Time in Distributed Systems
Cooperation and Communication Models

J.-P.Thomesse Z. Mammeri L. Vega

CRIN - CNRS URA 262
ENSEM, 2, av. de la Forêt de Haye
F-54516 Vandœuvre-lès-Nancy
email: {thomesse,mammeri,vegasaen}@loria.fr

Abstract

Dealing with time constraints in the design of distributed real-time systems is a challenge in the future trends on distributed computing. Time constraints must be considered, projected and derived onto all the layered structure components of the distributed system. This time constraints consideration, projection and derivation must be also done along all the stages of the life cycle. We focus how time constraints are involved in cooperation and communication aspects. We show one way to express time constraints with a temporal logic oriented specification, and some operating mechanisms to deal with them at the implementation stage.

1 Introduction

This paper deals with time considerations in distributed real-time systems. The problem of time constraints (TCs) is considered for a large number of years by a lot of people but essentially at the specification stage or in operating systems, through scheduling, and in protocols particularly in the MAC layer. The significative results in these domains are however insufficient to solve globally the problem. The main preoccupation is always to find development methods and mechanisms to implement reliable distributed real time systems and applications.

Motus [19] introduced several points of view to consider time aspects: time and specifier, time and verifier, time and implementor, from specification to implementation. The TCs have been studied and a lot of models have been proposed to solve the problems at the specification stage, with abstraction of the distribution [21, 1, 13, 24].

The possible validations are related to properties as lifeness, fairness, safety, but the notions of network or distribution (and then the time delays introduced) do not appear explicitely.

But when a solution must be designed, it's important to take into account characteristics of hardware, of operating systems, and of networks. Therefore we have introduced the time and designer point of view [32].

In this paper, we will focus on the design stage, between the specification and the implementation ones. At this stage, a solution must be defined by distributing the application on hosts, and exchanges on communication networks. The problem is to choose a good distribution according to the communication services provided by the profile, or to choose the profile according to the application needs.

Our approach starts from the application needs analysis, and from the distribution which generates more or less complex communication transactions constrained by time. Indeed, if exchanges are time constrained, that comes from application TCs which must be clearly expressed before to be managed in order to be met by the operating system and the communication protocols.

Our objectives are:

- to have models in order to verify at the design stage, before implementation, if the TCs will be met or not in the final solution, at least under certain hypothesis,
- to define operating mechanisms adapted to the applications constraints, in order to verify on line if TCs are met or not.

The paper is structured as follows: In the second section, we will describe the design development stage introducing the notions of application architectures (functional, support and operating) with a way to consider time aspects as constraints, characteristics or properties. In the third section we will specify more

precisely the TCs essentially related to the communication in order to define in the fourth section, the models of cooperation and/or communication. In the fifth section, we focus on operating mechanisms and open issues.

2 Time and application architectures

In this section, we will focus on time consideration in development stage. Different application architectures will be defined, and the concepts introduced. Starting from a functional architecture explaining TCs, the designer has to define an operating architecture with properties corresponding to the previous constraints and based on a support architecture with temporal characteristics.

2.1 Architectures

In this subsection we would like to introduce three kinds of architecture which are of importance for the design of any system: an application, a protocol, a subsystem [8, 29, 36].

Functional Architecture

A functional architecture is the result of a specification starting from a requirements document. It abstracts the implementation, and particularly the distribution. It may be obtained by any method but it must include the specification of functions, the typed relations between them, the TCs on functions or on relations between functions.

Support Architecture

A support architecture is composed of the description of hardware, of communication and operating systems, of all that is necessary for the implementation.

Operating Architecture

An operating architecture is the result of the mapping of a functional architecture on a support one.

2.2 Constraints and characteristics

Three words are commonly used to indicate time aspects: characteristics, constraints, properties. They are more or less randomly used and it clouds one's understanding.

Characteristics

A time characteristic is a known information, something known, for example, the reaction time of an actuator, the processing time of an action, and so on. Others characteristics are expressed in a function of time: a speed, a flow.

A time characteristic is a physical greatness expressed in time units, or with a time function, which certain values are known to choose elements for the solution and to implement this solution (constant, or minimum, maximum, average values).

Constraints

The term constraint is reserved to express what is to be verified in a solution before its design. For example a reaction time must be less than a given value, an action must be triggered at a given date. A constraint is a specification element.

Properties

The word property is reserved to express that a solution meets a constraint. It has then a characteristic conform to the constraint.

Conclusion

The three words have been defined according to development stage. A functional architecture expresses time constraints. A support architecture expresses time characteristics. An operating architecture must have properties corresponding to the constraints.

The same view may be applied on a layered system. At the run time a (N+1)-layer function requests a service to the N one. The service request is supposed constrained by TCs. The N-layer uses the characteristics of (N-1) to meet the constraints and to implement the mechanisms to verify on line their respect.

At the end of a specification, usually some verification or validation may occur if some formal model has been used. These validations are related to properties as lifeness, fairness, safety, which are all implementation independent. The knowledge of such properties is very important but not sufficient. To be sure that TCs are really met, it is necessary to know the real solution and to take physical or real time into account.

3 Time constraints modeling

About TCs in distributed systems a lot of papers have focused on the deadline of message transmission by different MAC protocols (for example [16]), and on the deadline of tasks by scheduling (for example [6, 10]). Analogies between tasks and messages have been evoked and studied [5]. But for the designer, the problem is to consider TCs at the application level, according to the real needs of application processes and of their cooperation. Therefore, we introduced some properties of solutions which are constraints at the design stage beginning. These properties are formally defined using a quantified temporal logic. They are based on the definition of objects submitted to events and of group of events (time coherences of events, actions and data).

3.1 Objects

The behavior of a system is modeled by the dynamics of its internal/external actions and data. It is the environment that imposes TCs over the system behavior, thus over its actions or data. An action is any internal or external treatment. A data is a piece

of information which is treated as a whole (a single measure, an N-PDU).

The main concept that allows us to model the dynamics of actions and data is the *event* concept. An event is a sequence of changes produced over a system condition along its behavior evolution. Each change over the condition is called an event occurrence [3]. The event could be internal or external [7] representing a change on the state of the system or its environment and they usually trigger a related processing.

The event concept allows us to model a system by the occurrences along the time axis of its external observable events. These events are the objects allowing us to describe the system behavior and the constraints over this behavior. So, in order to describe constraints over actions and data, we identify first the TCs over generic events in order to allow the possibility to associate a semantic according to any kind of actions and data.

3.2 Time constraints types

Two ways are used to indicate TCs related to system behavior:

- non-quantified time modeling approach referring to relationships linked to causality or any order between objects without duration concept. With this time we can only express behavioral relationships between system events [18, 23].
- quantified time modeling approach referring to instants or dates and intervals or time windows, durations or periods, for example earliest date, deadline, maximum, minimum or average duration, period. This modeling approach allows the expression of timeliness properties on the system behavior [13, 1, 21].

Here we are interested in the second one. We show in this section some constraints associated to events.

3.3 Constraints and events

Basic notions

An event is a changes sequence of a state or a condition. Example E_1: the temperature T_1 is greater than T_c. Each time, it goes from false to true, it's an occurrence of the event E_1. The constraints on such an event are constraints on their occurrences. An event is defined by:

an event is a couple (logic condition, value) which occurrence is the time instant when the logic condition over the assertion changes from false to true.

In a more formal way, in order to express in the following sections time properties over events without ambiguity, we could define an event as follows, but first we introduce some notations.

Notations: $\mathcal{E}$ is the set of predicates representing system events. VAR is a set of local variables with a range of values $type(v) = \{v_0, ..., v_n\}$. t is a time variable with a range of values $type(t) = \{t_0, ..., t_n, ...\}$.

Definition 1 *One event E / $E \in \mathcal{E}$, is a 2-ary predicate $E(t, v)$ where $v \in VAR$ and t is its time variable. We note E_i the $i-th$ occurrence of event E and $(E_i)_{i=1,...,n} \in O_E$ is the set of the event occurrences over the time:*

$E_0(t_0, v_0) \rightarrow ... \rightarrow E_i(t_i, v_i) \rightarrow ...$

where $\rightarrow$ indicates the successor and $\forall i,\ E_i \rightarrow E_{i+1}$. We have a time-stamping function

$d : O_e \rightarrow type(t)$ *where* $\forall i\ d(E_i) = t_i$

The time interval, or time window, is one of the basic concepts that allows us to express TCs. We define the following syntax for the time windows.

Definition 2 *A time window ΔT is a bounded interval $[t_s, t_e]$ where t_s is the start instant, t_e is the end instant and*

$\Delta T = |t_s - t_e|$ / $t_e, t_s \in N$ *and* $t_e - t_s > 0$

$t_s \equiv start(\Delta T)$

$t_e \equiv end(\Delta T)$

We can identify three basic constraints over one event occurrence: the earliest time, the deadline time and the time window constraint. Using definitions 1 and 2, we define these TCs over events as following.

Earliest time constraint. An earliest time constrains one event occurrence to occur after this instant.

Definition 3 *For the $i-th$ occurrence of event E with an associated earliest TC t_e, the respective logic assertion is*

$E_i \wedge (d(E_i) \geq t_e)$

Deadline constraint. The deadline constrains the occurrence to occur before this date.

Definition 4 *For the $i-th$ occurrence of event E with an associated deadline constraint t_d, the respective logic assertion is*

$E_i \wedge (d(E_i) \leq t_d)$

Time window constraint. It is a combination of the two above constraints.

Definition 5 *For the $i-th$ occurrence of event E within a ΔT, the respective logic assertion is*

$E_i \wedge (start(\Delta T) \leq d(E_i) \leq end(\Delta T))$

Single events and constraints

The constraints relating occurrences of a same event may be described by their temporal relationships with the system clock.

Periodicity specifies exact distance between two successive event occurrences.

Definition 6 *One event E is a periodic one with ΔT_p period iff*

$\forall i\ (E_i \wedge \bigcirc E_i = E_{i+1}) \Rightarrow (d(E_i) + \Delta T_p = d(E_{i+1}))$

Minimal arrival rate [12] expresses the minimal distance between two event occurrences (assumption about the rate of stimuli from the environment).

Definition 7 *A minimal rate of occurrences noted* ΔT_{min} *is defined by*

$\forall i \ (E_i \wedge (\bigcirc E_i = E_{i+1})) \Rightarrow (d(E_i) + \Delta T_{min} \leq d(E_{i+1}))$

Jitter is a time window constraint applied to a periodic event. It expresses the permissible time window drift between two event occurrences (assumption about the jitter data packet arrivals on multimedia applications [34]).

Definition 8 *One event E has a jitter constraint* $\Delta T_j = \Delta T_{max} - \Delta T_{min}$ *iff*

$\forall i \ (E_i \wedge \bigcirc E_i = E_{i+1}) \Rightarrow (d(E_i) + \Delta T_{min} \leq d(E_{i+1}) \leq d(E_i) + \Delta T_{max})$

Dependent events and time constraints

Some events are independent. But a lot of events are related with other internal or external events. In [7], a constraint classification is done by identifying minimal and maximal constraints over internal and external ones. This classification shows the reactive aspect of this kind of systems which must be able to react to environment changes. So, the environment events called stimuli, indicating environment changes, trigger a reaction of the system to produce related responses.

We use this distinction in order to define causal relation between events. We define two temporal relationships between causal or related events.

Response time. It indicates the timing of the occurrence of two events linked by a causality relationship: one cause event (stimulus) generates the production of another consequence event (response). The most usual cases are:

1. Maximal duration or deadline between a cause and its effect events (see definition 9).
2. Exact distance (e.g., delay). It is defined as in definition 9 but the inequality is replaced by an equality.

Definition 9 *A response TC between a stimulus* E^s *and its associated response* E^r*, is expressed as maximal time bound* ΔT_r

$E_i^s \Rightarrow \Diamond(E_i^r \wedge |d(E_i^s) - d(E_i^r)| \leq \Delta T_r)$

Time coherence. We have time coherence on a group of two or more linked events if their occurrences are within a time window.

Definition 10 *The occurrences i of a set of events* $E_i^{j=\{1,\ldots,n\}}$ *are called time coherent within a time window* ΔT_c *iff*

$\forall j \ (E_i^j \Rightarrow (start(\Delta T_c) \leq d(E_i^j) \leq end(\Delta T_c)))$

,

3.4 Actions, data and constraints

Some constraints on events being defined, it seems possible to extend these concepts to other objects as actions or tasks and data. An action can be temporally described by start event (for its beginning) and an end event (for its termination). The deadline constraint for an action is seen as a constraint on the end event. An earliest date constraint is seen as an earliest constraint on the start event. Considering an action attached to an indication (in OSI sense [31]), a deadline constraint on the response is similar to the deadline constraint on the action.

A parallel approach (or dual) may be considered with data. A data is the result of an action. The production instant or event may be confused with the end event of the production action. A consumption of a data may be assimilated to the beginning of the concerned task. Considering the life time of a data, it's a duration between production and consumption instants. We assume that all the concepts previously defined for events are also valid for actions and data, specially coherence.

These considerations are not here formally defined, (see [35]), but they show informally how TCs expressed in a specification on actions or on data, may be translated into TCs on events, in order to be managed in a unique way as well at the design stage as at the running one. These concepts are used to define the relationships between the entities cooperating in a distributed system. The cooperation between application processes is expressed in different ways, client-server, producer-consumer, ... Constraints on APs or on their relationships are expressed in terms of event TCs. The main events are communication requests, indications, responses, and confirmations.

We propose a way to facilitate the mapping of functional architecture on a support one, by using cooperation and communication models. The cooperation models are used to describe the wished behavior with TC, the communication ones are used to define the profile mechanisms of support architecture. The mapping is then the operation which associates communication transactions with cooperation ones.The verification of the properties is made by considering the TCs on cooperation models and the characteristics of the communication ones.

4 Cooperation models

We have defined TCs on events, independently of their nature and of their semantics. We have chosen this way, in order to be able further to associate various senses with the events. An application is specified in different ways, starting from actions, or from

data. A communication mechanism is seen from actions point of view associated to requests, or to indications, or from transfered information one (data or PDUs). An event may be the reception of an indication, the sending of a request. An event is generally associated with an action, and the action produces an event when it terminates its execution. The action is, according to classification, either a communication action, or an application action. This consideration leads to the well known *Client/Server* behavior.

Nevertheless, it is possible to consider the result of action rather than actions themselves, for example, data must be produced at the same time. The TCs on events are then translated into TCs on data, where the significant events are production of data, sending, reception, consuming, ... At the application design stage, we will consider these facts and we would like verify, at running stage, that the constraints are met or not. This consideration leads to the *Producer/Consumer* behavior.

Each of these two behaviors is the basis of four cooperation models, according to the number of the participating entities in the exchange. These models are described in the following section.

4.1 Producer(s)/Consumer(s) models

This model has a data oriented semantics. It allows us to describe the data exchanges between distributed sites and to express the time constraints and properties associated to them.

The communication types derived from the producer/consumer model are function of the number of participants that exchange data between them according to this model. Thus, we have four possible communication types: one to one, one to N, N to one and N to M, where N and M are the number of involved processes whose temporal properties are discussed in the following paragraphs.

Producer/Consumer model

This basic model represents the transfer of data between the *producer* that provides data to a *consumer*.

In real time system, data have a validity constraint called *life time*. The life time of data is a time window defined by the actions performed over this data: production, transmission and consumption. Another time problem is the rate control [25].

Producer/Multi-Consumer model

This model represents a communication exchange between one producer and several consumers that require the same data at the same moment provided by the producer. As in the above section we have a life TC over data that is function of the periodic, or non-periodic nature of its associated actions. In addition we have the problem of the *time coherence* of the data copies located in the distributed consumers.

Multi-Producer/Consumer model

Here the interaction between tasks is defined between several producers, each of one is a provider of one data, and there is one consumer that requires at the *same time*, all these data provided by the producers. So, we have the same constraints of producer/multi-consumer model, but the *time coherence* constraint is associated to the productions of data at each distributed producer.

Multi-Producer/Multi-Consumer model

We can reduce this cooperation model as a combination of the two above models.

4.2 Client(s)/Server(s) models

The semantics of the client/server model is an action oriented one. As in the previous section, the cooperation models derived from the client/server model are function of the number of participants requesting and providing services according to this model. Thus, we have the same four possibilities: one to one, one to N, N to one and N to M, where N and M are the number of involved processes.

Client/Server model

It is a one-to-one elementary model. It represents the interaction between the requesting and the performing of an action, or a service, between a client and a server.

The functional structure is distributed using the notion of service [20]. We can consider a service as the execution of an action (cf. section 3.4) because both of them perform an activity that provides a result.

The TCs associated with this model have a time response nature. We can distinguish four deadlines over the causal events of a client/server transaction: *request-indication*, *indication-response*, *response-confirmation* and *request-confirmation*.

Client/Multi-Server model

This cooperation model represents a communication type composed by many servers that provide services to one client. The client needs the servers activities results to perform its own ones. Here the problem is the *centralized* time coherence of the actions requested by one site and the *distributed* time coherence in the execution of the actions performed by the distributed servers. So, it is the same problem for the time coherence of indications and responses.

Multi-Client/Server model

Here we have a cooperation model constituted of several clients and one server. The server provides services to the clients in order to provide them the results that they need to realize their own activities.

We find in this case the problem of the *distributed* time coherence on the clients services requests, and the *centralized* time coherence on the execution of these actions by the server.

Multi-Client/Multi-Server model

This model is the case of a cooperation activity where we can find the two above models.

4.3 Examples

After this short presentation about the cooperation models, we would like to analyze some examples.

Producer/Multi-Consumer model

The constraints in this data exchange are:

- the deadline on the reception of indications that can be expressed as follows. Considering the event set $(Ind^j)^{j=\{1,...,n\}}$ as the indications set produced by a request event Req. So at the occurrence i we have a maximal response time ΔT_{max}:

$$Req_i \Rightarrow \Diamond(\forall j\ (Ind_i^j \wedge (d(Ind_i^j) \leq d(Req_i) + \Delta T_{max})))$$

- the time coherence on the reception of the indications. Considering the occurrence i of a set of events $(Ind^j)^{j=\{1,...,n\}}$ as the indications set that must be received within a time window ΔT_c:

$$\forall j\ (Ind_i^j \Rightarrow (start(\Delta T_c) \leq d(Ind_i^j) \leq end(\Delta T_c)))$$

Client/Server

The four TCs on the client/server model, introduced in section 4.2, are expressed as follows. Considering the next causality relations between the events on a client/server transaction i:

$Req_i \Rightarrow \Diamond(Ind_i \Rightarrow \Diamond(Rsp_i \Rightarrow \Diamond(Cnf_i)))$

The deadlines constraints associated are:

$Req_i \Rightarrow \Diamond(Ind_i \wedge |d(Req_i) - d(Ind_i)| \leq \Delta T_{ri})$

$Ind_i \Rightarrow \Diamond(Rsp_i \wedge |d(Ind_i) - d(Rsp_i)| \leq \Delta T_{ir})$

$Rsp_i \Rightarrow \Diamond(Cnf_i \wedge |d(Rsp_i) - d(Cnf_i)| \leq \Delta T_{rc})$

$Req_i \Rightarrow \Diamond(Ind_i \wedge |d(Req_i) - d(Cnf_i)| \leq \Delta T_t)$

Client/Multi-Server

We can see that its constraints are a combination of those issued from the client/server model and the producer/multi-consumer model. So, considering a set of $j = \{1, ..., n\}$ servers, we have first the deadlines constraints:

$\forall j\ (Req_i^j \Rightarrow \Diamond(Ind_i^j \wedge |d(Req_i^j) - d(Ind_i^j)| \leq \Delta T_{ri}))$

$\forall j\ (Ind_i^j \Rightarrow \Diamond(Rsp_i^j \wedge |d(Ind_i^j) - d(Rsp_i^j)| \leq \Delta T_{ir}))$

$\forall j\ (Rsp_i^j \Rightarrow \Diamond(Cnf_i^j \wedge |d(Rsp_i^j) - d(Cnf_i^j)| \leq \Delta T_{rc}))$

$\forall j\ (Req_i^j \Rightarrow \Diamond(Cnf_i^j \wedge |d(Req_i^j) - d(Cnf_i^j)| \leq \Delta T_t))$

Then we have time coherence constraints over the set of indications, responses and confirmation events, as follows:

$\forall j\ (Ind_i^j \Rightarrow (start(\Delta T_1) \leq d(Ind_i^j) \leq end(\Delta T_1)))$

$\forall j\ (Rsp_i^j \Rightarrow (start(\Delta T_2) \leq d(Rsp_i^j) \leq end(\Delta T_2)))$

$\forall j\ (Cnf_i^j \Rightarrow (start(\Delta T_3) \leq d(Cnf_i^j) \leq end(\Delta T_3)))$

5 Operating mechanisms and open issues

In the previous sections we have presented the notions of architectures, of time constraints and of cooperation models. A functional architecture specifies time constraints. A support architecture has time characteristics. The mapping of the first one on the second one must have the right properties associated with the constraints.

At the mapping stage (design stage) some validation must be obtained [2] but often such a validation is not deterministic. [2] has shown that time interoperability could be of two types: the so-called IOP_{must} and IOP_{may} which express respectively that interoperability is always or may be sometimes guaranteed. But it is not sufficient for real-time applications. Therefore, mechanisms must be included in the communication profile as well as strategies in the traffic control [9, 26, 28].

We give here some ideas and solutions in order to manage the different time constraints present in a given transaction.

5.1 Temporal qualification of data

As previously mentioned, communication between remote entities may be achieved according to several models. This section especially deals with the basic communication model, i.e., the producer/consumers one. Message scheduling depends mostly on the protocols of the network, and especially on the MAC (medium access control) protocol. The mechanisms proposed in this section are general and they are not designed for a particular network.

The communication between a producer and a consumer is achieved according to several steps: production of data value, it passes through the stack of the communication layers at the producer station, it is transmitted on the medium, it passes through the stack of the communication layers at the consumer station, consumption of the data value.

In a real-time context, production and consumption operations but also the communication layers must respect certain TCs to guarantee that the end-to-end (i.e., producer/consumer) cooperation will meet the whole application TCs. It is necessary to clarify the temporal data validity for each data with regard to the production and consumption time windows to ensure the data consumption coherence. A produced variable value is valid during a particular time window with regard to each consumer, called Temporal Validity Window. The consumer must terminate or begin its consumption operation before the end of the temporal validity window. In consequence, emission,

receipt and consumption operations must be scheduled by taking into account the end of the temporal validity windows.

To facilitate the analysis of temporal validity of values, and to have a good knowledge about the TCs, we associate a time window with each step; this time window (TW) will specify the time interval during which the step must be started and finished. Like that, we may specify the following TWs: a TW for the production step, a TW for each communication layer for the producer station, a TW for each communication layer for the consumer station, a TW for the consumption step.

One end-to-end TW, or one end-to-end delay, is often (or even, usually) associated with a communication relationship (i.e., one TW is associated with all the steps of a producer/consumer communication). We propose the use of several TWs to effectively master real-time communications [17].

A temporal status is associated with each step of the communication. These statuses enable to know if the TCs assigned to the corresponding steps (or time windows) are satisfied, or not.

The temporal status associated with each communication step is elaborated by an entity controlling the respect of the TCs, this entity is called TCCE (Timing Constraint Control Entity). TCCEs modeled by a general state machine are presented in [15].

In consequence, the consumer does not receive only a variable value but a message containing the produced variable value and several temporal statuses. If all the temporal statuses are set to True, then the variable value is valid for consumption, otherwise it is invalid, and the consumer may know why the value is invalid.

Finally, we note that some kind of such temporal mechanisms have been validated and implemented in the FIP network. Further work has to be carried on to deal with temporal data validity in other communication models such as client/server and client/multi-server models.

5.2 Clock synchronization

One of the principal functions of real-time systems is to provide adequate mechanisms for measuring the time instants at which particular events occur, the duration of time intervals between events. These functions become particularly critical, as the occurrence of the same event may be observed from such inherently asynchronous devices as a number of different processors.

Data have often meaning only when associated with time, and the temporal qualification of data is intrinsically based on time. This implies, in the context of a real-time context, that data must be time-stamped at different times of their life. The direct consequence of this is that all nodes in a distributed system should have access to a globally agreed real-time clock.

Example

Multi-Producer/Consumer - Client/Multi-Server. Let the constraints *All productions must be done within a* ΔT_p and *All transmissions must be done within a* ΔT_p. A client sends a request to N servers, if the following hypothesis is made: *All indications are produced within a given* ΔT, the previous mechanisms in 5.1 may be used to verify if productions are done with their TCs without clock synchronization. The control of the correct transmission in time is done by the client.

But without the hypothesis, it is necessary to time-stamp the events in order to control the TCs are met and then a clock synchronization is necessary [11, 14]. The previous hypothesis is equivalent to a clock synchronization mechanism.

In this section we have only overviewed some of the main operating mechanisms which are used to support real-time applications. Others as scheduling of tasks [4, 6, 30, 33, 37] and PDUs are of importance too (see for example [10, 5, 22, 27, 38]).

6 Conclusion

At the design stage, we have to choose a support architecture, then the well suited mechanisms, the right scheduling of tasks, of traffic, the right clock synchronization algorithm, the right TC verification mechanisms, in order to obtain a safe operating architecture.

We would like to focus on the fact that it is important to can do these choices and the mapping with a maximum confidence, i.e. with proofs as formal as possible, as quantified as possible that the operating architecture will be correct. It will be necessary to optimize it to obtain the best one according to criteria (costs, place, power supplying, ...).

This paper is too short to go into long considerations and complete formal explanations about the communication and cooperation models. It intends to present an approach for linking operating mechanisms and physical characteristics to specification needs.

Our approach, based on cooperation models allows the designer to express his needs in terms of communication services at which are attached TCs. We may then look to the capability of identifying and measuring the characteristics which are of importance to meet the constraints. The main goal now is to be able to define TCs derivation.

A lot of other problems have not been neither treated nor even evoked, task placement, dynamic vs. static scheduling, tasks and PDUs scheduling, dynamic configuration, quality of service regulation and so on. It will be of importance in the future to associate, to integrate all these aspects in a real design tool for distributed real-time applications.

Acknowledgments

The authors would like to thank their colleagues of the working group on real-time in the "Groupe de Recherche" PRS of the "Centre National de la Recherche Scientifique" in France, for their discussions on some topics presented in this paper.

References

[1] R. Alur and T. A. Henzinger. Logics and models of real time: a survey. Technical Report TR-92-1262, Department of Computer Science, Cornell University, Ithaca, NY 14853-7501, January 1992.

[2] Y. Benkhellat, M. Siebert, and J.-P. Thomesse. Interoperability of sensors and distributed systems. *Journal Sensors and Actuators, Elsevier Sequoia*, 2:247–254, Octobre 1992.

[3] M. Benmaiza. *Le concept d'évènement dans la spécification et la programmation de systèmes temps réel.* PhD thesis, Université Henri Poincaré, CRIN, Nancy (France), Mars 1984.

[4] C. Cardeira and Z. Mammeri. Ordonnancement de tâches dans les systèmes temps réel et répartis. *Revue RAIRO Automatique, Productique, Informatique Industrielle, APII*, 28(4):353–384, 1994.

[5] C. Cardeira and Z. Mammeri. A schedulability analysis of tasks and network traffic in distributed real-time systems. Measurement Journal, 1995. No. 15, Elsevier.

[6] C.C. Cheng, J.A. Stankovic, and K. Ramamrithan. Scheduling algorithms for hard real-time systems. *Real-time systems Newsletter*, 3(2):1–24, 1989.

[7] B. Dasarathy. Timing constraints of real-time systems : constructs for expressing them, methods for validating them. *IEEE Transactions on Software Engineering*, SE-11(1):80–86, January 1985.

[8] J.L. Delcuvellerie. *Ingénierie des Systèmes d'automatisation de production : connaissance en conception des architectures des systèmes informatisés d'automatisation et outil d'analyse de la robustesse aux défaillances d'architectures réparties.* PhD thesis, Institut National Polytechnique de Lorraine, Centre de Recherche en Informatique de Nancy, Nancy (France), mars 1989.

[9] ISO TCCA group. Interim report of the TCCA group of ISO/TC 184/SC 5/WG2 on Time-Critical Communications Architectures. Report N 254, April 1991.

[10] H. Kopetz. Scheduling in distributed real time systems. TR 1/86, Mars, Austria, January 1986.

[11] H. Kopetz. Clock synchronization in distributed real-time systems. *IEEE trans. on Computers*, C-36(8):933–940, August 1987.

[12] R. Koymans. Specifying real-time properties with metric temporal logic. *The Journal of Real-Time Systems*, 2:255–299, 1990.

[13] R. Koymans. Specifying message passing and time-critical systems with temporal logic. In *Lecture Notes in Computer Science (651)*. Springer-Verlag, 1992.

[14] L. Lamport. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52–78, 1985.

[15] P. Lorenz and Z. Mammeri. Real-time software architecture: application to FIP fieldbus. In *IFAC Workshop on algorithms and architectures for real-time control, AARTC'95*, Ostend, Belgium, 31 May 1995.

[16] N. Malcolm and W. Zhao. Hard real-time communication in multiple-access networks. *Journal of Real-time systems*, 8:35–77, 1995.

[17] Z. Mammeri and P. Lorenz. Integration of temporal mechanisms in communication protocols for time-critical distributed systems. In *12th IFAC Workshop on Distributed Computer Control Systems, DCCS'94*, pages 7–13, Toledo, Spain, September 1994.

[18] Z. Manna and A. Pnueli. The modal logic of programs. In *Colloq. Aut. Lang. Prog.*, pages 385–409, Berlin, 1979. LNCS 71, Springer-Verlag.

[19] L. Motus. Time concepts in real-time software. In *the International Workshop on Real-Time Programming (WRTP'92)*, pages 1–10, Bruges (Belgium), June 1992.

[20] B.J. Nelson. *Remote Procedure Call.* PhD thesis, Carnegie Mellon University, Computer Science Department, 1981. (Technical report CSL-81-9 at Xerox PARC).

[21] J.S. Ostroff. A verifier for real-time properties. *The Journal of Real-Time Systems*, 4:5–35, 1992.

[22] F. Panzieri and R. Davoli. Real-time systems: a tutorial. In *Performance Evaluation of Computer and Communication Systems*, pages 435–462. LNCS (729), 1993.

[23] A. Pnueli. The temporal logic of programs. In *18th IEEE Annual Symposium on the Foundations of Computer Science*, pages 46–57, Providence, R.I., New York, November 1977.

[24] A. Pnueli and E. Harel. Applications of temporal logic to the specification of real time systems. *Lecture Note in Computer Science*, 331:84–98, September 1988.

[25] M. Raynal. *La communication et le temps dans les réseaux et les systèmes répartis.* Eyrolles, Paris, 1991.

[26] M.G. Rodd and S.F. Al-rowaihi. Temporal modelling of real-time communication protocols based on a processor/channel approach. *Journal of Real-time systems*, 6:243–262, 1994.

[27] M.G. Rodd and W. Zhao. RTMMS- An OSI-based real-time messaging system. *Journal of Real-Time Systems*, 2:213–234, 1990.

[28] L. Sha, S.S. Sathaye, and J.K. Strosnider. Scheduling real-time communication on dual-link networks. In *Proceedings IEEE Real-time systems sympo*, pages 188–197, Phoenix, Arizon, December 1992.

[29] F. Simonot-Lion and C. Verlinde. Importance d'un cadre de référence dans la mise en place d'un système automatisé de production. In *Actes de la Conférence sur Automatisation Industrielle vol. I*, Montreal, juin 1992.

[30] B. Sprunt, L. Sha, and J.P. Lehoczky. Aperiodic task scheduling for hard real-time systems. *Journal of Real-Time Systems*, 1:27–60, 1989.

[31] ISO TR/8509 Technical Report. Information Processing Systems - Open Systems Interconnection - Service Conventions. International Organisation for Standardization, 1987.

[32] J.P. Thomesse. Time and industrial local area networks. In *the 7th Annual European Computer Conference on Computer Design, Manufacturing and Production (COMPEURO'93)*, pages 365–374, Paris-Evry (France), May 1993. Organized by the IEEE Computer Society.

[33] K. Tindell, A. Burns, and A. Wellings. Allocating Hard Real-Time Tasks: An NP-Hard Problem Made Easy. *Journal of Real-Time Systems*, 4:145–165, 1992.

[34] D. Towsley. Providing quality of service in packet switched networks. In L. Donatiello and R. Nelson, editors, *Joint Tutorial papers of Performance'93 and Sigmetrics'93*, pages 560–586. Lecture Notes in computer Science (729), 1993.

[35] L. Vega and J.P. Thomesse. Temporal properties in distributed real-time applications. In *13th Workshop on Distributed Computer Control Systems, DCCS'95*, Toulouse (France), September 1995. IFAC. To appear.

[36] C. Verlinde, E. Georgel, and J.P. Thomesse. Hierarchical and functional architecture for control systems. In *IECON 15th Annual conference*, pages 462–468, Philadelphia, November 1989. IEEE Industrial Electronics Society. Vol. I: Signal processing and system control.

[37] J. Xu and D.L. Parnas. On satisfying timing constraints in hard-real-time systems. In *Proc. of the ACM SIGSOFT'91 Conf. on Soft. for Critical Systems*, pages 132–146, New Orleans, December, 1991.

[38] Q. Zheng and K.G. Shin. Fault-tolerant real-time communication in distributed computing systems. In *22nd Intern. Sympo. on Fault-tolerant Computing*, pages 86–93, Boston, October 1992.