Enhancing Real-Time Communication over COTS Ethernet switches

Ricardo Marau, Luís Almeida, Paulo Pedreiras
DET/IEETA, Universidade de Aveiro, Portugal
{lda,pedreiras,marau}@det.ua.pt

Abstract

Switched Ethernet arose in the last decade as a means to increase global throughput with parallel switching paths, segment the network and create isolated collision domains, thus reducing the non-determinism of the original shared Ethernet. However, COTS Ethernet switches still suffer from a few drawbacks that affect negatively their real-time communication capabilities. For example, there can be overflows in ports queues with consequences across ports, priority levels and virtual LANs, and the number of priorities is too short for any kind of priority-based scheduling. Moreover, switches present extra latencies and jitter due to the need to interpret frame addresses and also due to different internal architectural solutions. In this paper we propose using the Flexible Time-Triggered communication paradigm to enhance the temporal behavior of Ethernet switches with respect to periodic streams. We explain the system architecture and we present a formulation of the global periodic traffic scheduling problem handled by the FTT master. Simulation and experimental results show the advantages of using such synchronized framework.

1. Introduction

Since the early 90s that the interest in switched Ethernet has been growing steadily, being a means to improve global throughput, implement traffic isolation and reduce the impact of the non-deterministic CSMA/CD arbitration of original Ethernet. Switches, unlike hubs, provide a private collision domain for each of its ports, since they are not directly connected to each other. When a message arrives at a switch port, it is buffered, analyzed concerning its destination, and moved to the buffer of the destination port. If that port is busy, the message is queued in memory and transmitted later (Figure 1). Switches may use several queues associated with different priority levels (IEEE 802.1D). The number of distinct priority levels is limited to 8, but many current switches that support traffic prioritization offer an even further limited number. The scheduling policy used to handle the messages queued at each port also has a strong impact on the network timing behavior [4].

Currently, most switches are fast enough handling message arrivals so that queues do not build up at the input ports. However, queues may always build up at the output ports whenever several messages arrive in a short interval and are forwarded to the same destination port. This situation may lead to overloads in which the output queues use up all the available memory, causing further messages to be discarded. Despite not frequent, this situation may easily occur in communication protocols relying on multicast and/or broadcast data dissemination, such as those based on the producer-consumer model (e.g., Ethernet/IP-Industrial Protocol [2]) or on the publisher-subscriber model (e.g. RTPS [8]). Consequently, the use of switches, only by itself, is not enough to guarantee real-time behavior [7].

Moreover, switches add an extra latency when compared to hubs because of the need to process the frame addresses. Furthermore, this latency is variable, mainly due to internal architectural aspects, with significant variations among different manufacturers and models.

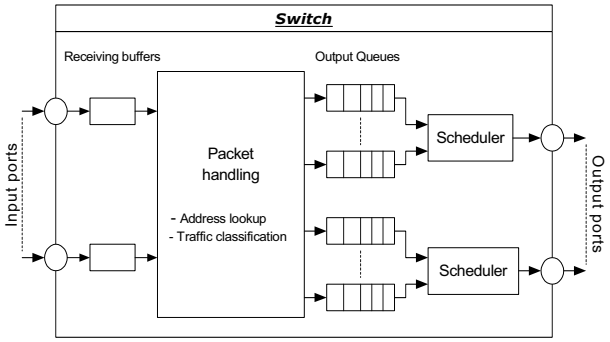

Figure 1: Typical switch internal architecture.

Nevertheless, switches do alleviate the impact of the non-determinism inherent to Ethernet's CSMA/CD medium access control (MAC) and open the way to efficient implementations of real-time communication over Ethernet. In this paper we propose using the Flexible Time-Triggered communication paradigm to enforce global coordination among periodic streams, thus controlling the load submitted to the switch at each instant and avoiding the potential queue overflow problems. The same paradigm has already been used over shared Ethernet to overcome the non-determinism of its MAC leading to the so called FTT-Ethernet protocol [6]. Herein we propose an adaptation of that protocol to micro-segmented networks, i.e., based on switches and with only one station connected to each port, which we will call FTT-SE. As common to all FTT implementations, the advantages of this protocol are the global traffic coordination in a common timeline, the possibility for fast and atomic on-line updates to the set of streams, the possibility to support wide ranges of streams periods and the possibility to enforce any traffic scheduling policy.

This work was partially supported by the European Commission (NoE ARTIST2, IST-2-004527) and by Universidade de Aveiro, Portugal.

In the next section we review the existing approaches to provide real-time communication over switched Ethernet, while in section 3 we address the enhancements that we propose for COTS-based switched Ethernet systems. Therein we briefly present the previous FTT-Ethernet protocol as well as the new FTT-SE, which is fine-tuned to exploit the benefits of micro-segmented Ethernet networks. Section 4 presents the scheduling model beneath the FTT-SE synchronous communication and provides a condition that allows building EC-schedules on-line that respect the cyclic structure of the protocol while bounding the respective memory requirements. Section 5 presents simulation and experimental results that show the effectiveness of the protocol and Section 6 concludes the paper.

2. Real time with Switched Ethernet

This section briefly reviews some of the most relevant techniques to enforce a real-time behavior on switched Ethernet. A deeper discussion can be found in [14].

One approach consists in **enhancing the switch** with traffic control and scheduling capabilities. E.g. Hoang *et al* [3] propose the inclusion of EDF traffic scheduling and on-line admission control inside a switch. The EtheReal protocol [9] presents a similar architecture, also based on a modified Ethernet switch.. PROFINET Isochronous Real Time (IRT), a new PROFINET real-time profile [12], employs a distributed cyclic time-slotting scheme encompassing a deterministic time-triggered phase and an asynchronous phase for non-real-time traffic. Another class of techniques consists in using a **traffic shaper** in each node to limit the burstiness and amount of the load submitted to the network and prevent memory overflows, e.g. as proposed in [10].

Master/slave techniques may also be used to achieve real-time behavior, since in these architectures the master initiates every transactions and thus has complete control of the load submitted to the network. For example, the EtherCAT protocol [12] uses this technique together with specialized switches and an open-ring topology. Another example is the ETHERNET Powerlink protocol (EPL) [1], where a master node explicitly triggers each transaction according to a table schedule.

Finally, s**tandard switched Ethernet infrastructures,** relying on plain COTS switches, network interface cards (NIC) and IP stacks, can also be used, e.g., as in Ethernet/IP [2]. Avoiding overloads and achieving timely behavior in this case requires a careful analysis by the system designer since there are no run-time mechanisms to enforce it. The PEAC protocol [13] is also based on COTS hardware but uses adapted network drivers implementing a cyclic framework, synchronized by a time master, with a TDMA phase for periodic real-time traffic, and an asynchronous phase for non-real-time sporadic traffic, typically IP. This last approach presents a few similarities with the one proposed in this paper but with the TDMA periodic schedule replaced by on-line scheduling.

3. FTT-SE: an enhancement of FTT-Ethernet

As seen in the previous section, there are several ways to achieve real-time communication over switched Ethernet. However, some of them are based on non-standard hardware, a solution that conflicts with some of the key arguments supporting the use of Ethernet in real-time applications (cost, availability, compatibility with general purpose LANs). Therefore, we focus on COTS-based solutions, only, but still aiming at a high level of traffic control towards more predictable timing behavior. Particularly, we propose adapting FTT-Ethernet, originally developed to operate over shared Ethernet, which allows tighter traffic control than existing solutions based on COTS switches, e.g. Ethernet/IP or traffic shaping. The use of the FTT architecture brings other important benefits such as the support for arbitrary traffic scheduling policies, priority levels beyond the eight levels specified in IEEE 802.1D, offsets among streams reducing latency and jitter, on-line admission control and bandwidth management and, finally, avoiding memory overflows inside the switch.

On the other hand, FTT-Ethernet is still a master/slave protocols and, as such, introduces an additional overhead caused by master polls. However, the FTT architecture employs an improved technique, called master/multi-slave, according to which the master addresses several slaves with a single poll, considerably alleviating the protocol overhead.

3.1 Brief review of FTT-Ethernet

FTT protocols organize the communication in fixed duration slots called Elementary Cycles (ECs), with one master message per cycle called Trigger Message (TM), which contains the periodic schedule for that EC. The periodic messages, called synchronous, are synchronized with the periodic traffic scheduler. The protocol also supports aperiodic traffic, called asynchronous, which is managed in the background, in the time left within the EC, after the periodic traffic (Fig 2).

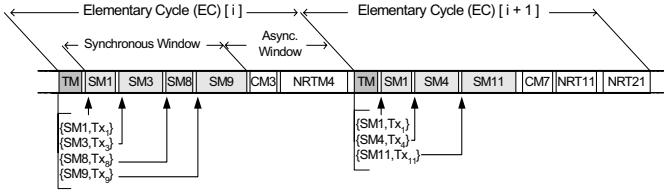

Figure 2. The EC structure in the original FTT-Ethernet.

The traffic scheduling activity is carried out on-line and centrally in the master and the periodic traffic schedules are disseminated by means of the TM. Since the traffic scheduling is local to one node, it is easy to enforce any kind of scheduling policy, as well as perform atomic changes in the communication requirements. This last feature allows for on-line stream admission and removal under guaranteed timeliness as well as on-line bandwidth management. Similarly to the freedom in the traffic scheduling policy, the

specific bandwidth management scheme can also be any. These features are the kernel of the FTT paradigm and are the justification for the *flexible* attribute.

3.2. FTT-SE for micro-segmented networks

Although it is possible to seamlessly deploy FTT-Ethernet over hub or switch-based Ethernet networks, important efficiency gains may be achieved by tailoring the FTT-Ethernet protocol to take advantage of the distinctive features of micro-segmented switch-based topologies, namely the absence of collisions and the existence of parallel transmission paths.

The inherent absence of collisions that results from the existence of private collision domains for each port leads to a noteworthy simplification of the protocol implementation in the slave nodes, which no longer needs to enforce collision-free medium access. Messages are transmitted immediately after decoding the TM, with the switch taking care of the serialization. Consequently the content of the TM itself is also simplified, since the specification of the transmission instants is no longer needed.

On the other hand it becomes possible to take full advantage of multiple transmission paths by abandoning the pure broadcast architecture of FTT-Ethernet as long as we provide the FTT master with information about the nature of the data exchanges regarding the type of addressing (unicast, multicast and broadcast) and which end nodes are involved. With this information the master can compute which messages follow disjoint paths (i.e., non overlapping source and destination nodes) and thus build schedules that exploit this parallelism, increasing the aggregated throughput.

This new feature corresponds to moving from the broadcast-based producer/consumer cooperation model in FTT-Ethernet to a publisher/subscriber scheme in FTT-SE. The master keeps a data structure with the currently existing groups of publisher/subscribers, with the identification of the respective streams and the associated physical addresses and ports. Specific calls issued by the publishers and subscribers allow creating groups and binding nodes to groups. Two different cases must be considered according to the type of switches uses. For non-multicast switches only unicast and broadcast streams can be considered. For true multicast switches the standard Internet Group Multicast Protocol (IGMP, RFC 2236) is used to setup up multicast groups. The binding process for subscribers uses IGMP(Internet Group Management Protocol) messages sent explicitly to the FTT master, which are also snooped by the switch, allowing both the master and the switch to build coherent forwarding tables. The master must be correctly configured to the type of switch being used.

Figure 3 shows the communication system architecture, with the FTT master attached to one switch port and scheduling the transmission of the remaining stations.

3.3. Handling aperiodic transmissions in FTT-SE

Unconstrained aperiodic communication may generate bursts that fill in output queues, leading to long priority inversions in typical FIFO queues and possibly to queues overflow and consequent packet losses. One way to improve this situation is constraining the transmission of aperiodic traffic in the nodes using traffic shaping or smoothing. This way, transmission instants are not constrained but the amount of traffic generated within any interval is bounded.

Alternatively, a more robust and timely but less efficient mechanism is the one used originally in FTT-Ethernet, based on polling. In this case, the transmission instants are adequately planned by the global scheduler but synchronization delays will increase the response times.

However, there is yet a more efficient approach that can be obtained without over constraining the transmission instants of aperiodic traffic, i.e., using switches with two priority levels. In this case, the lower priority level can be assigned to the aperiodic traffic that may be transmitted without being polled, substantially reducing its response time. Nevertheless, even in this case there can be priority inversions in the output ports caused by the non-preemptive nature of packet transmission, but these are bounded to one packet. Moreover, adequate mechanisms are still required to constrain the burstiness of the asynchronous traffic to prevent buffer overflows and consequent interference with the high priority periodic traffic [7][10].

FTT-SE can use any of the mechanisms above, depending on the requirements of each application. The polling approach is more adequate for situations requiring precise timeliness. When the non-preemption blocking is tolerable, the dual-priority approach seems better suited. The asynchronous communication mechanisms within FTT-SE, however, are outside the scope of this paper and will not be further addressed.

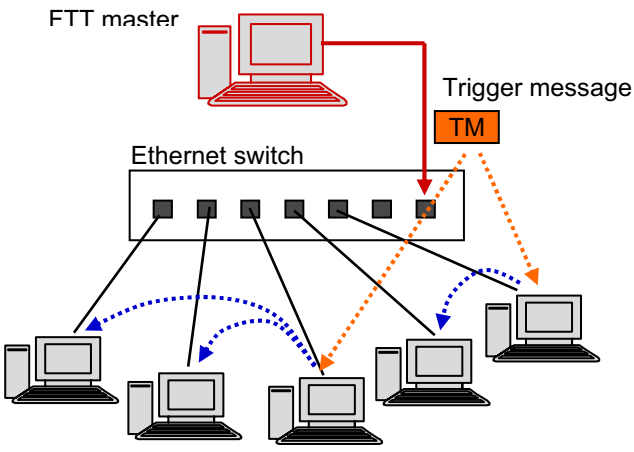

Figure 3. FTT-SE system architecture.

4. The periodic scheduling model in FTT-SE

The scheduling carried out by the FTT master may take into account individual priorities of each message, possibly dynamic priorities, e.g., for EDF scheduling, which are neither restricted nor correlated to the eight priority levels defined in 802.1D. This way, FTT-SE supports strict priority

scheduling within each of the priority levels defined in the standard, thus also including single priority switches (one priority level). The term *strict*, though, can only be applied at a coarse time scale, with a resolution of ECs, since priority inversions within the EC can occur.

4.1. The traffic scheduling model

Concerning the scheduling model, there are N periodic streams (SM_i) which are stored in a structure called *SRT* as shown in (1).

$$STR = \{SM_i: SM_i(C_i,D_i,T_i,O_i,S_i,\{R^1_i..R^{ki}_i\}),\ i=1..N\} \quad (1)$$

C_i is the transmission time of each instance, D_i is the stream deadline, T_i is the stream period and O_i the offset. Both D_i, T_i and O_i are expressed as integer numbers of ECs. Then, S_i is the sender node and $\{R^1_i..R^{ki}_i\}$ is the set of ki receivers for this stream. The calculation of C_i deserves a special note because the protocol automatically fragments large messages in a sequence of packets that are scheduled sequentially by the master. This is particularly useful to transmit regularly large amounts of data such as video frames. The fragmentation threshold is defined per stream.

The set in the *SRT* is scheduled by the FTT master according to any on-line policy, implementing a single queue of ready periodic streams. This queue is used to build the EC-schedule that will be encoded in the TM and broadcast through the switch. The TM will cause all nodes that are senders in this EC to feed the scheduled streams to the switch through a set of M upload links l^u_j. Streams sent by each node are queued locally until they are transmitted. When these streams arrive at the switch they are conveyed, with latency ε, to the output ports and queued for transmission in the M download links l^d_j (Fig. 4).

The transmission of each packet is non-preemptive, as usual, but the transmission of long messages, i.e., those with multiple packets per instance, can be preempted between packets.

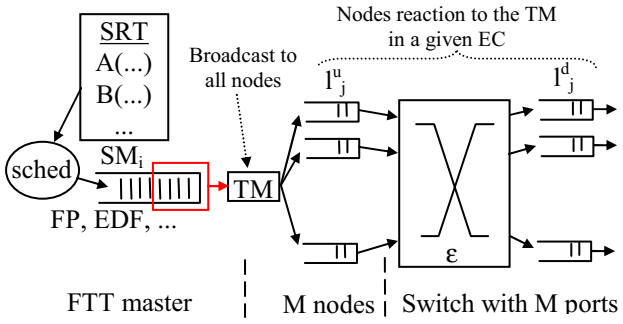

Figure 4. The scheduling model with FTT-SE.

The scheduling problem is two-fold. In one hand, it is necessary to build the EC-schedules so that the transmission of the periodic messages fits within the synchronous window of that EC. On the other hand, it is important to determine schedulability bounds adequate for each scheduling policy in this scheduling model. In this paper we address the former problem, only. The second problem bears some resemblances with multiprocessor scheduling and is part of on-going work.

4.1. Building EC-schedules

Herein we address the mechanisms for constructing the EC-schedules considering the multiple queues associated to the M upload and download links.

The first aspect that must be noted is that, since most current switches are full duplex, transmissions in download links overlap with those in the uplinks but shifted an amount of time corresponding to the switch latency ε. The second aspect is that the initial constraint of limiting the communication activity to the synchronous window, whose maximum duration is *LSW*, means that no link can be used more than *LSW*–ε (Fig. 5). *tr* is the turnaround time, i.e., the time needed by the stations to decode the TM and start their own synchronous transmissions.

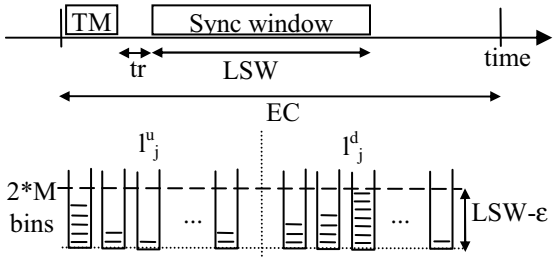

Figure 5. Constraining the synchronous traffic to the synchronous window.

A third aspect is that the transmissions in the downlinks are causal with respect to those in the uplinks and thus must always occur after them, at least an amount of time ε. This means that, if a set of packets arrive close to the end of the synchronous window at a given queue, their transmission might extend beyond the end of that window, even if the total load in that downlink is lower than *LSW*. Therefore, when constraining the traffic in the downlinks to *LSW* it is necessary not only to account for the traffic duration but also for the respective transmission instants.

Therefore, given these aspects, we can use one bin associated to each link during the EC-schedule construction, where the transmission times of the respective streams are accumulated. Then, for the uplinks, considering that the respective transmissions occur immediately after *tr* and in sequence, it is just necessary to check the load in each bin against the *LSW*-ε threshold. In the downlinks, as referred above, we need to consider the finishing instant of the latest transmission (f) and check it against the end of the synchronous window (*LSW*). To determine such instant, we need to keep track of the transmission instants of the previous packets sent in that queue, detect whether there is an overlap and add the respective transmission time (Fig. 6).

Once any of the limits is overridden, the stream that caused it is kept in the ready queue and the EC-schedule is closed and encoded in the TM. The conditions to stop the EC-schedule construction and advance to the next EC are given in (2). The one on top concerns the uplinks and the one below concerns the downlinks.

$$\begin{cases} \max\limits_{j}\left(\sum\limits_{SM_i \in l_j^u} C_i\right) \le LSW - \varepsilon \\ \max\limits_{j}\left(\max\limits_{SM_i \in l_j^d}(f_i)\right) \le LSW \end{cases} \quad (2)$$

The memory requirement in any node μ^n_j or port μ^p_j during the synchronous window, in bytes, is then upper bounded by (3), where r stands for the links transmission rate, considered equal for all links.

$$\max_{j=1..M}(\mu^n_j, \mu^p_j) < (LSW-\varepsilon) * r/8 \quad (3)$$

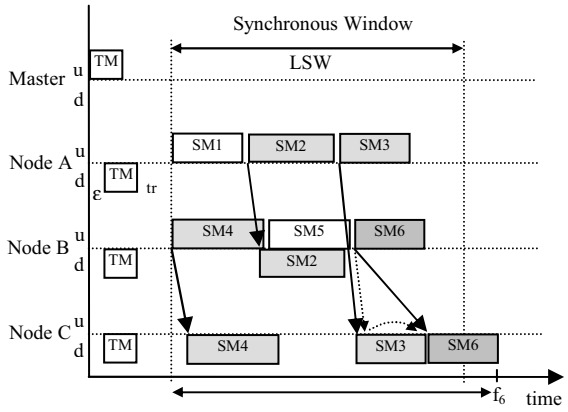

Figure 6. Causality constraint in the downlinks.

5. Simulation and experimental results

In order to test the efficiency of FTT-SE we conducted both simulations and experiments on a real platform. The simulations were made to allow assessing the efficiency in the use of the aggregated switch throughput, while the experiments targeted the verification of the implementation correctness as well as the assessment of the gains, in terms of jitter control, obtained with FTT-SE when compared to a common non-controlled use of a switch.

5.1. Simulation results

The traffic scheduling model used in FTT-SE (Sec. 4) enforces a strict priority order in the scheduling of messages, even if it leads to the insertion of idle-time in the synchronous windows of the ECs. This happens whenever the scheduler moves on to the next EC while there is still capacity left in some links and the ready queue in not empty, yet. Such idle-time introduces a degradation of the efficiency in the use of the switch aggregated throughput.

Therefore, to assess such degradation we carried out several simulations with randomly generated message sets using both RM and EDF scheduling. The operational parameters considered an EC duration of 5ms and a maximum synchronous windows duration (*LSW*) of 85% of the EC, i.e., 4.25ms. The message sets were generated with uniform distributions according to the following parameters: period T in [1,4] ECs, deadline equal to period, single packet messages with transmission time C corresponding to a payload in [1200, 1450] data bytes, *Publisher* chosen from {A, B, C, D, E, F, G, H} and *Subscriber* chosen from {A, B, C, D, E, F, G, H, Broadcast}\{*Subscriber*} and considering three different cases, no broadcasts, 50% broadcasts and 100% broadcasts. The two first cases allowed verifying the capability of using parallel forwarding paths. Moreover, despite the protocol supporting the specification of activation offsets the simulations considered a synchronous release of all messages since we were interested in detecting worst case response times.

The message sets were generated in order to obtain a given utilization value of the most loaded link. Thus, new messages were continually appended to the set until one link reached the predefined maximum load. This generation method was used because it prevents queue overflows.

Each of the generated sets was simulated using both EDF and RM scheduling policies. The simulations were carried out until a deadline was missed or when the macro cycle had elapsed in which case the set was considered schedulable. The ratio of schedulable sets for EDF and RM with respect to the total number of generated sets is shown in Fig. 7. This ratio is shown as a function of the total load submitted to the switch, corresponding to the generated sets. In general, as expected, EDF (bottom) generates more schedulable sets than RM (top) despite the difference being relatively small (less than 10% of the schedulability ratio).

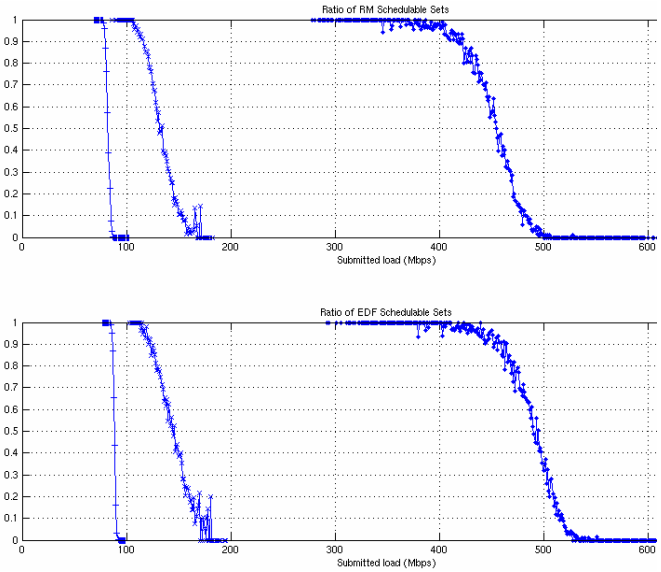

Figure 7. Schedulable sets versus the aggregated submitted load with EDF (bottom) and RM (top).

Also as expected, the switch utilization with broadcast traffic is rather low since parallel forwarding paths are not exploited. There can still be a small level of parallelization between the uplinks and downlinks inherent to full duplex but it is rather limited. Conversely, without broadcasts the switch allows for a substantial increment in the utilization of its aggregated bandwidth. In this case, there were 8 publishers connected to the switch through 100Mbps ports and generating traffic further constrained by the synchronous window with a maximum duration of 85% of the EC. With these circumstances, the maximum aggregated throughput is

680Mbps. The figures show that EDF and RM are capable of successfully scheduling all generated sets with aggregated utilizations of 55 and 50% of that absolute maximum and, in some cases, up to 80% and 73%, respectively. Given that the operational parameters of the simulations are realistic, these values show that FTT-SE is capable of efficiently exploiting the switch aggregated capacity.

The values obtained with 50% broadcasts are intermediate values that illustrate the penalty that they cause. It is interesting to observe that the schedulability ratio grows approximately 50% when moving from the 100% broadcasts to the 50% broadcasts case but when moving to no broadcasts, such improvement is near 450%. This indicates that broadcasts impose a severe penalty on schedulability even if in low number.

5.2. Experimental results

A prototype implementation of the FTT-SE protocol was carried out on the RT-Linux real-time operating system over the Ethernet layer provided by the LNet network stack. Several practical experiments were carried out to verify the correctness of the implementation as well as the level of jitter control. The experimental platform, shown in Fig. 8, comprises eleven computers interconnected by an Allied Telesyn model 8024 Ethernet switch, with 24 ports and 2 priority levels. The computers included one Celeron at 2.2GHz, one Pentium III at 550MHz, six Celeron at 735MHz as well as three SBCs with Pentium MMX at 266MHz. The network interface cards (NICs) used were Intel 8255 and 3Com 3C905B.

ID	C (Bytes)	T(=D) (ms)	$maxJ_w$ (µs)	$maxJ_{wo}$ (µs)
2	1000	1	483	996
7	1000	1	174	984
8	1000	1	170	1003
3	3840	3	92	932
1	3840	4	893	559
4	3840	4	316	446
5	3840	4	1000	521
6	3840	4	137	561
9	1480	8	4132	436

Table 1. Message set used in the FTT-SE experiments.

One of the computers was dedicated to the FTT master, nine computers were data publishers, publishing one message each, and the last computer was a subscriber of all those nine message streams. Only one subscriber is used in this experimental assessment to maximize the messages concurrency in a single link, creating a worst case jitter situation. The message set is detailed in Table 1 and mixes messages with different activation rates as well as single- (messages 5 and 7 to 9) and multi-packet (messages 1 to 4 and 6). The total load submitted by this set is approximately 68,6Mbps.

Concerning the operational configuration of FTT-SE, the EC duration was set to 1ms and the *LSW* to 85% of the EC, i.e., 0,85ms. The traffic scheduling was RM.The same experiments were also carried out with the publishers sending information at the same rate but without the transmission control mechanisms provided by FTT-SE. The interarrival instants of all messages at the subscriber node were recorded for both of these configurations referred to as with and without FTT.

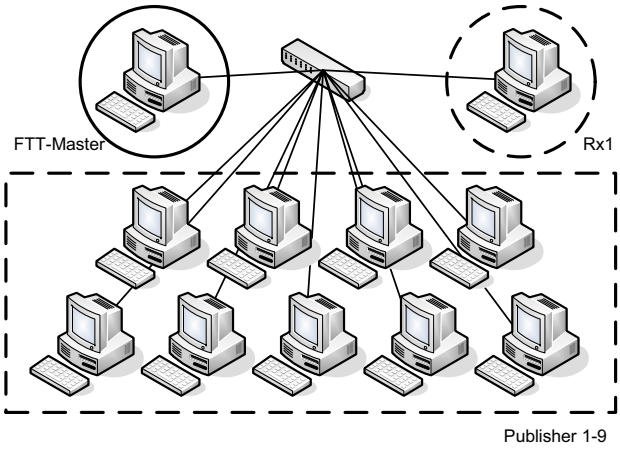

Figure 8. The experimental platform.

Before presenting the experimental results it is important to assess the quality of the measuring tool. In this case we took timestamps directly from the high precision timer (TSC) of the Pentium processors to measure the regularity of the reception of the transmitted streams. These timestamps are, however, influenced by several factors that cause distortions to the measurements, e.g., OS-related jitter, network device drivers and packet switching jitter. This is referred to as *infrastructure jitter* and it is inherent to the measurement process. On the other hand, the purpose of the measurements is to assess the jitter induced by traffic scheduling and to compare it with and without FTT-SE. We call this one, *traffic scheduling jitter*.

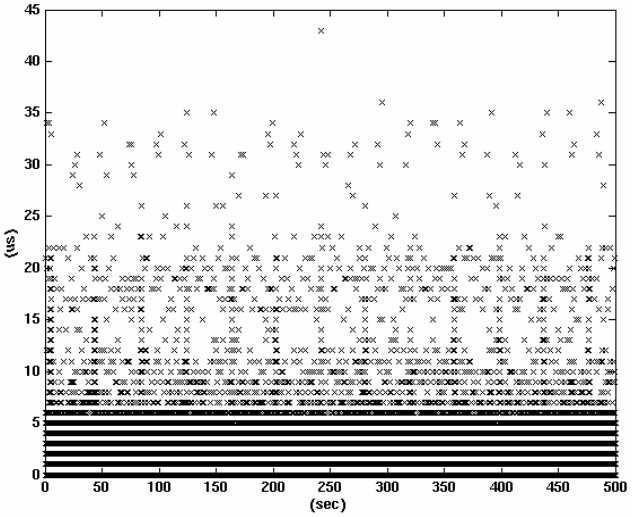

Figure 9. The infrastructure jitter.

To measure the infrastructure jitter we carried out a preliminary experiment with one single but long packet message (1500B Ethernet payload) sent every EC (1ms) by one of the slower computers, during 560s. The transmission

was controlled by FTT-SE. In this case there is no traffic scheduling jitter and thus the jitter observed is only caused by the infrastructure. Fig. 9 shows this infrastructure jitter which is lower than 5µs for 99% of the samples, with a single occasional maximum of 43µs. These values seem very good taking into account that the whole FTT transmission control process is included in this experiment.

After having quantified the infrastructure jitter, we carried out the reception regularity measurements of the streams in Table 1 during approximately 5min. The maximum measured jitter is also shown in the same table in columns $maxJ_w$ and $maxJ_{wo}$, for the cases of with and without FTT-SE, respectively. The results are somehow curious. In the latter case, i.e., in the absence of transmission control, the clocks of the various publishers are not synchronized and the respective relative drifts contribute for the jitter. This effect is clear in the following Figures 10 to 12 that show the interarrival times of messages 1, 7 and 9, and the respective spread. On the other hand, the jitter induced by FTT is almost discrete, caused by interference among the streams in the FIFO queues in each EC.

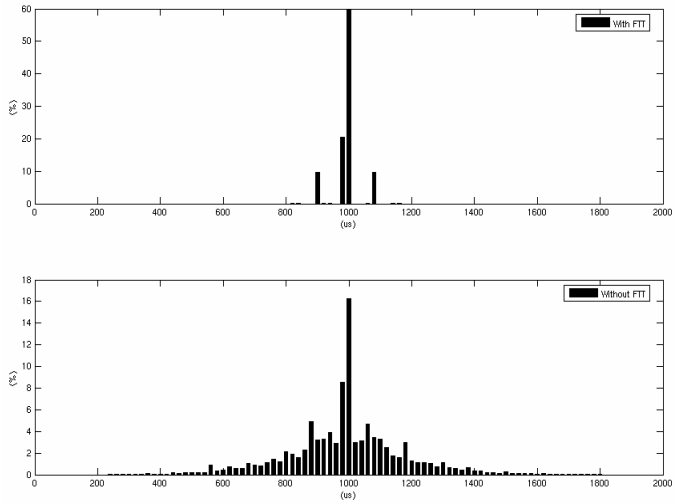

Figure 10. Histogram of interarrival times for message 7.

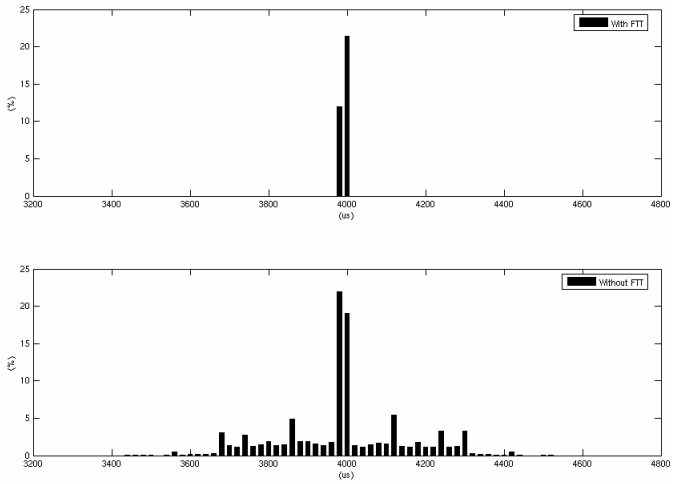

Figure 11. Histogram of interarrival times for message 1.

Moreover, without FTT there are larger bursts of submitted load that seems to be the cause for the higher jitter measured in the faster variables (Table 1). Conversely, with FTT, the submitted load is always within the capacity of the synchronous window of each EC and thus the jitter tends to be smaller for faster variables. Nevertheless, the switch FIFO queues preclude any jitter control within the synchronous window. This is because most of the traffic is submitted at the same time and it becomes extremely difficult to predict the enqueuing order. In some special cases, for example with multi-packet messages 3 and 6, the jitter value is especially low because it refers to the last packet, which is sent later in the EC. This causes that packet to be enqueued after all others and thus in a relatively constant position in the FIFO queue. Message 9 deserves a special note because, as expected, its priority is the lowest with RM and the jitter is very high, corresponding to several ECs.

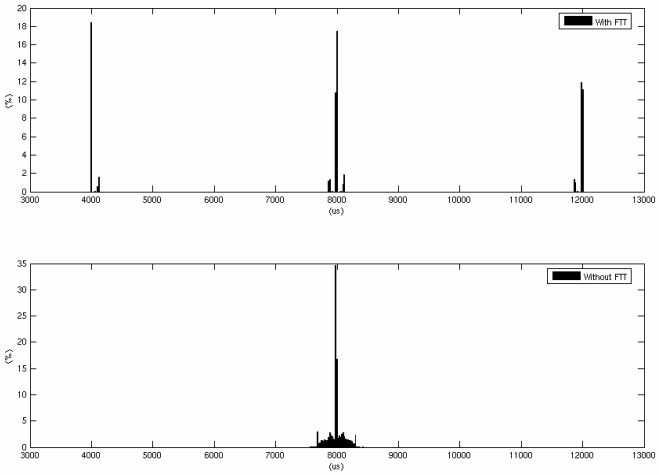

Figure 12. Histogram of interarrival times for message 9.

Finally, the transmission control enforced by FTT-SE may also be beneficial under highly bursty loads. Without the transmission control the level of contention at the receivers might be too high for many network device drivers, which simply crash. The transmission control of FTT-SE prevents this abnormal situation by maintaining the submitted load under manageable levels per EC, obviously at the expense of enlarging the processing time for the same load. Nevertheless, this is sufficient to avoid such crashes and keep the system running. This phenomenon has been observed several times during the practical experiments.

6. Conclusion

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. Meanwhile, several techniques were proposed to overcome such difficulties but they require specific hardware, are inflexible with respect to communication parameters or do not enforce timeliness guarantees. Therefore, in this paper we propose using the FTT paradigm to achieve flexible communication with high level of control to guarantee

timeliness and provide adequate queues management in micro-segmented Ethernet networks. This resulted in the FTT-SE protocol. We briefly explained its mechanisms and provided a result that allows building EC-schedules that respect the duration of the synchronous window. Finally, several simulation and experimental results were obtained that exhibit the efficiency of the proposed approach in terms of using the aggregated switch throughput.

References

[1] ETHERNET Powerlink protocol, available at http://www.ethernet-powerlink.org

[2] Ethernet/IP (Industrial Protocol) specification, available at http://www.odva.org

[3] Hoang, H. Jonsson, M., Hagstrom, U., Kallerdahl, A. "Switched Real-Time Ethernet with Earliest Deadline First Scheduling - Protocols and Traffic Handling". Proc of WPDRTS 2002, the 10th Intl. Workshop on Parallel and Distributed Real-Time Systems. Fort Lauderdale, Florida, USA. April 2002.

[4] Jasperneit, J., P. Neumann. "Switched Ethernet for Factory Communication". Proceedings of ETFA2001 – 8th IEEE International Conference on Emerging Technologies and Factory Automation. Antibes, France. October 2001.

[5] Moldovansky, A., Utilization of Modern Switching Technology in Ethernet/IP Networks, Proc. of the 1st Int. Workshop on Real-Time LANs in the Internet Age, RTLIA'02, Vienna, Austria. Published by Edições Politema, Porto, Portugal, 2002.

[6] Pedreiras, P., Gai, P., Almeida, L., Buttazzo, G. "FTT-Ethernet: a Flexible Real-Time Communication Protocol that Supports Dynamic QoS Management on Ethernet-Based Systems". IEEE Trans. on Industrial Informatics. Vol. 1, N. 3, August 2005.

[7] Pedreiras, P., R. Leite, L. Almeida. Characterizing the Real-Time Behavior of Prioritized Switched-Ethernet. RTLIA'03, 2nd Workshop on Real-Time LANs in the Internet Age, (satellite of ECRTS'03), Porto, Portugal, July 2003.

[8] RTPS (Real-Time Publisher/Subscriber protocol) part of the IDA (Interface for Distributed Automation) specification, available at http://www.ida-group.org

[9] Varadarajan, S., Chiueh, T. "EtheReal: A Host-Transparent Real-Time Fast Ethernet Switch". Proc of the 6th Int Conference on Network Protocols, pp. 12-21. Austin, USA. Oct 1998.

[10] Loeser, J., H. Haertig. "Using Switched Ethernet for Hard Real-Time Communication". Proc Parallel Computing in Electrical Engineering, International Conference on (PARELEC'04), pp. 349-353, September 07 - 10, 2004, Dresden, Germany.

[11] EtherCAT Technology Group. http://www.ethercat.org/

[12] Real-Time PROFINET IRT. http://us.profibus.com/profinet/07

[13] Bonaccorsi, A. L. Lo Bello, O. Mirabella, A. Pöschmann, P. Neumann. A Distributed Approach to Achieve Predictable Ethernet Access Control in Industrial Environments. 5th IFAC International Conference on Fieldbus Systems and their Applications. July 7-8 2003. Aveiro, Portugal.

[14] P. Pedreiras, L. Almeida. Approaches to Enforce Real-Time Behavior in Ethernet. in *The Industrial Communication Systems Handbook*, R. Zurawski (ed). CRC Press, ISBN:0-8493-3077-7, 2005.