes from the listeners. When at least one listener ready and at least one listener asking failed are received, the bridge forwards a *listener ready failed* back to the talker, indicating that there is at least one path with sufficient resources and at least another with insufficient resources to carry out the communication.

When a bridge transmits the listener attribute through a port, it registers its transmission in the *listener ready table* of said port. When a talker receives a listener ready or a listener ready failed it knows that the stream was successfully created for at least one listener and can start the transmission.

It is important to note that talker and listeners must periodically transmit a talker advertise and a listener ready, respectively, to announce their intention to keep communicating.

3 Qualitative comparison

In this section we will carry out the qualitative comparison of the AC mechanisms of the protocols from three different points of view, reliability, flexibility and performance.

Concerning reliability, we will study the effects derived from temporary faults in the communication channel, paying special attention to the loss of AC messages, i.e. slave requests and master commands in FTT and talker advertise and listener ready in AVB. We will also analyse the consequences of permanent faults in the nodes and the communication channel, paying special attention to the presence of *Single Points of Failure* (SPoF). Regarding flexibility, we will study the mechanisms and services related to the AC that each protocol implements to increase the overall adaptivity of the system. Finally, regarding the performance comparison, we decided to study the overhead caused by each protocol in terms of the number of messages needed to complete the creation of a stream in a network with a single transmitter and receiver.

Note that in these comparisons we consider both FTT implementations that have been mentioned above, namely FTT-SE and HaRTES. Furthermore, these comparisons are done assuming a mono-hop architecture for all protocols.

3.1 Comparing reliability

Faults can be classified by their persistence, that is, whether they are temporary or permanent. Furthermore, we can also differentiate faults that happen in the communication channel from those happening in the nodes.

Networked systems are specially vulnerable to temporary faults happening in the communication channel, called transient faults, since the probability of suffering said faults is high when compared to permanent faults, due to electromagnetic interference in the links. Nevertheless, none of the presented protocols consider specific mechanisms to face the loss of messages caused by transient faults. The consequences derived from message loss are different depending on the protocol and its specific implementation (whether it is an FTT-SE or a HaRTES implementation in the case of FTT) and on the message that is affected by the fault.

In the FTT-SE implementation, the transmission of asynchronous messages in the slaves is scheduled using a signalling mechanism. This mechanism consist in the transmission of a *signalling* message to the master at the beginning of each EC informing of pending asynchronous messages. The master uses the signalling message of every slave to schedule the transmission of asynchronous messages. The transmission of the asynchronous messages is also triggered by the TM. Thus, if the signalling message that notifies a pending slave request is lost, the master will not schedule its transmission. Thus, the AC will be delayed until a signalling message reaches the master. Note that the signalling message is not used in the HaRTES implementation, since the master has a complete view of the communication and can apply frame forwarding policies as explained above. Thus, slave requests in HaRTES are immediately transmitted when issued by the application.

In both FTT-SE and HaRTES if a slave request message is affected by a fault the AC will not take place for said request and, therefore, the slave that sent said request will not be able to create, attach or modify the stream. Furthermore, since FTT does not consider retransmissions for asynchronous messages, the slave request will never be processed or applied.

Concerning the master command message, the effects derived from its loss may be different depending on the specific implementation of FTT. More specifically, in FTT-SE the master is connected to the switch through a single link, therefore if the master command is lost in that link none of the slaves will receive the update information and the NRDBs will remain inconsistent with the SRDB. On the other hand, in both FTT-SE and HaRTES, the master command needs to reach the slaves via their dedicated links. In case transient faults affect only a subset of these links, it is possible that some nodes receive the master command message while others do not receive it. This situation would lead to inconsistencies between NRDBs and SRDB. Again, since in FTT no retransmissions are considered for asynchronous messages, the inconsistencies derived from the master command loss will be irreversible.

Regarding AVB, both talker and listeners must periodically send a message to announce their intention to keep communi-

cating. Thus, if a talker advertise is lost in the link that connects the talker to the bridge the AC will be delayed until one of the following transmissions reaches the destination.

Nevertheless, no retransmissions are considered for messages forwarded by the bridge. Therefore, if a talker advertise message is lost in some of the links when forwarded by the bridge only nodes receiving it will be able to attach to the stream as listeners. If the message that is lost after being transmitted by the bridge is the listener attribute, the talker will not be able to initiate the communication even if the resources are reserved along the path between the talker and the listeners. Moreover, since the messages are lost after being registered in the output ports of the switch, further talker advertise and listener ready messages with the same stream identifier are not forwarded through those ports, which makes it impossible for nodes to later attach to the communication.

Temporary faults in nodes are not considered in any of the presented protocols, and their analysis is left as future work.

With respect to permanent faults we looked in particular to the possible presence of SPoF. Concerning FTT-SE, the communication channel, the master node and the switch are SPoF and no mechanisms are considered to deal with the failure of any of those components. Therefore, a link affected by a permanent fault would lead to the isolation of the node attached to it, preventing it from communicating. In case the link affected by the fault connects to the master node, the whole communication would be affected, since the TM would not reach the salves preventing them from communicating.

In case of a permanent fault in the master, again, the communication among slaves would not take place due to the lack of TM. Finally, in case of a permanent fault in the switch the communication would be impossible, since the switch is responsible for forwarding all messages.

Regarding the HaRTES architecture, both, the communication channel and the HaRTES switch represent SPoF. In the project *Fault Tolerance for Flexible Time-Triggered* (FT4FTT) several mechanisms were proposed to eliminate the existing SPoF of the HaRTES architecture [6], by means of space, time and information redundancy.

Finally, in AVB we can also find two SPoF, the communication channel and the bridge. No mechanisms to eliminate those SPoFs are considered in AVB. Nevertheless the IEEE is currently working on the definition of the second generation of AVB, also known as *Time Sensitive Networking* (TSN) which will provide support for space redundancy of the communication channel and for replication of critical messages.

3.2 Comparing flexibility

The AC is said to provide protocols with flexibility, since it allows to dynamically change the QoS provided by the network. Nevertheless, different levels of flexibility can be achieved depending on the specific implementations and the services considered by the protocols.

All three protocols allow for the creation and deletion of streams. Regarding changes in streams, both FTT-SE and HaRTES allow for slaves to dynamically change the QoS requirements of an exiting stream by sending a slave request message to the master. When the master receives the slave request it evaluates whether the resources are sufficient to carry out the change in all the necessary links and, if so, it broadcasts the new QoS parameters to all slaves to update their NRDBs. Conversely to FTT, in AVB nodes that want to change the QoS provided for a given stream must first remove the existing stream. Moreover, after removing the stream nodes must wait an amount of time predefined by the protocol before requesting its creation with the new parameters. During this time other nodes can perform the reservation of resources for other streams and exhaust the resources, what would prevent the creation of the updated stream.

On the other hand, in both FTT and AVB streams must belong to a class. Specifically, a stream in FTT-SE and HaRTES can be synchronous or asynchronous, while an AVB stream can belong to class A or B. All streams belonging to the same class share resources and follow the same policies. Neither FTT-SE, HaRTES nor AVB provide support for dynamically changing the class a stream belongs to, instead the stream must be removed, a new stream must be created and all nodes involved must then attach to the new stream.

Regarding the responsibility for requesting the modification of the network (be it adding, deleting or modifying a stream), both FTT-SE and HaRTES allow for any slave, even if it is not purporting to participate in the exchange of messages of the stream, to send slave requests to the master to modify the information of any stream. This way legacy applications can be deployed in a transparent manner on top of FTT if there is an FTT-node that performs the requests. Nevertheless, in AVB this is not possible since the node sending the talker advertise must be the same one to later transmit through that stream.

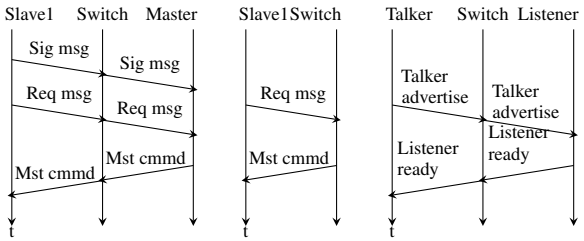
Finally, both FTT and AVB allow for the transmission of non-compliant traffic through the network. This traffic is transmitted in both cases as best-effort traffic when bandwidth is available after transmitting FTT synchronous and asynchronous traffic or AVB class A and class B traffic.

3.3 Comparing performance

To compare the performance obtained by each approach we decided to measure the overhead caused by the AC in terms of the number of messages exchanged. We will consider a simple scenario where only two nodes want to communicate, one as transmitter and the other one as receiver of the information.

During the FTT AC, slaves send at least three slave requests, one to request the stream creation, one for the publisher and one for each subscriber to attach to the stream. Since in FTT slaves can send several slave requests in the same message, a slave that wants to create a stream can also request to be attached to it in the same message. When the slave requests are successfully completed, the master sends a master command for each slave request announcing the result of the AC.

In FTT-SE, as has been mentioned before, slaves that want to attach to a stream must first send a signalling message to inform



(a) FTT-SE messages (b) HaRTES messages (c) AVB messages

Figure 5: Messages exchanged during the Admission Control.

the master about pending slave requests. The master schedules the transmission of the asynchronous messages carrying the slave requests and triggers it in the slaves by means of the TM. Therefore, as seen in Figure 5a in FTT-SE the AC requires the transmission of six messages per participant, that is, twelve messages for our simple scenario. Note that the TM is not considered in this calculation since it is not an AC-specific message, but it is transmitted every EC.

Regarding HaRTES, as depicted in Figure 5b, just two messages are exchanged per participant during the AC, since the signalling mechanism is no longer used. Thus, four messages are exchanged in our simple scenario.

These calculations have been carried out considering that each slave sends their own requests, nevertheless as mentioned before, FTT allows for a single slave to transmit all the requests. Therefore, in such a scenario it would be possible to further reduce the number of messages needed.

With respect to AVB, as depicted in Figure 5c at least four messages are exchanged during the AC. First, a talker that wants to communicate sends a talker advertise message to the bridge, that will in turn check for the availability of resources in each output port and then forward the corresponding talker attribute (talker advertise or talker failed) through them. Then, nodes receiving a talker attribute transmit a listener ready or a listener asking failed through the incoming port to the bridge, that checks again for the availability of resources and sends the corresponding listener attribute to the talker.

Table 1: Summary of the qualitative comparison.

	Reliability		Flexibility			Perf.
	Trans. Faults	Perm. Faults	QoS change[a]	Class change	Legacy nodes	Overh.[b]
FTT-SE	✗	✗	✓	✗	✓	12
HaRTES	✗	✓	✓	✗	✓	4
AVB	✗	✗	✗	✗	✗	4

[a] Dynamically changing the QoS of an already existing stream.
[b] Minimum number of massages exchanged during the Admission Control.

Therefore, we can conclude that the overhead in the communication introduced by FTT-SE during the AC is significant when compared to HaRTES or AVB.

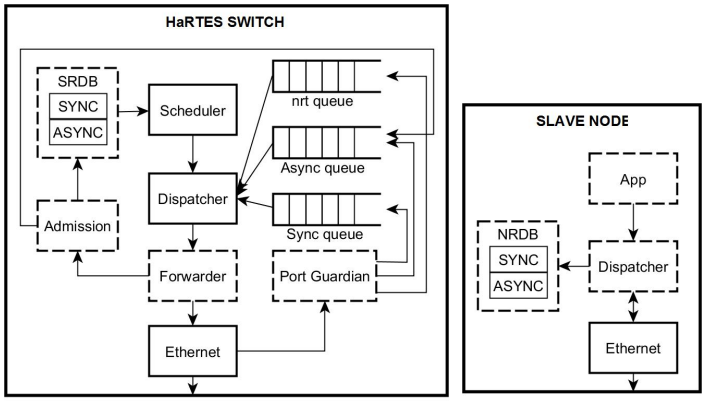
4 Quantitative study of HaRTES

In this section we describe a simulation model of the AC in the HaRTES implementation of FTT and a further performance analysis carried out using said model. The work described in this Section is a first step of the quantitative comparison of the ACs of these protocols, that would be completed in the future.

Taking a look at Table 1 we can see that the HaRTES implementation of FTT has clear advantages when compared to the FTT-SE implementation. For this reason we decided to just study how the AC performs over HaRTES. Furthermore, there is a growing interest on extending the HaRTES switch specification in order to support AVB traffic [12]. This interest is derived from the need to implement highly-dependable and real-time architectures that are compatible with Internet technologies. The model next described becomes the first step towards a further comparison between HaRTES and AVB, in order to establish the advantages and disadvantages presented by each protocol and set the starting point for further integrations.

4.1 Modelling HaRTES

The AC model we present in this work is based on a preliminary model of HaRTES built on top of the OMNeT++ INET framework [7] that is being developed in the University of Banja Luka. This preliminary version did not count with the required AC mechanism, therefore all streams were statically defined during the initialization phase and could not be modified or removed at run-time.



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

Figure 6: Simulation models for the HaRTES switch and the slave nodes [7].

Figure 6a shows the model of the HaRTES switch, while Figure 6b shows the model of a slave node. Each box in the figures represents a module in OMNeT++. The modules of the mentioned preliminary model that have been modified in this work are represented in the figures using dashed lines. Note that even though the initial model already included the Admission module, this module did not implement any functionalities and was there as a hook for future implementation,

only. Moreover, the transmission of asynchronous messages, necessary for transmitting slave requests and master command messages, was not fully supported by the model when this work started. Next we will try to give an insight into the modules modified during this work and the implementation of the associated messages required for the AC.

Concerning the model for a slave node, as depicted in Figure 6b three modules were modified during this work, the `App` module that contains the application the slaves will execute during the functioning of the system; the `Dispatcher` module, responsible for the forwarding of application messages, for the classification of messages coming from the network between data or FTT control messages and the later forwarding of data messages to the application and the processing of FTT messages; and the `NRDB` module that contains the information related to the state of the active streams in the network.

As already mentioned, in HaRTES slave nodes are responsible for requesting the creation, modification and deletion of streams, that is, any slave request must be triggered at the application level. Thus, the `App` module in the slaves had to be modified in order to support the creation and transmission of slave requests.

Regarding the `Dispatcher` module, it was adapted to handle the transmission of slave requests to the Ethernet module. Moreover, it was also modified to support the processing of master command messages received from the switch as a result of the AC.

Typically, the processing of master command messages will lead to the addition or the modification of information in the `NRDB` module. Therefore, this module was extended to support the deletion of existing streams. It is important to note that the functions used to add streams to the `NRDB` were already implemented in the preliminary model, even though this addition was done off-line, during the initialization phase, due to the lack of slave requests and schedulability analysis. Nevertheless, we did implement functions to allow for the deletion of streams from the `NRDB`. Furthermore, since the modification of streams can be easily achieved deleting the existing entry in the table and creating a new one with the new parameters, it was not necessary to implement a specific function to modify the streams' parameters.

Regarding the HaRTES switch model, four modules were modified as depicted in Figure 6a. The `Port Guardian` module is responsible for filtering and classifying the messages coming from the network, sending them to the appropriate queue or discarding them when necessary. The `Forwarder` module is responsible for deciding whether each message must be sent to the `Admission` module or to the `Ethernet` layer. It also prepares each message to be transmitted through the network by adding the corresponding header. In what concerns the `Admission` module, it is responsible for accepting or rejecting slave requests, by checking for the availability of resources in the network through a schedulability analysis. The `SRDB` module contains the information of the existing streams used to carry out the schedule in every EC.

As already mentioned concerning the AC, the master that is embedded inside the switch is responsible for processing slave requests, checking for the availability of resources and informing the slaves about the result. Since both slave request and master command messages are asynchronous messages the HaRTES switch model had to be adapted to support the reception and transmission of this type of messages. More 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 carrying the result of the AC to the slaves.

The `Admission` module was adapted to process slave requests and to carry out specific actions depending on the type of request. Specifically, when receiving an addition or modification request, the `Admission` module executes the implemented schedulability analysis. Conversely when it is a deletion request, the module simply deletes the entry from the `SRDB` table. Finally, whenever the master decides to actually perform a modification in the network, after a slave request a master command message is sent to inform the slaves about the changes.

Concerning the `SRDB` module it is important to note that this module is actually an instance of the same module used for the `NRDB` in the nodes. Therefore, its functionalities within this model are the same and all changes applied to the `NRDB` were also applied to the `SRDB`.

Finally, all FTT-specific messages, such as the trigger message, synchronous messages, etc. must be previously defined in order to use them in the model. Thus, the definition of FTT messages was extended to include slave request and master command messages. The definition of these messages does not affect any module in the switch nor the node models, but it is orthogonal to both of them.

4.2 Schedulability Analysis

As already mentioned, the `Admission` module of the HaRTES switch model was modified to include the schedulability analysis responsible for deciding if there are enough resources in the network to apply a requested change. More specifically, during 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 [10]. These schedulability analyses are utilisation-based, that is, the sum of the utilisation ratio of the streams in every link must be under a given limit. The utilisation of a stream corresponds to the amount of resources that will be needed to transmit its messages. Since FTT is based on an EC-schedule, that is, the transmission of a message must be completed within the EC it was scheduled, the utilisation limit is scaled to account for the Synchronous Window and the EC lengths, to ensure that there are enough resources in the corresponding window of the EC.

Since HaRTES uses full-duplex links to connect the switch to the slaves, the communication analysis must account for

the different loads of the links. More specifically, as shown in Figure 7, the utilisation of the uplink (link through which the node sends messages to the switch) of the transmitter may be different from the utilisation of the downlink (link used by the master to transmit messages to a node) of the receiver, depending on the number of nodes involved in the communication and the messages transmitted by each of them.

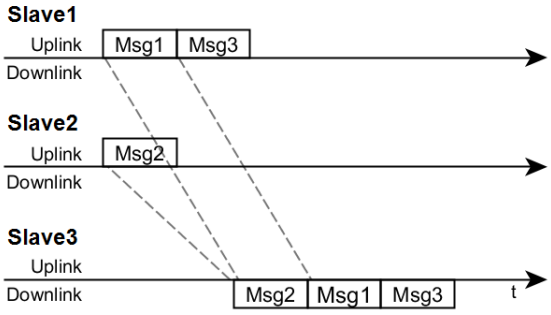


Figure 7: Representation of the uplinks and downlinks of three slaves during the transmission of several messages.

As seen in Figure 7 the utilisation in the uplink only depends on the load generated by the slave in each EC. Therefore, for each node and EC the analysis consists in adding the utilization of all the streams to be transmitted by said node in said EC and checking whether this sum exceeds the threshold. In contrast, since the transmission of messages in the downlinks is affected by several factors, the analysis is not so simple for said links. Observing the transmission of Msg3 in Figure 7, we see that the transmission instant of the message in the downlink does not just depend on the load of the downlink, but also on the moment it was transmitted through the uplink. Thus, the analysis must account for the delay that the message may suffer in the uplink due to the transmission of other messages, such as Msg1 in our example. Also, the analysis must account for the link and switch delay, represented in the Figure by the dashed lines. Further discussion on the analyses is out of the scope of this work, for more detail the reader can refer to [10].

4.3 Experiments and results

As mentioned above, we want to study the performance of the AC in HaRTES. To this, we must first identify all the elements that may have an impact on the performance. First, since the communication in FTT is divided in ECs and the schedule is done on an EC-basis, the duration of the EC may impact the performance of the whole system, including the AC. It is important to note that the duration of the EC is not fixed, but it may change depending on several factors, like the type of traffic or application. Since the duration of the EC limits the number of messages that can be transmitted and processed, the duration should vary depending on the dynamics of the system and the specific application. Thus the experiments must account for different durations of the EC.

Another element that may affect the performance of the AC is the load of the network. This is because the load has a direct impact on the time needed to complete the schedulability analysis, and thus on the AC. This impact is due to the fact that the analysis is utilisation-based, which means that it is necessary to calculate the sum of the utilisation rate of all the streams that transmit through a link and check if it is under a certain limit. Since the information related to the streams is saved in the SRDB, in order to perform the analysis the master must read all the entries of the SRDB several times, searching for the streams transmitted through the affected links. Thus, when the load of the network increases so do the size of the SRDB and the time required to compute the analysis.

Here we want to study the impact that these two parameters have on the performance of the AC. To that end we measure the time that takes to complete the AC from the instant a slave sends a slave request until that slave finishes processing the master command and updates its NRDB. In HaRTES slaves can transmit their asynchronous messages at any moment of the EC, as soon as they are produced by the application. Nevertheless, the master only reads the received slave requests during the Asynchronous Window to ensure the timeliness of the protocol. Moreover, since the master command message is asynchronous, its transmission is confined to the Asynchronous Window of the first EC with enough bandwidth available after completing the schedulability analysis. Thus, the master command may be sent during the same EC as the slave request or in a later EC, depending on the duration of the schedulability analysis.

Since in the normal operation of the system the load of the network may change with time, we carried out several tests simulating different network loads. Specifically, we considered three loads, labelled as low, medium, and heavy, which correspond to durations of the schedulability analysis of 1, 50 and 100 ms respectively.

Note that OMNeT++ is an event-based simulator and therefore it only simulates the passage of time when events occur. Moreover, events in OMNeT++ are represented by the transmission of messages. Since the schedulability analysis is not an event, simulated time does not pass during its execution. Therefore, we modified the Admission module in order to force the sequential processing of slave requests and to account for the execution time of the analysis. This was done using a queue and a timer. All slave requests arriving to the Admission module are saved in the queue. Afterwards, the timer is set with a predefined time that depends on the type of request, for instance a delete request does not require the schedulability analysis to be performed while a creation or modification request do; and the load of the network. When the timer expires it triggers the instantaneous analysis of the first slave request in the queue. This process is repeated as long as there are slave requests in the queue pending to be processed.

On the other hand, we also want to study the impact that concurrent slave requests may have on the duration of the AC. In HaRTES the master processes as many slave requests as possible every EC. To study the impact of concurrency we considered two scenarios, one where slave requests are sent

and processed sequentially and a second scenario in which all the requests are issued at the same time.

Table 2: Experiment parameters

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

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

Table 2 shows an overview of the datasets used in the experiments. We repeated the experiments using each dataset for several durations of the EC, more specifically, we repeated the experiments for EC durations of 1, 2, 5, 10, 20, 50 and 100 ms. The results are grouped by the dataset used.

The network used in all the experiments is composed of one HaRTES switch connected to ten slave nodes, labelled with an ID that goes from 1 to 10. All the slaves in the network are set to request the creation of a new stream. In the first case (no concurrency) the transmission of slave requests is sequential. To achieve that, we set an order for the transmission of requests, that corresponds to the ID number assigned to the slave. Each slave waits for the reception of the master command of the previous request to trigger the transmission of its own slave request, that is sent right after receiving the TM of the following EC, during the Synchronous Window. In the second scenario (with concurrent requests) all the slaves are set to transmit their request at the same time, at the beginning of the EC during the Synchronous Window.

Regarding the duration of each window of the EC, for the sake of simplicity we considered the length of the Synchronous and the Asynchronous Windows to be equal. Therefore, each window represents the 50% of the EC time remaining after the transmission of the TM.

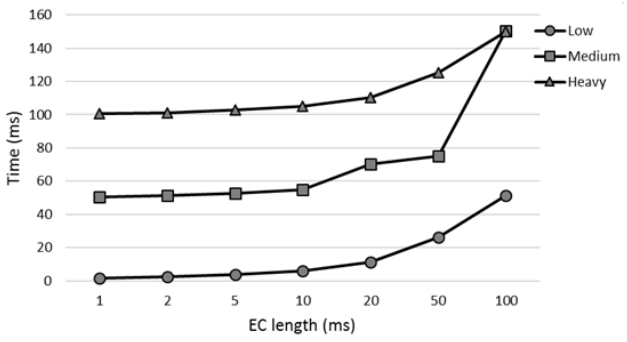


Figure 8: Mean duration of the Admission Control for different EC lengths and network loads, without concurrent requests.

Figure 8 shows the mean duration (the average of the duration for the requests of each slave) 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. Having longer EC lengths increases the number of frames that can interfere with the slave requests, increasing the time required to carry out the AC. 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.

Table 3: 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

In order to understand why the EC length has such an impact even in networks with low load we can look at Table 3, that presents the numerical results of the experiment. We can see that in most cases the time to complete the AC corresponds to the duration of the schedulability analysis plus half of the EC length. As said before, slave requests are transmitted right at the beginning of the Synchronous Window, while the processing of requests by the master is delayed until the Asynchronous Window starts, to protect the timeliness of the synchronous traffic. Therefore, we can see that the size of the windows, and thus the size of the EC, has a severe impact on the performance of the AC.

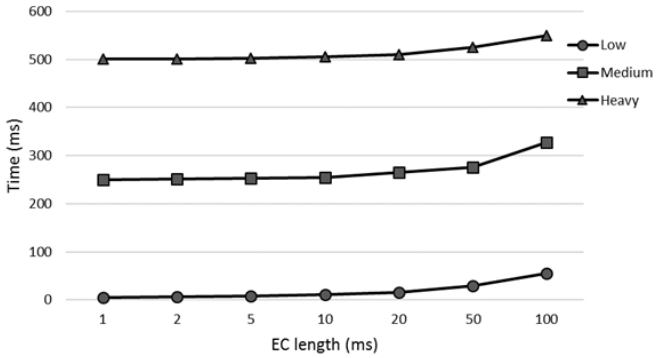


Figure 9: Mean duration of the Admission Control for different EC lengths and network loads, with concurrent requests.

Figure 9 shows the mean duration of the AC for different EC lengths with concurrency. Since all the requests are issued at the same time and they are processed sequentially, the last request to be processed by the master must wait for the other requests to be analysed. 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, to further analyse the impact of concurrency in the performance of the AC, we observed the maximum duration

Table 4: Maximum duration of the AC for different EC lengths network loads and concurrent requests.

EC(ms)	1	2	5	10	20	50	100
Low	11.01	13.23	16.39	23.38	30.01	60.01	110.01
Medium	501.01	502.01	505.01	510.01	531.234	550.01	655.67
Heavy	1001.01	1002.01	1005.01	1010.02	1020.01	1050.01	1100.01

of the AC, shown in Table 4 . 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. Although it could be possible to use a worst-case approach for trying to bound the AC response time, the fact remains that in the most general case without imposing restrictions on the number of requests that each slave can send, such bound will not be found. Additionally, even restricting the number of requests, the bound that might be obtained would be truly pessimistic, specially in the case of heavy loads. The resulting lack of predictability in practice is particularly critical since the AC 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 slave 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 already been rejected, thus delaying the processing of requests related to other streams. Hence, it is crucial to add to the specification of the protocol the notification of rejected requests by the AC to slaves.

5 Conclusions

In the last years there has been a growing interest in using Ethernet as the technology for developing distributed embedded systems, even in the automation, avionics and automotive domains. Moreover, recently industry has shown interest in developing multimedia-based applications in these fields, for both entertainment and assistance. In these areas multimedia applications coexist with traditional distributed control applications, generating a great diversity of traffic in the network. On top of that, these novel applications can be launched at any moment during the system's operation, what means that the number of participants may change dynamically.

Nevertheless, Ethernet only supports best-effort traffic, and thus, control traffic can not be transmitted with adequate timing guarantees. Moreover, Ethernet does not provide mechanisms to dynamically change the network in runtime. Several protocols were proposed to overcome these drawbacks, such as Flexible Time-Triggered and Audio Video Bridging.

In environments that may change in unpredictable manners adaptivity and reconfigurability are of utmost importance to provide adequate real-time guarantees. Admission Control allows for the on-line reservation of resources along the path between a transmitter and a receiver, providing real-time capacities by preventing unbounded packet delays or packet losses. Moreover, Admission Control provides flexibility by supporting the modification of the network's parameters in runtime.

In this work we described the Admission Control mechanisms implemented in three Ethernet-based communication protocols, namely FTT-SE, HaRTES and AVB. Furthermore, we presented a qualitative comparison of the AC mechanisms from three different perspectives: reliability, flexibility and performance. Table 1 presents an overview of said comparison. Concerning reliability, we observed that none of the presented protocols provides adequate an implementation that tolerates transient faults in the channel, what can lead to the loss of messages related to the AC. Moreover, HaRTES is the only protocol that currently counts with mechanisms to tolerate permanent faults, both in the channel and the nodes. Nevertheless, the IEEE is currently working on the specification of a new set of standards to overcome some of the reliability problems of the communication channel, but not of the nodes.

Regarding flexibility, we observed that both FTT implementations allow for on-line changing the QoS of the streams, while AVB forces the deletion and redefinition of the affected streams, reducing the reactivity of the system. Moreover,FTT also allows for legacy nodes to send hard and soft real-time traffic by delegating the definition of streams to a single slave, conversely to AVB.

Finally, concerning the performance we concluded that FTT-SE introduces a significant overhead in the communication when compared to HaRTES and AVB, in terms of the number of messages exchanged during the process.

On the basis of the results obtained we decided to further analyse the HaRTES implementation of FTT, since it had exhibit better results than FTT-SE. To that aim, we implemented a simulation model in the OMNeT++ INET framework for the Admission Control of HaRTES. This model is based on a preliminary model of HaRTES that is currently being developed in the University of Banja Luka and includes the models for both the HaRTES switch and the slave nodes.

We used the model to study the impact that the length of the Elementary Cycle has on the duration of the Admission Control. We carried out several tests considering different EC lengths, network loads and levels of concurrency of the slave requests. The results obtained show that the duration of the EC and the duration of each of its parts (Synchronous and Asynchronous Windows) have an important impact on the performance of the AC, since the time needed to complete the AC increases with

the length of the EC and the Synchronous Window. We also observed that, as expected, the level of concurrency affects the performance, that decreases as the number of concurrent requests increases.

Moreover, with the obtained results we can conclude that it may be impossible for slaves to determine the time that will take for the AC to be completed. This is specially critical since HaRTES does not count with mechanisms to notify slaves when a slave request has been rejected and therefore it may be impossible for slaves to differentiate when a request has been lost, delayed or simply rejected by the master. We recommended that this notification of rejected request is included in the HaRTES specification.

Finally, as part of the future work we plan to extend this performance analysis to include the Stream Reservation Protocol implemented in the AVB standards, in order to complete the comparison among the protocols discussed in this work.

Acknowledgements

This work was supported by project TEC2015-70313-R (Spanish MINECO/FEDER), by 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.

I wish to express my gratitude to Doctor Mladen Knezic for his inestimable participation and assistance during the execution of this work. I would like to thank my supervisors Julián Proenza, for his support and guidance during this work and during my journey, and Luís Almeida, for giving me the opportunity to work, learn and grow with him.

Last but not least, I want to thank to all my friends, specially those in Portugal who turned this work into a fantastic adventure, and my family, for the support and care they provide me with in every step I take.

Appendix A Publications of this work

- I. Álvarez, L. Almeida and J. Proenza, A First Qualitative Comparison of the Admission Control in FTT-SE, HaRTES and AVB, In *Proceedings of the 12th IEEE World Conference on Factory Communication Systems (WFCS)*, Aveiro, 2016. doi: 10.1109/WFCS.2016.7496524
- I. Álvarez, M. Knezic, L. Almeida and J. Proenza, A first Performance Analysis of the Admission Control in the HaRTES Ethernet Switch, In *Proceedings of the 21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, 2016.

Bibliography

[1] The INET Framework—An open-source OMNeT++ model suite for wired, wireless and mobile networks.

[2] *IEEE Standard for Local and metropolitan area networks, Virtual Bridged Local AreaNetworks, Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*, IEEE Std. 802.1Qav, 2009.

[3] *IEEE Standard for Local and metropolitan area networks, Virtual Bridged Local Area Networks, Amendment 14: Stream Reservation Protocol*, IEEE Std. 802.1Qat, 2010.

[4] *IEEE Standard for Local and metropolitan area networks-Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, IEEE Std. 802.1AS, 2011.

[5] Gonzalo Carvajal, Luis Araneda, Alejandro Wolf, Miguel Figueroa, and Sebastian Fischmeister. Integrating Dynamic-TDMA Communication Channels into COTS Ethernet Networks. *IEEE Transactions on Industrial Informatics*, PP, 2016.

[6] David Gessner, Julian Proenza, and Manuel Barranco. A Proposal for Managing the Redundancy Provided by the Flexible Time-Triggered Replicated Star for Ethernet. In *Proc. 10th IEEE Int. Workshop on Factory Communication Systems (WFCS)*, Toulouse, France, May 2014.

[7] 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.

[8] Herman Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The Time-Triggered Ethernet (TTE) Design. In *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (Seattle, Washington: TU Wien)*, page 22–33, May 2005.

[9] Lucia Lo Bello. Novel trends in Automotive Networks: A perspective on Ethernet and the IEEE Audio Video Bridging. In *Proc. 19th IEEE Conf. on Emerging Technologies & Factory Automation (ETFA)*, September 2014.

[10] 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*, pages 1–10, Sept 2010.

[11] Paulo Pedreiras and Luís 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.

[12] R. Sousa, P. Pedreiras, and P. Gonçalves. Enabling IIoT IP Backbones with Real-Time Guarantees. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–6, Sept 2015.

[13] András Varga. The omnet++ discrete event simulation system. In *ESM'01*, 2001.