VALIDATION OF THE NETWORK CALCULUS APPROACH FOR THE PERFORMANCE EVALUATION OF SWITCHED ETHERNET BASED INDUSTRIAL COMMUNICATIONS

Jean-Philippe Georges, **Thierry Divoux**, **Eric Rondeau**

Centre de Recherche en Automatique de Nancy (CNRS UMR 7039)
Université Henri Poincaré, Nancy 1, Faculté des Sciences
BP 239, F-54600 VANDŒUVRE Cedex, France
<firstname.name>@cran.uhp-nancy.fr

Abstract: The objective of this work is to validate the modelling of a switched Ethernet architecture in order to be able to evaluate the maximum end to end delays. The major interest is to apply these results in an industrial or technical context, where some of the communications are strongly time-constrained. This paper describes a model of an Ethernet switch. It also presents the analytical formulas issued from the Network Calculus theory which upper bound the maximum time for crossing such a switch, and an algorithm to determine the maximum end to end delays of the time-critical messages over the whole network. This work is validated by an experimental application which proves that the worst case estimated by the Network Calculus can be really reached and that this method is not so pessimistic.

Keywords: Network Calculus, performance evaluation

1. INTRODUCTION

Nowadays, Ethernet is more and more used in industrial environment despites its non determinisim due to its medium access mechanism. Several works have been led in order to solve this problem. The first idea was to consider that the data flow is sufficiently quickly handled, compared with the time constraints of the industrial applications. The micro-segmentation with full-duplex switches (Alves *et al.*, 2000) which shift collisions to congestion, combined with the appropriate real-time scheduling and fault-tolerance techniques (Song, 2001) also must enable the use of Ethernet in safety-critical applications with hard real-time constraints. Some other proposals concern the modification or the extension of the access method (Kopetz *et al.*, 1989)(Pedreiras and Almeida, 2002)(Höller *et al.*, 2003), or the traffic smoothing (Kweon *et al.*, 1999) (Caponetto *et al.*, 2002). Finally, some works focus on the topology optimization (Georges *et al.*, 2004*a*). But all these considerations do not prove that all the frames are effectively received under a predefined bound. Our approach is the following. A good knowledge of the topology and the traffic enables by using the Network Calculus theory to determine analytically the maximum end-to-end delays for crossing the network. The objective of this paper is to validate this theory by comparing it to measurements on a real experimentation. This will authorize to know *a-priori* if the time contraints of the application will be always satisfied (even in the worst-case). In other papers we have presented some methods improving the network performances : they integrate the Network Calculus to the topology optimization (Georges *et al.*, 2004*a*) or take into account the priority management (Georges *et al.*, 2004*b*).

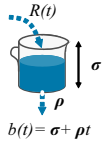
The section 2 details how to model the incoming traffic existing in industrial environment by using the leaky bucket concept. In section 3, a switch modelling is described. The section 4 describes the mathematical process to determine the maximum end-to-end delays for the messages to cross the whole network. These results are compared with the measurement done on a real experimental platform (section 5). They show that the worst-case obtained by the Network Calculus is sometimes reached in the real world. So that the theory is not so pessimistic contrary to the general feeling about this approach.

2. MODELLING OF THE TRAFFIC

An analysis of the traffic handled by industrial networks enables to identify several kinds of frames regarding their periodicity and their temporal deadlines :

- the control traffic is usually periodic and is hardly time-constrained. It depends on the time cycles defined by the control devices such as programmable controllers. It is well known in terms of volume and frequency.
- alarms are sent asynchronously, but have also to be received under a predefined bounded time. Each of them is clearly defined, but the occurrence of their transmission is unknown.
- the traffic to configure industrial devices (programs up or downloading) and the general management traffic (for maintenance, production management) are also asynchronous, but do not have to respect strict hard time constraints. This traffic is unknown in terms of volume and transmission occurrence.

Since the objective is to guarantee some deterministic performances of the network, the incoming traffic has to be bounded. For this, the leaky bucket controller concept (Le Boudec and Thiran, 2001) (figure 1) is used. The burstiness constraint (Cruz, 1991) imposes the traffic generation to be bounded by an affine function $b(t)$, in which a burst value σ is associated to a constant rate ρ.



$$b(t) = \sigma + \rho t$$
$$\forall x, y;\ y \geq x,\ \ x \geq 0, \text{ then}$$
$$R \sim b \Leftrightarrow \int_x^y R(t)\, dt \leq \sigma + \rho (y - x)$$

Fig. 1. The *leaky bucket controller* concept

$b(t)$ can be assimilated ($\sim$) to the upper bound of the number of bits sent at time t, σ is the maximum amount of traffic that can arrive in a burst, i.e. the frames max. length, ρ is an upper bound on the long-term average rate of the traffic and $R(t)$ is the instantaneous rate of the stream.

To take into account the capacity of the links C, the previous affine function is completed with a stability constraint $b(x) \leq Cx$. It means that the arrival of data cannot be greater than the capacity C of the link. So, we have

$$b(t) = \min\{Ct, \sigma + \rho t\} \quad (1)$$

3. SWITCH MODELLING

Switches are modelled as presented on the figure 2. It is the result of a study described in (Georges *et al.*, 2003) in which several models are built and compared**.** It is constituted of a sequence of three basic elements : one multiplexer, one queue and one demultiplexer. In order to take into account the internal speed of a switch, the three following capacities will be used : C, the throughput inside the switch, C_{in}, the throughput of arrival of data on input ports and C_{out}, the output speed.

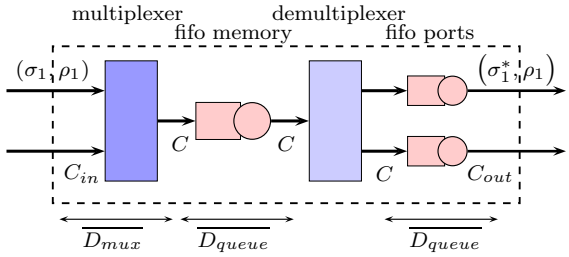


Fig. 2. Switch model

Since the study is about Ethernet networks, the scheduling policy has to be non preemptive. Moreover, it is considered that the components are work-conserving systems, i.e. they cannot have vacations and their forwarding policy is the best-effort. In this paper, the FIFO forwarding policy is chosen in order to respect the fairness of Ethernet: a FIFO memory to model the switching process and FIFO queues to model the transmission function of the output ports. This model has been improved in (Georges *et al.*, 2004*b*) in which it is extended with the priority management.

4. MAXIMUM END-TO-END DELAYS FOR CROSSING A SWITCH

The Network Calculus is a deterministic theory of the queuing systems. The main contributions are (Cruz, 1991),(Le Boudec and Thiran, 2001),(Chang, 2000). (Le Boudec and Thiran, 2001) have used it to improve delay bounds of the differentiated services model of the IETF for different schedulers. This response-time analysis is more and more used to study Ethernet networks temporal performances (Georges *et al.*, 2004*b*), (Jasperneite *et al.*, 2002).
The quantity of data processed in a device is called the *backlog* (Le Boudec and Thiran, 2001, definition 1.2.1) and could be seen as the congestion

value of the device. It is defined at time t by the amount of data which is already arrived minus the amount of data processes at the same time. In the network calculus theory, delays correspond to the time needed to process the backlog. So, upper bounded delays depend in the worst case on the maximum backlog expression. For instance, (Cruz, 1991, theorem 4.1) has defined that the transit delay of a 2 input-ports FIFO multiplexer is upper bounded by

$$\overline{D_{mux,1}(t)} = \frac{\max\left[b_1(t) + b_2(t + L/C_2) - C_{out}t\right]}{C_{out}}$$

For queues, the Cruz's proposition can be extended as shown in (Georges *et al.*, 2003) to :

$$\overline{D_{queue}} = \frac{1}{C_{out}} \frac{(C_{in} - C_{out})}{C_{in} - \rho_{in}} \sigma_{in}$$

Finally, it is assumed for the demultiplexer that the 'routing step' is instantaneously achieved, so that $\overline{D_{demux}} = 0$. The complete study (Georges *et al.*, 2003) gives the temporal behaviour of each component constituting the switched communication system. Therefore, the maximum delay for crossing an Ethernet switch is upper-bounded by:

$$\overline{D_{switch}} = \overline{D_{mux}} + \overline{D_{queue_{(memory)}}} + \overline{D_{queue_{(ports)}}}$$

It is now necessary to determine the maximum end-to-end delays over a complete switched communication system. A method to resolve these upper-bounds is proposed in the next section.

5. EXTENSION : MAXIMUM END-TO-END DELAYS FOR CROSSING A SWITCHED ETHERNET NETWORK

The maximum delay for crossing a switch $\overline{D}$ depends on the leaky bucket parameters : the maximum amount of traffic σ that can arrive in a burst and the upper-bound ρ on the long-term average rate. Consequently, we need to know the (σ, ρ) envelop at each point of the network. As shown by the figure 3, the problem is that we only know the initial arrival curve (σ^0, ρ^0). The other arrival curves, for example after crossing one switch (σ^1, ρ^1) have to be determined.

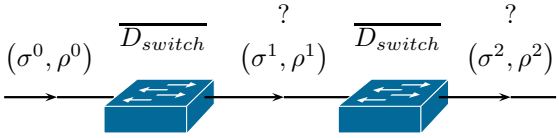


Fig. 3. Burstiness along a switched network.

Our contribution consists in applying the Cruz theory to a network model based on switches modeled as in figure 2. In order to resolve the evolution of the burstiness constraint of a flow, Cruz extends the previous method. For a system for which the arrival of data is constrained by b_{in} ($R_{in} \sim b_{in}$) and for which the delay $\overline{D}$ for crossing the system is finite ($\overline{D} < +\infty$), he shows that the output of data is constrained by b_{out} ($R_{out} \sim b_{out}$) as :

$$b_{out}(x) = b_{in}\left(x + \overline{D}\right) \tag{2}$$

which gives by using (1):

$$\sigma_{out} = \sigma_{in} + \rho_{in}\overline{D}, \quad \rho_{out} = \rho_{in} \tag{3}$$

In the case of the figure 3, the arrival curve after crossing the first switch will be also defined by $(\sigma^1, \rho^1) = (\sigma^0 + \rho^0\overline{D_{switch}}, \rho^0)$.

This analysis is based on the fact that the arrival rate stays constant and that the delay is translated in a supplementary burst (*seconds in bits*). An example will be given in section 5 to illustrate the method. Since the routing strategy is fixed at any point of switched Ethernet networks, all of the previous upper-bounded delays are translated in upper-bounded output burstiness and the difference of burstiness between the input and the output of the network will enable to determine end-to-end delays. All of these points are put together in the following algorithm. Its philosophy is to derive in a first time the delay equation in output burstiness equation, then to compute the output burstiness of each stream at each point of the network and finally to obtain an upper-bounded delay from the end-to-end burstiness difference.
The end-to-end delay resolution steps are:

(1) Identify all streams on each station and determine the initial leaky bucket values.
(2) Identify the route of each stream.
In switched Ethernet networks, paths are determined by the spanning tree.
(3) On each switch, formulate all streams output burstiness equations as described in the equation (3). By convenience, it is suggested to choose the notation σ_i^j, where i is the stream identifier and j is the number of crossed switches. At the beginning, the substream i is represented by σ_i^0.
(4) Define the equation systems under the mathematical form $a_n\sigma_1 + b_n\sigma_2 + ... + z_n\sigma_m = \delta_n$.
(5) Calculate the burstiness values σ_i.
(6) From the equation (3), determine the end-to-end delay with

$$\overline{D_i} = \frac{\sigma_i^h - \sigma_i^0}{\rho_i}$$

where h is the number of crossed switches.

The first step of this algorithm corresponds to our first assumption : even if the incoming traffic is unknown, a simple pessimistic characterisation is possible. For each stream, the amount of data transmitted on the network by the source at time t must be bounded by an (σ, ρ) envelop. Information about traffic or traffic smoother variables can be used. The second step is due to the second assumption : the routing strategy is supposed

fixed. In switched Ethernet networks, it is true since the spanning tree protocol eliminates the loops and gives a single path to go from one node to another. In this part, we only have to look at the spanning tree to learn the switches that will be crossed by each stream. Now the initial conditions have been validated and the algorithm goes on with the formulation of the burstiness for each stream all along the network. In fact, the output burstiness of a stream is written by using the equations presented in the section 4. The equation system written at the step 4, is resolved at the step 5. Now since all burstiness variables are known, it is possible to determine a delay bound between two points of the network. As shown at the last step of the algorithm, the equation (3) is used to determine an upper-bounded end-to-end delay since it gives $\overline{D} = (\sigma_{out} - \sigma_{in})/\rho_{in}$. The end-to-end delay is proportional to the burstiness increasing all along the network.

6. EXPERIMENTAL VALIDATION

In order to confirm the validity of the model, a set of experimental measurements has been carried out. In these experiments, the time to transmit one frame from the sender to the receiver on a switched Ethernet network is measured. These results will be compared to the bound given by the network calculus applied on the model proposed in this paper and also compared with the delays provided by a network simulation tool.

As presented by (Pasztor and Veitch, 2001), one-way measurements of delays are not quite simple since measurement errors are possible. It is mainly due to timing problems as the non synchronisation of clocks and the process scheduling. The first issue is to be sure that the two monitors (sender and receiver) have the same time reference. In a first approach, we choose to execute them on a same computer with two Ethernet interfaces. Consequently, the two processes have to obtain a privileged access to the processor and the concurrence between them has to be controlled. Using the `sched_setscheduler` *Linux* system call, the scheduling policy is set to `SCHED_FIFO` and the priority of the two processes to `sched_get_priority_max`. Moreover, in order to ensure a good scheduling between these two processes, the sender suspends its execution just after sending one frame by calling the `usleep` event up to the next frame. It enables the receiver to quickly react to one frame arrival event. Consequently, we choose to study the model presented in this paper over a first platform (figure 4).

The platform is constituted of one *Cisco Catalyst 2912 XL* switch and three PCs. The network links are always configured at $10Mb/s$ in the full-duplex mode. Communications are generated by

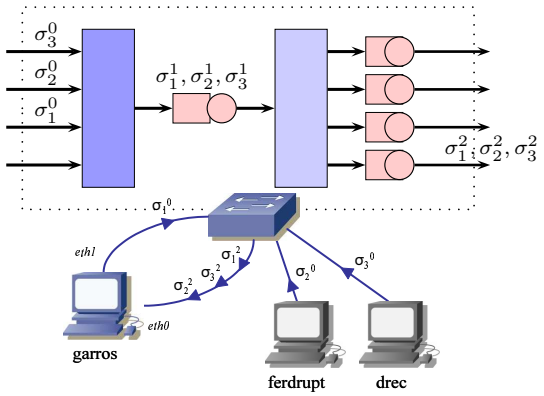


Fig. 4. A first experimental platform.

an algorithm running on the three stations. This algorithm enables to build Ethernet frames : the network output interface, the MAC destination address, the frame length and the inter-arrival time of the frames and to send it on the network. *Garros* periodically sends frames of 72 bytes (the minimum data length in an Ethernet frame) from the first Ethernet interface (`eth1`) to its second Ethernet interface (`eth0`). The period is fixed at 10 ms. In order to load the switch, a background traffic is generated : *ferdrupt* and *drec* send frames of 1526 bytes (the maximum data length in an Ethernet frame) each 5 ms to the `eth0` interface.

We apply now the method steps of the end-to-end delay resolution previously presented. First, we identify the initial leaky bucket values of each stream. There are three streams in this network : from *garros*/`eth1` to *garros*/`eth0` (stream 1, initial leaky bucket $b_1^0(t)$), from *ferdrupt* to *garros*/`eth0` (stream 2, $b_2^0(t)$) and from *drec* to *garros*/`eth0` (stream 3, $b_3^0(t)$). The parameters of the traffic generators give :

$$b_1^0(t) = \sigma_1^0 + \rho_1 t = 72 + 7200t$$
$$b_2^0(t) = \sigma_2^0 + \rho_2 t = b_3^0(t) = \sigma_3^0 + \rho_3 t = 1526 + 305200t$$

Next, we have to identify the route of each stream. Here, paths are simple as shown on the figure 4 : there is only one switch to cross. Moreover, we have to consider the paths inside the switch, i.e. in the switch model (figure 4). At the input, streams which arrive from different input ports are put together into the shared memory by the multiplexer, and they are forwarded on the same output queue corresponding to the interface `eth0` of *garros*. To identify the delay generated by the multiplexer and the queue, we will note the output burstiness of each stream respectively on the multiplexer and the queue : σ_1^1, σ_1^2 for the stream 1; σ_2^1, σ_2^2 for the stream 2 and σ_3^1, σ_3^2 for the stream 3. For each of the basic element of the switch model, we formulate all streams output burstiness. It gives the equation system at the top of this page. These equations are obtained by using

$$
\begin{cases}
\frac{C}{\rho_1}\sigma_1^1 = \left(\frac{\rho_1}{C}+1\right)\sigma_1^0 + \sigma_2^0 + \frac{\rho_1+\rho_2}{C-\rho_3}\sigma_3^0 + \left(L_3 - (\rho_1+\rho_2)\frac{L_3}{C_3} + \rho_2\frac{L_2}{C_2}\right) & (1.1)\\
\frac{C_{out}\left(C-(\rho_1+\rho_2+\rho_3)\right)}{\rho_1\left(C-C_{out}\right)}\sigma_1^2 = \left(\frac{\rho_1\left(C-C_{out}\right)}{C_{out}\left(C-(\rho_1+\rho_2+\rho_3)\right)}+1\right)\sigma_1^1 + \sigma_2^1 + \sigma_3^1 & (1.2)\\
\frac{C}{\rho_2}\sigma_2^1 = \sigma_1^0 + \left(\frac{\rho_2}{C}+\frac{\rho_1+\rho_3}{C_2-\rho_2}\right)\sigma_2^0 + \sigma_3^0 + \left(\rho_1\frac{L_1}{C_1} + \rho_3\frac{L_3}{C_3}\right) & (2.1)\\
\frac{C_{out}\left(C-(\rho_1+\rho_2+\rho_3)\right)}{\rho_2\left(C-C_{out}\right)}\sigma_2^2 = \sigma_1^1 + \left(\frac{\rho_2\left(C-C_{out}\right)}{C_{out}\left(C-(\rho_1+\rho_2+\rho_3)\right)}+1\right)\sigma_2^1 + \sigma_3^1 & (2.2)\\
\frac{C}{\rho_3}\sigma_3^1 = \sigma_1^0 + \sigma_2^0 + \left(\frac{\rho_3}{C}+\frac{\rho_1+\rho_3}{C_3-\rho_3}\right)\sigma_3^0 + \left(\rho_1\frac{L_1}{C_1} + \rho_2\frac{L_2}{C_2}\right) & (3.1)\\
\frac{C_{out}\left(C-(\rho_1+\rho_2+\rho_3)\right)}{\rho_3\left(C-C_{out}\right)}\sigma_3^2 = \sigma_1^1 + \sigma_2^1 + \left(\frac{\rho_3\left(C-C_{out}\right)}{C_{out}\left(C-(\rho_1+\rho_2+\rho_3)\right)}+1\right)\sigma_3^1 & (3.2)
\end{cases}
$$

the section 4 results and the equation (3). The two first lines represent the burstiness evolution of the stream 1. (1.1) corresponds to the multiplexer and (1.2) corresponds to the queue. In the same way, (2.1) and (2.2) are relative to the stream 2 and (3.1), (3.2) to the stream 3. The equation system which describes such a small network shows that for a more complex architecture, the dimension of the system will roughly increase. It is due to the fact that the burstiness evolution is determined at each point of the network and that there is no aggregation of streams. When the burstiness values σ are computed, the next step is to calculate $\overline{D_1} = \frac{\sigma_1^2 - \sigma_1^0}{\rho_1}$ in order to determine the maximum end-to-end delay for the stream 1. The result gives : 3080 μs. This reference of 3080 μs will now be compared with the experimental measures (figure 6(a)). Then, we used the simulation in which a network device such as a switch is modelled by using buffers to represent input and output ports on which are specified both buffer sizes and buffer processing times. It also uses internal buses for moving frames from one port to another one. On this example, the simulation gives a maximum delay of 450 μs. First, if we compare it to the bound provided by the Network Calculus, we can conclude that the over-estimation of the network calculus is very important. But if we compare now this bound to the experimental results, we can remark that even if almost all measures are very inferior to this bound and to the simulation value, some measures tend to the calculus bound. Indeed, 56 % of the measures are inferior to 450 μs (the average of the measures is 664 μs). Some observed delays grow up to 2832 μs, i.e. nearly 6 times more. That shows that a simple capacity analysis (the load of the link between the switch and the network interface `eth0` of *garros* is less than 50 %) is not sufficient to ensure that an Ethernet network will respect the industrial requirements. Moreover, it shows that calculus bounds are closed to the greatest measures since the over-estimation is only about 8 % of the maximum delay measured. Finally, it shows that the background traffic which rises the switch load would increase the delays in the worst-case.

Another experimentation extends the topology (figure 5): two switches have now to be crossed. The station named *garros* is fitted with two Ethernet cards (`eth0` and `eth1`), each one connected to a different switch. The two other stations are directly connected to one different switch. Links are configured at 10 Mb/s in full-duplex.

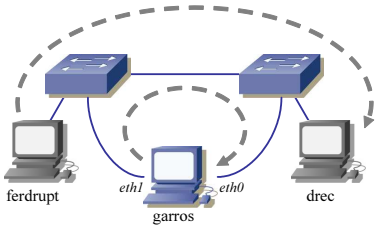


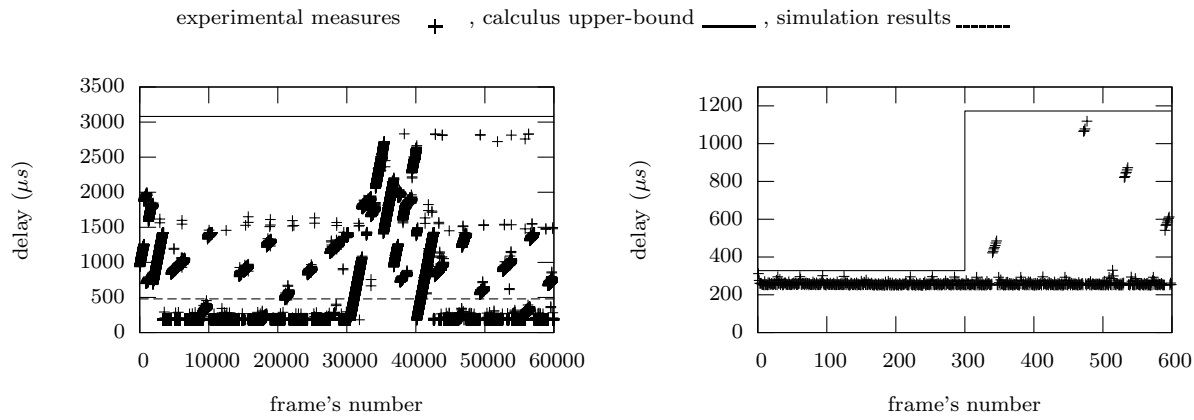
Fig. 5. A second experimental platform.

Garros will send frames of 72 bytes of data each second, from `eth1` to its second interface `eth0` during 10 minutes. After 300s, *ferdrupt* sends frames of 1026 bytes of data to *drec* each 10 ms. End-to-end delays measures are plotted on the figure 6(b). It shows again that the measures confirm the upper-bound given by the network calculus. Indeed, for the first 5 minutes when using the calculus we determine that end-to-end delays will be limited to 328 μs, the measures remain inferior to 312 μs. And for the last 5 minutes (with the background traffic), we compute a bound of 1173 μs when the maximum delay observed is 1119 μs. In fact, the over-estimation is less than 5 % of the maximum delay measured in the two cases. Moreover, the graph shows that our approach enables to take care about real worst situations. During the second part of the experimentation, even if more than 91 % of the measures remain inferior to 328 μs (the calculus upper-bound in the first part), we observe that a simple background traffic could significantly increase the delay as determined by the calculus.

To conclude, during these series of experimentation, the bounds obtained by the calculus theory are validated by the measures. The association of the network calculus and of the switch modelling proposed here, enables to have a good idea of the performance of this kind of network.

7. CONCLUSION

The experimental results presented in this paper show that the computed maximum end-to-end



(a) First platform

(b) Second platform

Fig. 6. End-to-end delays of frames sent by garros (stream 1) in μs

delays are very closed to the worst-case measures. It has two meanings. First, the proposed method really bounds the Ethernet performances even if it is non-deterministic. Secondly, it proves that the use of the Network Calculus for industrial networks is not so much pessimistic. Therefore, this paper is a contribution to the debate on Ethernet based industrial communications.

ACKNOWLEDGEMENTS

This work has been performed in the framework of the FP6 IST STREP project entitled "Networked Control Systems Tolerant to fault : NeCST" (`www.strep-necst.org`) which is funded by the European Community.

REFERENCES

Alves, M., E. Tovar, G. Fohler and G. Buttazo (2000). CIDER - envisaging a cots communication infrastructure for evolutionnary dependable real-time. In: *12th Euromicro Conf. on Real-Time Systems*. pp. 19–22.

Caponetto, R., L. Lo Bello and O. Mirabella (2002). Fuzzy traffic smoothing : Another step towards statistical real-time communication over Ethernet networks. In: *Intl. Workshop on Real-Time LANs in the Internet Age*.

Chang, C.S. (2000). *Performance Guarantees in Communication Networks*. Telecommunication Networks and Computer Systems. Springer Verlag.

Cruz, R. (1991). A calculus for network delay, part I : Network elements in isolation and part II : Network analysis. *IEEE Transactions on Information Theory* **37**, 114–141.

Georges, J.-P., T. Divoux and E. Rondeau (2003). Comparison of switched Ethernet architectures models. In: *IEEE Conf. on Emerging Techn. and Factory Automation*. pp. 375–382.

Georges, J.P., N. Krommenacker, T. Divoux and E. Rondeau (2004*a*). Designing suitable switched Ethernet architectures regarding real-time application constraints. In: *11th IFAC Symp. on Information Control Problems in Manufacturing*.

Georges, JP., T. Divoux and E. Rondeau (2004*b*). A formal method to guarantee a deterministic behaviour of switched Ethernet networks for time-critical applications. In: *IEEE Intl. Symp on Computer Aided Contr Syst Design*.

Höller, R., T. Sauter and N. Kerö (2003). Embedded SynUTC and IEEE 1588 clock synchronization for industrial Ethernet. In: *IEEE Conf. on Emerging Technologies and Factory Automation*. pp. 422–426.

Jasperneite, J., P. Neumann, M. Theis and K. Watson (2002). Deterministic real-time communication with switched Ethernet. In: *4th IEEE WFCS*. pp. 11–18.

Kopetz, H., A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft and R. Zailinger (1989). Distributed fault-tolerant real-time systems. *IEEE Micro* **9**(1), 25–40.

Kweon, S.-K., K.G. Shin and Q. Zheng (1999). Statistical real-time communication over Ethernet for manufacturing automation systems. In: *5 IEEE Real-Time Tech and App Symp.*

Le Boudec, J.-Y. and P. Thiran (2001). *Network Calculus, A Theory of Deterministic Queueing Systems for the Internet*. Springer Verlag.

Pasztor, A. and D. Veitch (2001). A precision infrastructure for active probing. In: *Proceedings of the PAM*.

Pedreiras, P. and L. Almeida (2002). Flexibility, timeliness and efficiency over Ethernet. In: *2002 International Workshop on Real-Time LANs in the Internet Age*.

Song, Y.Q. (2001). Time constrained communication over switched Ethernet. In: *IFAC Int. Conf. on Fieldbus Systems and their Applications*. pp. 152–169.