Université de Liège

Faculté des Sciences Appliquées

Systèmes et Automatique

Institut d'Electricité Montefiore, B28
Université de Liège au Sart Tilman
B-4000 Liège 1 (Belgique)

Esprit Project 5341

OSI95

The OSI95 Transport Service with Multimedia support

***

The Tick-Tock Case Study for the Assessment of Timed FDTs

L. Léonard, G. Leduc and A. Danthine

[LLD 94] L. Léonard, G. Leduc, A. Danthine, **The Tick-Tock Case Study for the Assessment of Timed FDTs**, *The OSI95 Transport Service with Nultimedia Support*, A. Danthine, University of liège, Belgium (ed.), Research Reports ESPRIT - Project 5341 - OSI95 - Volume 1, Springer-Verlag, 1994, pp. 338-352.

Date: July 1994
R94-07

The Tick-Tock Case Study for the Assessment of Timed FDTs

Luc Léonard[1] Guy Leduc[2] and André Danthine
[1] Research Assistant of the Belgian National Fund for Scientific Research (F.N.R.S.)
[2] Research Associate of the F.N.R.S.
Université de Liège, Institut d'Electricité Montefiore, B 28, B-4000 Liège 1, Belgium
Email: leonard@montefiore.ulg.ac.be

The initial purpose of this paper was to design a case study to assess LOTOS-T [MFV 93] which is a temporal extension of LOTOS developed within the ESPRIT II OSI95 project. However, we think that it can be useful for any proposed timed FDT.

It consists of a protocol composed of two entities and an underlying service provider, whose behaviour is mainly based on various timing constraints such as time-out, isochronism, rate-control, …

The selection of the mechanisms was guided by the two following characteristics:

- *Realism: the selected mechanisms have been inspired by similar and existing protocol mechanisms or service facilities, even if we have tried to (over)simplify them in order to focus the case study on the timing constraints.*
- *Temporal modelling facilities: this means that the specificity and the variety of the timing constraints are intended to assess whether timed FDTs have enough power and flexibility to tackle a maximum number of aspects of timed behaviours. Of course, we do not pretend to have explored all possible timed behaviours which may exist in protocols. We have simply tried to cover a broad spectrum of them.*

Keywords: time, FDT, case study, assessment.

1 Introduction

The main reason for the development of FDTs was to obtain a way to describe services and protocols, in a manner that would be non ambiguous, clear and precise. This required them to be expressive enough, to allow the description of the behaviour of any system. Most of them indeed have the expressive power of a Turing machine. This also required them to be designed so that the description of most of the classical mechanisms could be made clear, with light constructs, and without having to use "tricks". However, initially, most of the FDTs were conceived without integrating explicitly any notion of time. In other words, it was not possible to describe easily the influence of time on the behaviour of a system. The recourse to extremely heavy and complicated artificial mechanisms was needed, leading to specifications of an unacceptable complexity.

At the present time, because of an increasing number of protocols including mechanisms where time plays a non negligible role, the real need to remedy to this

problem has already led to the creation of many "timed" FDTs, most of which being either new timed process algebra or quantitative time extensions of well-known asynchronous process algebra. For example: ACP_ρ [BaB 90], $ACP_{\tau\varepsilon}^t$ [Gro 90], ATP (Algebra of Timed Processes) [NRS 90, NiS 91, NSY 91][1], CIRCAL [Mil 85], Estelle [ISO 9074], LOTOS-T [MFV 92], Meije [AuB 84], PADS (Process Algebra for Distributed Systems) [Azc 90], SCCS [Mil 83], SDL [CCITT Z100], TCCS [MoT 90], TIC [QAF 90], TiCCS (Timed CCS) [Wan 90, Wan 91], TiCSP (Timed CSP) [ReR 88, Ree 90], Timed-arc PN [Wal 83], Timed PN [MeF 76], TinLOTOS (Timed-Interaction LOTOS) [BLT 90], TLOTOS [Led 92], Timed LOTOS [LeL 92], U-LOTOS & T-LOTOS [BoL 92], TPCCS [Haj 90, Han 91], TPL (Temporal Process Language) [HeR 91].

As we can see, there exist many proposals, and interesting novelties appear regularly. This situation reflects the diversity of the philosophies envisaged about the way time should be introduced in the FDTs, and the variety of the options explored in order to realize them. An overview and synthesis may be found in [NiS 92], a classification is also proposed in [LeL 93].

But questions thus arise: are all the choices equally valuable? What are the advantages and disadvantages of each one? What are the improvements a given formalism would require? We think that the most instructive way to get pieces of answers to these questions is simply to try to apply these FDTs to a concrete case. The main reason is that this seems to be the only way to determine the difficulties that could arise, and the deficiencies or the needs they will bring to light.

However, until now, most of the researches on timed FDTs did remain within a quite theoretical scope, the most complex example of practical use being often a simple alternating bit protocol (other examples are also treated in [Han 91]). Anyway, applying a timed FDT to the specification of a real system is not the ideal way to assess it. Two main reasons for this:

- It is likely that a real system will confront with only a limited number of problems, among the ones a timed FDT should be able to treat.
- Usually, the majority of the features of a real system are not concerned by timing aspects, and one would waste much time in specifying them.

It thus appeared to us that the definition of a case study, specially designed for the assessment of timed FDTs, would be useful, in particular to the authors of such FDTs. In this perspective, our work of assessment of LOTOS-T [MFV 93], developed within the ESPRIT II OSI95 project, gave us the opportunity to propose the case study we present in this paper. We did not focus it on LOTOS-T, and we think it could be of interest for any proposed timed FDT.

It consists of a protocol that groups together several mechanisms, chosen according to two main criteria:

- Either because they correspond to "classical", often encountered, mechanisms.
- Or because they illustrate the need for some facilities that, according to our knowledge of several formalisms, appear to be among the usual deficiencies we noticed.

---

[1] A new formalism based on ATP but with some different properties (e.g. the possibility of a dense time domain) is presented in [NSY 92]

The well-known time-out or the rate control are examples of the first type mechanisms. Also, the present development of the R.N.I.S. confronts us with the specification of isochronous systems, with discrete variations of the period.

Among the special difficulties, let us point out:

- The problems related to the definition of non-deterministic delays. Such delays are an useful abstraction, for instance to describe a propagation time into a transmission medium.
- The possibility for the temporal data to be treated and manipulated as any other data type. Many mechanisms (for instance congestion avoidance mechanisms) adapt their behaviour according to temporal information they deduce from the observation of their environment.
- The freedom to treat time as a dense or as a non dense domain. Some timed FDTs are restricted to a discrete time domain. Using a discrete time domain requires a careful definition of the grain of time of the specified system, in order to be able to express any timed constraint w.r.t. this basic unit of reference. This turns out to be inconvenient in a practical design, in which the grain of time is tightly bound to the abstraction level of the specification. Therefore, when a process is refined, this may require the selection of a smaller grain of time, and the whole specification needs to be rewritten for consistency. Such problems can be avoided by using a dense time domain, like the rational numbers.
- And, last but not least, as this request usually appears to be both the most fundamental one and the most difficult to tackle: the ability to express some constraints on the occurrence of interactions between concurrent processes. This capability is, in particular, of prime importance to allow for the specification in a structured style (see next chapter).

Although we have tried to keep it realistic, this case study is mainly focused on the timing aspects, and we have lightened it from the details that were not relevant to this concern.

2 Expressiveness and Style

Before describing the case study, we would like to come back to an important point, and bring some more ideas about it: What is a "good" timed FDT? And thereby, what could be a good specification of this case study?

As we already said above, one should not think that the (theoretical) expressive power is the sole criterion to assess the capabilities of an FDT. This is a very naive approach which does not give any interesting results. The reason is that most *non*-temporal FDTs (e.g. Basic LOTOS, CCS, … but not Petri Nets) have already the maximal expressing power, i.e. they can express Turing machines, and thus are able in particular to express quantitative timed behaviours.

In other words, every timed FDT permits probably to describe the case study we present here. But what we really need is an FDT that provides compact and readable solutions to these problems. The keywords are therefore “flexibility”, “facility”, “readability” and “modularity” rather than “(theoretical) expressive power”.

Another keyword is also often encountered in this context: the *style* of a formal description. In LOTOS, four styles have been proposed [VSS 88], each one presenting its own advantages. Among them, the most useful are usually the constraint-oriented style and the resource-oriented style. The constraint-oriented style (for more information, see [Bri 89]) is the most important during the first stage of the design because it is structured and extensional. That means that it allows the production of abstract, modular and easily extensible specifications, which are fundamental properties at this stage. The resource-oriented style is the one that best allows the structured description of possible implementations. This concept of style could be extended to some other FDTs which have a parallel composition operator à la LOTOS. The idea is that one should be able, when it is possible, to choose the most convenient style in accordance with the aim of the specification. In particular, the structure brought by the styles appears to be of paramount help, if not simply mandatory at all, to master the complexity of large systems .

So, we think that the objective when trying to extend FDTs with time, must not simply be to allow clear and light descriptions of timed behaviours but also to provide the capability to choose the style of them. Trying to describe the case study according to different styles is thus a good test in order to determine whether this requirement is met or not. According to our own experience, the most difficult problems would be faced with the constraint-oriented style.

3 The Tick-Tock Case Study

The Tick-Tock system is a protocol composed of two entities and an underlying service, which we will simply call *sender*, *receiver* and *service*. Let us recall that although we have tried to keep it realistic, it is lightened from the details that are not relevant to timing aspects.

It will first be defined in a simpler version, limited to its main features, in order to exhibit the essential constraints that need to be described. Then, as options, some more complex mechanisms will be presented, which could be encountered in practice and create new problems.

The three elements of the system will be described mainly as "black boxes", through the different constraints imposed to their observable behaviours, and without any detail on their internal organization. This suggests to try to specify them, as far as possible, in a structured and abstract style (e.g. the constraint-oriented style in LOTOS), in order to reflect at best the separation of timing concerns. If some a priori independent mechanisms cannot be specified independently of each other but must be combined in a single process, this should be considered as a shortcoming or, stated otherwise, as a lack of support of modularity. Such intertwinings would be especially annoying for the description of real (i.e. large) protocols. But, of course, one is invited to check that specifying in the other styles is possible too.

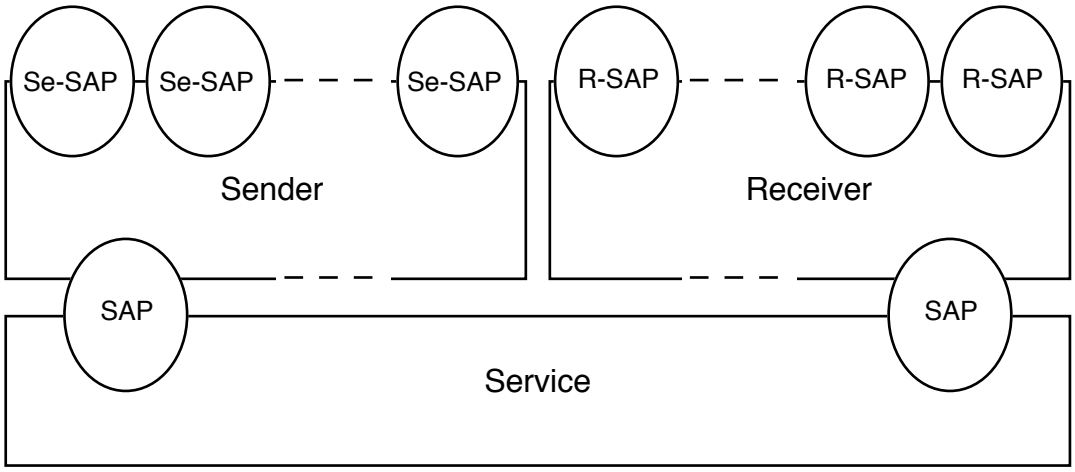

Fig. 3.1 Sketch of the system

The parameters that appear in the definition will never be given a precise value: everyone is free to realize its own implementation. However, we will sometimes point out some obvious rules that should be satisfied to preserve a sensible functioning.

We will use, in the sequel, a referential time unit, simply called "unit". Note however that this "unit" is not an elementary grain of time: the times expressed in the following description could be fractions of a unit.

4 Presentation of the Simple Version of the System

4.1 Service

In order to keep things simple, we have restricted the *service* specification to its interactions with its two users, *sender* and *receiver*, through their respective SAPs. Under these conditions, *service* can be characterized as follows:

Service Primitives: They carry a 48-byte cell as parameter. Primitives are instantaneous and atomic events. Our system is so simplified (the exchanges are always done between the same two SAPs) that no other parameter needs to be specified. In the sequel, these exchanges between the service and its users will simply be referred to as cells instead of primitives.

Isochronism: The given service is isochronous: a cell from *sender* is only accepted at some precise, punctual instants, that follow one another regularly in time, with a given rate (period: π units). An opportunity of emission can be neglected by *sender*. Just one cell can be exchanged at any instant.

Spacing Between the Deliveries: There is always a delay of at least α units between two successive deliveries at a same SAP.

Transmission Delays: A cell is always delivered between τmin and τmax after its emission.

Immediate Acceptation: A cell offered to *receiver* must be immediately accepted by *receiver*. If it is not possible, *service* losses the cell immediately.

Loss Free Transmission: The previous point describes the only way a cell received from *sender* can be lost: no cell is lost during the transit through *service*.

FIFO-Ordering of Cells: The cells arrive in their emission order.

The last two constraints - "Loss free transmission" and "FIFO-ordering of cells" - are not strictly necessary and might seem less realistic. However, they help to avoid unnecessary complications either in the specification of the service or in the definition of other mechanisms that we will meet later.

4.1.1 Comments About *Service*

It is mandatory that a delivered cell be accepted immediately by *receiver* or be lost. With just one constraint (Transmission delays) to determine the instant of arrival of a cell, expressing this urgency would have been easy with most of the formalisms. That is the reason why we have tried to introduce an additional constraint (the minimal delay α between two successive deliveries), that can modify the result of the first one, and thereby makes it impossible to impose urgency unilaterally. So, this problem of expressing urgency will have to be solved as a joint effect of the two constraints, a task for which most of the formalisms are not directly kited out (remember that, with concern of clarity, a structured and abstract style is suggested to specify the system, which supposes a separate representation of the different temporal constraints).

Let us notice that there is no incompatibility between the constraint "transmission delays" and the constraints "spacing between the deliveries" and "FIFO-ordering of cells", as long as the period of admission π is greater than or equal to the minimal delay between two deliveries α.

Besides this difficulty, the modelling of *service* also faces the problem of describing an isochronous system and of an undetermined delay between two bounds.

4.2 Sender

This entity multiplexes on *service* the data received from its n users, with which it communicates through their respective Se-SAP. It is characterized as follows:

Received Data: *Sender* receives from its users primitives that contain 3 parameters :

- a source address (20 bytes)
- a destination address (20 bytes)
- a SDU (we propose, for reasons of simplicity, that the size of each SDU be a multiple of 40 bytes)

Minimal Delay Between Primitives on a Given Se-SAP: On a given Se-SAP, every primitive must be at least spaced out from the previous one by a delay that is proportional to the size of its own SDU (β units per 40 bytes).

Emitted Data: *Sender* emits 48-byte cells on *service*, through their common SAP.

Emission of a Cell: *Sender*, when it desires to emit a cell, waits for an opportunity from *service*. When *service* is ready, the operation occurs immediately. If it has nothing to emit, *sender* just neglects the offers of *service*.

Segmentation of a SDU: Each SDU received by *sender* is emitted on *service* as a sequence of consecutive 48-byte cells. This sequence corresponds to the segmentation of the SDU into 40-byte fragments, completed with a 6-byte header and a 2-byte trailer. The sequence begins with a cell that carries the source and destination addresses.

Identification of the SDU: Each SDU receives an identification number that, from SDU to SDU, is increased of one unit, following the order determined by the instant of occurrence of the primitive. This number is written in the second and third bytes of the header of each cell.

Order of Emission of the Cells Related to Different SDUs: All the cells related to a SDU are emitted consecutively. The order of emission between the cells from different SDUs follows the one determined by the instant of occurrence of the primitives.

Order of Emission of the Cells Related to a Same SDU: The fourth byte and the first six bits of the fifth byte of the header of each cell indicate the rank, in the initial SDU, of the 40-byte fragment carried. These values are set to zero for the first cell of the sequence (the one that carries the source and destination addresses).

Emission of the Cells Related to a SDU: The order of emission of the cells corresponds to the one indicated by the fourth byte and the first six bits of the fifth byte of their header.

Minimal Delay Before the Emission of a Cell: A cell that is ranked x in the order of emission of the cells related to its SDU cannot be emitted before a delay $(x * \upsilon)$ after the occurrence of the primitive.

Types of Cells: The last two bits of the fifth byte of the header of each cell indicate its type as follows:

00 -> the cell carries the source and destination addresses ;
01 -> the cell carries a fragment of a SDU that is not the last one ;
10 -> the cell carries the last fragment of a SDU.

In the case of a 40-byte SDU, a 10-type cell follows directly the 00-type one.

Urgency of the Emission: *Sender* emits its cells as soon as it is allowed to, according to the previous rules.

4.2.1 Comments About *Sender*

Sender presents mainly two potential difficulties. First, one must be able to express an ASAP (As Soon As Possible) constraint between *sender* when it wants to emit and *service* when it agrees to receive a cell. This case is a typical example of the need for such constraints.

Next, one must try to express in a light and clear way, how the delay between two consecutive primitives on a same Se-SAP is related to the size of the SDU carried by the last one. In particular, this supposes the ability to treat time values as data.

4.3 Receiver

This entity de-multiplexes on its users the data received from *sender* through *service*.

Received Data: *Receiver* receives 48-byte cells from *service* through the SAP that joins them.

Frequency of Receptions: *Receiver* always listens to *service*, except during a period of "deafness" of ϕ units after the receipt of a cell.

Prolongation of the Deafness: An undetermined prolongation of the "deafness" may occur, because of internal congestion reasons.

Emitted Data: *Receiver* offers to its users primitives that carry two parameters: a source address and a SDU.

Creation of the Primitives: *Receiver* proposes a primitive for every SDU properly received and reassembled. This means that all the cells of the sequence have been received successively and in the right order. These conditions can be checked thanks to the parameters carried by the cells.

Emission Point of a Primitive: A primitive is proposed on the R-SAP that corresponds to the received destination address.

Discipline of Exit: If more than one SDU happen to wait on a same R-SAP, they will be proposed one by one, according to a FIFO discipline.

Minimal Delay Between Primitives on a R-SAP: On a R-SAP, every primitive is at least spaced out from the previous one by a delay proportional to the size of its SDU (ϖ units per 40 bytes).

4.3.1 Comments About Receiver

As it is defined here, *receiver* does not present special difficulties. Let us notice that it would be logical for the usual "deafness" delay ϕ to be smaller than the minimal delay α between the delivery of two consecutive cells by *service*.

5 Description of Additional Features

5.1 Adaptation of the Access Period to *Service*

The period between two consecutive interaction offers made by *service* may vary in time. The aim is to adapt the access to *service* to the presumed needs of *sender*, estimated from the use *sender* makes of the actual capacities he has at its disposal.

The mechanism proposed here segments the stream of proposals into consecutive and separated sequences of 10 proposals, at the end of which *service* is allowed to modify the period. Two main rules apply:

- at the end of a sequence, if all the offers have been accepted (10 cells have been emitted), the period is divided by two.
- at the end of 3 consecutive sequences, during which the period has not been updated by the previous or the present rule, the period is multiplied by a coefficient determined thanks to the following table, according to the number of proposals effectively used among the 30. (Multiplying by 1, in the case 26 --> 30, is also considered to be an update.)

Table 5.1 Determination of the multiplying coefficient

Number of proposals used			Coefficient
0	------->	10	2
11	------->	20	3/2
21	------->	25	6/5
26	------->	30	1

Bounds are however imposed on the possibilities of variation of the period, which must always remain between η and ψ units. Remember that initially, the period is supposed to be equal to π units.

5.2 "Crash" of *Service*

At any instant, without any reason, *service* may "crash". All the cells in transit are then lost and *service* needs an unpredictable delay before restarting. It restarts free of any cell and with a period π.

5.3 Rate-Control at the Access to the Service Provided by *Sender*

In section 4.2., a minimal delay between primitives occurring at a Se-SAP has been introduced. It may be interpreted as a kind of rate-control or any other kind of upper limit on the capabilities of *sender*. In this section, two other mechanisms are introduced which do not replace the minimal delay, but add rate-control mechanisms of a different nature. They impose an upper bound to the average rate *sender* accepts. Their effects are described by the next two additional constraints:

Individual Control on Each Se-SAP: Through each Se-SAP, during any period of κ units, the amount of information exchanged in the SDUs must not exceed φ blocks of 40 bytes. In other words, at any instant, a primitive may occur at a given Se-SAP only if the size (expressed in blocks of 40 bytes) of the SDU it carries is smaller than or equal to φ minus the sum of the sizes of the SDUs carried by all the primitives that have occurred at this Se-SAP since κ units of time (or since the start of *sender* if it has been functioning for less than κ units of time).

However, a primitive that carries a SDU with more than φ blocks of 40 bytes (let us say ζ blocks, with $\zeta > \varphi$) can be accepted if it is spaced out from the previous primitive on the Se-SAP (or from the start of *sender*) by a delay that is at least of $((\zeta / \varphi) * \kappa)$ units.

Figure 5.1 illustrates this mechanism, with $\kappa = 4$ and $\varphi = 20$[2].

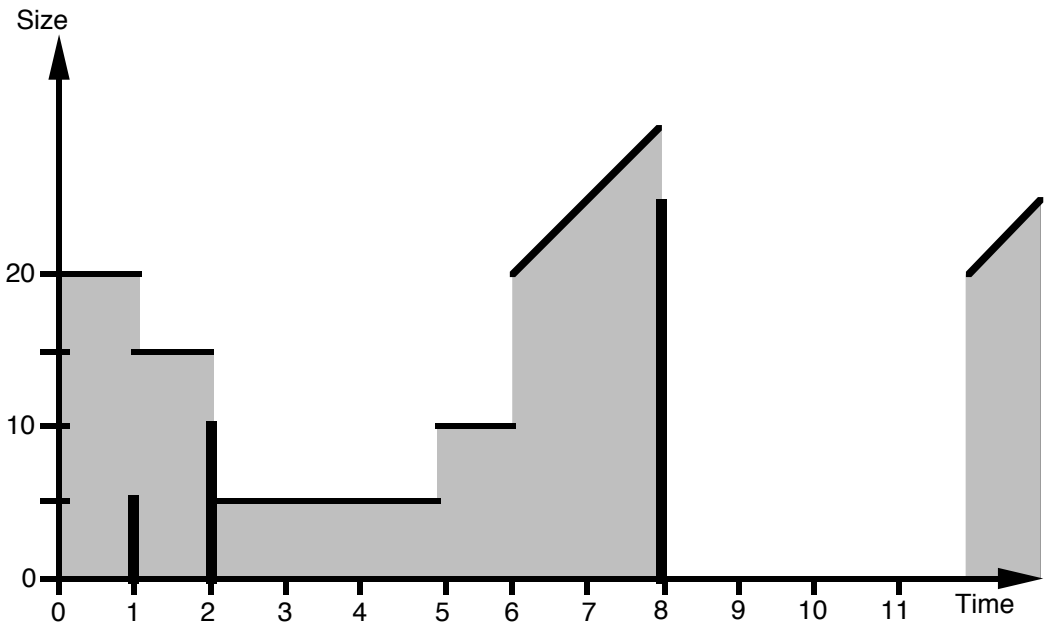

Fig. 5.1 Illustration of the individual control mechanism

[2] Note that κ and φ are not necessarily natural numbers.

A vertical bold line indicates the occurrence of a primitive at a given instant, and the size of its SDU. The horizontal or oblique lines above grey surfaces indicate, at each time, the maximal possible size of SDU that could be accepted. Initially, the maximal size is 20 (blocks of 40 bytes). Primitives carrying a SDU of more than 20 blocks are not yet allowed. At time 1, a SDU of size 5 is accepted and the maximal authorised size decreases of 5 units. At time 2, a SDU of size 10 is accepted and the maximal authorised size decreases of 10 units. At time 5, i.e. 4 time units after its occurrence, the first SDU stops being taken into account in the sum of the sizes, and the maximal authorised size grows of 5 units. At time 6, the same occurs for the second SDU. At this time, i.e. 4 time units after the occurrence of the last primitive, primitives carrying a SDU of more than 20 blocks begin to be accepted. The maximal authorised size grows proportionally to the elapsed time. At time 8, such a primitive occurs. Its effect is that for 4 units of time, i.e. until it stops being taken into account in the sum of the sizes, no primitive at all can be accepted.

Global Control of all the Se-SAPs: Through the whole set of Se-SAPs, during any period of λ units, the amount of information exchanged in the SDUs must not exceed μ blocks of 40 bytes.

However, a primitive that carries a SDU with more than μ blocks of 40 bytes (let us say ζ blocks, with $\zeta > \mu$) can be accepted if it is spaced out from the previous primitive on all the Se-SAPs of *sender* (or from the start of *sender*) by a delay that is at least of $((\zeta / \mu) * \lambda)$ units.

This mechanism is totally similar to the previous one but cares for all the Se-SPAs at a time.

5.4 Rate-Control on the Reception of Cells by *Receiver*

A rate-control mechanism, aimed at preventing the congestion of *receiver*, limits both its capability to accept cells from *service* and to keep SDUs not yet accepted by the users. The constraints "frequency of receptions" and "prolongation of the deafness" of the simplified definition are forgotten and replaced by the following ones.

First Sufficient Condition to Accept a Cell: A cell proposed by *service* is accepted if, at this moment[3], the difference between the number of blocks of 40 bytes already received and the number of blocks of 40 bytes already evacuated (the notion of evacuation will be explained later) by the entity since the beginning of its functioning is smaller than θ. The only bytes taken into account when counting the received or evacuated blocks are the ones from the exchanged SDUs, thus excluding the ones from the headers, the trailers or the source and destination addresses.

Second Sufficient Condition to Accept a Cell: When the first condition alone does not apply, a cell proposed by *service* is however accepted if, and only if, there exists at least one SDU among the ones proposed to the users that has been waiting for more than ρ units to be accepted. The oldest of the SDUs in this situation is immediately

[3] Remember that the offers of service are "punctual", they must be accepted immediately or never.

deleted and the next SDU on the same R-SAP (according to the FIFO rule), if it exists, takes its place.

Let us insist on the fact that one deletes an old enough SDU only if and when it is expressly necessary.

Evacuation of Bytes: There are three ways to evacuate some bytes:

- the bytes of a complete SDU are considered evacuated at the instant of occurrence of the primitive that transmits it from *receiver* to its user ;
- the bytes of the already received fragments of a still incomplete SDU are considered evacuated at the moment when one notices that a loss occurred, that definitely prevents a correct reassembling of the SDU (this case is detailed in the next point) ;
- when there is a lack of place to receive a coming cell, we saw that a complete SDU, waiting for delivery, can possibly be deleted if it is waiting for more than ρ units.

Evacuation Policy of the Useless Fragments: The main rule is simple: the already received fragments of a still incomplete SDU (a "last fragment" 10-type cell is not yet arrived) are evacuated if, and when, a cell arrives that is not the expected one, either because it does not carry the right sequence number, or because it carries an identification number different from the current SDU's one. The faulty cell is also thrown away, except if it is a "first fragment" 00-type cell, that announces the arrival of a new SDU. Also, if there is currently no pending SDU, either because the last cell arrived was of 10-type, or after the arrival of an unexpected cell that is not a 00-type one, each new cell arriving that is not of the 00-type will be rejected.

Another rule is added to this main one, that prevents from keeping fragments waiting for the arrival of a new cell, when it is known that, without loss, this cell had to arrive within a certain time. A time-out mechanism is then implemented, that is restarted after the arrival of each new cell and evacuates the pending fragments when it expires. The value of this time-out theoretically depends on the access period to *service* that, as we saw in a previous improvement, may vary in time in a non negligible way. *Receiver* thus tries to estimate this period, and at each restarting, a new time-out value is recalculated, according to the next formula: (2 * *min-delay* + 3 * *marge*). In this formula, *min-delay* is the smallest delay between two consecutively received cells noticed since (29 * η) units and *marge* is the difference between the maximal and the minimal possible transmission delays from *sender* to *receiver* (*marge* = τmax - τmin). If some cells are lost, or if their emissions were spaced out by a long period, it might happen that *min-delay* be quite important, or even do not exist, because less than two cells were received for the previous (29 * η) units. The time-out value must however always remain smaller than (ψ + *marge*) units. It will then be reduced to this value if it happens to be greater, or if *min-delay* does not exist (η and ψ are the previously defined minimal and maximal bounds on the access period to service).

5.5 Comments About the Additional Features

As announced, these features present different, and quite complex mechanisms that could be practically encountered. Especially, the rate-control of *receiver*, and the adaptation of the access period to *service* request the ability for the formalism to treat the temporal values as any data type, and the "crash" scenario of *service* poses a problem that, according to us, could sometimes be difficult to solve in a smart way: an interruption that can occur at any time without any reason.

6 Conclusion

The objective of this case study was to propose a large panel of temporal mechanisms that could be faced while specifying real protocols or services and that a temporal FDT should thus be able to describe in a smart way. In comparison to an already existing, real protocol or service, this case study presents, in our opinion, two advantages:

- all its features not directly relevant to the time have been (over)simplified;
- it offers a wider scope of possible aspects of time modelling.

We thus think, and hope, that it could be of interest to anyone who would like to test a temporal FDT in a more concrete situation, as did the *Abracadabra* service and protocol and the *Daemon Game* examples [ISO 10167] for the "classical" FDTs.

References

[AuB 84] D. Austry, G. Boudol, **Algèbre de Processus et Synchronisation,** *Theoretical Computer Science 30* (1984) 91 - 131 (North-Holland, Amsterdam).

[Azc 90] A. Azcorra-Saloña, **Formal Modeling of Synchronous Systems,** Ph. D. Thesis, ETSI Telecomunicación, Universidad Politécnica de Madrid, Spain, Nov. 1990.

[BaB 90] J.C.M. Baeten, J.A. Bergstra, **Real time process algebra,** Rept. No. P8916b, University of Amsterdam, Amsterdam, March 1990.

[BLT 90] T. Bolognesi, F. Lucidi, S. Trigila, **From Timed Petri Nets to Timed LOTOS,** in: L. Logrippo, R. Probert, H. Ural, eds., *Protocol Specification, Testing and Verification X*, (North-Holland, Amsterdam, 1990) 395-408.

[BoL 92] T. Bolognesi, F. Lucidi, **LOTOS-like process algebras with urgent or timed interactions,** in: K. Parker, G. Rose, eds, *Formal Description Techniques IV,* (North Holland, Amsterdam, 1992), 249-264.

[Bri 89] Ed Brinksma, **Constraint-oriented specification in a constructive formal description technique,** in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg, eds., *Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness, LNCS 430* (Springer - Verlag, Berlin Heidelberg New York, 1990) 130-152.

[CCITT Z100] CCITT-Study group X,**Specification and Description Language (SDL), Z.100**, (Blue Book, Vol. X, Fascicle X.1, 1988).

[Gro 90] J. F. Groote, **Specification and Verification of Real Time Systems in ACP,** in: L. Logrippo, R. Probert, H. Ural, eds., *Protocol Specification, Testing and Verification X*, (North-Holland, Amsterdam, 1990), 261-274.

[HaJ 90] H. Hansson, B. Jonsson, **A calculus for communicating systems with time and probabilities**, in: *11th IEEE Real-Time Systems Symposium*, Orlando, Florida, 1990, IEEE Computer Society Press

[Han 91] H. Hansson, **Time and Probability in Formal Design of Distributed Systems,** Ph. D Thesis, DoCS 91/27, Uppsala University, Dept. of Computer Science, P.O. Box 520, S-75120 Uppsala, Sweden.

[HeR 91] M. Hennessy, T. Regan, **A temporal process algebra,** in: J. Quemada, J. Mañas, E. Vazquez, eds., *Formal Description Techniques III,* (North-Holland, Amsterdam, 1991) 33-48.

[ISO 8807] ISO/IEC-JTC1/SC21/WG1/FDT/C, **IPS - OSI - LOTOS, a Formal Description Technique Based on the Temporal Ordering of Observational Behaviour**, IS 8807, February 1989.

[ISO 9074] ISO/IEC-JTC1/SC21/WG1/FDT/B,**Information Processing Systems - Open Systems Interconnection - Estelle, a formal description technique based on an extended state transition model**; IS 9074, July 1989.

[ISO 10167] ISO/IEC DTR 10167, **Guidelines for the Application of Estelle, LOTOS and SDL**, January 90.

[Led 92] G. Leduc, **An upward compatible timed extension to LOTOS,** in: K. Parker, G. Rose, eds, *Formal Description Techniques IV* (North Holland, Amsterdam, 1992), 217-232.

[LeL 93] G. Leduc, L. Léonard, **A timed LOTOS supporting a dense time domain and including new timed operators.** in: M.Diaz, R.Groz eds., *Formal Description Techniques V*, (North-Holland, Amsterdam, 1993), 87-102.

[MeF 76] P. Merlin and D. Farber. **Recoverability of communication protocols - implications of a theoretical study**. *IEEE Trans. on Computers*, 24(9):1036-1043,1976.

[MFV 93] C.Miguel, A.Fernandez, L.Vidaller, **Extending LOTOS towards performance evaluation.** in: M.Diaz, R.Groz eds., *Formal Description Techniques V*, (North-Holland, Amsterdam, 1993).

[Mil 83] A.J.R.G. Milner, **Calculi for Synchrony and Asynchrony,** *Theoretical Computer Science*, Vol. 25, No. 3, July 1983, 267-310 (North-Holland, Amsterdam).

[Mil 85] G. Milne, **CIRCAL and the Representation of Communication, Concurrency and Time,** *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 2, April 1985, 270-298.

[MoT 90] F.Moller, C. Tofts, **A temporal calculus of communicating systems,** in: J.C.M. Baeten, J.W. Klop, eds., CONCUR '90, *Theories of Concurrency: Unification and Extension, LNCS 458* (Springer - Verlag, Berlin Heidelberg New York, 1990) 401-415.

[NRS 90] X. Nicollin, J.-L. Richier, J. Sifakis, J. Voiron, **ATP: An algebra for timed processes,** in: M. Broy, C.B. Jones, eds., *IFIP Working Conference on Programming Concepts and Methods*, Sea of Gallilee, Israel (North-Holland, Amsterdam, 1990).

[NiS 91] X. Nicollin, J. Sifakis, **The Algebra of Timed Processes ATP: Theory and Application,** Rept. No. RT-C26, Projet Spectre, LGI-IMAG, Nov. 1991.

[NiS 92] X. Nicollin, J. Sifakis, **An Overview and Synthesis on Timed Process Algebras,** in: K.G. Larsen, A. Skou, eds., *Computer-Aided Verification III,* (LNCS 575, Springer-Verlag, Berlin Heidelberg New York, 1992) 376-398. Also in: LNCS 600.

[NSY 91] X. Nicollin, J. Sifakis, S. Yovine, **From ATP to Timed Graphs and Hybrid Systems,** in: J.W. de Bakker, C. Huizing, W.P. de Roever, G. Rozenberg, eds., *Real-Time: Theory and Practice* (LNCS 600, Springer-Verlag, Berlin Heidelberg New York, 1992) 549-572.

[NSY 92] X. Nicollin, J. Sifakis, S. Yovine, **Compiling Real-Time Specifications into Extended Automata,** in *IEEE/TSE Special issue on Real-Time*, Vol 18, n°9

[QAF 90] J. Quemada, A. Azcorra, D. Frutos, **A timed calculus for LOTOS,** in: S. T. Vuong, ed., *Formal Description Techniques II,* (North-Holland, Amsterdam, 1990) 195-209.

[Ree 90] G.M.Reed, **A Hierarchy of Domains for Real Time Distributed Computing,** in: M. Main, A. Melton, M. Mislove, D. Schmidt, eds., *Mathematical Foundations of Programming Semantics* (LNCS 442, Springer-Verlag, Berlin Heidelberg New York, 1990) 80-128.

[ReR 88] G.M.Reed, A.W. Roscoe, **A Timed Model for Communicating Sequential Processes,** *Theoretical Computer Science 58* (1988) 249 - 261 (North-Holland, Amsterdam).

[VSS 88] C.A. Vissers, G. Scollo, M. van Sinderen, **Architecture and Specification Style in Formal Descriptions of Distributed Systems,** in: S. Aggarwal, K. Sabnani, eds., *Protocol Specification, Testing and Verification, VIII* (North-Holland, Amsterdam, 1988), 189-204.

[Wal 83] B.Walter, **Timed Petri Nets for Modelling and Analysing Protocols with Real-Time Characteristics**, in: H. Rudin and C.H. West, eds., *Protocol Specification, Testing and Verification III,* (North-Holland, Amsterdam, 1983) 149-160.

[Wan 90] Y. Wang, **Real-Time Behaviour of Asynchronous Agents,** in: J.C.M. Baeten, J.W. Klop, eds., *CONCUR '90, Theories of Concurrency: Unification and Extension, LNCS 458* (Springer - Verlag, Berlin Heidelberg New York, 1990) 502-520.

[Wan 91] Y. Wang, **CCS + Time = an Interleaving Model for Real Time Systems,** in: J. Leach Albert, B. Mounier, M. Rodríguez Artalego, eds., *Automata, Languages and Programming 18,* (LNCS 510, Springer-Verlag, Berlin Heidelberg New York, 1991) 217-228.