OASIS formal approach for distributed safety-critical real-time system design

Jean-Sylvain Camier, Damien Chabrol,
Vincent David, Christophe Aussaguès

CEA, LIST
BP 65, GIF SUR YVETTE CEDEX, F-91191 FRANCE
firstname.lastname@cea.fr

Abstract. OASIS provides an environment for real time multitasking and communication design, as well as an execution environment based on a safety oriented embedded real time kernel. The formal approach of real-time design avoids many difficulties: it allows implementing efficient advanced real-time functionalities without any safety loss. The concepts and methodology presented in this paper ensure the most important safety properties. Within this framework, our goal is to rely on formal and algebraic tools that can automatically bring the proof of correctness for safety-critical design issues. Such a constructive approach can easily speed up the system development by the formalization of the off-line analysis.

1 Introduction

In most key industries which require high dependability systems, distributed real time systems have found widespread use to ensure safety critical functions. In a distributed system, consisting of several independent nodes communicating via a network, the architecture of the communication network is the core of the system, and strongly conditions the capability of the system to fulfill the real time and safety requirements (Shin, 1994). It is a major difficulty to obtain a complete deterministic safety critical real time system conforming with very high safety requirements and allowing both rigorous and flexible development, including verification and validation. Providing these properties to a system has often been possible only by the use of specific hardware. Our contribution leads to the definition of a constructive system engineering approach that can deal with deterministic, fault-tolerant, safety critical real time distributed architectures on standard hardware.

To prepare the next generation of sensitive systems or embedded systems in transport and energy domains, our contribution is based on the OASIS approach developed by the CEA (French Atomic Energy Commission) and AREVA-NP (Nuclear Power Firm), in cooperation with EDF (Electricité de France). This approach provides rules and methods to design and implement safety-critical real-time systems over single processor architecture (Aussaguès and David, 1998). Work has been carried out to extend OASIS to distributed architectures. A deterministic and real-time protocol based on TDMA and its optimized version, called A-TDMA have been developed complying with the time-triggered context of OASIS.

2 The OASIS approach

The use of parallelism techniques in the design and the implementation steps of safety critical systems, while still ensuring strong safety properties, is the main focus of the OASIS approach (David et al., 1998). It proposes rules, formal methods and a set of tools for safety critical application engineering, allowing design and parallel programs implementation with fully deterministic and predictable behavior and thus guaranteeing specified dependability properties.

The execution model is deduced from the programming model, leading to the realization of a micro-kernel and the way of producing the embedded runtimes; it is based on a time-triggered architecture. The OASIS kernel architecture is composed of a generic runtime wrapping that interfaces the tasks and the micro kernel, plus a real time determinism and dependability-oriented micro kernel which implements data exchanges for the communications and a graph-execution control of each task code sequences.

There are only specifications of the behavior of the tasks: the system call principle does not exist. After the code generation, one function is associated with each real time behavioral element. Their semantic is compiled in the runtime and their sequence is defined in their state-transition diagram. It guarantees responses in specified times using a safety-oriented multitasking execution, and has predictable and reproducible behaviors. The main idea of the OASIS approach is, first of all to develop multitask programs which are sporadic or cyclic, regular or irregular and then to demonstrate that some properties of well functioning descended from the specifications of the system can be proved and are in accordance with the constraints of qualification defined by safety standards and recommendations.

2.1 A Multitasking Time-Triggered System

In the OASIS approach, an application is viewed as a static set of communicating agents (real-time tasks) that interact to perform their functions. An agent is an autonomous execution entity consisting of a varying number of elementary actions (noted EA), the sequence of which complies with certain logical conditions 1. In a time-triggered approach, the system observes its environment and initiates processing operations at recurring, predetermined instants in time. In OASIS, these instants can be viewed as points in the globally synchronized time of the time triggered system.

2.2 Formal verification

In the OASIS approach, the ΨC formal language allows the developer to express the complete and formal behavior of an agent where the pure algorithmic parts are written in the ANSI C language. The goal of the ΨC language is to allow a formal description of the elements of the OASIS approach (parallelism, temporal synchronizations, communications) without breaking the algorithm coding rules of the developer.

The chain of development is operational since March 2000 and includes:

- the complete compilation of the agents and of their associated state-transition diagrams,
- the interface of the agents with the system layer (parameters of system calls to the system layer, communication buffers),

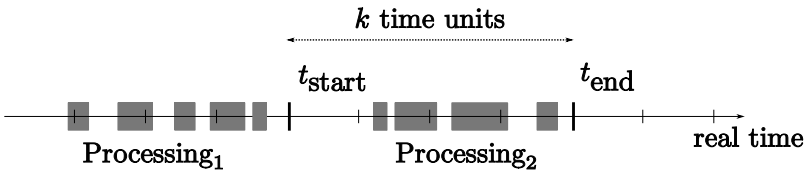

FIG. 1 – *EA processing within its time interval: each elementary action performed by an agent takes place between two temporal synchronization points of the time-triggered system, not necessarily contiguously. These points are the beginning date, which is the earliest instant when the elementary action can be started, and the ending date (deadline) which is the latest date when the EA must be terminated. These two dates allow to explain a formal duration of the elementary action. This formal duration is constant and represents a constraint for the actual duration of execution.*

- the calculation of each buffer worst case sizes,
- the link edition for each agent, between the agents and with the system layer,
- the generation of static memory protection tables to enable the protection of the whole application and the run-time (that is the code resulting from the sizing),
- the schedulability proof when worst case execution times of each EA are given.

2.3 Determinism fundaments

Determinism is one of the basic properties of OASIS. Determinism means that the behavior is unique and unvarying. Each task has a behavior independent of the target architecture. An off-line sizing of all the physical supplies which are necessary for the running can be done. Due to unvarying behavior, adding or deleting a task which has no direct interaction with other tasks, does not affect the application behavior.

Strict observation principles Communication mechanisms in OASIS are based on these following strict observation principles :

- A production processing and its data are encapsulated in the structure of a real-time agent: data are generated by only one producer, the owner agent,
- A data in production is hidden to the consulting agents,
- A data has a new value when its associated production processing reaches its deadline. The data is then available for the consulting agents,
- Past values are non modifiable by either the producer or the consumers,
- A consulting agent observes past values as they are at its beginning date. These past values are then qualified as visible.

Temporal Variables Each temporal variable defines a real-time data flow: its values, available to any agent that consults them, are stored and updated by a single writer, the owner agent, at a predetermined temporal rhythm as it is shown in Figure 2. This rhythm is expressed in the source code as a regular time period.

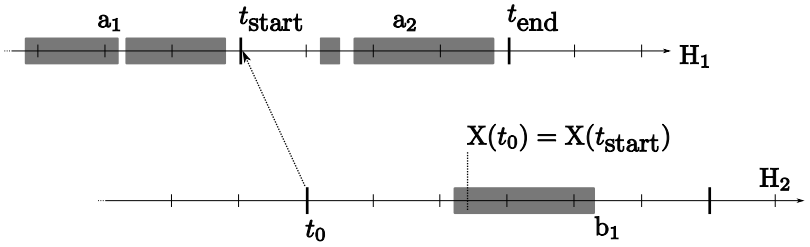

FIG. 2 – *Real-time data flow: assuming that an agent has a temporal variable X, regardless of the values between dates t_{start} and t_{end}, if another agent* observes *the value of X at date t_0, the X value "observed" is always its past value $X(t_0) = X(t_{start})$.*

Real-Time Messages In addition to conventional sending attributes of a message (content, recipient, etc.), messages are *typed*, checked during compilation and their visibility date is specified by the sender (see Figure 3). A recipient agent has queues for receiving purposes which can only accommodate messages of the same type and has an associated time limit, which allows a validity period to the received messages. To achieve determinism, the sequence of messages sorted by visibility date is made to be total.

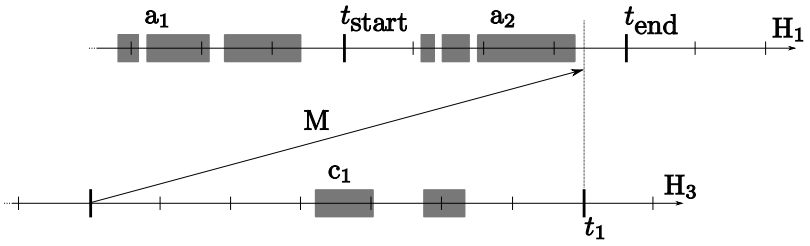

FIG. 3 – *Real-time messages: if message M is sent by the second agent with a visibility date t_1, then it cannot be consumed before the t_{end} date, since $t_1 > t_{start}$.*

2.4 Fault-tolerant mechanisms

Safety-critical applications must be designed according to sizing and execution hypothesis. OASIS provides a safety-oriented real-time kernel which ensures an early fault detection and confinement. The implemented mechanisms ensure that any applicative error, regardless of location, will have a deterministic impact on the system (Aussaguès and David, 1998).

Fault detection The OASIS kernel ensures detection of:

- Non-conforming agent behaviours. Each agent is monitored by using the state-transition diagram extracted during the compilation stage. This diagram describes all of the logical and temporal behaviours of an agent,
- Usual exceptions (fault addressing),
- Non respect of temporal constraints (deadline and quota),
- Attempts of memory overflow.

Indeed, a rigorous memory space segmentation is created at three levels (micro-kernel, system layer and agents) based on the two following mechanisms: the privileged mode of processors which are limited only to critical run-time sections; and the memory management unit (MMU) which enables a memory protection between the agents, the system layer and the micro-kernel.

Fault confinement The kernel and the runtime are segmented and put in place according to a patented process (CEA/Framatome-ANP David and Delcoigne, 2000). In opposition to most current systems, the asynchronisms of the execution do not have any impact on the determinism. In the safety domain, each processing must have a bounded duration and an upper bound can be calculated. The knowledge of the upper bounds of time execution is not used to tune the real-time behavior of the system but only to size the targeted architecture.

3 Distributed architectures implementation

The OASIS environment is geared by an extension to provide distributed safety-critical real-time systems with the same properties. The objective is to increase performance and modularity while satisfying compliance to the safety requirements. In order to design a distributed safety-critical system with the OASIS chain of development, the model for single processor has been extended to distributed ones with the two following constraints:

- The distributed system remains in conformance with the OASIS model, in particular with the determinism property,
- The underlying hardware architecture (eg. network topology) must be as transparent as possible for application designer.

3.1 Distributed architecture requirements

In traditional networks, maximizing the bandwidth or minimizing the average delay of transmission are both criteria of performance. In the context of safety-critical real-time network, these criteria are not enough. Indeed, such networks must satisfy timeliness and be in conformance with safety requirements, especially determinism and fault tolerance.

Timeliness A real-time communication network must ensure the complete termination of a transmission before a predetermined instant, its deadline. If this deadline has expired, the communication is considered as failed even if the transmission ends after this instant.

Determinism This safety principle is directly linked to the causal principle: an event leads to the same effects for given causes and conditions. In others words, in the network domain, this means that the network scheduling must be predictable and reproducible in both temporal and logical domains. However, determinism is sometimes just interpreted according to two variants: *temporal determinism* (the communication delay must be bounded, predictable) and *system determinism* (a transmission must succeed or a network error detection must be raised, if the communication is missing or corrupted). Even if two different variants exist, the safety problem implies both, including reproducibility.

Fault tolerance A fault tolerant system must be able to ensure its functions even if software or hardware errors occur. Therefore, network errors must be efficiently detected and confined in order to avoid error propagation. The network architecture must be able to bring fault resistance in order to carry on safe transmission.

3.2 Formal implementation of the distributed execution model

In the OASIS design model, in order to use temporal variable or message mechanisms, it is necessary to indicate respectively the updating period date or the visibility date. The updating date and the visibility date indicate the instant when the data should be available for consumers. It also represents the last instant for receiving the data. By this way, in the OASIS approach, the communication latencies are never considered as null. These temporal parameters provide a way of integrating temporal constraints and communication latencies due to the network at the design level. No change of the design model is required. The underlying target hardware architecture is transparent for the application designers.

Single processor architectures On single or shared-memory multiprocessor architectures, the effective delay for putting data at the disposal of the consumer consists only in copying the memory buffers. For example, in Figure 4, an agent (that calls the sending procedure during d_1) sends a message M with the visibility date t_0. During the s effective delay, the system layer of the producer agent stores the message in its local buffer. On the consumer side, the system layer copies the message from the producer buffer to the consumer one before the execution of the consumer agent.

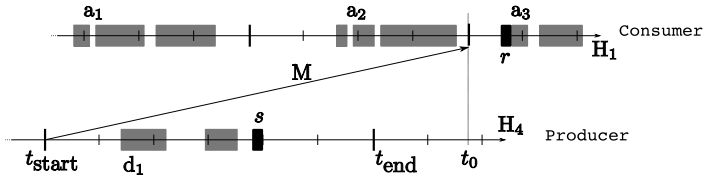

FIG. 4 – *Effective delays of communication management in a single board architecture.*

Distributed architectures On distributed architectures, the communication latencies are particularly significant in comparison with the application rhythms. Then, time resource for

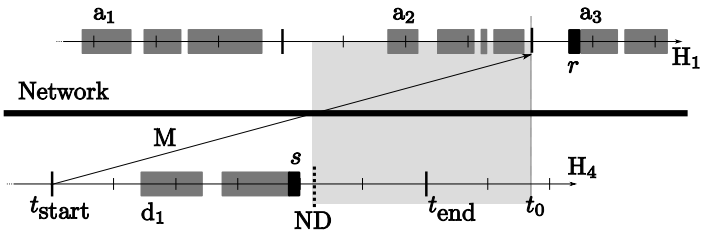

FIG. 5 – *Communication model on distributed architectures.*

the communication management must be freed. Without modifying the OASIS concepts, this operation consists in adding a network deadline before the availability date. A network temporal window is then applied and sized off-line in order to include the communication delay.

In Figure 5 the agents communicate through a local area network. A network deadline (ND) is set before the visibility date t_0 which allows to free a network window which has been represented by the gray rectangle, thus guaranteeing the data to be ready for the preparation of the corresponding packet. For the application designers, this additional deadline is automatically computed and transparent: the source code is unchanged.

3.3 TDMA

Time Division Multiple Access-based protocols have been identified as interesting protocols with the ability of ensuring fault-tolerant and timeliness communication. By way of illustration, the use of Ethernet network components is not an additional requirement for the OASIS chain of development. Thus, any reliable real-time protocols, such as FlexRay, which uses TDMA protocol and is able to ensure the requirements, are capable of being integrated into OASIS distributed systems.

TDMA divides the *global time* into a sequence of time slots, each of them assigned to a single unit of the network. A periodic sequence of time slots is called a cycle. These assignments and the time slot sizes are stored in a static schedule table that is computed off-line during an analysis stage called *network scheduling*. Communication latencies are predictable and can be computed. Indeed, TDMA ensures that only one board at a time uses the network bus. However, because of the jitter brought by clock drifts (see Figure 6.) and usual sources of asynchronism, several transmissions (by cycle repetition) can not be performed at the same effective instants S:

$$-\rho \leq S' - S \leq \rho + \sigma \quad \textit{modulo} \quad C \tag{1}$$

where S and S' are effective instant of transmission computed off-line, ρ the clock synchronization accuracy, σ the transmission precision and C the network cycle period. According to the equation (1), the network occupation of one transmission must therefore be upper bounded such as the computed time slot is the sum of its associated transmission window, the underlying network featuring *minspace* (minimum interframe spacing required by the physical network) and the *packet* transmission latency.

$$T_e = T_s + \|W\| + minspace + \|packet\| \tag{2}$$

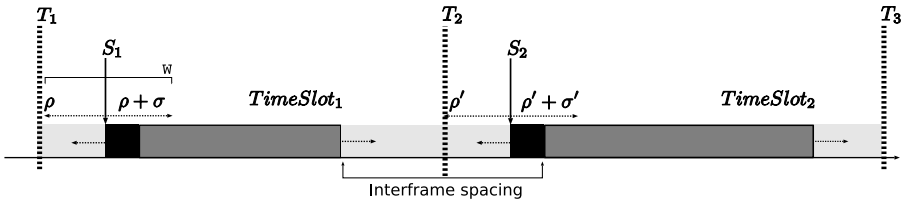

FIG. 6 – *TDMA scheduling: effective transmission intervals are coloured in dark gray and silent space reserved to ensure a safe communication in light gray.*

where T_s is the earliest starting transmission instant and T_e is the latest ending transmission instant.

The transmission window W is the interval during which the transmission command must occur, in order to meet the time slot ending instant, computed from an upper bound ρ and σ, such as:

$$W = [S - \rho, S + \rho + \sigma] \tag{3}$$

Determinism and timeliness Since TDMA brings a static network scheduling, its behaviour is therefore reproducible in temporal and logical domain. To ensure the timeliness, the network scheduling must be computed in order to set each time slot before the deadline where transmitted data must be available for distant consumers.

Fault Tolerance Network fault tolerance begins from the ability to detect errors coming from the network. the error detection is simple and efficient. Since the network scheduling is static, each unit has a perfect knowledge of the network behaviour. The error detection is therefore based on a qualitative approach where missing or swapping of packets are critical errors. Moreover, the network is supervised in order to detect disregarded execution and sizing hypothesis such as the absence of collision or the bounded clock synchronization accuracy.

3.4 Implemented TDMA optimizations

Advanced-TDMA consists in allowing an unique node to initiate a transmission whenever the bus is busy. A carrier sense mechanism is required to automatically perform the next transmission. The time wasted between two packets is therefore optimized to its possible minimum which is the inter-frame space of the network specification (see Figure 7). Network sizing of such a protocol must be rigorous in order to ensure safe and efficient communication. During network scheduling, time slots are sized like in TDMA except starting and ending transmission instants which are postponed if possible. Indeed, the objective is to parallelize an effective transmission with the transmission window of the next communication. This optimization is therefore performed in accordance with the following conditions in order to preserve safety and timeliness requirements.

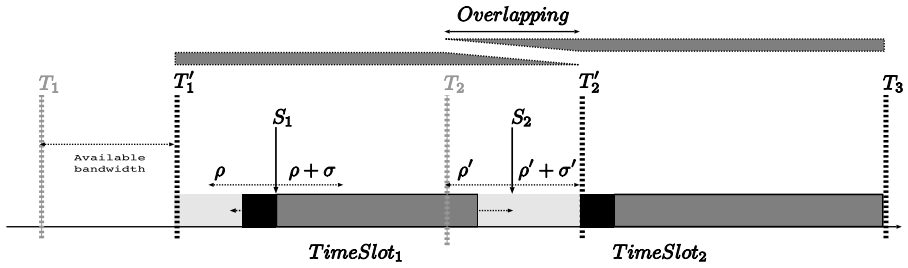

FIG. 7 – *Advanced-TDMA. Time slot 1 is postponed from interval (T_1,T_2) to (T'_1,T'_2).*

- Transmission windows must never be overlapped one over another otherwise collision or transmission swapping may occur:

$$W_i \cap W_j = \varnothing \quad (i \neq j) \tag{4}$$

- Time slot overlapping must be performed only between two consecutive ones. If more than one transmission window overlaps a time slot in accordance with condition 1, a collision inevitably occurs.
- In accordance with condition 1 and 2, the time slot, overlapped by the transmission window of the next time slot, must have a length large enough, such as:

$$\|Slot_1\| > \|W\| + \|W'\| + minspace \tag{5}$$

  where packet 1 is sent before packet 2

- Optimization must not be performed for the first transmission to ensure timely transmission. A synchronization barrier must be identified in the network by all the boards to prevent drift propagation. It corresponds to an instant where no communication can begin before and end after it.

A-TDMA improves the bandwidth use by avoiding silent spaces, and thus remains in conformance with the network requirements. A bandwidth is won and made available for others units.

3.5 Formal communication scheduling

From the communication constraints which can be extracted from the design model and the mapping of the distributed tasks, the network runtimes are automatically produced during compilation stage (Chabrol et al., 2005). The network runtime includes the OASIS protocol information, all of the sizing information of the packets to be networked, the static time slots scheduling and the formal instants of transmission. OASIS supplies all the tools at an industrial level for an automatic chain of development. Indeed, each agent is monitored by using the state-transition diagram extracted at the compilation stage. This diagram translates all the logical and temporal behaviours of an agent, discretizing each temporal synchronization point

that an algebraic tool can deal with in order to manipulate, schedule and size the desired communication. Such diagrams are used to compute the network requirements for each agent, systems of algebraic equations with linear equalities and inequalities are solved to schedule the communication, taking into account the constraints to use the A-TDMA protocol to its best advantage. Work on automatic formal network sizing and scheduling will be detailed in further papers. The profit is *flexibility*, allowing an easier application development, *scalability*, merging execution model core properties and some best suited underlying topologies.

4 Perspectives

In order to ensure the functions defined by the application requirements, which would be the same as in a degraded mode, the whole system must be designed to tolerate the single failure of any of its components. Otherwise, the whole system may fail.

4.1 (Δ,k)-graph approach

An interesting problem in graph theory is to draw graphs in which both the degrees of the vertices and the diameter of the graph are small (Gérard Memmi, 1982). A good network from a communication point of view should have a small diameter to allow all pairs of processors to communicate quickly. This concern with network topology leads to take full advantage of the capabilities of a distributed computer system. A (Δ,k)-graph is a graph of diameter Δ whose nodes are each of degree k or less. Distributed computer systems can be modeled using graph theory to evaluate potential network topologies. This approach deals with the design of communication architecture with a concern for fault tolerance. This introduces the *vulnerability* notion of the diameter, also called the (Δ,Δ',k,s)-problem (Rodriguez, 2006). We are looking for (Δ,k)-graph such that the subgraphs obtained by deleting any set of s vertices have a diameter of at most Δ' (see Figure 8 for an example of such (3,3,4,1)-graph).

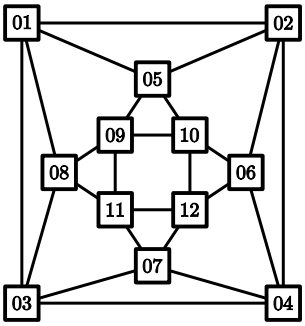

FIG. 8 – *Cuboctahedron quartic ($\Delta = 3, k = 4$)-graph.*

The following section uses these kind of interesting topologies for a distributed system in order to implement synchronization algorithms, the convergence of which can be formaly proved.

4.2 Algebraic clock synchronization

New kinds of clock synchronization algorithms have emerged (Ramanathan et al., 1990), based on rate-based synchronous diffusion algorithms. The main issue of such algorithms is the need to know if the *average* value will effectively converge. A well chosen connected graph $\mathcal{G}$ can by itself bring the proof of convergence, as well as an indication of its convergence speed for clock synchronization or spreadcast algorithms. The network is represented as a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with A being its adjacency matrix. Let $C = [c_1^t, c_2^t, \ldots, c_n^t]^T$, C represents the time readings of each node in the network at the time t. Since we are computing the average value for all nodes in the system, the clock value diffusion formula can be escribed by applying the following square matrix $R = (r_{ij})$ on the clock reading vector, where r_{ij} can be chosen randomly provided $\sum_{j \neq i} r_{ij} \leq 1$ and $r_{ii} = 1 - \sum_{i \neq j} r_{ij}$.

$$r_{ij} = \begin{cases} 0 & \text{if } n_i \text{ and } n_j \text{ are not neighbors} \\ r_{ji} & \forall (i,j) \in [1, \ldots, n]^2 \end{cases} \tag{6}$$

Algebraic relations exist between the adjacency matrix of the connected graph $\mathcal{G}$, the largest eigenvalue of R which lead to the proof of the convergence of such algorithms (Li and Rus., 2006). Perron-Frobenius theorem sorts the $(\lambda_1, \ldots, \lambda_n)$ real eigenvalues in such way:

$$1 = \lambda_1 > \lambda_2 > \lambda_3 > \cdots > \lambda_n = -1 \tag{7}$$

The convergence of the diffusion algorithm depends on the condition:

$$1 > \lambda_{max} = max(|\lambda_2|, |\lambda_n|) \tag{8}$$

Suppose $1 > \lambda_{max}$, the iteration for the whole network will effectively converge to the synchronized clock vector.

5 Conclusion

OASIS can be used to implement safety-critical multitasking real-time systems satisfying the increasing need for complexity while simplifying their design. The concepts and methodology presented ensure the most important safety properties namely the data coherency, an unvarying and deterministic real time behavior, and in-depth fault-detecting and fault-confining hardened mechanisms, which is why OASIS is a true formal approach to the design of deterministic and multitask real-time applications. Our goal is to rely on formal and algebraic methods that can automatically bring the proof of correctness for safety-critical design issues. Such a constructive approach can speed up the system development by the formalization of the off-line analysis. The presented system architectures can offer resilience to a single Byzantine fault, as well as immediate proof of convergence for the clock distribution. A prototype is already operational with Ethernet components and allows performing an efficient and dependable system. Another step will be the algebraic formalization of algorithms introducing gossiping in communication schemes (M.J. Dinneen, 1991) leading to optimality for broadcasting with minimal connectivity, which means *sizing* at optimal cost.

References

Aussaguès, C. (1999). *Placement Optimal de Tâches pour les Systèmes Parallèles temps-réel Critiques*. Ph. D. thesis, CEA, LIST / University of Marseille.

Aussaguès, C. and V. David (1998). A method and a technique to model and ensure timeliness in safety critical real-time systems. In *ICECCS*, pp. 2–12.

CEA/Framatome-ANP David, V. and J. Delcoigne (2000). Patent wo 02/39277 a1, security method making deterministic real-time execution of multitasking applications of control and command type with error confinement.

Chabrol, D., G. Vidal-Naquet, V. David, S. Louise, and C. Aussaguès (2005). Oasis - a chain of development for fault-tolerant deterministic distributed real-time systems. *6th Braunschweig Conference Automation, Assistance and Embedded Real-Time Platforms of Transportation*.

David, V., J. Delcoigne, E. Leret, A. Ourghanlian, and P. Hilsenkopf (1998). Safety properties ensured by the oasis model for safety critical real time systems. *IFIP-SAFECOMP'98 Conf.*.

Gérard Memmi, Y. R. (1982). Some new results about the (d, k) graph problem. *IEEE Trans. Computers*.

Li, Q. and D. Rus. (2006). Global clock synchronization in sensor networks. *IEEE Transactions on Computers*, 214–226.

M.J. Dinneen, M.R. Fellows, V. F. (1991). Algebraic constructions of efficient broadcast networks.

Ramanathan, P., K. Shin, and R. Butler (1990). Fault-tolerant clock synchronization in distributed systems. *IEEE Computer, vol. 23, pp. 33-42*.

Rodriguez, J. (2006). The (δ,δ',s,t)-diameter of graphs: A particular case of conditional diameter. *Discrete Applied Mathematics, Volume 154, Issue 14*.

Shin, K.G. ; Ramanathan, P. (1994). Real-time computing: a new discipline of computer science and engineering. *Proceedings of the IEEE 82,* (1), 6–24.