INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team ADEPT

Asynchronous Distributed Environments, Protocols, and Time

Rennes

THEME 1C

Table of contents

1. Team

Scientific Leader
Michel Hurfin [CR, INRIA Rennes]

Administrative Assistant
Lydie Mabil [TR, INRIA Rennes]

Researcher Cnrs
Emmanuelle Anceaume [CR]

Researcher INRIA
Gérard Le Lann [DR, INRIA Rocquencourt]

Faculty Members of the University of Rennes I
Maria Gradinariu [Associate Professor]
Achour Mostéfaoui [Associate Professor]
Michel Raynal [Professor, until November 2003]

External Researchers
Jean-Pierre Le Narzul [Associate Professor, ENST Bretagne]

Visitors
Roy Friedman [Poste d'accueil INRIA, Technion, Israel]

Postdoctoral Fellow
Xiaojun Ma [French Research Ministry]

Ph.D. students
Frédéric Dang Ngoc [Grant France Télécom, until September 2003]
Eric Mourgaya [Fellowship INRIA, until July 2003]
Julien Pley [Fellowship MENRT]
Philippe Raïpin Parvédy [Fellowship MENRT]
Matthieu Roy [Fellowship ENS]
Gwendal Simon [Grant France Télécom]
Frédéric Tronel [Engineer INRIA Rhones Alpes, until July 2003]

2. Overall Objectives

Distributed computing is becoming increasingly important in various economic sectors. Today's computers are rarely isolated. Instead, they are implicitly integrated in local or wide area networks (LAN or WAN) and most of the computations require them to communicate with each other and share resources (hardware and data). In the near future large-scale Internet-based applications will rely on massive interconnection of heterogeneous and distributed computing facilities that are potentially mobile (with frequent connections and disconnections that have to be handled) and unreliable (with different types of failures that may impact the program behavior).

In this general setting, the ADEPT research group aims at addressing some fundamental problems arising from the design of distributed applications. Solutions to these essential problems (which are not tied to a particular application instance) are at the heart of many generic services that need to be offered by systems or middlewares.

Several topics related to distributed computing, such as observation, synchronization, data consistency, fault tolerance and security have started to be tackled by the research community over the last decades. As a practical consequence, many resulting significant advances are now integrated in mature distributed platforms to support distributed applications. All these works have led to the identification of powerful abstractions and the design of numerous algorithmic solutions. Clear and unambiguous specifications of fundamental problems have been proposed and unanimously adopted by the whole community. A general approach based on the idea of specifying both the problems and the environments is now widely accepted. Most of the problems'

specifications are based on a classical decomposition of the properties into three classes: safety properties which ensure that something bad does not happen, liveness properties which ensure that something good eventually happens and timeliness properties which characterized critical real-time distributed services. Given a particular problem, its study requires to have a complete knowledge of the underlying distributed computing environment (communication primitives, model of synchrony, failure model, mobility and evolution, ...). Roughly speaking, modeling the computation system allows the identification of the minimal assumptions under which the correctness of a distributed algorithm can be established.

In a distributed system, a problem can be solved easily in a particular model while it can be proved to have no solution in a weaker model. Therefore, one of our research objectives is to determine the borderline between the problems that are solvable and the problems that are not solvable in a particular environment. When an impossibility result is identified, either a weaker problem or stronger assumptions have to be considered. On the contrary, when multiple solutions to a same problem can be found, two other objectives consist on one hand in understanding the relationships between them (classification setting) and on the other hand, in comparing them (definition of quality criteria). All the correct solutions can be distinguished by additional metric or subjective criteria such as minimality of the set of necessary and sufficient conditions, efficiency (time and message cost), robustness, maintainability, scalability, extensibility, flexibility, accuracy, compactness, simplicity, clarity or elegance. For many problems, the definition of an optimal trade-off cannot be done statically by a compiler. Therefore, providing adaptive strategies to take advantage of the existence of various algorithmic solutions and thus to always benefit from the most appropriate one is for us a promising research direction.

Even if past advances in distributed computing had made it possible to run nowadays complex distributed applications, some aspects still require further investigations whereas new market demands and technological progress are introducing new challenges, requiring constant investment in research and development. In particular, dependability, mobility, adaptability, flexibility and size factors are new challenges that have now to be overcome. New generations of middlewares have to provide high level services that mask the inherent difficulties of a distributed system which, in the worst case, can be large, heterogeneous, asynchronous, unreliable, and mobile. The definition of a virtual machine, independent of the underlying architectures, aims to reduce the cost and the time required to build new softwares.

In this context, the ADEPT research group follows two main objectives.

- Our first objective is to reach a deeper understanding of fundamental problems that arise in distributed computing. Our scientific contributions aim at meeting the new challenges confronting the designer of distributed algorithms. Therefore, during the study of a particular problem, we make the choice to combine several unfavorable assumptions together (asynchronous systems, unreliable systems, large scale systems, frequent connections and disconnections, ...).
- Our second objective is to validate and to promote the use of the algorithmic solutions we have designed. For this, we conduct experimental evaluations by developing middleware services that integrate our know-how and experience in distributed computing. This prototyping activity leads us to consider technical and operational problems as well as methodological issues. The feedback that we get helps us to define new directions in our research activity. The main benefit is to maintain a close link between our research activities on fundamental problems and the new advances continuously made in the field of software engineering. This is of particular importance to be able to provide flexible and adaptive solutions.

3. Scientific Foundations

3.1. Introduction

The researches conducted in the ADEPT research group are focusing on four main topics.

- First, we are continuing our study of the consensus problem, a problem shown to be at the core of agreement problems in distributed systems. New important results have contributed to strengthen the knowledge on this subject accumulated by the team over the last decade.
- Second, we are interested in fault tolerance issues and we design and develop middleware services to cope with dependability problems. In our approach, these high level services are reduced to agreement problems solved in a modular, flexible and adaptive manner. We consider mainly asynchronous systems because the combination of asynchrony and failure occurrences actually creates an uncertainty on the state of the application as perceived by a process (uncertainty of failure detection and the consequential impossibility of reaching agreement).
- When we address one of the two above topics, we refer to a (synchronous or asynchronous) distributed system composed of a set of well-identified processors. At any time, the identity and the localization of all the cooperating processors is known. A third research topic aims at defining new abstractions to cope with large-scale systems where frequent connections and disconnections occur. All proposed solutions can be used in peer-to-peer systems.
- Finally, the notion of time is at the core of our last research topic. Our goal is to study how time can be perceived and used in algorithmic solutions (in particular through timeout mechanisms or timeliness constraints). The goal is to determine how and at which steps in the design process of a real-time application the concept of time has to be considered.

3.2. Consensus in Distributed Systems

Key words: *distributed computing*, *agreement problem*, *consensus*, *failures*, *failure detector*, *condition-based approach*.

In many distributed applications, all the processes that cooperate to achieve a common goal have to share a common view of the state of the system. To build such a common view, processes have to execute an agreement protocol during which each process proposes its own partial view and gets a final value which must be the same for every one. Among all the agreement problems (atomic commit, atomic broadcast, election, membership, etc), the consensus problem is the simplest paradigm. Unfortunately, it has been shown that the consensus problem has no deterministic solution in a purely asynchronous distributed system. To design correct agreement protocols, one needs either to weaken the problem or strengthen the underlying system by adding synchrony assumptions.

The *consensus* problem is a central paradigm of fault-tolerant distributed computing informally stated as follows. Each process of a set of n processes proposes a value, and each non-faulty process has to decide a value (termination) in such a way that a decided value is a proposed value (validity) and the non-faulty processes decide the same value (agreement). *Uniform consensus* is a stronger problem in the sense that it requires that no two processes decide distinct values (uniform agreement). So, uniform consensus prevents a faulty process from deciding differently from the non-faulty processes. Consensus is important for several reasons. Consensus is an abstraction of the agreement part of several fault-tolerant distributed computing problems. It constitutes a basic building block on top of which solutions to those problems can be designed. Consensus is also important because of its relation to problem solvability. It has been shown that some distributed agreement problems can be solved in some system (with a given fault model) if, and only if, consensus can be solved in that system (e.g., atomic broadcast and consensus are equivalent problems in asynchronous distributed systems prone to process crashes). Yet another justification of the importance of consensus lies in the principles and mechanisms that underlie the design of most consensus protocols.

Understanding those basic principles and mechanisms can provide system designers who have to cope with unreliable underlying distributed systems with a better understanding on the way the unpredictability and uncertainty inherent to those systems can be mastered.

Asynchronous distributed systems are characterized by the absence of bounds on processing time and message transfer delay. Differently from synchronous systems, consensus (hence, also uniform consensus) cannot be solved in asynchronous systems prone to even a single crash failure [57]. This impossibility result constitutes another motivation to study the consensus problem, as it questions the nature of the borderline separating synchronous systems from asynchronous systems. Several approaches have been proposed to allow for the design of deterministic protocols that circumvent this impossibility result (we do not consider here probabilistic protocols such as the one of Ben-Or). Roughly speaking, there are two approaches. The first one consists in enriching the underlying asynchronous system with an additional mechanism such as a leader oracle (e.g., Paxos) or a failure detector as defined by Chandra and Toueg so that consensus can be solved in the resulting augmented system. The second one (called condition-based) consists in determining sets of input vectors (an input vector has one entry per process, each entry containing the value proposed by the corresponding process) for which consensus can be solved despite up to t process crashes.

3.3. Fault Tolerance in Distributed Systems

Key words: *distributed computing*, *fault tolerance*, *agreement problem*, *group communication*, *atomic broadcast*, *failure*, *flexibility*, *adaptability*.

The study of fault-tolerant mechanisms based on processor redundancy is an active research topic of our research group. Defining efficient solutions to manage sets of replicas located on different machines requires to have a precise knowledge of the consistency problems that arise in distributed systems. Reaching an agreement among the different replicas of a critical server is one of the crucial problems we are faced to. Solving such basic problems allows to provide high level services as for example group communication services which are essential to manage the evolution of the set of replicas. The type of the faults (simple crash failures, timing and malicious failures), the characteristics of the environment (synchronous or asynchronous), the way the set of replicas evolves (size, speed of the replicas' changes, partial or complete knowledge of the changes) are the main parameters that we need to consider to converge to an homogeneous and adaptive set of replication services.

Building fault tolerant applications in an asynchronous distributed system is a major challenge and a complex endeavor. Redundancy is a key design principle to achieve this goal by masking failures. Two schemes, namely passive and active replication, are commonly used to replicate servers that fail independently. In both cases, the group abstraction is particularly interesting: different copies of a critical server form a group that is in charge of executing the requests submitted by clients. Intuitively, if one machine hosting a replica fails, the service will still be available (thanks to the other replicas). In a distributed system, a group consists of a finite set of processes that communicate either together or with external processes (open group). A request to the group is always broadcast to all the members. Regarding message ordering, one primitive, namely atomic broadcast, is of particular interest. It ensures that all sent messages are delivered in the same order. The membership of a group can change according to the desire of its current members to leave the group, to the desire of external processes to join the group, or to the underlying system behavior (crashes, disconnections, etc.). Processes belonging to a same group have to maintain a common view of the current composition of the group. This first problem is addressed by the group membership service. With regard to all these problems, processes belonging to the same group have, from time to time, to reach a unanimous decision. Using a solution to the consensus problem, as a building block to implement atomic broadcast and group membership, is an interesting approach that exhibits many advantages. The impossibility result of Fischer-Lynch-Paterson (FLP) can be circumvented at this level.

Many protocols have been designed to cope with crash failures. But such failures constitute just one particular type of failures. Considering more severe failures (like omission or malicious failures) is a major challenge. One possible approach consists in defining new protocols specially designed to consider these new

kinds of failures. A second approach consists in adding external components to an existing protocol (that already tolerates crash failures) so that this protocol is able to support more severe failures. In that case, no modification of the original protocol is necessary.

Part of our research consists in designing flexible modular and adaptive group communication services that can tolerate various kind of failures. With this goal in mind, we consider component-ware approach and try to benefit from the new advances in software engineering when we design adaptive algorithmic solutions.

3.4. Large Scale, Dynamic Open Systems

Key words: *distributed computing*, *large scale systems*, *mobility*, *peer-to-peer systems*, *ad-hoc networks*, *sensor networks*, *virtual reality*, *self-organization*.

The major feature of all recent scalable systems is their extreme dynamism in structure, content and load: nodes are continuously joining and leaving the system, there is no central entity in charge of their organization and control, and there is an equal capability, and responsibility entrusted to each of them to own data. To cope with such characteristics, these systems must be able to spontaneously organize toward desirable global properties.

Different kinds of self-organization can be addressed. In peer-to-peer systems, self-organization is handled through protocols for node arrivals and departures based on a fault-tolerant overlay network as in Pastry, or through localization and routing infrastructure as in OceanStore. In ad-hoc networks, solutions have been proposed for a self-organizing public-key management system that allows users to create, store, distribute, and revoke their public keys without the help of any trusted authority or fixed server. The idea of self-organization or related variants as self-configuration, self-healing or reconfiguration was studied in [62] or [60]. The former work proposes the concepts of self-healing and self-configuration in wireless networks, while the latter one focuses on the concept of reconfiguration of a metamorphic robotic system with regard to a goal configuration. Finally, one could argue that the self-stabilization framework [54][56] specifies the self-organization of highly dynamic networks. Actually, the nature of the dynamic networks cannot fit the self-stabilization process essentially because self-stabilization deals with systems affected by infrequent and transient faults, which make them capable to converge toward a stable pre-defined configuration. This is orthogonal to self-organizing systems which should be relatively insensitive to perturbations or faults, and should have a strong capacity to restore themselves, essentially because these systems thrive on fluctuation or “noise”.

3.4.1. Peer-to-Peer systems

The Peer-to-Peer phenomenon started in 1999 with the launch of Napster, a service that allowed users to share audio files (MP3s). While offering a distributed storage model, Napster relied on a centralized server in charge of storing the index of files available within the user community. Similar systems such as Gnutella, FreeNet, KaZaA and others have been developed. Unlike Napster, these systems have adopted a completely decentralized approach both for the lookup service and for the files storage (making difficult the censorship...), and have evolved from audio to video and software content. While decentralized, these systems have scaling problems. For example, Gnutella floods searches until a time to live (TTL) query parameter is reached, Freenet uses a random walk based algorithm that can fail to retrieve existing files, etc. To cope with this problem, current systems such as CAN, Chord, Pastry, Tapestry, Viceroy, etc. have been developed supporting hash table functionalities (DHT systems) - mapping keys to values on Internet-like scales. The core of these DHT systems is a named-based routing algorithm. The DHT nodes form an overlay network in which each node has several neighbors. When a lookup(key) operation is executed, the lookup is routed through the overlay network to the peer responsible for that key. Each of the proposed DHT systems employs a different routing algorithm that differs in many respects, but the main one is their difference in terms of routing geometry: CAN (when the dimension is taken to be $log(n)$) routes along a hypercube, Chord, Pastry, and Tapestry routes along a ring, and Viceroy uses a butterfly network. These geometries have differing degrees of flexibility in choosing their neighbors and their next hop.

3.4.2. Cellular/Ad-hoc and Sensor Networks

In Cellular Networks, a base station manages an area of world called *cell*. Mobile hosts are connected with the base station of the cell to which they belong. Moreover the mobile hosts cannot establish a direct connection between them, they rely on a base station to communicate. Ad-hoc Networks differ from Cellular Networks as it relies on no specific infrastructure. Each mobile host can send messages to the nodes within its transmission range. Moreover, it can route messages on behalf of others. That is, two mobile hosts can communicate either over a direct wireless link or over a sequence of wireless links including one or more intermediate nodes. Sensor networks are systems in which numerous compute and sensing devices are distributed within an environment to be studied. Sensor networks have been proposed for a range of engineering, scientific and defense application. While some sensor networks have static sensor positions, dynamic sensor networks include mobile nodes and wireless communication between them.

3.4.3. Shared Virtual Reality Systems

In general, a Virtual World means a technology for moving through and interacting with a three-dimensional computer-generated environment such that the experience is perceived to be real. Shared Virtual Environments are inhabited by interconnected *Entities* driven by users (avatars) or by computer (virtual objects). Entities, characterized by a position in the virtual world, enter and leave the world, move from one virtual place to another and interact in real-time. In the real world, which virtual environments emulate, entities have a limited area of interest. Thus, a person on foot typically has an area of interest of only several hundred meters. So, in virtual environments, entities need only to be aware of each other within immediate surroundings. In virtual reality systems a specific problem to be solved is the optimal *partitioning of the shared virtual world*.

3.5. Time in Distributed Computing

Key words: *real-time*, *timeliness properties*, *asynchronous model*, *synchronous model*.

Many distributed applications are characterized by real-time constraints. Obviously, such applications have to be executed on a synchronous system that exhibits strong timing assumptions. Yet, during the first phases of the design process, an asynchronous model of computation can be considered.

We want to explore several ideas related to the concept of time from a middleware perspective. Time is often a key concept for application developers and in many application domains Timeliness is an important QoS attribute that has to be addressed by the next generation middleware. Even, when the goal is not to build real-time applications, timing behaviors are assumed to be more or less predictable. For example, to cope with crash failures, unreliable failure detectors mechanisms based on timeouts are used.

Due to the absence of a shared physical clock, time measurement is a difficult problem in a world wide area network. In the past, two opposite strategies have been explored. On one hand, synchronization algorithms have been proposed to tackle the problem of constructing a global time. On the other hand, much work was done on replacing the notion of global time by that of causality: logical clocks (Lamport clocks and Vector clocks) have been defined to capture the dependencies information.

Nowadays, large-scale distributed infrastructures are composed of a huge amount of mobile and fixed devices. In such a context, frequent disconnections (that are either intentional or accidental disconnections) prevent the computation of a single view of the whole system. For example, in peer-to-peer systems, a processor can only manage a partial view of the system (limited to its neighborhood) otherwise the main part of its activity will be devoted to maintaining an up-to-date knowledge of the entire system. This restriction prohibits the use of classical synchronization algorithms as well as the use of vector clocks that are based on the a priori knowledge of all the involved computers. Also when trying to build composable services, slight inaccuracies in various components can easily build up causing violations of the QoS specification for the composed service.

Our main objective is to investigate the concept of time in face of the new challenges which have emerged in the field of distributed computing. More precisely, following the strategies used in the past, we want to address three different research directions by answering three questions:

Time-free approach: Is it possible to avoid the use of timing assumptions whenever an application has no explicit timeliness requirements? As mentioned previously, a failure detector mechanism is a typical example of a problem that requires usually the definition of timeouts to be solved. Some results obtained recently demonstrate that relying on simple stability properties is enough to solve this problem without having to tune timeout periods. The objective is to avoid all the troubles due to a bad estimation of the timeout value.
Late-Binding approach: Is it possible to postpone the use of timing assumptions until the last steps of an application's design ? Component-based approaches are now used to develop software on top of middleware. Therefore, it is important to create components whose liveness and safety properties can be proved independently of the timing properties of the environment in which they will be plugged. Timeliness requirements have to be considered as late as possible during a timing and scheduling analysis.
Mixing approach: As mentioned previously, in large-scale distributed infrastructures, processors will have to be managed by small groups in order to cope with scalability issues. Processors will be members of the same group if they have nearby physical locations, if they share some data or if they have recently interacted together. Depending on the criteria (physical and logical), a process can belong to several groups at a given time and switch from one group to another. In each group, there can be a different concept of time and consequently new concepts of synchronization have to be studied to address time composability and interoperability issues.

4. Application Domains

4.1. Software for Telecommunication and Space Industries

Key words: *distributed computing*, *telecommunication*, *space*, *fault tolerance*, *group communication*, *middelware*, *real-time application*.

The potential benefits of distributed applications are more and more apparent in many economic sectors. Yet a broad set of open problems have still to be faced. Through our contacts with telecommunication and space industries, we observe in these two different fields a growing interest in distributed computing that lead these partners to express more and more complex requirements. More precisely, the adequacy between the properties ensured by their applications (that are getting increasingly stronger) and the assumptions about their systems (that are getting more and more weaker) becomes questionable.

In these context, the activity of the ADEPT research team aims at contibuting to the emergence of new fundamental services which can be integrated within systems and middlewares. More precisely, we put the stress on the following user's requirements:
Fault-tolerance: In many application domains, distributed entities have to interact with each other in a robust manner. In that case, fault tolerance is a key requirement. a broad range of failures (from benign failures up to malicious failures) have to be tolerated.
Flexibility and adaptativity: The new generation of distibuted services has to be adaptive. To achieve these goal, algorithmic solutions have to benefit from the recent advances in software engineering (componentware approach, $ldots$) and vice-versa.
Real-Time: The way timeliness properties are adressed is far from being optimal. New strategies have to be defined to improved the performance of the solutions, to simplify their development and to facilitate the combination of timing requirements with other non-functional requirements (fault-tolerance, ...).
Large scale dynamic systems: In large systems, applications involving dynamic and mobile sets of participants start only to emerge. Algorithmic solutions developed in a more static context cannot be simply adapted. New approaches and new abstractions are necessary.

Prototyping activities performed during 2003 aims at developping and experimenting a fault tolerant group communication service that is used on one side to implement an active replication service and on the other side to provide a task allocation mechanism in a GRID. Our goal is not to focus on a particular application but rather to consider generic services that have an universal dimension.

5. Software

5.1. Eden : a Group Communication Service

Participants: Michel Hurfin, Jean-Pierre Le Narzul, Xiaojun Ma, Frédéric Tronel.

Key words: *distributed computing*, *group communication*, *middleware*, *fault tolerance*.

Eden is a configurable group-based toolkit, which aims at developing reliable, object-oriented, distributed applications. Eden implements a group communication system using agreement components. Atomic broadcast, group membership and view synchrony are considered as agreement problems, which can be reduced to a basic problem, called the Consensus problem. Through the use of a "generic agreement component", implementing a generic consensus algorithm, and through some adequate component compositions, Eden provides an elegant and modular way of implementing the fundamental services of a group communication system.

OpenEden is an implementation of the Fault Tolerant CORBA specification based on the use of the Eden group communication framework. It relies on an interception approach (portable interceptors) for supporting active replication of CORBA objects. At the client side, a request is intercepted by a client interceptor which redirects the request to a gateway; this gateway is in charge of managing the communication with the group of CORBA objects. At the server side, requests are also intercepted and reordered thanks to the use of the EDEN toolkit. The OpenEDEN approach is portable, transparent (for the programmer) and non-intrusive for the ORB system.

5.2. Paradis: Resource allocation in a Grid

Participants: Michel Hurfin, Jean-Pierre Le Narzul, Julien Pley, Philippe Raïpin Parvedy.

Key words: *distributed computing*, *grid computing*, *task allocation*, *middleware*, *fault tolerance*.

PARADIS is an "operating system" for a Grid dedicated to genomic applications. Genomic applications are time-consuming applications that can be splitted into a huge number of independant tasks. PARADIS is used to reliably allocate ressources to these tasks.

In PARADIS, we consider that the network which is globally asynchronous, is composed of synchronous subnetworks called *domains* (in practice, these domains correspond to LANs). To improve the fault tolerance and the efficiency of computations on the Grid, we try to benefit as much as possible from the synchronous properties of communications within a domain and to avoid as much as we can the communications within domains. To avoid a flood of the Grid, only one node per domain is allowed to communicate with the other domains. This node is called the *proxy*. In order to provide an easy access to the Grid from anywhere, the applications can be launched through web portals.

The implementation of PARADIS relies on the ADAM library. Agreement components are used by proxies to share a common view of the evolution of the Grid. Decisions are used to solve, despite failures, the group membership problem and the resource allocation problem.

6. New Results

6.1. Consensus in Distributed Systems

Participants: Emmanuelle Anceaume, Roy Friedman, Eric Mourgaya, Achour Mostéfaoui, Philippe Raïpin Parvedy, Michel Raynal, Matthieu Roy.

Key words: *distributed computing*, *consensus*, *crash failures*, *Byzantine failures*, *failure detector*, *oracles*, *accuracy property*, *condition-based approach*.

6.1.1. Failure Detectors

Unreliable failure detectors provide information on process failures. On the one hand, failure detectors allow to state the minimal requirements on process failures detection that allow to solve problems that cannot be

solved in purely asynchronous systems. But, on the other hand, they cannot be implemented in such systems: their implementation requires that the underlying distributed system be enriched with additional assumptions. The usual failure detector implementations rely on additional synchrony assumptions (e.g., partial synchrony). In [35], we propose a new look at the implementation of failure detectors and more specifically at Chandra-Toueg's failure detectors. The proposed approach does not rely on synchrony assumptions (e.g., it allows the communication delays to always increase). It is based on a query-response mechanism and assumes that the query/response messages exchanged obey a pattern where the responses from some processes to a query arrive among the $(n-t)$ first ones (n being the total number of processes, t the maximum number of them that can crash, with $1 \leq t < n$). When we consider the particular case $t = 1$, and the implementation of a failure detector of the class denoted $\diamond\mathcal{S}$ (the weakest class that allows to solve the consensus problem), the additional assumption the underlying system has to satisfy boils down to a simple channel property, namely, there is eventually a pair of processes (p_i, p_j) such that the channel connecting them is never the slowest among the channels connecting p_i or p_j to the other processes. A probabilistic analysis shows that this requirement is practically met in many asynchronous distributed systems.

Additionally to this implementation, we studied stacking implementation of failure detectors. Let the scope of the accuracy property of an unreliable failure detector be the minimum number (k) of processes that may not erroneously suspect a correct process to have crashed. Usual failure detectors implicitly consider a scope equal to n (the total number of processes). Accuracy properties with limited scope give rise to the classes of failure detectors called $\mathcal{S}_k$ and $\diamond\mathcal{S}_k$. We have investigated the following question: "Given $\mathcal{S}_k$ and $\diamond\mathcal{S}_k$, under which condition is it possible to transform their failure detectors into their counterparts with unlimited accuracy ($\mathcal{S}$ and $\diamond\mathcal{S}$)?". In [14], we have answered this question in the following way. We first present a particularly simple protocol that realizes such a transformation when $t < k$ (where t is the maximum number of processes that may crash). Then, we show that there is no reduction protocol when $t \geq k$.

Finally, we tried to address the issue of the relative power of $\diamond\mathcal{P}$ (*eventually perfect*) and $\diamond\mathcal{S}$ (*eventually strong*). Both classes include failure detectors that eventually detect permanently all process crashes, but while the failure detectors of $\diamond\mathcal{P}$ eventually make no erroneous suspicions, the failure detectors of $\diamond\mathcal{S}$ are only required to eventually not suspect a single correct process. In such a context, we have addressed in [52] the following question: "Are there one-shot agreement problems that can be solved in asynchronous distributed systems with reliable links but prone to process crash failures augmented with $\diamond\mathcal{P}$, but cannot be solved when those systems are augmented with $\diamond\mathcal{S}$?" Surprisingly, we show that the answer to this question is "no". An important consequence of this result is that $\diamond\mathcal{P}$ cannot be the weakest class of failure detectors that allows to solve one-shot agreement problems in unreliable asynchronous distributed systems.

Failure detectors are a particular type of oracles. Systems can be equipped with other appropriate oracles to circumvent the FLP impossibility result. Several oracles (unreliable failure detector, leader capability, random number generator) have been proposed, and consensus protocols based on such ad-hoc oracles have been designed. In [28], we present a generic consensus framework that can be instantiated with any oracle, or combination of oracles, that satisfies a set of properties. This generic framework provides indulgent consensus protocols that are particularly simple and efficient both in well-behaved runs (i.e., when there are no failures), and in stable runs (i.e., when there is no failure during the execution although some processes can be initially crashed). In those runs, the protocols terminate in two communication steps (which is optimal). Indulgence means that the resulting protocol never violates its safety property even when the underlying oracle behaves arbitrarily. Interestingly, the protocol can also allow processes to decide in one communication step in some specific configurations.

6.1.2. Evaluation of the condition-based approach

As the consensus safety properties have never to be violated (this is a "minimum" requirement if we want to solve consensus!), the FLP impossibility result basically says that an asynchronous consensus protocol cannot always guarantee the termination property. So, given a consensus protocol, an interesting question is "Does this protocol terminates very often?". The quality of service of failure detectors have been studied and the metrics investigated are "how fast" failures are detected and (2) "how well" false detections are avoided. In a

recent paper [36], we tried to answer an analogous question in the context of the condition-based approach. In some sense, a condition-based consensus protocol does "its best" with respect to termination: it guarantees termination when the input vector belongs to the condition and there is no more than t crashes, or when there is no crash at all (whatever the input vector). In the other cases it can sometimes terminate according to the failure pattern, the perception each process has of the failure pattern, and the fact that the input vector is not "too far" from the condition. (A condition could be seen as a "consistent heuristics" with respect to termination.) So, the aim is to evaluate the "quality" of the condition-based approach with respect to the consensus termination property.

To answer the question "Does a condition-based protocol terminate very often?", we consider the following parameters: λ (probability that a process crashes), n (total number of processes), and t (an a priori fixed value). Usually, protocols explicitly use the parameter t in their code. This allows a process to wait for values from $n - t$ processes, and so to construct a partial view of the input vector. But, t is only a parameter assumed to be an upper bound on the number of faulty processes. So, when more than t processes crash, processes can block forever, despite the fact that they could have terminated if they had considered a greater value for t. To summarize, there are two causes for a condition-based protocol not to terminate: (1) there are more than t process crashes, or (2) the input vector does not belong to the condition. The main lessons learned by answering these questions are the following:

- Both the conditions $C1$ (which tries to decide the maximal proposed value) and $C2$ (which tries to decide the proposed value that appears the most often) are good with respect to the protocol termination property. Moreover, $C1$ is better than $C2$.
- Given a condition C and a pair (n, λ) characterizing a system, it is possible to compute a value of t that maximizes the probability that the protocol terminates. That value of t is usually much smaller than the upper bound $\lfloor (n-1)/2 \rfloor$.

6.1.3. Conditions in a synchronous systems

Synchronous systems provide upper bounds on processing time and message transfer delay. Those bounds allow both consensus and uniform consensus to be solved for any value of t ($t < n$), the upper bound on the number of processes that can crash. A synchronous consensus protocol proceeds by successive rounds. During each round, the processes that have not crashed exchange messages. They stop and decide when they have reached a round such that they know they all have converged to the same decision value. It has been shown that both consensus and uniform consensus require $t + 1$ rounds in the worst case. The fact that, in a practical setting, failures are rare has motivated the design of early deciding protocols. These protocols are characterized by the fact that the number of rounds they require depends on the number f of processes that have actually crashed during the run ($0 \leq f \leq t$). It has been shown that early deciding consensus requires at least $f + 1$ rounds, while early deciding uniform consensus requires at least $\min(f + 2, t + 1)$ rounds.

In [37], we investigating the use of the condition-based approach to solve the uniform consensus problem in synchronous systems, and discovered that t-acceptable conditions that allow to solve consensus in asynchronous systems are exactly the conditions that allow to solve consensus in a synchronous system in a single round when the input vector belongs to the condition and if additionally $f = 0$. Otherwise, the condition-based uniform consensus protocol terminates in two rounds when the input vector belongs to the condition, whatever the value of $f \leq t$. Finally, if the vector does not belong to the condition, the protocol terminates in $t + 1$ rounds. A second step protocol combines early decision and a condition. This protocol enjoys the previous termination property (termination in one or two rounds when the input vector belongs to the condition), and terminates in at most $\min(f + 2, t + 1)$ otherwise.

While the first protocol shows that the t-acceptable conditions can be used to expedite decision in a synchronous system, the second protocol shows that this advantage can be combined with the early deciding approach without additional cost. This shows that synchronous systems can benefit from the condition-based approach. These results establish a bridge relating synchrony and asynchrony. Namely, the conditions that

allow to solve uniform consensus in an asynchronous system are exactly the conditions that allow to solve it optimally (in terms of number of rounds) in a synchronous system.

6.1.4. Byzantine faults

We have considered the Consensus problem in asynchronous distributed systems where up to t processes can exhibit a Byzantine behavior. A Byzantine process can deviate arbitrarily from its specification whereas a crashed process simply halts executing its code. A way to solve the consensus problem in such a context consists of enriching the system with additional oracles that are powerful enough to cope with the uncertainty and unpredictability created by the combined effect of Byzantine behavior and asynchrony. Considering two types of such oracles, namely, an oracle that provides processes with random values, and a failure detector oracle, we present in [53] two families of Byzantine asynchronous consensus protocols. Two of these protocols are particularly noteworthy: they allow the processes to decide in one communication step in favorable circumstances. The first is a randomized protocol that assumes $n > 5t$. The second one is a failure detector-based protocol that assumes $n > 6t$. These protocols are designed to be particularly *simple* and *efficient* in terms of communication steps, the number of messages they generate in each step, and the size of messages. So, although they are not optimal in the number of Byzantine processes that can be tolerated, they are particularly efficient when we consider the number of communication steps they require to decide, and the size of the messages they use. Moreover, the proposed protocol do not require "heavy" mechanisms such as "message proofs", certificates, or any kind of application level signatures. In that sense, they are practically appealing.

6.2. Fault Tolerance in Distributed Systems

Participants: Emmanuelle Anceaume, Michel Hurfin, Jean-Pierre Le Narzul, Xiaojun Ma, Julien Pley, Philippe Raïpin Parvédy, Frédéric Tronel.

Key words: *distributed computing*, *fault tolerance*, *group communication*, *crash failure*, *Byzantine failure*, *component*, *grid computing*.

6.2.1. A Component Approach for Reliable Distributed Programming

Componentware is a promising paradigm for helping developers to build distributed applications. The idea behind this paradigm is to allow a developer to configure and assemble components rather than writing complex, specific and somewhat obscure code that uses system services to allow objects to inter-operate. Based on this paradigm, we have designed ADAM, a library of agreement components for reliable distributed programming.

ADAM is based on a generic and adaptive solution to the consensus problem that can be customized to cope with the characteristics of the environment as well as the properties of the reliable distributed abstractions that have to be ensured.

The central component of the ADAM library is the Generic Agreement Component (GAC); it is used by a range of components implementing the fundamental agreement services mentioned above. The GAC implements a generic fault-tolerant consensus algorithm. The originality of GAC lies in the "versatility" parameters used to customize the consensus algorithm and adapt it to a particular agreement protocol. To be operational, GAC has to be instantiated through the definition of a concrete agreement component. Thanks to a set of methods provided by this component, the behavior of the agreement algorithm can be tuned to fit the exact needs of the agreement problem to be solved. The ADAM library currently includes the most important components for reliable distributed programming: group membership management, atomic broadcast, view synchrony.

Details about this activity can be found in [27].

6.2.2. Design of a Fault Tolerant CORBA Architecture

OPEN EDEN is a fault tolerant CORBA architecture based on agreement components. A critical CORBA service is made reliable by replicating a CORBA object on different sites of the distributed system. The set of

replicas forms a CORBA object group. EDEN provides a group communication system to OPEN EDEN, using agreement components from the ADAM [58] library.

The design of OPEN EDEN has been driven by the desire to use only portable techniques to implement the interaction between the EDEN system and CORBA. We selected portable interceptors [61][55], a mechanism specified by the OMG that allows to transparently add service code to the ORB. Compared to other approaches based on system-level interceptors (like in Eternal [59]), our approach is more portable since it can be easily migrated to different architectures. Transparency (for the application programmer) is another benefit of our approach; requests to a replicated service are transparently intercepted and redirected to the EDEN group communication system. This work is described in the paper entitled "Open Eden: a Portable Fault Tolerant CORBA Architecture", presented at the ISPDC conference.

6.2.3. Grid computing

The major aim of a Grid is to federate several powerful distributed resources within a single virtual entity which can be accessed transparently and efficiently by external users. As a Grid is a distributed and unreliable system involving heterogeneous resources located in different geographical domains, fault-tolerant resource allocation services have to be provided. In particular, when crashes occur, tasks have to be reallocated quickly and automatically, in a completely transparent way from the users' point of view.

Four members of the ADEPT team have designed and developed PARADIS, a system based on the consensus building block of Adam and implemented in a Grid dedicated to genomic applications. These time-consuming applications can be split up into a huge number of independent tasks which can be allocated independently on different domains. Load strategies can be plugged to PARADIS thanks to a bid mechanism used for the task allocation.

6.3. Large Scale, Dynamic Open Systems

Participants: Emmanuelle Anceaume, Roy Friedman, Maria Gradinariu, Eric Mourgaya, Michel Raynal, Matthieu Roy, Gwendal Simon, Frederic Dang Ngoc.

Key words: *distributed computing*, *large scale systems*, *mobility*, *peer-to-peer systems*, *ad-hoc networks*, *sensor networks*, *virtual reality*, *self-organization*.

In the context of scalable and highly dynamic systems, we have propose a formal specification of the self-organization notion [21]. For the best of our knowledge, this has never been done before in the area of scalable and dynamic systems despite a non negligible use of this term. This specification is based on the three principles that govern dynamic systems. The first one concerns the exchange of information or resources with the environment (components are possibly capable to infinitely often retrieve new information/resources from components around them). The second one is the dynamics of these systems (components have the ability to move around, to leave or to join these systems based on local knowledge). The third principle is the specificity of the components: Among all components of the system, some have huge computation resources, some have large memory space, some are highly dynamic, some have broad centers of interest. By looking at these systems as a mass of components, we obviate any differences that might exist between individual components, differences which make the richness of these systems.

All these tenets have as common seed the locality principle, that is some range of effect, both in terms of interaction and knowledge. We formalize this idea, leading first to the notion of *local self-organization*. Intuitively, a locally self-organizing system should force components to be adjacent to components that improve or at least maintain some property or evaluation criterion. By imposing the system to be locally self-organizing at all its nodes and by ensuring that despite the frequent and unpredictable connections/disconnections, the knowledge within the system progressively enriches, we then formalize the notion of *self-organization*.

In [21], we also consider the location data problem in peer-to-peer systems. As described hereabove, two main approaches have been proposed to tackle this challenging problem, the flooding approach and the distributed hash table (DHT) approach. We propose a different approach which exploits the self-organization property of the system to advance our performance objective of minimizing data localization latency. We build

the data localization service on top of a multi-criteria self-organization layer. This layer is made up of three sub-layers, each of them dynamically organizing the topology of the system according to a given criteria. Such an approach enables nodes to decide whom to contact and when to add or drop a connection based on local knowledge only. By resorting on this layered organization, the retrieval of a data having the maximal similarity with the data owned by a given node needs $O(1)$ computation steps.

Another problem on which focused our research is data dissemination in mobile dynamic systems. The publish/subscribe method is the most inclusive strategy to establish communication between the information providers (publishers) and the information consumers (subscribers). In [22] we study the publish/subscribe problem in the particular context of peer-to-peer networks. We give a formal definition of publish/subscribe systems. We then use the publish/subscribe communication paradigm to design deterministic protocols (topic and content-based) for peer-to-peer networks. Our protocols are designed on top of an innovative information dissemination scheme, and can cope with the anonymity and mobility of both publishers and subscribers, weak-connectivity, and polarization, which are some of the characteristics of peer-to-peer networks. Moreover, in our solutions, every node could play a role of both publisher and subscriber. The algorithms are designed completely independent of the underlying routing substrates. The key advantage of our protocols is that they are scalable without additional re-organization cost.

In [25] we pool together the mobile systems and analyze them from different angles including architecture and computability aspects. We show that mobile systems (i.e. cellular systems, ad-hoc networks, peer-to-peer systems, virtual reality systems or cooperative robotics) are basically confronted with the same problems, hence it is useless to maintain the actual false barriers. Due to the important similarities between the different mobile systems we claim that the next logical step in building a common view is to design a model which conceptually should unify them by providing an abstract description of the parameters which distinguish these systems from classical distributed ones.

Actual implementations of Shared Virtual Reality systems are based on central servers. In [32], we present a totally distributed architecture where all entities participate to the creation of an adequate topology based on peer-to-peer network. We describe the algorithms involved in the management of the relationships between entities in this topology.

6.4. Time in Distributed Computing

Participants: Emmanuelle Anceaume, Michel Hurfin, Gérard Le Lann.

Key words: *real-time*, *timeliness properties*, *asynchronous model*, *synchronous model*.

In the context of a contract with CNES, we have started to investigate the problem of designing distributed services characterized by both real-time and faut-tolerance requirements [48][49].

In [44], we examine how computer system problems can be derived from real application problems, with a particular focus on the relevance of some assumptions, especially those related to computational models. Then, we compare these models, ranging from pure synchronous to pure asynchronous semantics, to conclude that synchrony does not necessarily dominate asynchrony whenever one is concerned with real operational systems. The issue as to whether asynchronous solutions can be considered for designing and building real-time distributed dependable systems is addressed. A priori, time free solutions are antagonistic with proving timeliness properties. We show how to circumvent this apparent contradiction via the late binding principle. This principle, as well as drawbacks of synchronous solutions, are illustrated.

In [40], we present a systems engineering methodology for constructing certifiable real-time distributed systems. In the proposed approach, an architectural and algorithmic solution to an application problem is designed by considering the "weakest" models including the weakest asynchronous computational model and multimodal arrival model. Furthermore, timeliness properties are described using Jensen's benefit accrual predicates. Once a system solution is designed, timeliness properties are established by constructing necessary feasibility conditions that are expressed as non-valued predicates. The predicates are quantified and verified to produce the specification of a certified solution. We illustrate the approach by considering a packet transmission problem that desire soft timeliness. We present a certifiable solution to this problem that consists

of switched Ethernet, a soft real-time packet scheduling algorithm (that was previously developed), and feasibility conditions.

7. Contracts and Grants with Industry

7.1. ags Contract: Generic Architectures for Satellite Systems (2002-2004)

Participants: Emmanuelle Anceaume, Michel Hurfin.

Key words: *distributed computing*, *space*, *fault tolerance*, *real-time*.

With the collaboration of Gérard Le Lann (Inria Rocquencourt), C. Gallet-Delporte (LIAFA - Paris VII) and H. Fauconnier (LIAFA - Paris VII), we are studying the design of a generic architecture for the future satellites.

This work, supported by the CNES, started in 2002. By the end of 2003, we have proposed an architecture meeting the specifications of the CNES regarding the model of their system, the hypothesis, and the properties that have to be guaranteed.

This study aims at designing a middleware that enables to make the application design independent from the environment. This middleware has to take into account the specificities of the space constraints as well as alleviate the transition from a centralized design to a distributed one.

Finally, among all the requirements imposed by the CNES, hard real time and dependability (especially in presence of Byzantine failures) are the two critical ones.

7.2. speeral Contract (2002-2005)

Participants: Emmanuelle Anceaume, Maria Gradinariu, Eric Mourgaya, Matthieu Roy.

Key words: *telecommunication*, *distributed computing*, *large scale systems*, *mobility*, *peer-to-peer systems*, *ad-hoc networks*, *sensor networks*, *virtual reality*, *self-organization*.

It should be clear now that “traditionnal” distributed systems and dynamic and scalable distributed systems differ in many terms. Because of their extreme dynamicity in structure, content and load, we cannot model the behavior of the nodes with the traditional ones. We have to revisit the notion of faulty and correct behavior, as well as the notion of fault. Furthermore, we have to wonder whether agreement services make sense in such systems, and if yes, how to revisit them, in terms of specification and solution.

All these questions are studied in this collaboration with FT R&D. This study started at the beginning of 2003 and will last for 30 months.

8. Other Grants and Activities

8.1. National Project

8.1.1. ACI GénoGRID (2003-2004)

Participants: Michel Hurfin, Eric Mourgaya, Jean-Pierre Le Narzul, Xiaojun Ma, Julien Pley, Philippe Raïpin Parvédy.

In the context of the ACI GRID (Actions Concertées Incitatives - Globalisation des Ressources Informatiques et des Données) program, supported by the French ministry of Research, the GénoGRID project has started in December 2001 for a duration of three years. Two Irisa teams are involved in this project: ADEPT and SYMBIOSE.

The major aim of the project is to federate several powerful distributed resources within a single virtual entity which can be accessed transparently and efficiently by external users. As a Grid is a *distributed* and *unreliable* system involving heterogeneous resources located in different geographical domains, fault-tolerant resource allocation services have to be provided. In particular, when crashes occur, tasks have to be reallocated quickly and automatically, in a completely transparent way from the users’ point of view.

The Grid we consider is deployed over the Internet. Even if this network is globally asynchronous, it is composed of synchronous subnetworks called *domains* (in practice, these domains correspond to LANs). To improve the fault tolerance and the efficiency of computations on the Grid, we try to benefit as much as possible from the synchronous properties of communications within a domain and to avoid as much as we can the (asynchronous) communications between domains. In order to provide an easy access to the Grid from anywhere, the applications can be launched through web portals.

8.1.2. CNRS Specific Action: Distributed Algorithms and Applications (2003-2004)

Participants: Emmanuelle Anceaume, Michel Hurfin, Michel Raynal.

Two researchers of LIAFA, namely Carole Delporte-Gallet and Hugues Fauconnier, have proposed the creation of a national community of researchers working on distributed computing. This initiative will receive a support from CNRS during one year.

8.2. International Cooperations

8.2.1. China (Southeast University)

Participants: Michel Hurfin, Jean-Pierre Le Narzul, Xiaojun Ma, Julien Pley, Philippe Raipin Parvedy.

Following the end of the LIAMA project entitled "Fault Tolerant Corba" (2000-2002), strong relationships have been maintained with Professor Yun Wang of Southeast university (Nanjing). A new cooperation project has been set up during the visit in China by Michel Hurfin in March 2003. After he received his Ph.D. degree from Southeast university at the end of 2002, Xiaojun Ma has applied successfully for a post-doctoral position in our institute (grant from the research ministry). During 2003, he has worked on the design and the implementation of a group communication service.

8.2.2. United States of America (University of Santa Barbara)

Participants: Achour Mostéfaoui, Matthieu Roy, Michel Raynal.

A three year grant (july 2001/july 2004) from NSF and INRIA has been obtained for a cooperation with Professors D. Agrawal and A. El Abbadi from the University of Santa Barbara (California) to carry research on datawarehouses in the context of asynchronous distributed systems.

9. Dissemination

9.1. Teaching Activities

- Several members of the ADEPT research team belong to the university of Rennes I or to ENST Bretagne (a telecommunication engineering school). Therefore, an important part of their time is devoted to teaching to engineers and master students.
- Achour Mostéfaoui is member of the director board of the engineer diploma (DIIC) of the Department of Computer Science (IFSIC) of the University of Rennes 1. He heads the software engineering part of this diploma.
- Jean-Pierre Le Narzul has the responsibility for organizing several teaching units at ENST Bretagne (RSM Department). He gives lectures on both distributed computing and object-oriented language. He is also involved in the setting of programs for continuous training.
- Michel Hurfin gives lectures on fault tolerance and distributed computing to students of two engineering schools: ENST Bretagne (Brest, 6 hours) and Supelec (Rennes, 9 hours).
- During his stay at Southeast University in China (March 2003), Michel Hurfin has given a lecture on fault tolerance in distributed systems to master students.

9.2. Presentations of Research Works

- Emmanuelle Anceaume is the main organizer of the seminars entitled "Networks and Systems" that are periodically held in our institute. Since the creation of this thematic seminar serie in 2000, more than seventy presentations have been made by members of the IRISA institute and by visiting researchers.
- Roy Friedman has given two talks entitled "Group communication" and "Fault-tolerance in CORBA" during the second French-Mexican summer school on cooperative distributed systems (ESRC 2003) organized at IRISA in october 2003.
- Michel Hurfin has been invited to give a tutorial on agreement problems and a talk about grid computing during the 21st Brazilian Symposium on Computer Networks (SBRC) organized in Natal, in May 2003, by the Brazilian Computing Society and by the National Computer Network Laboratory.
- Michel Hurfin has given a talk entitled "agreement problems" during the second French-Mexican summer school on cooperative distributed systems (ESRC 2003) organized at IRISA in October 2003.
- Gérard Le Lann has given a talk entitled "Asynchronous Middleware for Achieving Adaptivity and Dependability" during the 43rd meeting of IFIP Working Group 10.4 (Dependable Computing), Cape Verde, January 2003.
- Gérard Le Lann was an invited lecturer at the 31st Spring School in Theoretical Computer Science, on "Distributed Algorithms", organized by University of Paris 7 and EPFL (Switzerland) in Porquerolles, France, May 2003. The Lecture title was: "Distributed Real-Time Computing: Where Do We Stand?".
- Gérard Le Lann has given a talk entitled "Time Optimal Consensus and Coordination in Distributed Real-Time Systems in the Presence of Failures" during the IliaTech meeting on "New solutions for the construction of computing systems out of COTS products", INRIA Rocquencourt, June 2003.
- Gérard Le Lann has given two invited presentations at the Workshop on Advanced Avionics Architecture & Modules (A3M), ESA/ESTEC, Noordwijk, The Netherlands, September 2003. The titles of the presentations were "The TRDF (proof-based system engineering) method for integrated modular avionics development" and "Algorithmic solutions for A3M".
- Michel Raynal has been invited to give a talk on the Condition-based Approach in Distributed Computing during the 10th Int. Workshop on Expressiveness in Concurrency (Express'03), a satellite workshop of *CONCUR'03* that was held in Marseille in September 2003.
- Michel Raynal has given a talk entitled "consensus in synchronous systems" during the second French-Mexican summer school on cooperative distributed systems (ESRC 2003) organized at IRISA in october 2003.
- Michel Raynal was an invited lecturer at 31st spring school in Theoretical computer Science, on "Distributed Algorithms", organized by University of Paris 7 and EPFL (Switzerland) in Porquerolles, France, May 2003. His lecture was on "Distributed agreement problems".
- Michel Raynal has given the following seminars in 2003:

  - "Consensus in synchronous systems", January 2003, UNAM, Mexico
  - "Consistency of distributed data", May 2003, University of Connecticut, USA
  - "An information structure for indulgent consensus", May 2003, MIT
  - "Accord réparti", June 2003, EPFL, Lausanne

  - "Asynchronous renaming", September 2003, University of Nizhni Novgorod, Russia,
  - "Une visite guidée au pays du consensus asynchrone", November 2003, séminaire du LAAS-CNRS, Toulouse

- Members of the ADEPT research team have attended several conferences and workshops dealing with distributed computing (the reader is encouraged to refer to the bibliographic references for additional information).

9.3. Integration within the Scientific Community

- Maria Gradinariu was

  - member of the program committee of the *6th Symposium on Self-Stabilizing Systems* that was held in June 2003 in San-Francisco, California, in conjunction with DSN 2003.
  - member of the program committee of the *1st Workshop on Adaptive Distributed Systems* that was held in October 2003 in Sorrento, Italy, in conjunction with DISC 2003.

- Michel Hurfin was

  - member of the program committee of the *3rd International Workshop on Distributed Auto-adaptive and Reconfigurable Systems (DARES 2003)* that was held in May 2003 in Providence (RI) in conjunction with ICDCS 2003.
  - member of the program committee of the second French-Mexican summer school on cooperative distributed systems (ESRC 2003) organized at IRISA in October 2003.
  - member of the program committee of the *4th International Workshop on Distributed Auto-adaptive and Reconfigurable Systems (DARES 2004)* that will be organized in March 2004 in Tokyo (Japan) in conjunction with ICDCS 2004.
  - member of the program committee of the *18th International Conference on Advanced Information Networking and Applications (AINA 2004)* that will be organized in March 2004 in Fukuoka (Japan).

- Gérard Le Lann was

  - Co-organizer and co-chair of the Workshop on "Middleware for Adaptivity and Dependability", 43rd meeting of IFIP Working Group 10.4 (Dependable Computing), Cape Verde, January 2003.
  - Organizer of the IliaTech meeting on "New solutions for the construction of computing systems out of COTS products", INRIA Rocquencourt, June 2003.
  - Organizer of, and panelist at, the Panel on Mobile Ad-hoc Networks and Dependable Computing, Med-Hoc-Net 2003, Mahdia, Tunisia, 27 June.
  - Invited to join the External Advisory Board of the Center for Embedded Systems, Virginia Tech, VA, USA.

- Jean-Pierre Le Narzul is member of the program committee of the *third International Workshop on Assurance in Distributed Systems and Networks (ADSN 2004)* which will be held in Tokyo (Japan) in conjunction with ICDCS 2004.
- Achour Mostéfaoui was

  - member of the program committee of the *9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)* that was held in Porto Rico in May 2003.
  - member of the program committee of the *6th International Symposium on Programming and Systems (ISPS'03)* that was held in Alger in May 2003.

- Michel Raynal has been:

  - Member of the program committee of the *8th IEEE Int. Workshop on Object-Oriented Real-time Dependable Systems (WORDS'03)*, Guadalaraja, Mexico, January 2003.
  - Member of the program committee of the *6th IEEE Int. Symposium on Autonomous Decentralized Systems (ISADS'03)*, Pisa, Italy, April 2003.
  - Member of the program committee of the *2nd IEEE Int. Symposium on Network Computing and Applications*, Cambridge (MA), April 2003.
  - President of the program committee of the *8th Workshop on Future Trends of Distributed Computing Systems (FT DCS8)*, Porto Rico, May 2003.
  - International Liaison Co-Chair of the *23th IEEE Int. Conf. on Distributed Computing Systems*, Providence (RI), May 2003.
  - Member of the program committee of the *First Latin-American Symposium on Dependable Computing (LASDC'03)*, São Paulo, Brazil, October 2003.
  - Member of the Advisory and Publicity Committee of the *9th IEEE Int. Workshop on Object-Oriented Real-time Dependable Systems (WORDS)*, Capri Island, Italy, October 2003.
  - Member of the program committee of the *22th IEEE Symposium on Reliable Distributed Systems (SRDS'03)*, Firenze, Italy, October 2003.

- Michel Raynal is currently the chair of the Sterring Committee of the series of DISC symposia (whose proceeding are published in the LNCS series of Springer-Verlag) and a member of the Steering Committee of the series of SIROCCO colloquia (Colloquium on Structural information and communication complexity).

10. Bibliography

Major publications by the team in recent years

[1] O. BABAOGLU, E. FROMENTIN, M. RAYNAL. *A Unified Framework for Expressing and Detecting Run-Time Properties of Distributed Computations.* in « Journal of Systems and Software, Special issue on Software Engineering for Distributed Computing », number 3, volume 33, June, 1996, pages 287-298.

[2] M. HERMANT, G. LE LANN. *Fast Asynchronous Uniform Consensus in Real-Time Distributed Systems.* in « IEEE Transactions on Computers », number 8, volume 51, August, 2002, pages 931-944.

[3] M. HURFIN, A. MOSTÉFAOUI, M. RAYNAL. *A Versatile Family of Consensus Protocols Based on Chandra-Toueg's Unreliable Failure Detectors.* in « IEEE Transactions on Computers », number 4, volume 51, April, 2002, pages 395-408.

[4] M. HURFIN, N. PLOUZEAU, M. RAYNAL. *Detecting Atomic Sequences of Predicates in Distributed Computations.* in « Proc. of the ACM Conference on Parallel and Distributed Debugging », pages 32-42, San Diego, California, May, 1993, Reprinted in SIGPLAN Notices, vol. 28,12, December 1993.

[5] J. M. HÉLARY, A. MOSTEFAOUI, M. RAYNAL. *A general scheme for token and tree based distributed mutual exclusion algorithm.* in « IEEE Transactions on Parallel and Distributed Systems », number 11, volume 5, November, 1994, pages 1185-1196.

[6] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL. *Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems.* in « Journal of the ACM », number 6, volume 50, November, 2003, pages 922-954.

[7] M. RAYNAL, A. SCHIPER, S. TOUEG. *The Causal Ordering Abstraction and a Simple Way to implement it.* in « Information Processing Letters », volume 39, September, 1991, pages 343-351.

[8] M. RAYNAL, M. SINGHAL. *Logical Time: Capturing Causality in Distributed Systems.* in « IEEE Computer », number 2, volume 29, February, 1996, pages 49-57.

[9] A. SCHIPER, M. RAYNAL. *From group communication to transactions in distributed systems.* in « Communications of the ACM », number 4, volume 39, April, 1996, pages 84-90.

Doctoral dissertations and “Habilitation” theses

[10] A. MOSTÉFAOUI. *Résolution des problèmes d’accord en restreignant les données.* Habilitation à diriger des recherches, University of Rennes I (école doctorale Matisse), December, 2003.

[11] E. MOURGAYA. *Les problèmes d’accord : une approche comportementale.* Thèse de doctorat, University of Rennes I (école doctorale Matisse), July, 2003.

[12] M. ROY. *Synchronisation distribuée sans attente : application à la résolution des problèmes d’accord par contrainte des données.* Thèse de doctorat, University of Rennes I (école doctorale Matisse), November, 2003.

[13] F. TRONEL. *Application des problèmes d’accord à la tolérance aux défaillances dans les systèmes distribués asynchrones.* Thèse de doctorat, University of Rennes I (école doctorale Matisse), December, 2003.

Articles in referred journals and book chapters

[14] E. ANCEAUME, A. FERNANDEZ, A. MOSTEFAOUI, G. NEIGER, M. RAYNAL. *A Necessary and Sufficient Condition for Transforming Limited Accuracy Failure Detectors.* in « Journal of Computer and System Sciences (JCSS) », 2003, To appear.

[15] R. BALDONI, G. MELIDEO, J. HELARY, M. RAYNAL. *Efficient Causality-Tracking Timestamping.* in « IEEE Transactions on Knowledge and Data Engineering », number 5, volume 15, October, 2003, pages 1239–1250.

[16] A. K. DATTA, M. GRADINARIU, A. B. KENITZKI, S. TIXEUIL. *Self-stabilizing Wormhole Routing on Ring Networks.* in « Journal of Information Science and Engineering », volume 19, 2003, pages 401–414.

[17] C. DELPORTE-GALLET, H. FAUCONNIER, J.-M. HÉLARY, M. RAYNAL. *Early Stopping in Global Data Computation.* in « IEEE Transactions on Parallel and Distributed Systems », number 9, volume 14, September, 2003, pages 909–921.

[18] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL. *Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems.* in « Journal of the ACM », number 6, volume 50, November, 2003, pages 922–954.

[19] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL, M. ROY. *Condition-Based Consensus Solvability: a Hierarchy of Conditions and Efficient Protocols.* in « Distributed Computing », 2004, To appear.

[20] M. RAYNAL, L. RODRIGUES. *Atomic Broadcast in Asynchronous Crash-Recovery Distributed Systems and its use in Quorum-Based Replication.* in « IEEE Transactions on Knowledge and Data Engineering », number 5, volume 15, October, 2003, pages 1206–1217.

Publications in Conferences and Workshops

[21] E. ANCEAUME, M. GRADINARIU, M. ROY. *Self-organizing Systems. Case Study: peer-to-peer systems.* in « Proc. of the 17th Int. Symposium on Distributed Computing (DISC-03) », pages 1–8, Sorrento, Italy, October, 2003, Brief announcement.

[22] A. K. DATTA, M. GRADINARIU, M. RAYNAL, G. SIMON. *Anonymous Publish/Subscribe in P2P Networks.* in « Proc. of the Int. Parallel and Distributed Processing Symposium », pages 74–74, Nice, France, April, 2003.

[23] P. EZHILCHELVAN, M. RAYNAL. *An Optimal Atomic Broadcast Protocol and an Implementation Framework.* in « Proc. of the 8th Int. IEEE Workshop on Object-Oriented Real-Time Dependable Systems (WORDS-03) », IEEE, pages 32–39, Guadalajara, January, 2003.

[24] C. FETZER, M. RAYNAL. *Elastic Vector Time.* in « Proc. of the 23rd Int. Conference on Distributed Computing Systems », IEEE, pages 284–293, Providence, Rhode Island, May, 2003.

[25] M. GRADINARIU, M. RAYNAL, G. SIMON. *Looking for a Common View for Mobile Worlds.* in « Proc. of the 9th Int. Workshop on Future Trends of Distributed Computing Systems (FTDCS-03) », pages 159–165, San Juan, Puerto Rico, May, 2003.

[26] F. GREVE, M. HURFIN, J.-P. LE NARZUL. *Open Eden: a Portable Fault Tolerant CORBA Architecture.* in « Proc. of the 2nd IEEE Int. Symposium on Parallel and Distributed Computing (ISPDC-03) », IEEE, Ljubljana, Slovenia, October, 2003.

[27] F. GREVE, M. HURFIN, J.-P. LE NARZUL, X. MA, F. TRONEL. *ADAM: A Library of Agreement Components for Reliable Distributed Programming.* in « Workshop on Communication Abstractions for Distributed Systems (WCADC-03 in conjunction with ECOOP-03) », Darmstadt, July, 2003.

[28] R. GUERRAOUI, M. RAYNAL. *A Generic Framework for Indulgent Consensus.* in « Proc. of the 23rd Int. Conference on Distributed Computing Systems », IEEE, pages 88–95, Providence, Rhode Island, May, 2003.

[29] C. GUETTIER, G. LE LANN, J.-F. HERMANT. *Ad Hoc Sensor Networks, Constraint Programming and Distributed Systems.* in « IEEE Int. Conference on Information Technology, Research and Education (ITRE 2003) », IEEE, Newark, New Jersey, August, 2003.

[30] K. HORI, T. ENOKIDO, M. TAKIZAWA, M. RAYNAL. *Nested Invocation Protocol on Object-based Systems.* in « Proc. of the 6th Int. IEEE Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03) », pages 238–248, Tokyo, May, 2003.

[31] M. HURFIN, J.-P. LE NARZUL, J. PLEY, PH. RAÏPIN PARVEDY. *A Fault-Tolerant Protocol for Resource Allocation in a Grid dedicated to Genomic Applications.* in « Proc. of the Fifth Int. Conference on Parallel Processing and Applied Mathematics, Special Session on Parallel and Distributed Bioinformatic Applications (PPAM-03) », series LNCS, Springer-Verlag, Czestochowa, Poland, September, 2003.

[32] J. KELLER, G. SIMON. *Solipsis: A Massively Multi-Participant Virtual World.* in « Proc. of the Int. Conference on Parallel and Distributed Techniques and Applications (PDPTA-03) », pages 262–268, Las Vegas, Nevada, June, 2003.

[33] J. KELLER, D. STERN, F. DANG NGOC. *MAAY: A Self-Adaptive Peer Network for Efficient Document Search.* in « Proc. of the Int. Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA-03) », pages 864–870, June, 2003.

[34] D. LAVENIER, H. LEROY, M. MACWING, R. ANDONOV, M. HURFIN, PH. RAÏPIN PARVEDY, L. MOUCHARD, F. GUINAND. *GénoGRID: an Experimental Grid for Genomic Applications.* in « Health-Grid conference », Lyon, France, January, 2003.

[35] A. MOSTEFAOUI, E. MOURGAYA, M. RAYNAL. *Asynchronous Implementation of Failure Detectors.* in « Proc. of the Int. Conference on Dependable Systems and Networks (DSN03) », IEEE, pages 351–360, San Francisco, CA, June, 2003.

[36] A. MOSTEFAOUI, E. MOURGAYA, M. RAYNAL, PH. RAÏPIN PARVEDY. *Evaluating the Condition-Based Approach to Solve Consensus.* in « Proc. of the Int. Conference on Dependable Systems and Networks (DSN03) », IEEE, pages 541–550, San Francisco, CA, June, 2003.

[37] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL. *Using Conditions to Expedite Consensus in Synchronous Distributed Systems.* in « Proc. of the 17th Int. Symposium on Distributed Computing (DISC-03) », series LNCS, number 2848, pages 249–263, Sorrento, Italy, October, 2003.

[38] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL, M. ROY. *A Hierarchy of Conditions for Asynchronous Interactive Consistency.* in « Proc. of the 7th Int. Conference on Parallel Computing Technologies (PaCT-03) », series LNCS, number 2763, Springer Verlag, pages 130–140, Nizhni Novgorod (Russia), September, 2003.

[39] A. MOSTEFAOUI, M. ROY. *Single-Write Safe Consensus Using Constrained Inputs.* in « Proc. of the 10th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO-03) », pages 293–308, Umeå, Sweden, June, 2003.

[40] B. RAVINDRAN, G. LE LANN, J. WANG, P. LI. *A Systems Engineering Approach for Constructing Certifiable Real-Time Distributed Systems.* in « Proc. of the 6th IEEE Intl. Workshop on Object-Oriented Real-Time Distributed Computing (ISORC 2003) », IEEE, pages 105–112, Hakodate, Japan, May, 2003.

[41] M. RAYNAL. *The Renaming Problem as an Introduction to Structures for Wait-free Computing.* in « Proc.

of the 7th Int. Conference on Parallel Computing Technologies (PaCT-03) », series LNCS, number 2763, Springer Verlag, pages 151–164, Nizhni Novgorod (Russia), September, 2003.

[42] M. RAYNAL. *Token-Based Sequential Consistency in Asynchronous Distributed Systems.* in « Proc. of the 17th Int. Conference on Advanced Information Networking and Applications (AINA-03) », IEEE, pages 421–426, Xi'an, China, March, 2003.

[43] F. LE FESSANT, PH. RAÏPIN PARVEDY, M. RAYNAL. *Early Decision Despite General Process Omission Failures.* in « Proc. of the 22nd annual Symposium on Principles of Distributed Computing (PODC-03) », pages 222–222, Boston, Massachusetts, July, 2003, Brief announcement.

[44] G. LE LANN. *Asynchrony and Real-Time Dependable Computing.* in « Proc. of the 8th Int. IEEE Workshop on Object-Oriented Real-Time Dependable Systems (WORDS-03) », IEEE, pages 18–25, Guadalajara, Mexico, January, 2003.

[45] M. LE ROY, P. GULA, J.-C. FABRE, G. LE LANN, E. BORNSCHLEGL. *Novel Generic Middleware Building Blocks for Dependable Modular Avionics Systems.* in « Workshop on Spacecraft Data Systems », ESA/ESTEC (European Space Agency), Noordwijk, The Netherlands, May, 2003.

[46] PH. RAÏPIN PARVEDY, M. RAYNAL. *Reliable Compare and Swap for Fault-Tolerant Synchronization.* in « Proc. of the 8th IEEE Int. Workshop on Object-oriented Real-time Dependable Systems (WORDS-03) », pages 50–55, Guadalajara, January, 2003.

[47] PH. RAÏPIN PARVEDY, M. RAYNAL. *Uniform Agreement Despite Process Omission Failures.* in « Proc. of the Int. Parallel and Distributed Processing Symposium (IPDPS-03) », pages 212–212, Nice, France, April, 2003.

Internal Reports

[48] E. ANCEAUME, C. DELPORTE-GALLET, H. FAUCONNIER, M. HURFIN, G. L. LANN. *Rapport phase 1: capture des besoins.* Deliverable AGS - Architectures Génériques pour le Spatial, IRISA - LIAFA, June, 2003, 24 pages.

[49] E. ANCEAUME, C. DELPORTE-GALLET, H. FAUCONNIER, M. HURFIN, G. L. LANN. *Rapport phase 2: solution pour le problème AGS.* Deliverable AGS - Architectures Génériques pour le Spatial, IRISA - LIAFA, December, 2003, 96 pages.

[50] E. ANCEAUME, M. GRADINARIU, M. ROY, E. MOURGAYA. *Etat de l'art: Auto-organisation dans les réseaux dynamiques.* Deliverable SPEERAL - Modélisation et algorithmes pour les réseaux de pairs, IRISA, April, 2003, 14 pages.

[51] R. FRIEDMAN, A. MOSTEFAOUI, M. RAYNAL. *A Weakest Failure Detector-Based Asynchronous Consensus Protocol for $f<n$.* Research Report, number 1557, IRISA, September, 2003, to appear in the Journal Information Processing Letters.

[52] R. FRIEDMAN, A. MOSTEFAOUI, M. RAYNAL. *On the Respective Power of Eventual Failure Detectors to Solve One-Shot Agreement Problems.* Research Report, number 1547, IRISA, July, 2003.

[53] R. FRIEDMAN, A. MOSTEFAOUI, M. RAYNAL. *Simple and Efficient Oracle-Based Consensus Protocols for Asynchronous Byzantine Systems.* Research Report, number 1556, IRISA, September, 2003.

Bibliography in notes

[54] H. ATTIYA, J. WELCH. T. M.-H. COMPANIES, editor, *Distributed Computing : Fundamentals, Simulations and Advanced Topics.* 1999.

[55] R. BALDONI, C. MARCHETTI, L. VERDE. *"CORBA Request Portable Interceptors: Analysis and Applications".* in « Proceedings of the 3nd International Symposium on Distributed Objects and Applications (DOA 2001) », pages 208-217, sep, 2001.

[56] S. DOLEV. *Self-Stabilization.* The MIT Press, 2000.

[57] M. FISCHER, N. LYNCH, M. PATERSON. *Impossibility of Distributed Consensus with One Faulty Process.* in « Journal of the ACM », number 2, volume 32, April, 1985, pages 374-382.

[58] F. GREVE, M. HURFIN, J.-P. L. NARZUL, X. MA, F. TRONEL. *ADAM: A Library of Agreement Components for Reliable Distributed Programming.* in « ECOOP'2003 Workshop on Communication Abstractions for Distributed Systems », Darmstadt, Germany, July, 2003.

[59] P. NARASIMHAN. *Transparent Fault Tolerance for CORBA.* Ph. D. Thesis, 1999.

[60] J. WALTER, J. L. WELCH, N. M. AMATO. *Distributed reconfiguration of metamorphic robot chains.* in « Proc. of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC'00) », 2000, pages 171–180.

[61] Z. YANG, K. DUDDY. *CORBA: A Platform for Distributed Object Computing.* in « ACM Operating Systems Review », number 2, volume 30, April, 1996, pages 4–31.

[62] H. ZHANG, A. ARORA. *GS3 : Scalable self-configuration and self-healing in wireless networks.* in « Proc. of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC'02) », 2002, pages 58-67.