We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

11

Analysis of switched Ethernet for real-time transmission

Joan Vila-Carbó, Joaquim Tur-Massanet
and Enrique Hernández-Orallo
Universidad Politécnica de Valencia
Spain

1. Introduction

There has been a growing trend by industrial automation manufacturers of using Ethernet and IP networks at all levels of a factory, thereby replacing fieldbuses and other networks that were typically used in process control. All these networks have been also replaced by AFDX (Avionics Full-Duplex Switched Ethernet) in avionics.

Thus, there is growing interest in improving some features of Ethernet that have been pointed out as important drawbacks for its use in automation and real-time control, such as time-predictability and fault-tolerance. To this aim, industrial network manufacturers have released a new generation of managed switches that allow features like Quality of Service (QoS), configurable topologies such as rings with redundant paths to support loss of connectivity or hardware failures, and virtual private networks support (VPN) to enhance security and network isolation. In this paper we only focus on time-predictability issues.

A number of improvements to Ethernet temporal behaviour have been proposed in the literature. Most of them fall under three possible approaches: suppressing the collisions, reducing their number, and resolving them in a deterministic manner. An example of the first approach is using switches or changing the MAC. Reducing collisions can be done by means of bandwidth reservations. An example of deterministic collision resolution is the CSMA/DCR protocol (Le Lann, 1983). Although this classification is conceptually simple, other classifications that take into account technological issues have also been proposed in the bibliography.

The survey by (Decotignie, 2005) classifies solutions according to their degree of compatibility with standard Ethernet. According to this, solutions can be classified into incompatible, non-interoperable, interoperable homogeneus, and interoperable homogeneus. *Incompatible solutions* include solutions that modify the MAC protocol and cannot be implemented without modifying the Ethernet hardware. They are further classified in (Kurose et al., 1984). Some of these protocols have even been integrated into Ethernet chips (i.e., Intel 82596). An example could be the 100VG-AnyLAN, which is a real-time Ethernet evolution that eventually became standard as IEEE 802.12. *Non-interoperable solutions* include all the solutions that alter the protocol in such a way that a node that is compliant with the new protocol cannot operate in the presence of network nodes that do

not implement the alterations. Examples of this solution are TDMA-based protocols (Pedereiras, 2005) and EPL (Ethernet Powerlink), an open-source solution that is CANopen compatible. EPL can be implemented either on standard or modified hardware. *Interoperable homogeneus solutions* group all solutions that may coexist with standard products. Most of them offer guarantees under the assumption that all devices use the same modifications. They lose their temporal guarantees in the presence of unmodified (IEEE 802.3 compliant) nodes. Examples are REther (Venkatramani & Chiueh, 1994) and solutions that are based on traffic-smoothing. *Interoperable heterogeneus solutions* include those proposals that offer guarantees even in the presence of Ethernet nodes that do not implement the same modifications. Examples are switches with policing features (like EtheReal (Varadarajan, 2001)), or switches with prioritization features (like the ones defined in the IEEE 802.3 extension 801.3p).

The survey by (Felser, 2005) presents another classification of real-time Ethernet solutions according to the layer where modifications are introduced. This classification is illustrated in Fig. 1. The first approach is to concentrate all real-time modifications on top of TCP/IP. In a second approach, the TCP/UDP/IP protocols are bypassed, and the Ethernet functionality is accessed directly on top of Ethernet. Finally, in the third approach, the Ethernet mechanism and infrastructure itself is modified.

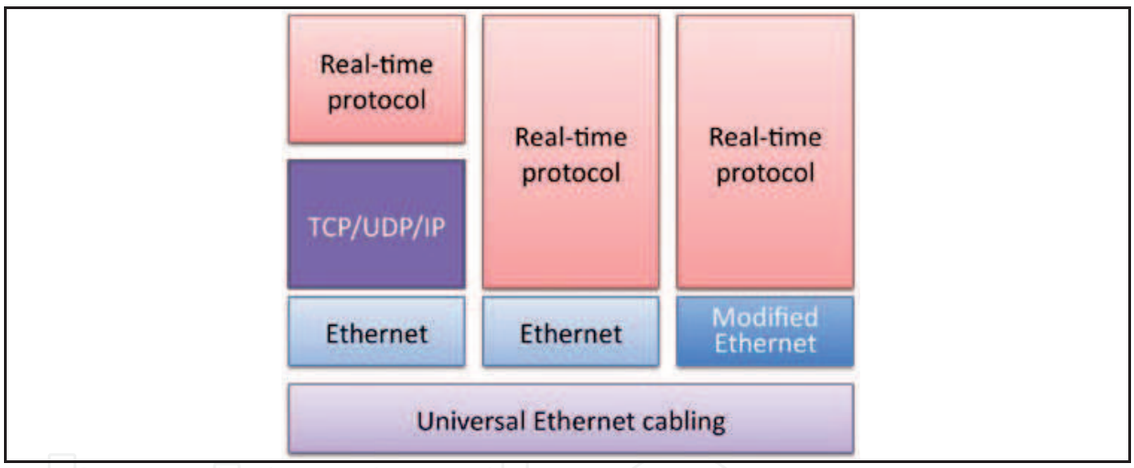


Fig. 1. Structure of real-time Ethernet solutions.

Examples of *solutions on top of TCP/IP* are: the Modbus/TCP standard with its real-time extension of the Real-Time Publisher Subscriber (RTPS) protocol; the EtherNet/IP protocol, that provides real-time communication based on priorities; or the P-NET on IP specification, which allows clients to access servers on IP networks. The *solutions on top of Ethernet* are typically those that impose some kind of time-slicing or time-slotting mechanism in the MAC layer. Examples are Ethernet Powerlink (EPL), TCnet (Time-critical Control Network), PA (Ethernet for Plant Automation) and Profinet. The *solutions that use modified Ethernet* equipment are those solutions that try to provide the typical topologies of factory automation (bus or ring topologies using Switched Ethernet, which is star oriented). This requires a switch in every connected device. The switching functionality is usually integrated inside the field device. The modifications are mandatory for all devices inside the real-time segment, but they allow non-RTE traffic to be transmitted without

modifications. Some examples are SERCOS (SErial Real-time COmmunication System Interface) (Schemm, E. 2004), EtherCAT, and Profinet IO. All the solutions referred by (Felser, 2005) became "IEC proposals for a publicly available specification for real-time Ethernet". They are briefly summarized in the referred survey and detailed in IEC documents (IEC, 2004).

This work focuses on the evaluation of what (Decotignie, 2005) calls *interoperable homogeneus solutions* and, more specifically, solutions based on hardware or software mechanisms that modify the MAC level to make it deterministic and allow the use of the TCP/IP on top of it. Examples are AFDX switches or industrial managed switches with traffic-control features. We also assume that some traffic-control features, like Linux Traffic Control (TC), are available in end-systems.

There are two main approaches for evaluating the considered solutions: *deterministic* techniques, which are based on determining analytical upper bounds; or *statistical* methods, whose goal is to obtain their probability distributions. We take both approaches into account in this work. The chapter is structured as follows: section 2 presents a taxonomy of the main solutions at the MAC level and also provides some analytical deterministic performance bounds, and sections 3 and 4 present the scenarios and the statiscal results of simulations using an industrial managed switch and Linux TC. Finally, section 5 outlines some conclusions.

2. Making Switched Ethernet predictable

The goal of this section is to present the main approaches for making Switched Ethernet deterministic by modifying the MAC level protocol. It also provides *deterministic bounds* for the performance of the proposed mechanisms, and identifies the key issues for performance.

Several methods have been proposed in the literature for obtaining deterministic bounds on network performance. The most widely used is the *Network calculus* theory (LeBoudec & Thiran, 2002). *Schedulability analysis* (Song et al., 2002) (Yiming & Eisaka, 2002) is also a well-known method, but the analysis is usually restricted to periodical flows using rate-monotonic (RM) or deadline algorithms (EDF). Finally, *Model checking* provides another method based on the use of timed automata for describing system temporal behavior. It requires a complete model of the system (which is not always available) and appropriate tools (such as UPPAAL) for evaluating it (Charara et al., 2006).

This section introduces three approaches for making Switch Ethernet predictable at the MAC level and provides their deterministic performance bounds. These are obtained analytically using the *Network Calculus* theory. The approaches considered are:

- *Traffic shaping*: based on bounding the traffic sent to an Ethernet switch.
- *Classes of Service*: based on traffic prioritization.
- *Virtual links*: based on bandwidth reservation.

Some of these methods may require the use of special switches. We refer to a *managed switch* as a switch that allows each of its individual ports to be controlled. Features vary with manufacturers and models. In general, they offer guarantees only if all network interfaces and switches allow the special features, and they may lose their temporal guarantees in the presence of unmodified nodes or switches.

2.1 Traffic shaping

Traffic shaping is a way to improve the predictability of Switched Ethernet that is based on bounding the traffic sent to the network. If all nodes limit their traffic, then it can be shown that the switch delay is bounded. A switch with *policing features* is a switch where non-conforming packets are delayed or eventually dropped. Since most Ethernet switches do not include them as built-in features, nodes need to be cooperative and avoid producing a traffic that exceeds the bounding specifications.

Traffic is usually shaped using a *token-bucket* (or leaky-bucket) specification. A token-bucket shaper with parameters *(r,b)* bounds the traffic to a long-term average rate *r* and a maximum burst size *b*. Practical implementations are usually based on defining a shaping interval T_s. If a shaper is flooded with packets, it transmits a block of size $s \leq T_s{*}r$ in *n* bursts of length $\leq b$ before the end of T_s. Token-bucket shaping is used for Variable Bit Rate (VBR) and non-strictly periodic traffic, like video. However, if all the *n* bursts are equally sized, the token-bucket shaping degenerates into strictly periodic traffic scheduling (with period T_s/n), which is really a particular case of traffic shaping.

The delay of an Ethernet switch using traffic shaping is bounded and can be calculated using Network Calculus. This method can be applied whenever the traffic is limited by an arrival curve. Using a token-bucket traffic specification, the arrival curve at the receiving port *k* of a switch is bounded by function $\alpha_k(t)=r_k{*}t+b_k$.

(Loeser & Haertig, 2004) showed that if the sum of the long-term average input rates for a switch transmit port does not exceed the network bandwidth C:

$$\sum_{k=1}^{N} r_k \leq C$$

then, the Ethernet Switch delay *d* and the buffer size B are bounded, respectively, by:

$$d = \sum_{k=1}^{N} \frac{b_k}{C} + t_{mux} \qquad B = \sum_{k=1}^{N} b_k + C \cdot t_{mux} \tag{1}$$

where t_{mux} is a fixed value provided by the vendor that includes the switching latency and the frame forwarding time.

These bounds were confirmed by several experiments performed using different commercial switches and operating systems (real-time DROPS and Linux using TC). The experiments showed that if the burst factor is low, the switch delay is reduced.

2.2 Classes of service

The concept of *Classes of Service* (CoS) is the basis for the Differentiated Services (DiffServ) architecture (Black et al., 1998). It is intended to allow network applications to share a network under different types of performance guarantees. For example, low-bandwidth sensors (like pressure sensors) and high-bandwidth sensors (like video cameras) produce highly different workloads, so they may also require different CoS with different performance guarantees. The concept of CoS allows the requirements of both traffic types to be jointly fulfilled.

A typical configuration of CoS would be the following:

- *Real-Time CoS*: suitable for low-bandwidth sensors. It would have to provide deterministic, hard real-time guarantees, like bounded deadlines and jitter.
- *Streaming CoS*: intended for video and high-bandwidth sensors and VBR streams. It would have to provide statistical guarantees, like average transmission rates.
- *Best-Effort CoS*: used by the rest of the traffic. No guarantees are provided, although some minimum bandwidth is always allocated.

CoS are accomplished through *traffic marking* and *prioritization,* so different CoS are assigned different marks and scheduling priorities.

Traffic marking can be done in several ways. For example, the managed switch used in this work (Sixnet SL-5MS) supports two mechanisms: the IEEE 802.1p tag priority field, that may be set from 0 to 7; or the ToS (Type of Service) priority field in the IP header, that may be set from 0 to 255. The switch provides four different priorities: urgent, expedited, normal, and background. All the priority tags need to be mapped to these four priorities.

Traffic prioritization can be done using two main scheduling policies: *non-preemptive priorities* or *fair scheduling*. The *non-preemptive priority* policy assures the lowest latency for high-priority data. With this policy, all packets in a higher priority queue are always sent before packets in a lower priority queue. This is done in a non-preemptive fashion, since the transmission of Ethernet frames cannot be interrupted once it has started. With *fair scheduling*, a weighted round-robin algorithm is used. This way, more high priority than low-priority packets get through, but low-priority packets are always assured some bandwidth. Unlike strict priorities, this policy avoids starvation. The Sixnet SL-5MS allows four weights: 1,2,4,8.

In Linux-based end-systems, traffic prioritization can be performed using Linux TC (Traffic Control), which is the Linux kernel mechanism that determines the way in which packets are sent. It offers a large set of functionality for multilevel traffic scheduling. Each traffic class can be assigned a different queuing discipline (Qdisc). There are two types of queuing disciplines: *classless* and *classful*. Classless Qdiscs only use one level of queuing, while Classful Qdiscs may have several. FIFO is the simplest classless Qdisc, but priority (PRIO) or token bucket-policies (TB) are also available.

Hierarchical Token-Bucket (HTB) policies are used in Linux TC to manage *spare capacity*. It is an adaptive mechanism for dynamically redistributing the unused bandwidth based on the concept of *token borrowing*: some given CoS, for example the Streaming CoS, may have children classes used for transmitting different streams. If a child class is sending at a rate below some given ceil-rate and the parent CoS has spare capacity, some other children classes requiring extra bandwidth may attempt to borrow tokens from the parent class in order to increase their transmission rates.

For switches with prioritization, a bound similar to the one presented in the previous section was obtained by (Zhang & Zhang, 2005). This paper assumes that the switch is able to discriminate frames, using the IEEE 802.1p traffic-marking feature, and the traffic is bounded by a token-bucket specification with parameters *(b,r)*.

The switch delay for traffic with priority level *k* is:

$$d_k = \frac{MTU + \sum_{j=1}^{k} b_j}{C - \sum_{j=1}^{k-1} r_j} + t_{mux} \quad (2)$$

where $j=1\ldots k$ are the priority levels that are higher than k, and MTU is the maximum transmission unit.

When a traffic flow crosses a switch with a delay d, the token-bucket parameters are modified as follows: $(r',b')=(r,b+rd)$. Thus, crossing several switches implies an increase of the burstiness b of the flow.

If a flow crosses m switches with delays $d^1,d^2,\ldots,d^m$ and if (r^{i-1},b^{i-1}) and (r^i,b^i) are the input and output flows at switch i respectively, then the end-to-end delay can be obtained by summing all the switch delays:

$$D = \sum_{i=0}^{m} d^i = \frac{b^m - b^0}{r} \quad (3)$$

A key issue when prioritizing traffic is the effect of *limited preemption*. Our experiments show that it is one of the key issues for the variability of transmission times. This effect can be more or less serious depending on the layer where packet prioritization is performed. The best situation is when packet scheduling is performed at the data-link layer, as occurs in most switches. In this case, the maximum non-preemptable data unit is the size of the MTU of an Ethernet frame, which is typically 1500 bytes. With a 100 Mbps Ethernet, this may cause a maximum delay of up to 120 μs. However, the prioritization performed by Linux TC occurs at the network level (IP level), so it schedules IP packets in a non-preemptive way. This means that the size of the IP packet has a strong impact on latencies. IP packet sizes are fixed by the upper layer protocols: UDP and TCP. When using UDP, the RFC791 recommends that hosts only send datagrams that are larger than 576 octets if they have assurance that the destination is prepared to accept them. However, in the end, IP packet sizes are fixed by the applications since UDP does not perform packet fragmentation. The maximum allowable size is 64 Kbytes. Such a packet might cause a delay of up to 5243 μs with a 100 Mbps Ethernet. Fortunately, this is not the usual case. For TCP, there is a handshake when opening a connection where two parties can agree on a MTU, using the TCP MSS option. The IP datagram size should match the minimum frame size of the underlying networks involved in a connection (for example, 1500 bytes in Ethernet and around 4470 in FDDI).

2.3 Virtual links

A *virtual link* (VL) is an abstraction for the partitioning of a network into multiple virtual instances that emulate isolated point-to-point networks. The 100 Mbps link of an end-system can support multiple virtual links. These virtual links share the 100 Mbps bandwidth of the physical link. VLs provide real-time performance guarantees in terms of bounded transmission delay and jitter. That is accomplished using the principle of *bandwidth reservation*.

One of the most powerful implementations of this concept can be found in AFDX (Avionics Full-DupleX switched Ethernet), which is Part 7 of the ARINC 664 Specification for the

exchange of data between Avionics Subsystems (Charara et al., 2006). AFDX is based on the full-duplex switched Ethernet solution. It is redundant, so each end-system is connected to two redundant networks. VLs in AFDX connect end-systems in a unidirectional way. This connection can be 1 source to N destinations. Switches use the VL identification to route the frames statically. Each VL in AFDX is defined using two parameters: the Bandwidth Allocation Gap (BAG) and the largest Ethernet frame (in bytes) that can be transmitted on the VL (*Lmax*). The BAG represents the minimum interval in milliseconds between Ethernet frames. It must be selected from a set of only 8 values *(2^n ms; n=0…7)*. For example, if a VL has a BAG of 32 ms, then Ethernet packets on that VL are never sent faster than one packet every 32 ms. If the VL has an *Lmax* of 200 bytes, then the maximum bandwidth on that VL is 50,000 bits per second (200* 8*1000/32).

The choice of BAG for a particular VL depends on the requirements of the AFDX ports that are being provided link-level transport by the VL. For example, suppose an avionics subsystem is sending messages on three AFDX communications ports that are being carried by the same VL. Let's assume the message frequencies on the ports are 10 Hz, 20 Hz, and 40 Hz, respectively. The total frequency of the combined messages (which are combined into the same VL) is 70 Hz. The average period of the message transmissions is 14.4 ms. Accordingly, to provide adequate bandwidth on the VL, a BAG that is less than 14.4 ms should be selected. The first available bag is 8 ms (125 Hz).

Fig. 2 shows the structure of the VL scheduler of AFDX with two main components: the packet *regulator* and the *multiplexor*.

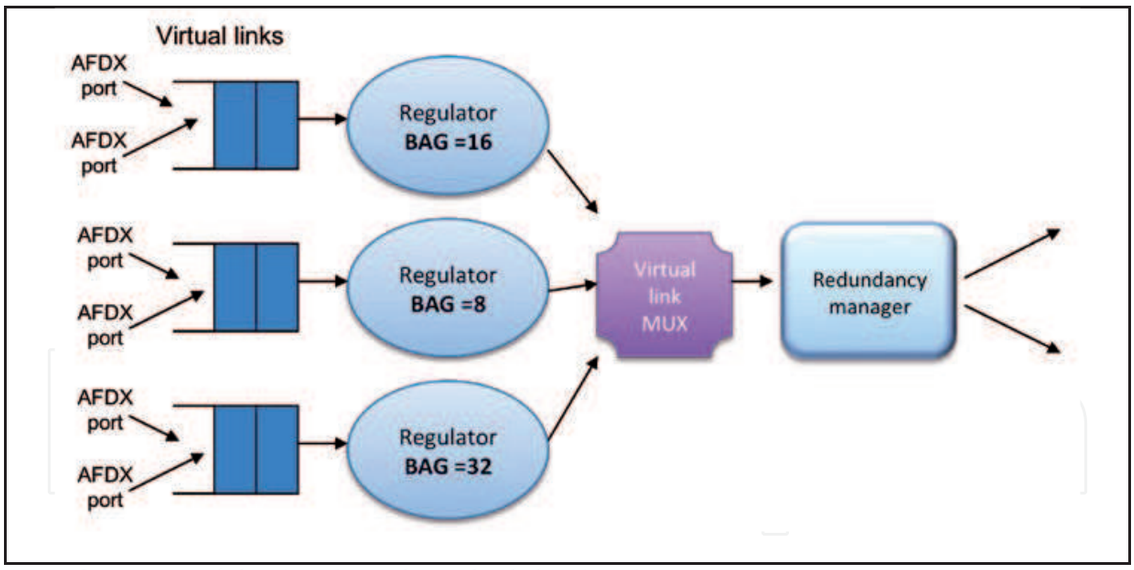


Fig. 2. Structure of the AFDX packet scheduler AFDX.

The traffic regulator performs as shown in Fig. 3. It has three main functions:

- *Traffic-shaping*: traffic is limited and frames are delayed to the next available BAG.
- *Traffic-policing*: frames that do not adjust to the specifications of the VL are dropped.
- *Traffic-filtering*: frames that do not belong to a VL are eliminated (the regulator acts as a firewall).

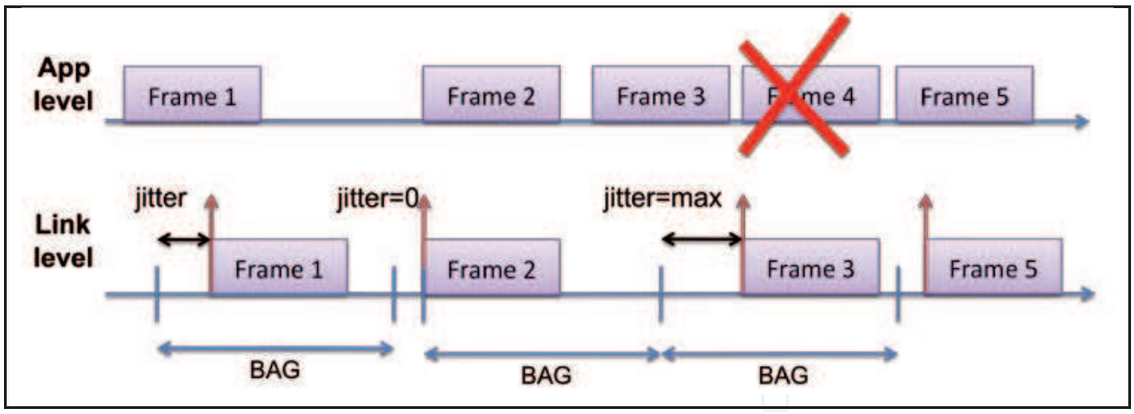


Fig. 3. Traffic regulation and jitter in AFDX.

The source end-system is required to enforce BAG restrictions for each outgoing virtual link. A number of VL scheduling algorithms can be used by the end-system.
Jitter is introduced when the regulator outputs are combined by the multiplexer (MUX); Ethernet frames arriving at input to the MUX at the same time will experience queuing delay. *Jitter* is defined as the interval from the beginning of the BAG to the first sent bit of the frame being transmitted at the virtual link maximum allocated bandwidth. The AFDX standard specifies a bound on this latency:

$$max\_jitter \leq min\left(500\mu s, \sum_{k\in\{VLs\}} \frac{(20 + Lmax_k) \times 8}{C} + 40\mu s\right) \quad (4)$$

where *C* is the link bandwidth, and 20 corresponds to the number of bytes for the preamble of the MAC frames. Note that the second term of the minimum represents the sender delay (d_{snd}) and it can be obtained from equation (1) by substituting b_k with $(20+Lmax_k)*8$ and the multiplexing delay t_{mux} with 40 μs.
The standard does not specify a maximum end-to-end delay bound, but it can be calculated using network calculus, as shown below. The end-to-end delay can be decomposed into the following delays:

- d_{snd}: sender delay. This is the delay between the call to the API sending function until the frame starts being transmitted to the output port. It is caused by the communication stack and VLs multiplexing. It is calculated using equation (1).
- d_{net}: network delay. It is composed mainly of two factors: the switch delay d_{sw} and the propagation delay d_{pro}, which is usually negligible. The switch delay is the sum of the delays at all the switches that a VL traverses. It mostly depends on the number and characteristics of VLs that share an output port and it can be calculated using equation (1) in a way similar to the sender delay d_{snd}. In a *store-and-forward* switch the time for buffering a complete packet would have to be added to d_{sw}. It is about 40 μs for a 500 bytes packet in a 100 Mps Ethernet, but it can be obviated in modern switches.
- d_{rcv}: receiver delay. This is usually a constant value.

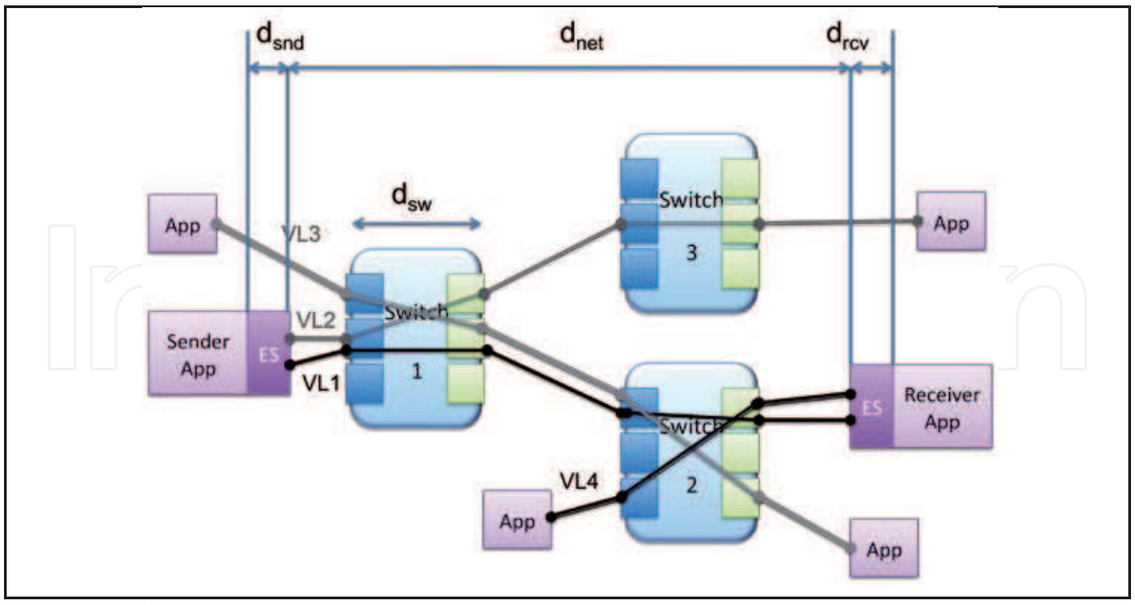


Fig. 4. AFDX sample network.

As an example, consider the network of Fig. 4 with four VLs and three switches. The VLs have the following parameters: VL_1($Lmax_1$=200, BAG_1=2ms); VL_2($Lmax_2$=400, BAG_2=16ms); VL_3($Lmax_3$=500, BAG_3=32ms); VL_4($Lmax_2$=50, BAG_2=4ms). Assume that t_{mux} = 10 μs in the switches, and d_{rcv} = 40 μs.

The end-to-end delay for VL_1 is composed by the sender delay d_{snd}, the sum of delays at switch 1 and switch 2 (d_{sw1} + d_{sw2}), the propagation delays d_{pro}, and the receiver delay d_{rcv}. To obtain the switch delays, the calculations must take into account the VLs that share some given output port with VL_1: VL_3 for d_{sw1}, and VL_4 for d_{sw2}. The sender delay d_{snd} can be calculated analogously, but now VL_1 shares the output link with VL_2. In summary, the end-to-end delay is bounded and is:

$$d = d_{snd} + d_{sw1} + d_{sw2} + d_{rcv} = 88 + 66 + 30 + 40 = 224\mu s$$

3. Evaluation of Switched Ethernet

The goal of this section is to evaluate the effectiveness of the implementations of the concepts of classes of service (CoS) and traffic-shaping on top of Switched Ethernet and Linux TC. This evaluation is based on statistics of real measurements. The experiments evaluate how a high-priority *real-time workload* is affected by a low-priority *background workload*.

3.1 Background

Before presenting the evaluation, we briefly review the main approaches for the *statistical analysis* of network performance. The most classical approach is *Queuing Theory*. However, it may not be a good choice for real-time analysis for several reasons: first, it uses Normal or Poisson inputs, which are not representative of the bursty and periodic characteristics of

real-time traffic (Song et al., 2002); second, the delay calculation is not adequate for several priority levels (Jasperneite et al., 2002). *Simulation* is another classical technique, but it is only valid when it covers a representative subset of the scenarios. Some other statistical methods proposed more recently are *Stochastic Network Calculus* (Jiang & Liu, 2008) and *Histogram Calculus* (Vila & Hernández, 2008) but they require a switch model, which is not always available.

A first evaluation of Switched Ethernet using strict periodic traffic that is based on simulation was done by (Pedreiras et al., 2003). The results showed that when the switches are heavily loaded, they may behave erratically with unpredictable delays, and they may drop high priority packets due to a lack of memory capacity. One of the most comprehensive and interesting evaluations was done by (Scharbarg & Fraboul, 2007). This evaluation was done using several switches, and it showed a comparison of several analytical and evaluation methods. The most accurate results were obtained via simulation. The major deviations from the Network Calculus were obtained with one single switch (up to 70%). A previous evaluation performed by (Jasperneite et al., 2002) also confirmed that analytical methods show noticeable deviations from simulations and the difficulty of characterizing the worst-case scenario. Another evaluation was performed by (Loeser & Haertig, 2004). It makes a comparison of analytical bounds and simulation results for the approach based on traffic-shaping.

3.2 Evaluation platform

The system platform for the evaluations of Switched Ethernet consists of several embedded Pentium III 1200 MHz boards running Linux with kernel version 2.6.23hrt (high-resolution timers). These boards are later referred to as *blade1*, *blade2*, etc. Every board is connected to two Ethernet networks: one is a conventional Ethernet, which is used for the traffic generated by the operating system (NFS and some other services); the other one is an industrial switched Ethernet network that is only used for traffic generated by the experiments. Using two networks guarantees that OS-generated traffic does not interfere with real-time traffic at all.

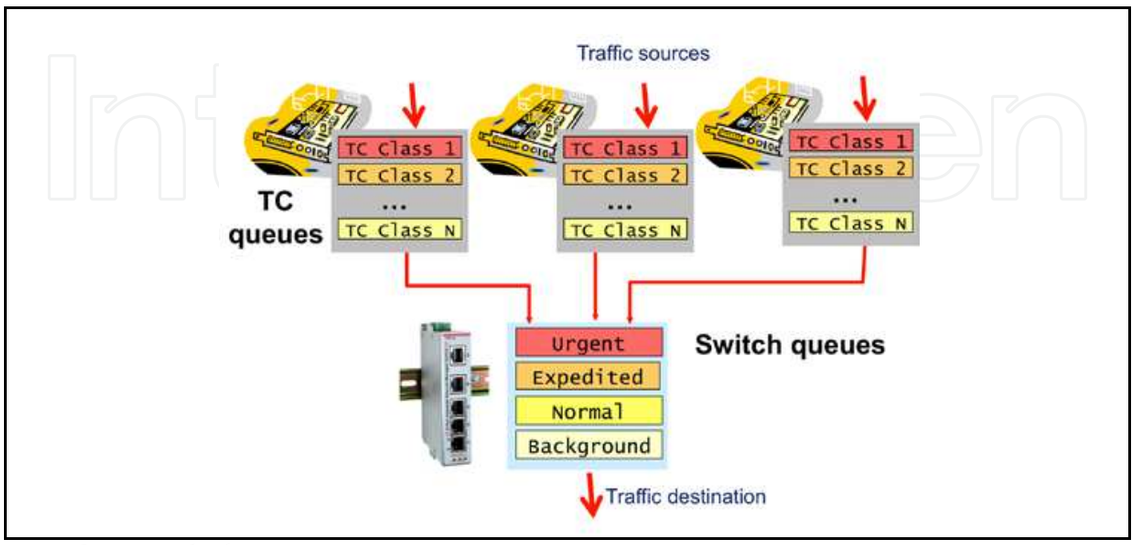


Fig. 5. System platform.

In the industrial network, traffic goes through two message queues (Fig. 5): the Linux TC queues (located at each outbound network interface) and the switch port queues.
The switch used in our benchmark is a Sixnet SL-5MS industrial 100 Mbps managed switch that provides up to four priority queues. The switch has been configured to support three CoS (Classes of Service) based on three non-preemptive priority levels: *Real-Time, Streaming,* and *Best-Effort.*
Linux TC has been configured to provide the same three CoS and, in some cases, to perform token-bucket shaping.

3.3 Metrics

The experiment metrics are the *round-trip delay* and the *delay jitter* of a real-time periodic workload. Messages are transmitted periodically from a sender process to a receiver process that, in turn, bounces the messages back to the sender.
The *round-trip delay* is defined as the time between the sending of a packet and the reception of the return message. It includes the network latency, the switch delay, the queue times, and the operating system overhead. In order to minimize this overhead, the sending and receiving tasks are scheduled at the maximum Linux user priority.
The *delay jitter* is defined as the absolute value of the instantaneous packet delay variation at the receiver process. This is the difference between successive packets. For example, consider that packets are transmitted every 1 ms. Thus, we expect to receive a packet 1 ms after the previous one. However, if the 2nd packet is received 0.8 ms after the 1st packet, the jitter is $|-0.2| = 0.2$ ms.
The evaluated metrics show the average value of the round-trip delay and the jitter and their dispersion in the form of probability distribution.

3.4 Workload definition

The workload consists of traffic flows with different classes of service, which in this evaluation were implemented using strict priorities for both the switch and Linux TC mechanisms:

- The *real-time workload* consists of UDP periodic messages. These have a fixed size of 570 bytes which is approximately the average of the Ethernet frame and a period, which is typically $0.5-1$ ms.
- The *background workload* is a workload whose only goal is to consume some network bandwidth and so "disturb" the real-time workload, possibly affecting its performance metrics. Different background workloads are used. They are parameterized by two factors: *packet size* and *bandwidth utilization*.

The parameters for characterizing the background workload are those that have been reported to have a large impact on Ethernet predictability. Packet size is important because the higher it is, the higher the effect of limited pre-emption is. On the other hand, bandwidth utilization is also very important, since the main recommendation of network manufacturers for preserving the network predictability is to keep bandwidth utilization low. For the background workload, different UDP and TCP workloads were used. However, results for TCP did not differ much from UDP, so only the results for UDP are presented.

The main difference between TCP and UDP could be that in TCP, the packet size is variable and the workload is shaped to an average bandwidth utilization and a maximum burst size. However TCP breaks these packets into fixed size IP packets (fragmentation) so, in the end, the situation is not much different from UDP with fixed size packets.

3.5 Evaluation scenarios

Another purpose of the experiments is to show the effectiveness of each of the two traffic control mechanisms: the managed switch and the Linux TC mechanisms. For this reason, several scenarios have been properly configured. They are shown in Fig. 6.

To evaluate the effect of Linux TC, the real-time traffic is sent between nodes *blade2* and *blade4*, and the background workload is sent from *blade2* to *blade3*. In this scenario, the real-time messages go through the following queues[1] (see Fig. 6, Contention at TC): TC2 → SP4 → TC4 → SP2. Messages from *blade2* to *blade4* contend with the background traffic at TC2, while the messages returning from *blade4* to *blade2* meet the ACKs generated by the background traffic in SP2. According to this, only TC affects the jitter of the messages arriving at *blade4*, while both TC and the switch affect the round-trip delay. However, the effect of ACKs is negligible. Thus, this scenario mostly measures the effectiveness of Linux TC.

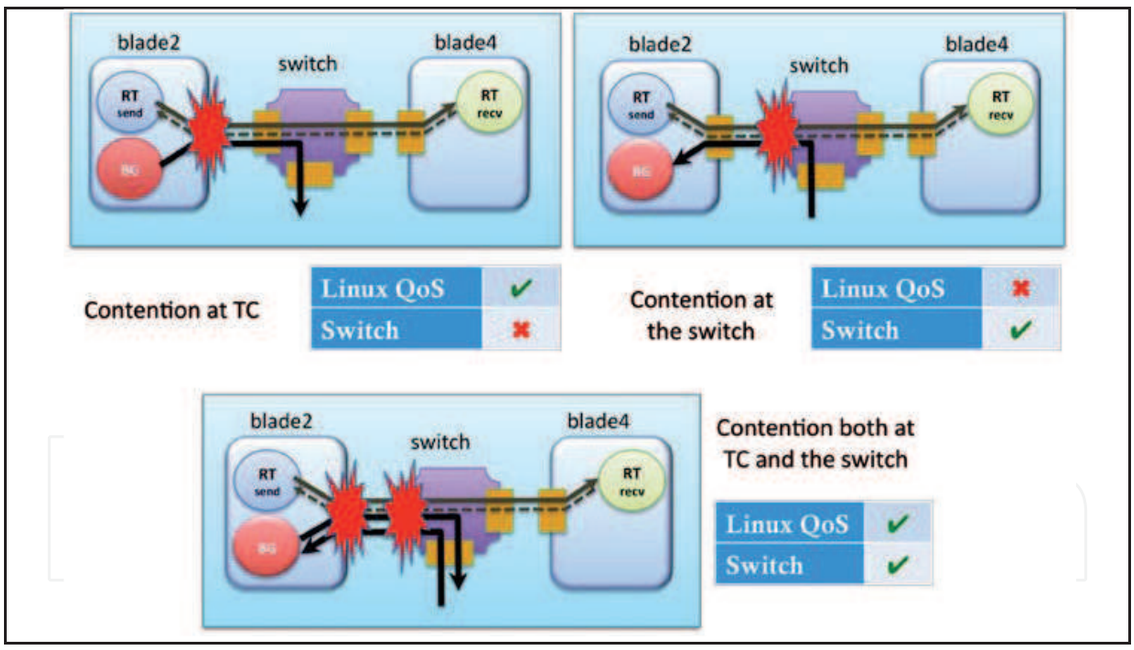


Fig. 6. Evaluation scenarios.

To evaluate the effect of the switch, the background workload is sent from *blade4* to *blade2* (see Fig. 6, Contention at the switch). Now, the real-time messages *blade2* → *blade4* contend with the background traffic in SP2, so this scenario measures the effectiveness of the switch.

[1] TCn stands for the queues of the outbound network interface of node n, while SPn stands for the queues of the switch port of node n.

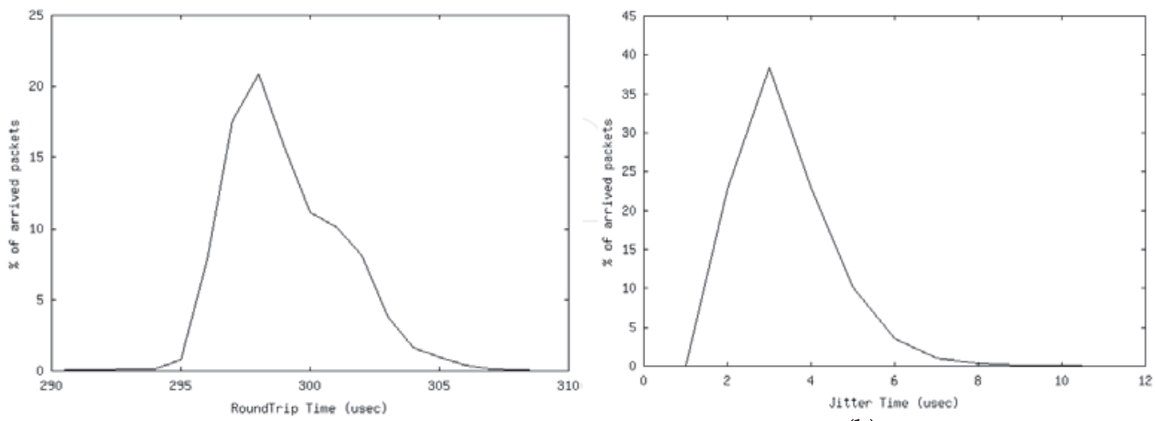
Finally, in a scenario where a background workload is sent from *blade2* to *blade3* and another background workload is sent from *blade3* to *blade2* (see Fig. 6, Contention both at TC and the switch), the real-time messages *blade2* → *blade4* contend with the background traffic in TC2 and SP2, so this scenario measures the combined effectiveness of Linux TC and the switch.

4. Evaluation results

This section presents the evaluation results of the managed Ethernet switch and Linux TC mechanisms and the influence of some parameters, like *packet size* and *bandwidth utilization* on the results. However, in order to be able to properly assess the results obtained in the evaluations, it is convenient to have some idea about the best and worst results that can be obtained. This is done through the analysis of the *idle* and *congestion* scenarios.
All the statistical results presented in this section were obtained by repeating the measures with 10^5 real-time messages.

4.1 Idle and congestion scenarios

The *idle scenario* is a scenario where the real-time workload is transmitted alone, without using any background workload. It is expected to provide the lowest achievable bounds on delay and jitter. The *congestion scenario* is a scenario where a background workload tries to saturate the network by sending packets at the maximum achievable speed. It is expected to disturb the real-time workload as much as possible and yield the worst possible results.
In the *idle scenario*, the real-time workload has a period of 1 ms and a total packet size of 570 bytes. That requires a bandwidth utilization of approx. 4.56 Mbps (4.56%). The probability distribution of the round-trip delays and the period jitter are shown in Fig. 7 (a) and (b), respectively. Round-trip times are in the range [290, 325] μs, with an average value of 300 μs. The standard deviation is very small. The average jitter value is about 3 μs and the maximum is 25 μs. The small variations in round-trips and jitter are mainly due to the unpredictability that the protocol stack (TCP, UDP, IP) and the OS drivers introduce.



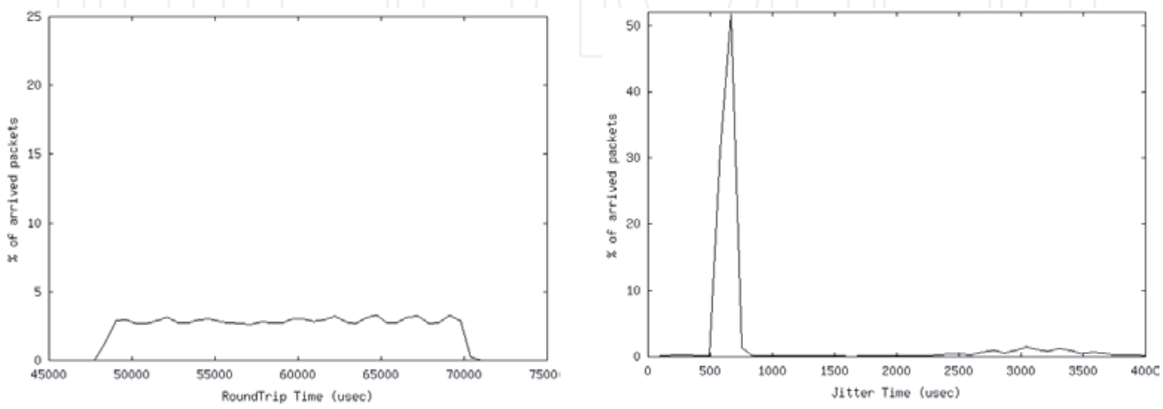
(a) (b)

Fig. 7. Idle scenario: (a) Round-trip, (b) Jitter.

The *congestion scenario* consists of the same real-time workload and a background workload generated by sending a TCP stream between the same nodes at the maximum achievable

speed. The background traffic saturated the outbound network queues of the sending node and the switch port queues of the sending and receiving stations. As a consequence, all the real-time messages suffered serious delays and even packet losses.

Fig. 8 (a) and (b) show the probability distributions of the round-trip delays and jitter, respectively. Both increase by several orders of magnitude compared to the idle scenario: the round-trip time is in the range [48000, 71000] μs, with an average of 58330 μs and a high standard deviation of 6283 μs. The jitter is in the range [0, 21429] μs with an average of 877 μs and a standard deviation of 844 μs.



(a) (b)

Fig. 8. Congestion scenario: (a) Round-trip, (b) Jitter.

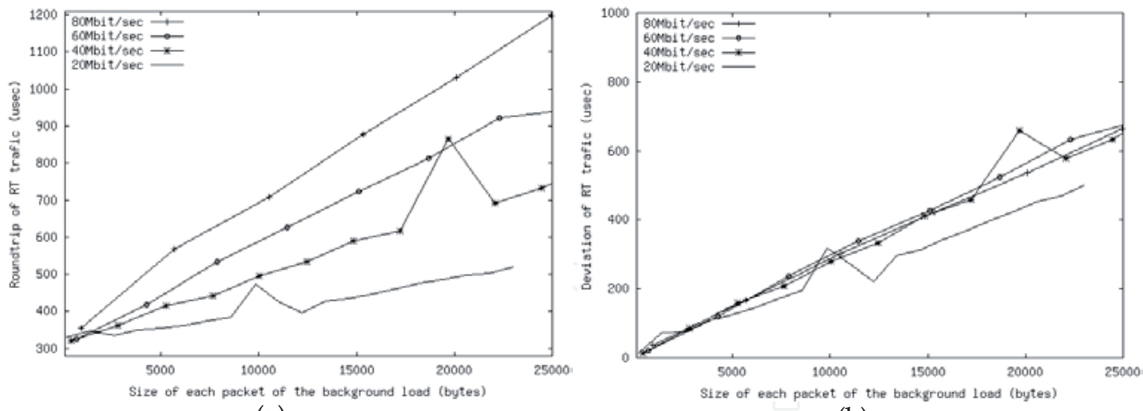
4.2 Effect of the packet size

The goal of this set of experiments is to show that the round-trip delay and jitter are affected by the packet size of the background workload. An increase in the packet size of the background workload is expected to increase the average transmission delay. This is due to the limited preemption effect: low priority Ethernet frames cannot preempt higher priority ones.

In Linux TC, this effect is much more serious since the TC scheduler treats UDP packets as indivisible; therefore, the preemption unit is the UDP packet and not the Ethernet frame. Experiments for measuring the effect of the packet size have been done using UDP because, unlike TCP, it allows setting a fixed packet-size. This is important since even though UDP packets are further fragmented into Ethernet frames, the Linux TC scheduler prioritizes UDP packets and not the Ethernet frames.

In the experiment, the real-time workload consists of UDP periodic messages of 570 bytes with a period of 1 ms. The background workload was characterized by a packet size in the range [0.1, 25] Kbytes and bandwidth utilization in the range [20, 80] Mbps (accomplished by varying the period of the background workload).

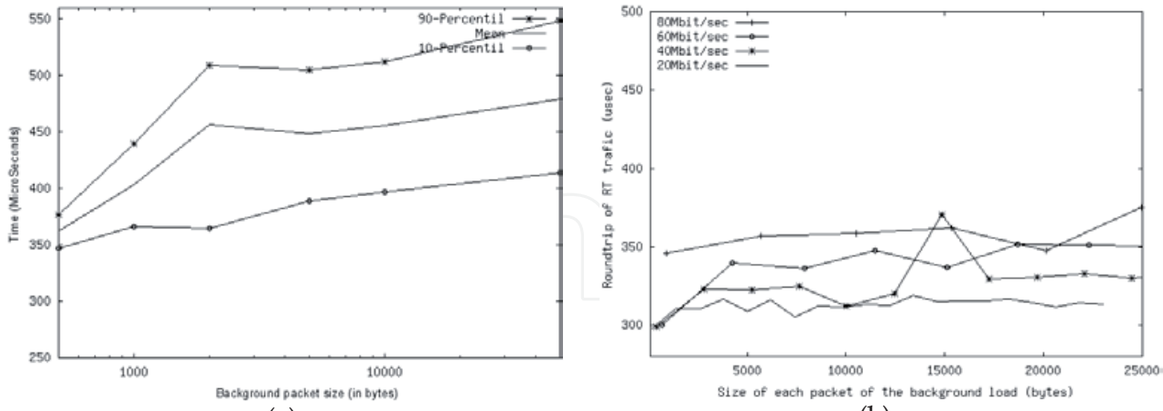
Results show that when Linux TC resolves traffic contention, round-trip delays grow linearly with the packet size, as shown in Fig. 9 (a). For example, when the bandwidth utilization increases from 1 to 80 Mb, it grows from about 300 μs to 1200 μs. The standard deviation (Fig. 9 (b)) also grows linearly up to 600 μs.



(a) (b)

Fig. 9. Effect of the packet size in Linux TC: (a) Round-trip delay, (b) Standard deviation

The switch proved to be much more effective for solving contention. This is shown through another experiment where the switch was used for prioritizing the traffic while Linux TC was used for traffic-shaping. In this experiment, we were aware that the real usefulness of Linux TC was traffic-shaping rather than traffic-prioritizing. The experiment was done using a highly bursty UDP background workload that was shaped, using a token-bucket filter at an average rate of 80 Mbps. UDP packets were fixed-size in a range of [0, 50] Kbytes. Fig. 10 (a) shows the results for the average round-trip times (on a logarithmic scale) and the percentiles 10 and 90. Delays grow linearly up to 1500 bytes (Ethernet frame size) and then remain almost constant about 460 μs, which is not much more than the idle scenario (460 μs). Fig. 10 (b) shows more clearly how the delay remains constant for larger packets and different bandwidth utilizations. The standard deviation also follows the same pattern and remains almost constant at about 45 μs for large packet sizes. Jitter shows similar results.

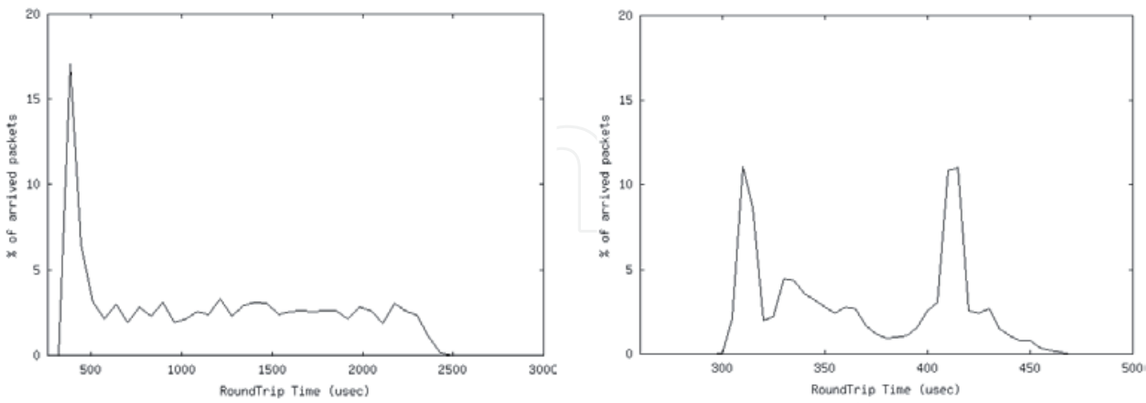


(a) (b)

Fig. 10. Effect of packet size with the managed switch: (a) Logarithmic BW=80, (b) Linear

The typical profile of the statistical distributions of the round-trip delays is shown in Fig. 11. Two cases are shown: when the contention occurs at Linux TC (Fig. 11 (a)), and when it occurs at the switch (Fig. 11 (b)). The distributions correspond to a background workload with a bandwidth utilization of 80 Mbps and an UDP packet size of 23 Kbytes, but they are

all very similar for any packet size. In general, distributions do not differ much, although, in Linux TC, the average is higher. Compared to the idle scenario, the dispersion of these distributions increases moderately.



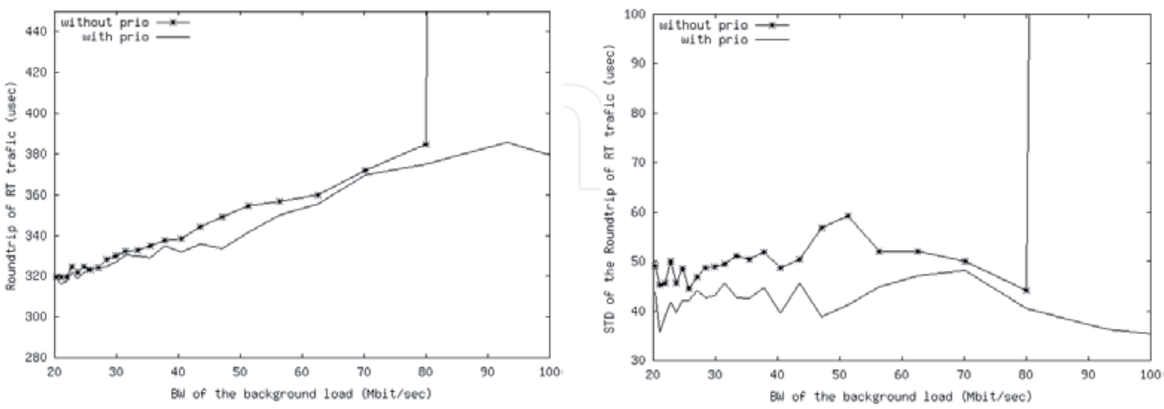
(a) (b)

Fig. 11. Round-trip distribution: (a) Contention at Linux TC, (b) Contention at the switch

4.8 Effect of the bandwidth utilization

These experiments show that increasing the bandwidth utilization of the background workload makes Switched Ethernet clearly less predictable, as reported by most manufacturers. The reason for this is that the average transmission delay increases with the bandwidth utilization because the length of packet queues grows. However, the experiment also shows that prioritizing real-time traffic suppresses this effect even for bandwidth utilizations close to saturation. This is because priorities keep high priority packets from having to wait for other queued packets.

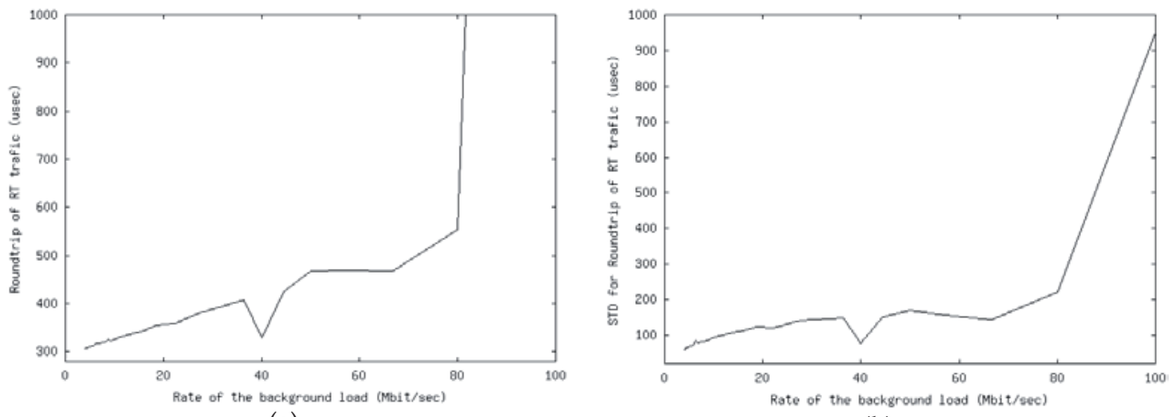
The real-time workload was the same as in the previous experiments. The background workload was a UDP flow with a packet size of 25 Kbytes and variable bandwidth utilization in the whole range (0 thru 100%).



(a) (b)

Fig. 12. Effect of bandwidth utilization with the switch: (a) Round-trip, (b) Std. deviation

The results of Fig. 12 (a) show the effect of prioritizing the real-time traffic in the switch. It shows the round-trip delays with and without prioritization. Prioritization is key for utilizations higher than 80%. Round-trip delays grow slowly from 320 μs to 380 μs for utilizations below 80%. For higher utilizations, priorities keep delays predictable, while lack of them make the system collapse. The standard deviation remains almost constant about 40 μs for the whole bandwidth range. In summary, switch prioritization is key for maintaining the performance close to an idle scenario in the whole range of bandwidth utilizations.
The effect of Linux priorities on this experiment is practically null. Fig. 13 only shows the round-trip delays with prioritization, since the results without prioritization are practically the same: round-trip delays grow from 320 μs to 550 μs when the bandwidth utilization increases from 20% to 80%. They grow dramatically for higher utilizations where results are close to the congestion scenario.



(a) (b)

Fig. 13. Effect of bandwidth utilization with Linux TC: (a) Round-trip, (b) Std. deviation

5. Conclusion

This work has presented a characterization of some approaches for making Switched Ethernet predictable. The solutions studied are based on hardware or software mechanisms that modify the MAC level and allow the use of the TCP/IP on top of it. Analytical performance bounds for these solutions have been presented. Statiscal results have also been presented for the case of an industrial managed switch using traffic prioritization features and end-systems running Linux with the TC packet scheduler.
The results show that the switch prioritization feature is key for maintaining Ethernet predictability with high bandwidth utilizations. On the other hand, the Linux TC priorities have a null effect since they schedule IP packets instead of Ethernet frames. The most useful feature of Linux TC was the traffic-shaping capability. Using traffic prioritization, average network delays and jitter have also shown to increase linearly with the packet size until reaching the Ethernet frame size, remaining constant for larger IP packet sizes. The standard deviation of the experiments also remained bounded through the whole utilization range.
In summary, Switched Ethernet has shown to be a good option for real-time transmission and maintaining a good level of predictability when using traffic-shaping and prioritization.

6. References

Black, D; Carlson, M; Davies, E; Wang, Z & Weiss W. (1998). *An Architecture for Differentiated Services,* RFC-2475. Dec. 1998.

Charara, H.; Scharbarg, J.-L.; Ermont, J. & Fraboul, C. (2006). Methods for bounding end-to-end delays on an AFDX network, *Proceeding of the 18th Euromicro Conference on Real-Time Systems*, pp. 202-212, ISBN 0-7695-2619-5, Prague Czech Republic, 30 July 5-7 2006, Published by IEEE.

Decotignie, J.D. (2005). Ethernet-based real-time and industrial communications. *Proceedings of the IEEE,* Vol. 93, No. 6, (June 2005) page numbers (1118-1128).

Felser, M. (2005). Ethernet-based real-time and industrial communications. *Proceedings of the IEEE,* Vol. 93, No. 6, (June 2005) page numbers (1102-1117).

Kurose, J.; Schwartz, M.; & Yemini, Y. (1984). Multiple access protocols and time-constrained communication. *ACM Computing Surveys*, Vol. 16, No. 1, (Mar. 1984) page numbers (43–70).

IEC International Electrotechnical Commission (2004) *Proposal for a Publicly Available Specification for Real-Time Ethernet,* documents IEC 65C/356a/NP, IEC 65C/341/NP, IEC 65C/360/NP, IEC 65C/359/NP.

Jasperneite, J; Neuman, P; Theis, M. & Watson K. (2002). Deterministic Real-Time Communication with Switched Ethernet, *Proceeding of the 4th IEEE International Workshop on Factory Communication Systems (WFCS02)*, pp. 11-18, ISBN 0-7803-7586-6, Vasteras Sweden, Published by Springer-Verlag. 2002.

Jiang, Y. & Liu, Y. (2008). *Stochastic Network Calculus.* ISBN 978-1-84800-126-8, Ed. Springer-Verlag. Berlin. 2008.

LeBoudec, J.-Y. & Thiran, P (2002).*Network Calculus: A Theory of Deterministic Queuing Systems for the Internet,* Lecture Notes in Computer Science, Vol. 2050. ISBN 3-540-42184-X, Ed. Springer-Verlag. Berlin.

Le Lann, G. (2004). *A deterministic multiple CSMA-CD protocol,* INRIA-Project Score, Internal Rep. PRO-1-002, Jan. 1983.

Loeser, J. & Haertig, H. (2004). Low latency hard real-time communication over switched Ethernet, *Proceeding of the 16th Euromicro Conference on Real-Time Systems (ECRTS04)*, pp. 13-22, ISBN 0-7695-2176-2, Catania Italy, 30 June-2 July 2004, Published by IEEE.

Pedreiras, P; Leite, R & Almeida L. (2003). Characterizing the real-time behavior of prioritized switched-Ethernet, *Proceeding of Proceedings 2nd International Workshop on Real-Time LANs in the Internet (RTLIA 2003)*, pp. 59-62, Porto, Portugal, July 2003.

Pedreiras, P.; Gai, P.; Almeida L. & Butazzo G.C.(2005). FTT-Ethernet: A flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems. *IEEE Transactions on Industrial Informatics*, Vol. 1, No. 3, (Aug. 2005) page numbers (162–172).

Scharbarg, J-L. & Fraboul, C. (2007). Simulation for end-to-end delays distribution on a switched Ethernet, *Proceeding of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007)*, pp. 1092-1099, ISBN 78-1-4244-0825-2, Published by IEEE. 2007.

Schemm, E. (2004). SERCOS to link with Ethernet for its third generation, Computing & *Control Engineering Journal*, Vol. 15, No. 2, (April- May 2004), page numbers (30-33)

Song, Y. ; Koubaa, A. & Lorraine L. I. (2002). Switched Ethernet for real-time industrial communication: modelling and message buffering delay evaluation, *Proceeding of the 4th IEEE International Workshop on Factory Communication Systems (WFCS02)*, pp. 27-35, ISBN 0-7803-7586-6, Vasteras Sweden, Published by Springer-Verlag. 2002.

Varadarajan, S. (2001). Experiences with EtheReal: A fault-tolerant real- time Ethernet switch, *Proceedings 8th IEEE Int. Conf. Emerging Technologies and Factory Automation*, pp. 183–194, Antibes-Juan les Pins, France. IEEE Computer, Society Press, 2001.

Venkatramani, C. & Chiueh T. (1994). Supporting real-time traffic on Ethernet, *Proceedings 15 th IEEE Real-Time Systems Symposium*, pp. 282–286, San Juan, Puerto Rico. IEEE Computer, Society Press, 1994.

Vila-Carbó, J. & Hernández-Orallo, E (2008). An analysis method for variable execution time tasks based on histograms. *Real-Time Systems Journal,* Vol. 38, No. 1, (Jan. 2008) page numbers (1-37), ISSN 0922-6443.

Yiming, A. & Eisaka, T. (2002). Support industrial hard real-time traffic with switched ethernet, *Embedded Software and Systems (ICESS05).* Lecture Notes in Computer Science Vol. 3820. pp. 671–682. ISBN 978-3-540-30881-2. Springer-Verlag. 2002.

Zhang, Q & Zhang, W (2005). Priority Scheduling in Switched Industrial Ethernet, *Proceedings of American Control Conference*, Portland, OR, USA. Jun 2005.

IntechOpen

IntechOpen

IntechOpen

Factory Automation

Edited by Javier Silvestre-Blanes

ISBN 978-953-307-024-7
Hard cover, 602 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Factory automation has evolved significantly in the last few decades, and is today a complex, interdisciplinary, scientific area. In this book a selection of papers on topics related to factory automation is presented, covering a broad spectrum, so that the reader may become familiar with the various fields, and also study them in more depth where required. Within various chapters in this book, special attention is given to distributed applications and their use of networks, since it is one of the most relevant subjects in the evolution of factory automation. Different Medium Access Control and networks are analyzed, while Ethernet and Wireless networks are looked at in more detail, since they are among the hottest topics in recent research. Another important subject is everything concerning the increase in the complexity of factory automation, and the need for flexibility and interoperability. Finally the use of multi-agent systems, advanced control, formal methods, or the application in this field of RFID, are additional examples of the ideas and disciplines that experts around the world have analyzed in their work.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Joan Vila-Carbo, Joaquim Tur-Massanet and Enrique Hernandez-Orallo (2010). Analysis of Switched Ethernet for Real-Time Transmission, Factory Automation, Javier Silvestre-Blanes (Ed.), ISBN: 978-953-307-024-7, InTech, Available from: http://www.intechopen.com/books/factory-automation/analysis-of-switched-ethernet-for-real-time-transmission

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen