A Synthesizable Ethernet Switch with Enhanced Real-Time Features

R. Santos, R. Marau, A. Vieira, P. Pedreiras, A. Oliveira
IEETA - DETI/University of Aveiro
3810-193 Aveiro, Portugal
{rsantos,marau,alexandrevieira,pbrp,arnaldo}@ua.pt

Luis Almeida
IEETA - DEEC/University of Porto
4200-465 Porto, Portugal
lda@fe.up.pt

***Abstract*—The use of switched Ethernet for safe real-time communication still suffers from undesired phenomena, such as blocking caused by long non-preemptive frames, lack of protection against errors in the time domain, couplings across virtual LANs and priority levels via internal switch shared resources. Recently, a few solutions were proposed to cope with such phenomena. One such solution is based on an enhanced switch following the Flexible Time-Triggered paradigm, which enforces strict service differentiation with any kind of traffic scheduling, blocking-free forwarding and timing errors confinement. In this paper we propose a new architecture following an hardware-software co-design approach that simplifies the development of the enhanced switch features by detaching the traffic scheduling from the traffic switching. The paper shows experimental results with an actual switch prototype that confirm the desired switch properties.**

I. Introduction

Nowadays, switched Ethernet architectures present attractive features such as large bandwidth, cheap network controllers, high availability, easy integration with Internet and a clear path of evolution. These features are fostering the expansion of switched Ethernet architectures to new application areas such as high-speed servoing, target tracking in military systems or even the control of electrical protection systems in substations. However, Ethernet switches are not typically designed to support the timeliness and safety requirements found in many of these emerging application areas. These limitations arise from aspects like blocking caused by long non-preemptive frames, lack of protection against errors in time domain, a limited number of priorities and possible memory overflows.

Supporting real-time communication with switched Ethernet has been a topic of intense research for several years. Some of the techniques proposed to overcome the limitations above mentioned rely on Commercial Of-The-Shelf (COTS) Ethernet switches while others use customized Ethernet switches. The first group includes many different techniques, most of them requiring software modifications in the end nodes. Techniques in this group include traffic shaping [1], master-slave protocols [2] [3] [4] that provide more efficient scheduling policies, QoS management and admission control features or simply limiting the generated traffic by application design [5]. However, by relying in COTS hardware, the sphere of control of these protocols is limited to the nodes that strictly comply with the associated protocol. This limitation has two important consequences. On one hand standard Ethernet nodes cannot be integrated in the network since non-conformant transmissions can jeopardize the real-time services. On the other hand the system timeliness also becomes vulnerable to malfunctioning nodes. A possible way to address these problems consists in integrating the traffic management and control mechanisms within the switch itself, creating a modified Ethernet switch. This direction, followed in proposals such as [6] [7] [8], allows obtaining gains in terms of performance and timeliness guarantees. This approach yields a relatively low level of intrusion because it is still possible using COTS hardware and standard software stacks in the end nodes, possibly with specific layers just for accessing the real-time services. Additionally, the integration of server-based traffic handling policies allows an efficient handling of asynchronous messages streams [9].

This paper describes the FPGA-based architecture of a modified Ethernet switch providing real-time communication services based on the Flexible Time-Triggered paradigm. The current architecture is an evolution of preliminary prototypes [10] [6] [11] that simplifies the development of the enhanced switch features by detaching the traffic scheduling from the traffic switching, the former being implemented in software in an external CPU and the latter being implemented in a dedicated FPGA. The paper describes the whole switch architecture and presents experimental results achieved with a working prototype that validate the desired real-time and traffic integration/isolation properties.

The paper is organized as follows. Section II presents the related work and contribution. Section III presents the rationale and scope for developing the new switch. Section IV presents the implementation of the switching part of the FTT-enabled switch using FPGA technology while Section V presents the experimental results with a working prototype. Finally, Section VI concludes the paper and points to lines of future work.

II. Related work and Contribution

FPGA technology presents a set of interesting features, such as very large logic capacity, flexibility of use and low NRE (Non-recurring engineering) costs, which makes it well suited to build customizable devices with specific properties for different application domains. Moreover, they exhibit a fast design cycle, are easily upgradable and permit fast prototyping due to high level modeling languages and synthesis tools.

Industrial and embedded device vendors are increasingly interested in providing Ethernet devices supporting real-time services. The flexibility and NRE costs of FPGA technology make it very appealing to the industrial and embedded device market. Additionally, according to [12], there is a demand for the integration of Modbus/TCP, Profinet IO and Ethernet/IP into a single device. The FPGAs are probably the best solution to this problem, given the possibility to integrate soft-core processors and IP cores.

For the case of real-time Ethernet protocols, one possible solution is to enhance the switches with tighter timing control and traffic classification capabilities. There are currently two protocols that use modified switches, TTEthernet and Profinet IRT.

TTEthernet (Time Triggered Ethernet) [8] [13] [14] is a new scalable real-time Ethernet platform that provides safe and real-time Ethernet communication services. The communication can be performed using three different traffic classes, Time-Triggered (TT), Rate-Constrained (RC) and Best-Effort (BE) messages. This protocol uses a modified switch that includes a dedicated TTEthernet controller and a high performance switching module based on an FPGA board [8].

The other protocol relying on customized Ethernet switches is Profinet IRT [7] [15] [16]. It is implemented with a modified switch that offers determinism in the transmission through explicit bandwidth reservation for the real-time data. The scheduling parameters are configured during the setup phase and they are obtained through the previous execution of a scheduling algorithm. On the other hand, this switch also allows the integration of Ethernet standard devices whose traffic is confined to dedicated time windows.

Despite the advantages presented by the above two protocols, they exhibit some limitations, e.g., they require a static pre-defined configuration for the real-time traffic, the on-line admission control is not generally available and miss the capability to adapt on-line to variable communication requirements as needed for dyamic quality of service management. Therefore, we propose an alternative based on the Flexible Time-Triggered (FTT) paradigm that is fully reconfigurable online. We propose an FTT-enabled switch providing support for arbitrary traffic scheduling policies, as well as online admission control, policing mechanism and dynamic quality of service management. Among the traffic scheduling policies, the switch integrates server-based scheduling techniques to efficiently handle real-time streams with arbitrary arrival patterns (video streams, alarms, sensors, ...) as well as legacy applications that are not prepared for the transmission control involved in synchronous communication. With these features, we believe this is the first Ethernet switch capable of providing a truely flexible solution for handling synchronous real-time, asynchronous real-time and non-real-time traffic with mutual isolation.

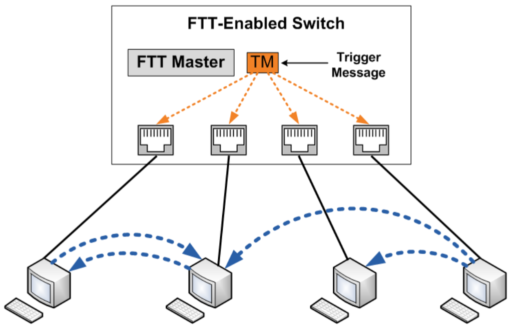

Fig. 1. FTT-Enabled Switch.

III. FTT-Enabled Switch

A. *Rationale*

The FTT-enabled switch is based on the Flexible Time-Triggered paradigm, with the FTT master included in the switch (Figure 1). Therefore, it uses the master/multi-slave transmission control technique, according to which a master addresses several slaves with a single poll message, considerably alleviating the protocol overhead when compared to the conventional master-slave techniques. The communication is organized in fixed duration slots called Elementary Cycles (ECs). Each EC starts with one poll message sent by the master, called Trigger Message (TM). The TM contains the schedule for that particular EC. Only the messages that fit within an EC are scheduled by the Master, thus memory overflows inside the switch are completely avoided. The FTT protocol defines three traffic classes: i) periodic real-time messages activated by the Master (referred to as *synchronous* since their transmission is synchronized with the periodic traffic scheduler); 2) aperiodic real-time traffic (called *asynchronous*), autonomously activated by the application within each node and 3) non real-time traffic. The synchronous and asynchronous traffic are transmitted within the real-time window and have guaranteed timeliness, while the non real-time traffic is scheduled in background, in the time left within the EC, in the so-called non real-time (NRT) window.

This solution with the FTT master included in the switch enables preserving all FTT attributes, but at the same time, it permits obtaining important gains in the following key aspects:

- A performance boost of the asynchronous traffic, which in this case is autonomously triggered by the nodes instead of being polled by the master node, while maintaining agreggated or per-stream temporal isolation;
- An increase in the system integrity since unauthorized transmissions can be readily blocked at the switch input ports, thus not interfering with the rest of the system.
- Seamless integration of non-FTT-compliant nodes without jeopardizing the real-time services.

B. *Architecture*

The functional architecture of the FTT-enabled Ethernet switch has been presented and discussed in earlier work [10] [6] [11]. It is basically formed by four main blocks, the *master* itself, which includes the System Requirements Data Base, the admission controller, the synchronous scheduler and Quality of Service (QoS) manager, the *input blocks* that classify, validate and filter the ingress traffic, the *global memory pool* that holds the messages of each class in independent subdivisions, and the *output blocks* that include three pointer queues, one for each traffic class, and asure the jitter-free transmission of the TM in each EC.

This architecture allows us maintaining a tight control on the traffic that enters the switch, including enforcing an adequate timing behavior and temporal isolation among traffic classes, whichever is the traffic arrival pattern. Therefore, nodes producing NRT or asynchronous traffic can use the switch transparently, as if it was a standard Ethernet switch, without needing any modification of the node software. The former traffic will not interfere with the latter and none will interfere with the synchronous traffic that might be flowing through the switch. This grants a high flexibility to the proposed solution for real-time communication, which efficiently combines those heterogeneous traffic classes with mutual temporal isolation. An appropriate FTT network driver is just required for the synchronous communication services and for setting up asynchronous channels.

IV. FTT-enabled Switch Implementation

The hardware architecture of the FTT-enabled switch using FPGA technology is shown in Figure 2. The FTT master, represented by the Master Unit, executes a complex set of operations and thus it is better suited for software implementation. On the other hand, there are several functionalities in the FTT-enabled switch that need predictability, determinism and speed in their execution thus being preferable to execute them in hardware. This group includes the reception, switching and transmission processes. This way, all blocks except the Master Unit are implemented in hardware (Figure 2). We will call this set of blocks the Switching Module.

The integration of these two parts, Master Unit and Switching Module, can be performed in different ways:

- The Master Unit runs in an independent CPU and the communication with the FPGA is carried out by a conventional communication mean available in the development board (e.g Ethernet, USB, PCI, ...);
- Utilization of an FPGA embedded processor to execute the Master Unit, either synthetizable or hardwired (e.g MicroBlaze, PowerPC).

To save resources in the FPGA, we currently followed the former approach, using an Ethernet port of the switch.

A. Hardware Implementation

Figure 2 represents the complete view of the FTT-enabled switch. The Switching Module is implemented on an FPGA except for the Ethernet PHYs that remain outside due to their electrical characteristics, timing requirements and wide availability of pre-built modules

1) MAC IP Core: Each Ethernet port has one associated Ethernet PHY directly linked to one Xilinx Tri-Mode Ethernet MAC soft core fully compliant with the IEEE 802.3 standard, which can operate at 10/100/1000 Mbits/sec and can be implemented on the programable logic resources of Xilinx FPGAs. This MAC core is highly configurable and provides reception/transmission control, core management and Ethernet PHY Media Independent Interface.

The MAC IP Core provides the clocks that enable the reception and the transmission of data. On the other hand, one independent main clock manages the switch core, composed by three main blocks (Memory Pool, Switching and Control Logic and Master Interface). The union of these different clock domains was possible using FIFOs (included inside the MAC Interface Unit) with independent clocks for the write and read sides. This way, for each port, the information received from the MAC IP Core is written in the corresponding FIFO with the reception clock provided by the same MAC. The reading of the information from the FIFO is performed with the main clock. The transmission of the data is processed in a similar way. The data ready for transmission in a specific port is written in the transmission FIFO with the main clock and read by the MAC with its transmission clock.

2) MAC Interface Unit: The MAC Interface Unit, specific for each switch port, implements all the logic that allows to configure the MAC IP Core. In the reception, it classifies, validates and handles the received data, writing it in memory. In the transmission, it reads the data from the memory, handles it and manages the interface with the MAC IP Core for the transmission.

The Reception Unit implements the reception interface of Ethernet frames coming from the MAC IP Core. The received data are inserted in the FIFO that separates the clock domains.

The next unit in the reception chain is the Classifier and Validate Unit that processes the received packets accordingly. For instance, the non-FTT packets are directly delivered to the Reception Buffer Unit. The FTT control packets, that comprise commands to the Master Unit, are delivered to the Master Interface. Finally, the remaining packets still pass through a validation process. If these packets are consistent with the EC-Schedule provided in the Trigger Message by the Master Unit, they are delivered to the Reception Buffer Unit, otherwise, they are trashed. Thus, the integrity of the switch and network is guaranteed.

The Reception Buffer Unit accumulates the incoming bytes to form a word N-bytes wide, where N is the number of ports. This way, the writing in the Memory Unit is carried out, per port, with N bytes at a time and at a rate N times slower than the arrival rate of the respective byte stream. After multiplexing all N ports, the writing frequency at the Memory Unit is equal to the bytes arrival rate at the individual ports, avoiding the need for higher frequency clocks. This technique is also used in the transmission chain by the Transmission Buffer Unit, which receives words N bytes wide from the Memory Unit and forwards the data to the respective FIFO one byte at a time. The reading frequency from the Memory

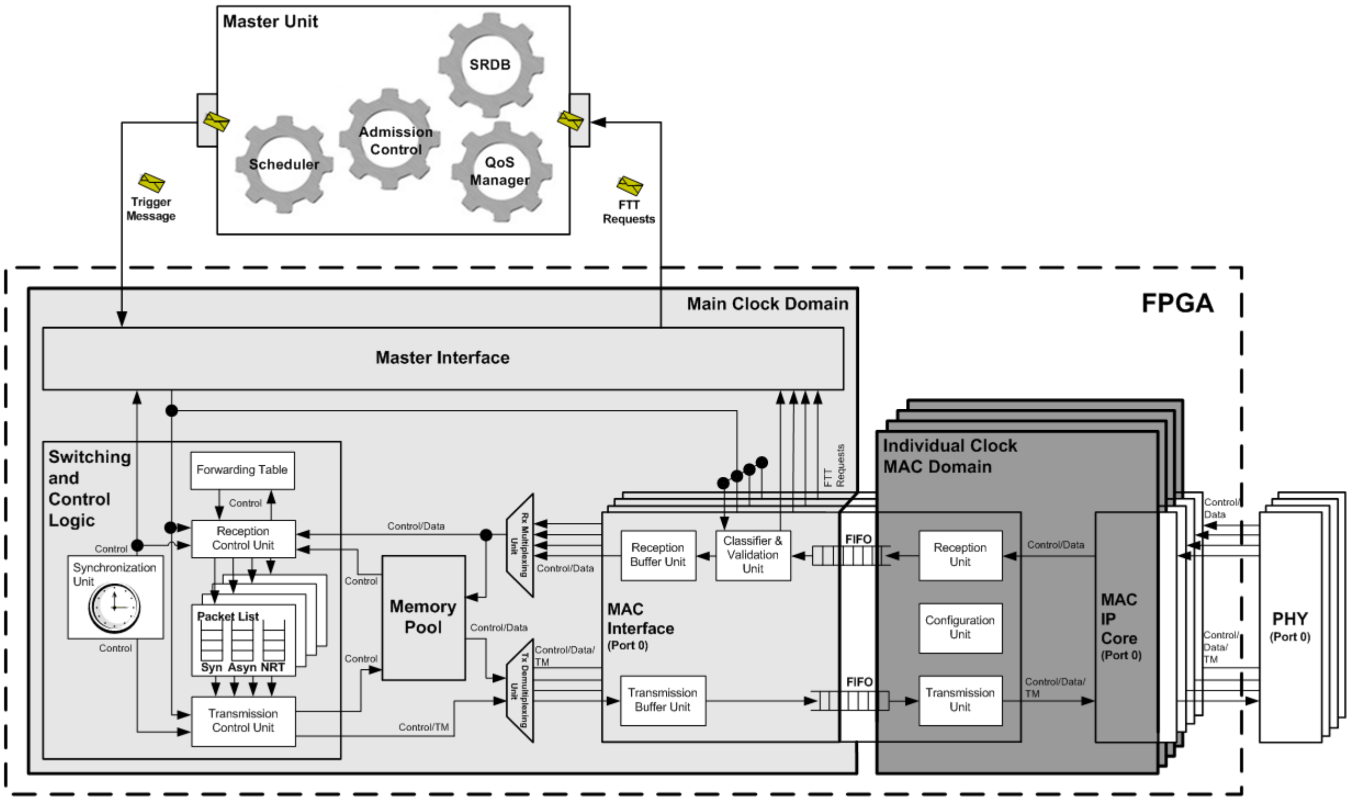

Fig. 2. FTT-enabled switch - complete view.

Unit is again equal to the byte transmission rate in the port FIFO.

The Transmission Unit reads the data from the FIFO, used to separate clock domains, and manages the sending of this data to the MAC IP Core. Finally, in a different chain, the Configuration Unit implements the logic used to configure the MAC IP Core at startup.

3) Control and Switching Logic Unit: The Control and Switching Logic Unit plays a central role within the switch, performing the reception, switching and transmission management based on the control and status signals provided by MAC Interface Unit. Moreover, it synchronizes all the switch and protocol operations. The Reception Control Unit manages the forwarding of the received packets to the output ports. This unit receives, for each Elementary Cycle, the corresponding Trigger Message, which includes the identification of all real-time data packets (FTT data) that will be received from each input port and sent by each output port (EC-Schedule). Thus, the FTT data packets are forwarded based on the contents of the Trigger Message, only. On the other hand, normal non-real-time packets (non-FTT data) are forwarded based on the Forwarding Table that is updated dynamically as in common switches. The actual forwarding is carried out by delivering the pointers of the respective packets to the Packet List Unit attached to the corresponding output port, and inserting them in the correct queue. This unit contains three queues, one for each traffic class (Sync RT, Async RT, and NRT), which contain the pointers to the packets that have to be sent in this EC.

4) Transmission Control Unit: The Transmission Control Unit controls the transmission of the packets in those queues respecting the corresponding phases in the EC, thus enforcing appropriate traffic confinement of the different traffic classes. Moreover, the Transmission Control Unit only transmits messages from the asynchronous or NRT queues if the time left within the respective windows is enough, thus preventing blocking of the Trigger Message and synchronous traffic. To start the transmission of a packet, this unit asks the Memory Unit to send its data to the MAC Interface Unit of the respective output port. One particular case is the transmission of the Trigger Message, which is sent directly to all MAC Interface Units (broadcast), at the beginning of each EC, as determined by the Synchronization Unit, in a blocking-free fashion thus with high precision. The Synchronization Unit controls the EC timing and requests EC-Schedules (Trigger Messages) from the Master Unit.

5) Memory Pool: The Memory Pool implements a dual port static Synchronous Random Access Memory - SRAM with separate clocks, control, address and data buses. One port (write only) is shared among all input ports and the other (read only) used by all output ports. The memory is segmented in blocks that allow to store packets with maximum size, and each block can store one packet, only. This mechanism is inefficient with short packets but simpler to manage, deterministic and not suffering from fragmentation. Therefore, it is enough to prove the concept.

6) Multiplexing/Demultiplexing Units: The Rx Multiplexing Unit is shared among all switch ports. It is basically a multiplexer (TDMA wheel) that allows all port uplinks to write the received messages data into the switch main memory. Contrarily, the TX Demultiplexing Unit is also shared among all switch ports and it is basically a demultiplexer that allows all ports to read data from the Memory Pool and write it on the corresponding downlink MAC.

B. Software Implementation

The Master Unit implements the SRDB, the QoS Manager, the Admission Control and the Scheduler. From an implementation point of view, these operations are algorithmically complex and make extensive use of dynamic lists, which are more efficiently implemented in software. Moreover, these operations basically correspond to the functionality of the Master in the FTT-SE protocol [2], which is a fully software implemented version of the protocol that works over COTS Ethernet switches.

Therefore, in order to re-use the FTT-SE Master as the Master Unit, the Master Interface was designed to provide the necessary standard interfaces that FTT-SE uses, namely one Ethernet port. Moreover, this port is dedicated to the communication with the Master Unit and does not integrate the communication ports managed by the Switching Module, which allowed us to incorporate the FTT-SE Master with minimal adaptations, offering substantial gains in development time. Thus, the FTT-SE Master is an autonomous component that can be connected to a COTS switch or to the Switching Module herein proposed, providing two different levels of service concerning traffic filtering, confinement, policing, temporal protection and compatibility with non-FTT nodes. This is a highly flexible and efficient solution.

One aspect that had to be worked out was the EC clocking. In the full software version of FTT-SE the EC timing is directly controlled by the Master. In the enhanced switch herein proposed the EC timing is controlled by the switch Synchronization Unit. This unit sends requests to the Master Unit that triggers the scheduling activity leading to the generation of the Trigger Messages containing the respective EC-Schedules, which are then used by the Switching Module.

This aspect is also relevant when considering multi-switch topologies. In such case, only one Master is used and the source of the EC clocking must be unique, too. Therefore, when using the enhanced switch, it must be the only one of such kind, with all others being COTS switches. Other alternatives to multi-switch architectures based on the enhanced switch are still under development.

Concerning the internal components of the Master Unit, the System Requirements Database (SRDB) deserves a special reference. It is a central repository for all the information related to traffic management, namely the messages attributes for both synchronous and asynchronous traffic, e.g., period/minimum inter-arrival time, length in bytes, priority if applicable, deadline and offset if applicable, plus information about the resources allocated to each traffic class, e.g., phase durations and maximum amount of buffer memory, and global configuration information, e.g., elementary cycle duration and bit rate.

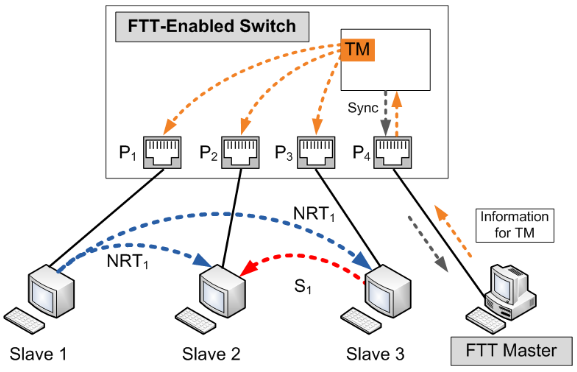

Fig. 3. Experiment 1: synchronous and NRT traffic.

The attributes in the SRDB can be updated on-line via FTT requests that are identified by the Switching Module and forwarded to the Master Unit. These requests can ask for addition or removal of messages to/from the SRDB, for example, in the course of on-line reconfiguration procedures, or even changing attributes of existing messages, for example, in the course of dynamic QoS management.

Finally, the Scheduler scans the SRDB on-line, every EC, and builds the list of real-time messages that must be produced in the following EC (EC-Schedule). This list is incorporated into the Trigger Message and sent to the Switching Module where it is buffered. The right transmission instant is defined by the Synchronization Unit, ensuring a precise timing.

V. Experimental Results

This section presents experimental results obtained from an implementation of the switch architecture presented in Section IV. This architecture was implemented in a NetFPGA board [17] that integrates a Virtex-II Pro XC2VP50 FPGA. This implementation uses 42% of total slices and the maximum frequency of operation is 127.13 MHz. Two experiments were carried out integrating different types of traffic in order to validate the switch traffic isolation capabilities.

In the first experiment, we addressed the temporal isolation between the synchronous traffic and the non-real-time traffic. Figure 3 illustrates the setup. The FTT-Master implements a 1ms elementary cycle, 41% of which assigned to the synchronous window, 42% to the NRT window and 16% for the guarding window. The remaining 1% is occupied by protocol overheads, for instance the TM transmission. The guarding window is a period of time at the end of the EC and before the next TM transmission, during which no new transmissions are allowed to start. Making this window as large as the largest packet in the network assures that the following TM transmission will never suffer blocking. In this scenario *Slave 3* sends a 1kB synchronous message to *Slave 2*, which is

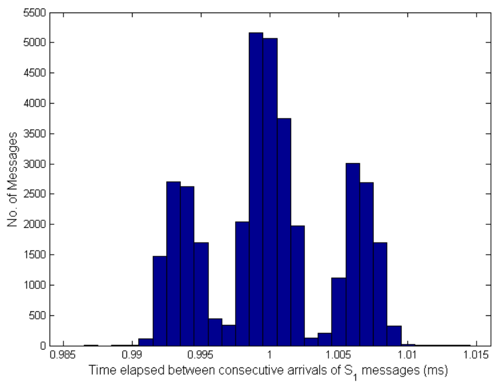

Fig. 4. Jitter affecting the Synchronous message.

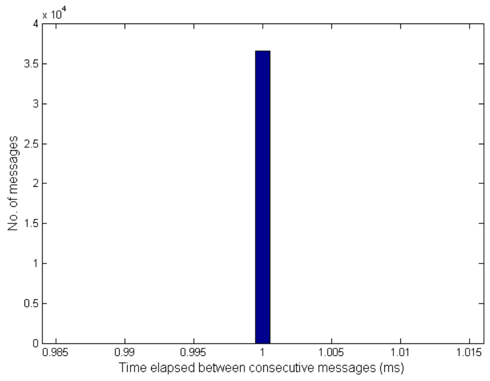

Fig. 5. Jitter affecting the Trigger Message.

scheduled by the FTT Master every EC. *Slave 1* transmits non real-time 800B multicast packets (*NRT*) to *Slave 2* and *Slave 3* separated by the minimum inter-frame gap, thus generating a load close to 100% in the respective uplink. On the other hand, the switch imposes, due to the NRT window length, a maximum of 42% NRT traffic load on the downlinks of *Port 2* and *3*. The exceeding traffic is discarded inside of the switch.

Figure 4 presents the jitter histogram that affected the synchronous message measured on the switch output in *Port 2*. The jitter magnitude was around $\pm 9\mu s$ and is originated in the PC architecture of the sender node. It is substantially inferior to what would be expected if there was interference from the NRT traffic ($\pm 66\mu s$ - transmission time of one NRT packet), which is not the case.

Still in the same experiment, we also measured the jitter affecting the transmission of the TM at the switch output, together with the remaining traffic. Figure 5 presents the histogram of the results achieved, which indicate a very high precision of the TM transmission with a jitter inferior to 50ns, even when some links were used at close to 50% capacity. This clearly shows the strict temporal isolation of the TM with respect to the remaining traffic, thus being a high precision temporal mark available in the system that can, for example, be used for clock synchronization purposes.

The second experiment is illustrated in Figure 6. In this

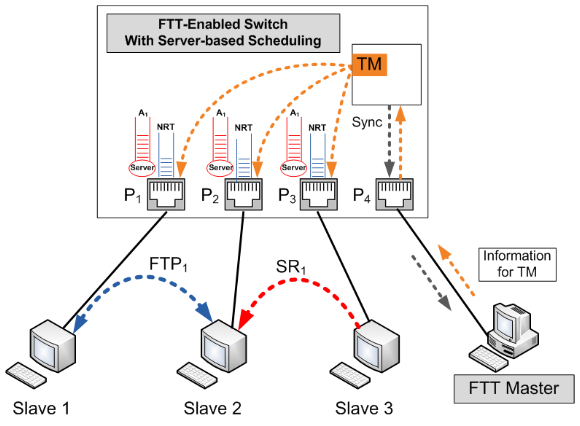

Fig. 6. Experiment 2: asynchronous and NRT traffic.

case, there is no synchronous window. *Slave 3* sends to *Slave 2* an asynchronous real-time stream with 1500B packets and an average inter-arrival interval of $200\mu s$. The respective source switches the stream on and off after a few seconds of silence or intense communication, respectively. Simultaneously, an *ftp* session is initiated to transfer a large file from *Slave 1* to *Slave 2*. The asycnhronous traffic is handled by a *sporadic server* such as proposed in [9]. Servers grant composability in the temporal domain to aperiodic traffic sources limiting their impact in the remaining traffic. The server capacity was set equal to the length of one message, i.e., the time to transmit one 1500B packet, and the replenishment period equal to $400\mu s$. With these parameters, the server is overloaded during the intervals of intense communication. However, the server parameters ensure that any two consecutive packets of this stream are never forwarded to *Slave 2* with less than $400\mu s$ separation. This server can be blocked by: an on-going NRT packet transmission (the worst case is the transmission of a packet with the maximum ethernet length - $123\mu s$); the guarding window ($123\mu s$), used to allow the TM to be transmitted without any interference; and by TM transmission itself ($7\mu s$). Therefore, the possible total interference can be up to $290\mu s$.

Figure 7 shows the histogram of the inter-transmission times of the asynchronous stream measured at the switch output in *Port 2*. As expected, the inter-transmission intervals are seldom below $400\mu s$, with occasional intervals that go down to $360\mu s$. The measurements have been made with a standard Linux PC, with the message time-stamping made at the application layer, inducing a precision around $40\mu s$ [2]. Thus the observed values are compatible with the accuracy of the measurement system. As expected, the shape of the inter-transmission intervals distribution is relatively irregular, due to the possible interference caused by the irregular length of the on-going NRT transmissions, as well as the guarding window and the TM transmission.

Finally, Figure 8 shows the effective bandwidth usage by the NRT and the asynchronous streams during nearly 43 seconds of consecutive operation in the output link of *Port 2*. We

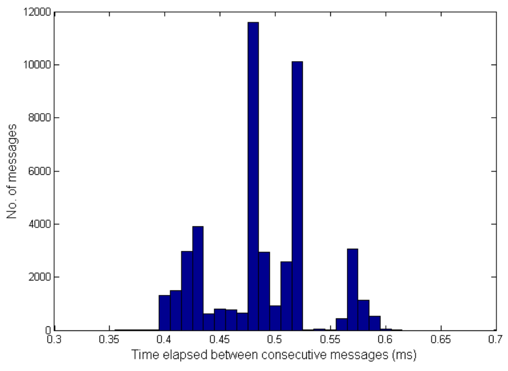

Fig. 7. Inter-transmission time of the asynchronous stream.

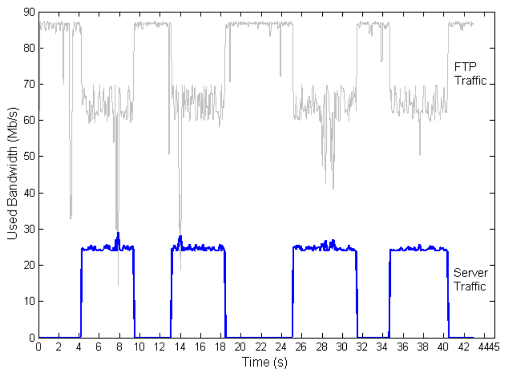

Fig. 8. Effective bandwidth usage.

can clearly see the higher priority given to the asynchronous stream server (lower graph) with respect to the NRT (*ftp*) traffic (upper graph). We also see the background priority level assigned to the *ftp* traffic, which uses what is left available. On the other hand, the bandwidth granted immediately by the server to the asynchronous stream when it starts transmitting intensely, reducing accordingly the bandwidth given to the ftp stream. Note when occurs pauses on the ftp transmission (at $8s$, $14s$ and $29s$) the asynchronous real-time stream uses more bandwidth, because there is no interference. When this happens, the maximum allowed number of server messages per cycle is transmitted.

VI. Conclusions and Future Work

The advent of switched Ethernet has opened new perspectives for real-time communication over Ethernet. However, a few problems subsist related with queue management policies, queue overflows and limited priority support. While several techniques were proposed to overcome such difficulties, the use of standard Ethernet switches constraints the level of performance that may be achieved. On the other hand, the growing availability of powerful programmable hardware devices and associated tools as well as IP cores of communication components opens the way to build customizable devices with properties that are better tuned to specific application domains.

Therefore, following such trend we proposed recently an FPGA-based enhanced Ethernet switch relying on the Flexible Time-Triggered paradigm that enforces strict temporal isolation of three traffic classes, provides seamless integration of non-FTT nodes without causing any interference on the periodic real-time traffic, also provides filtering of unauthorized transmissions at the switch ingress and generates a high precision time mark.

In this paper we presented an architecture for such enhanced switch that exploits the separation between the packet switching activity and the FTT Master functionality, the former being implemented in hardware (Switching Module) and the latter in software (Master Unit). This also allowed reusing the Master of the fully software implemented FTT-SE protocol with minor adjustments and great benefits in modularity and development costs. The paper also described the hardware implementation of the Switching Module, with a focus on the synchronization issues among the asynchronous units inside the switch.

On-going work addresses the full characterization of the overheads incurred by the proposed architecture concerning the interconnection between the Switching Module and Master Unit to derive performance limits. Two other issues will also be addressed, namely the adaptation of the enhanced switch to allow the coexistance of several units in the same synchronization domain and the replication of the Master. The design of specific gateways for connecting different synchronization domains will also be addressed.

VII. Acknowledgments

This project was partially supported by the Portuguese Government through grant SFRH/BD/32814/2006 and project HaRTES - PTDC/EEA-ACR/73307/2006 and by the European Community through the ICT NoE ArtistDesign - 214373. The authors also would like to thank Xilinx Inc. for the donation of the Tri-mode Ethernet MAC soft IP core, as well as ISE and ChipScope Pro FPGA design tools.

References

[1] J. Loeser and H. Haertig, "Using Switched Ethernet for Hard Real-Time Communication," in *PARELEC'04 - International Conference on Parallel Computing in Electrical Engineering*. Dresden - Germany: IEEE Computer Society, Sep. 2004, pp. 349–353.

[2] R. Marau, P. Pedreiras, and L. Almeida, "Enhancing Real-Time Communication over COTS Ethernet Switches," in *WFCS 06 - The 6th IEEE Workshop on Factory Communication Systems*. Turin - Italy: IEEE Computer Society, Jun. 2006.

[3] E. T. Group, "EtherCAT - Ethernet for Control Automation Technology," http://www.ethercat.org, Dec. 2007.

[4] "Ethernet Powerlink - online information," http://www.ethernet-powerlink.org/.

[5] O. D. V. Association, "Ethernet/IP," http://www.odva.org/.

[6] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida, "Designing a Costumized Ethernet Switch for Safe Hard Real-Time Communication," in *2008 IEEE International Workshop on Factory Communication Systems*. IEEE Computer Society, May 2008, pp. 169 – 177.

[7] PROFInet, "Real-Time PROFInet IRT," http://www.profibus.com/pn, Dec. 2007.

[8] TTTech, "TTEthernet," http://www.tttech.com/solutions/ttethernet/, Nov. 2008.

[9] R. Marau, N. Figueiredo, R. Santos, P. Pedreiras, L. Almeida, and T. Nolte, “Server-based Real-Time Communications on Switched Ethernet,” in *CRTS 2008: First International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*. Barcelona - Spain: IEEE Computer Society, 2008.

[10] R. Marau, P. Pedreiras, and L. Almeida, “Enhanced Ethernet Switching for Flexible Hard Real-Time Communication,” http://www.csem.ch/events/RTN06/RTN06.html, jul 2006, rTN 2006, 5th Workshop on Real-Time Networks, Dresden, Germany.

[11] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida, “FPGA-based Implementation of an Ethernet Switch for Real-Time Applications,” in *REC’09: V Jornada sobre Sistemas Reconfigurveis*, Monte da Caparica - Portugal, 2009.

[12] “Automation.Com Magazine,” http://www.automation.com/content/softing-uses-altera-fpga-for-real-time-ethernet.

[13] K. Steinhammer, P. Grillinger, A. Ademaj, and H. Kopetz, “A Time-Triggered Ethernet (TTE) Switch,” in *DATE’06 - Design Automation and Test in Europe*. Munich - Germany: ACM, Mar. 2006, pp. 794–799.

[14] M. Plankensteiner, “TTEthernet enabes the use of Ethernet networks in all applications,” *Embedded Control Europe*, pp. 12–14, 2008.

[15] P. Ferrari, A. Flammini, D.Marioli, and A. Taroni, “Experimental evaluation of PROFINET performance,” in *2004 IEEE International Workshop on Factory Communication Systems*, 2004, pp. 331–334.

[16] J. Feld, “PROFINET - Scalable Factory Communication for all Applications,” in *2004 IEEE International Workshop on Factory Communication Systems*, 2004, pp. 33–38.

[17] NetFPGA, http://www.netfpga.org/, May 2009.