INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team ADEPT

Asynchronous Distributed Environments, Protocols, and Time

Rennes

THEME COM

Activity Report

2004

Table of contents

1. Team

Team leader
Michel Hurfin [CR, INRIA Rennes]

Administrative assistant
Lydie Mabil [TR, INRIA Rennes]

Research scientist (Cnrs)
Emmanuelle Anceaume [CR]

Faculty members (University of Rennes I)
Maria Gradinariu [Associate Professor, Research Scientist INRIA since September 2004]
Achour Mostéfaoui [Associate Professor, team member until October 2004]

Faculty member (ENST Bretagne)
Jean-Pierre Le Narzul [Associate Professor]

Visiting scientists
Ajoy K. Datta [Professor, University of Nevada - USA, June 2004]
Fabíola Greve [Associate Professor, Federal University of Bahia - Brazil, June 2004]
Mikel Larrea [Associate Professor, University of the Basque Country - Spain, July 2004]
Roy Friedman [Scientific Researcher, Technion - Israel, January and February 2004]

Postdoctoral fellow (Inria)
Antonino Virgillito [starting September 2004]

Ph.D. students
Vincent Gramoli [Fellowship MENRT, since October 2004]
Julien Pley [Fellowship MENRT]
Philippe Raïpin Parvédy [Fellowship MENRT until September 2004 (then lecturer at the university of Rennes I)]
Matthieu Roy [Fellowship ENS until June 2004]
Aina Ravoaja [Fellowship INRIA and Brittany Region since October 2004]
Gwendal Simon [Grant France Télécom]

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 ions 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 characterize 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 impossible in a weaker model. Therefore, one of our research objectives is to determine the borderline between tractable and untractable problems within 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 the 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 metrics or subjective criteria such as minimality of the set of necessary and sufficient conditions, efficiency (time and message costs), 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 address several observation and synchronization problems through the study of a particular agreement problem, namely the Consensus problem. This problem and some of its variants are discussed within different approaches and with the emphasis on the design of algorithms to solve efficiently this fundamental problem in various specific environments. This intensive analysis carried out since a few years by several team members aims to have a deep understanding of this very important distributed service that allows to maintain strong forms of consistency within a well identified set of processes.
- Second, we are interested in dependability issues (reliability, availability, safety and security). More precisely we focus on replication mechanisms that may be used to build fault tolerant applications. Various model of failures are considered ranging from crash failure (the simplest) to Byzantine failure (the most sophisticated). To support the management of the distributed replicas, high level services are required. In our approach, all these services are reduced to agreement problems solved in a modular, flexible and adaptive manner. When failures can occur in the system, some processes (called faulty processes) can deviate from their specifications (crash, omissions, arbitrary behavior,...). Unfortunately, in an asynchronous systems, due to the lack of temporal assumptions, it is impossible to distinguish a correct process from a faulty one. As a consequence of this uncertainty, several useful agreement problems (consensus, atomic broadcast, group membership) have been proved to be impossible to solve in the pure asynchronous model. Parts of our activities aim to circumvent these impossibility result by weakening the problem definition or by strengthening the assumptions that are supposed to be satisfied by the environment.
- When we address one of the two above topics, we refer to a (synchronous or asynchronous) distributed system composed of a small set of well-identified processors. At any time, the identity and the localization of all the cooperating processors is known by any participant. A third research topic aims at defining new abstractions to cope with large-scale systems where frequent connections and disconnections occur. In this new context, it is impossible (or at least unreasonable) to try to compute a global view of the system that will need to be refreshed frequently. To cope with large scale systems, it is far better if each process maintains only information about its own neighborhood. Consistency between this multiple partial views have to be ensured. Some proposed solutions are generic and can be used in peer-to-peer systems, in sensor networks or in virtual reality 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

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

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 tractability. 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 the same 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 [37]. 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 randomized 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. In the second approach, weaker problems are considered. For example, in the condition-based approach, the idea 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. If the input vector does not belong to the condition, either the termination property or the validity property of the consensus problem may not be ensured. The k-set agreement problem is also another weaker problem: in this problem, processes are allowed to decide up to k different values.

3.3. Fault Tolerance in Distributed Systems

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

The study of fault-tolerant mechanisms based on processor redundancy is an active research topic in ADEPT. 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 such as group communication services which are essential to manage the evolution of the set of replicas. The type of 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 permanent 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. Moreover, a failure can be permanent or temporary. In some particular contexts (for example, space missions), the repair or the replacement of hardware is impossible or very expensive. Therefore, it should be possible to remove temporally a failed component from the set of operational components and to insert it again latter when the component has recovered.

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

Keywords: *ad-hoc networks*, *distributed computing*, *large scale systems*, *mobility*, *peer-to-peer systems*, *self-organization*, *self-stabilization*, *sensor networks*.

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 [39] and [38]. 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 [35][36] 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 available files 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 geographical area 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

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

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

Keywords: *distributed computing*, *fault tolerance*, *group communication*, *middleware*, *real-time application*, *space*, *telecommunication*.

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 contributing 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 distributed 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 addressed 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 developing 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

Keywords: *distributed computing*, *fault tolerance*, *group communication*, *middleware*.

Participants: Michel Hurfin, Jean-Pierre Le Narzul.

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

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

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

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 independent tasks. PARADIS is used to reliably allocate resources 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

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

Participants: Emmanuelle Anceaume, Roy Friedman, Michel Hurfin, Mikel Larrea, Achour Mostéfaoui, Julien Pley, Philippe Raïpin Parvédy, Matthieu Roy.

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 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. Classic failure detector implementations rely on additional synchrony assumptions such as partial synchrony. More recently, a new approach for implementing failure detectors has been proposed: it relies on behavioral properties on the flow of messages exchanged rather than on synchrony assumptions. We showed in [25] that these approaches are not antagonistic and can be advantageously combined. A hybrid protocol (the first to our knowledge) implementing failure detectors with eventual accuracy properties is presented. Interestingly, this protocol benefits from the best of both worlds in the sense that it converges (i.e., provides the required failure detector) as soon as either the system behaves synchronously or the required message exchange pattern is satisfied. This shows that, to expedite convergence, it can be interesting to consider that the underlying system can satisfy several alternative assumptions.

Additionally to this implementation, we studied stacking implementation of failure detector. 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 that we call $\mathcal{S}_k$ 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 ($\mathcal{S}_k$ and $\diamond\mathcal{S}_k$)?". In [12], 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 $t \geq k$.

We investigated in [22] the respective power of eventual failure detectors $\diamond\mathcal{P}$ (*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. We addressed the following question related to the comparative power of these classes, namely: "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, 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 enables solving one-shot agreement problems in unreliable asynchronous distributed systems.

In order to solve consensus, eventual failure detectors need a majority of correct processes. $\mathcal{P}^t \times \diamond\mathcal{S}$ has been shown to be the weakest realistic failure detector class needed to solve the consensus problem in an asynchronous distributed system prone to $t < n$ process crashes in which communication is by message-passing. However, the only protocol that is know to meet this bound is based on three layers of protocols construction, and is therefore not efficient. We presented in [15] a surprisingly simple and very efficient direct message-passing implementation of a $\mathcal{P}^t \times \diamond\mathcal{S}$-based consensus protocol.

In [13], we present an unreliable distributed timing scrutinizer (UDTS). This mechanism captures the state of the network and, based on its observation, estimates the waiting time that maximizes the efficiency of a round-based protocol regarding the number and/or the duration of its computational rounds. An UDTS is not an abstract oracle, it is implementable and reveals the properties of the network. We illustrate our approach by addressing the consensus problem in asynchronous distributed systems equipped with an unreliable distributed timing scrutinizer. The proposed protocol is conceptually simple. It benefits from the UDTS predictions to implement different convergence and decision conditions. This allows the protocol to be efficient in number and/or duration of rounds for stable networks.

In [33], we study the feasibility and cost of implementing failure detector classes $\mathcal{P}$ and $\mathcal{S}$ in systems with weak reliability and synchrony assumptions. We first give an algorithm that implements $\mathcal{P}$ in a system S_{sync} where both processes and communication links are synchronous, and any number of processes may crash. Then, we give an algorithm that implements $\mathcal{P}$ in a weaker system S_{strong} where processes are synchronous, but only the input and output links of an unknown correct process are timely (all other links can be asynchronous and/or lossy). Then, we give an algorithm that implements $\mathcal{S}$ in a weaker system S_{weak} where only the output links of an unknown correct process are timely. We also discuss the minimal link synchrony required to implement $\mathcal{P}$ and $\mathcal{S}$.

In [34], we revisit three $\diamond\mathcal{S}$-Consensus protocols proposed by (1) Chandra and Toueg, (2) Schiper, and (3) Hurfin and Raynal, respectively. We present the protocols following a generic and unified formalism, based on the roles that processes can play during a round of the protocol. When studying the protocols, we focus on two criteria: (1) their communication pattern, and (2) their insensitivity to erroneous suspicions (a concept that we define in the paper). Chandra-Toueg's protocol follows a centralized communication pattern, and is very sensitive to even a single erroneous suspicions. The other two protocols follow a decentralized communication pattern, and have mechanisms that make them insensitive to erroneous suspicions.

6.1.2. The condition-based approach

The *condition-based* approach for consensus solvability consists of identifying sets of input vectors, called *conditions,* for which there exists an asynchronous protocol solving consensus despite the occurrence of up to t process crashes. We investigated the largest set of conditions which allow us to solve the consensus problem in an asynchronous shared memory system. We showed that this class of conditions is structured into a hierarchy of more or less constrained conditions providing more or less efficient protocols [16].

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 [26], 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.

6.2. Fault Tolerance in Distributed Systems

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

Participants: Emmanuelle Anceaume, Fabiola Greve, Roy Friedman, Michel Hurfin, Jean-Pierre Le Narzul, Achour Mostefaoui, Julien Pley, Philippe Raïpin Parvédy.

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 to implement 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.

More details about this work can be found in [23]and [24].

6.2.2. 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. Since 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.

A presentation of PARADIS has been done at Supercomputing (Pittsburgh, November 2004) by members of the ADEPT team.

6.2.3. *Omission failures*

In [27], we consider synchronous distributed systems made up of n processes, where up to t can commit failures by crashing or omitting to send or receive messages when they should ("process omission" failure model). It presents a protocol solving uniform consensus in such a context. This protocol has several noteworthy features. First, it is particularly simple. Then, it is optimal both in (1) the number of communication steps needed for processes to decide and stop, namely, $\min(f+2, t+1)$ where f is the actual number of faulty processes, and (2) the number of processes that can be faulty, namely $t < n/2$. Moreover, (3) it ensures that no process (be it correct or faulty) executes more than $\min(f+2, t+1)$ rounds, thereby extending the decision lower bound to the full completion time. The design of a uniform consensus protocol with such optimality requirements was an open problem. Interestingly, as $\min(f+2, t+1)$ is a lower bound to solve uniform consensus in the synchronous crash failure model, the proposed protocol shows that uniform consensus is not "harder" in the omission failure model than in the crash failure model. The protocol is also message size efficient as, in addition to values, a message has to piggyback only n bits of control information.

6.2.4. *Byzantine failures*

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 [21] 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.

In [18] and [17], we address the problem of transient failures and propose a different model of failure: the scattered Byzantine failure model. In this model processes alternate correct and faulty periods. Specifically, during its faulty periods, a process behaves arbitrarily (one cannot expect anything from it during these periods) whereas during its correct periods, it behaves according to its specification. In that sense, the scattered Byzantine failure model generalizes the classical one. We characterize two reliable services guaranteeing timeliness properties in the presence of Byzantine failures, namely Clock Synchronization and Δ-Atomic Broadcast. We propose specifications of these services that meet the specificities of the scattered Byzantine failure model. These algorithmic solutions extend classical ones by adding statements that allow a process to leave a faulty period and come back to a safe state.

6.3. Large Scale, Dynamic Open Systems

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

Participants: Emmanuelle Anceaume, Ajoy K. Datta, Roy Friedman, Maria Gradinariu, Vincent Gramoli, Aina Ravoaja, Matthieu Roy, Gwendal Simon, Antonino Virgillito.

6.3.1. Caching

Internet based services is a potentially important application for MANETs (Mobile Ad Hoc Networks), as it can improve mobile users' perceived quality of service, reduce their energy consumption, and lower their air-time costs. In [20], We consider the problem of locating *cache proxies* in MANETs using several search techniques. We first examine several existing search techniques and a few novel ones including flooding, constrained flooding, a novel dynamic variation of probabilistic flooding, and BFS. These are superimposed on a Maximal Independent Set (MIS), a Connected Dominating Set (DS), and a novel adaptation of BFS-tree based overlays, where each of these overlays is maintained in a self stabilizing manner. We also compared the performance of these search techniques and overlays by extensive simulations.

6.3.2. Sensors

In [32], we design the *first, fully distributed, strictly localized, scalable,* and *self-$*$* solutions to the minimal connected sensor cover problem. The *Self-$*$* concept has been used to include many fault-tolerant properties like *self-configuring*, *self-reconfiguring/self-healing*, etc. We will present two *self-stabilizing* solutions, and show that these solutions are both self-configuring and self-healing. In a self-stabilizing system, every computation, starting from an arbitrary state, eventually reaches a state which satisfies the specification. Nodes achieve the global objective by using only local computations. Local algorithm based sensor networks are more robust, and scale better. The proposed solutions are space optimal in terms of number of states used per node. Another feature of the proposed algorithms is that the faults are contained only within the neighborhood of the faulty nodes. We also performed a comparison of the performance of the two proposed solutions in terms of the stabilization time and cover size metrics.

6.3.3. Dynamic Scalable Self-Managed Systems

In [19], we propose a middleware architecture and a generic orchestrating protocol for implementing distributed atomic transactions for large scale dynamic systems in a self-managing manner. In particular, the proposed solution is fully distributed, allows dynamic changes in the environment, and nodes are neither assumed to be aware of the size of the system nor of its entire composition. The architecture includes two modules and three services. The modules are expected to be instantiated and executed among relatively small sets of nodes in the context of a single transaction and, therefore, can be implemented using well-known distributed computing approaches. On the other hand, services are long lived abstractions that may involve all nodes and should be implemented using known peer-to-peer techniques. The proposed architecture is also interesting in the sense that it brings together several seemingly distinct research areas, including distributed consensus, group membership, notification services (publish/subscribe), scalable conflict detection (or locking), and scalable persistent storage. We also promote the use of oracles as a design principle in implementing the respective components of the architecture. Specifically, each of the modules and services are further decomposed into a “benign” part and an “oracle” part, which are specified in a functional manner. This makes the principles of our proposed solution independent of specific implementations and environment assumptions (e.g., it does not depend on any specific distributed hash tables or specific network timing assumptions, etc). Our contribution is therefore largely conceptual, as it focuses on defining the right architectural abstractions and on their orchestration, rather than on the actual mechanisms that implement each of its components.

6.3.4. Free riding

An emerging issue in P2P system is the lack of cooperation (free riding). Tracking this problem is very complex since we have to take into account a large set of features: large population, asymmetry of interest, collusion, "zero-cost identity", high turnover, and rationality. To tackle this problem, we adopt a game-theoretic approach [31]. We follow the principle that accepting a request should be rewarded, while refusing it should be punished. Following this principle, we propose an architecture made of two components, one being in charge of detecting free-riders, and the other one organizing nodes so that nodes are incited to cooperate.

7. Contracts and Grants with Industry

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

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

Participants: Emmanuelle Anceaume, Michel Hurfin.

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. At the beginning of 2004, 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 [28].

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)

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

Participants: Emmanuelle Anceaume, Maria Gradinariu, Matthieu Roy.

It should be clear now that “traditional” 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 [29][30].

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.

7.3. Assert Contract (2004-2006)

Keywords: *distributed computing*, *fault-tolerance*, *middleware*, *real-time*, *space*.

Participants: Emmanuelle Anceaume, Michel Hurfin.

ASSERT (Automated proof based System and Software Engineering for Real-Time) is an integrated project (IP) co-sponsored by the European Commission under the Information Society Technology (IST) priority within the 6th Framework Programme (FP6). The project addresses the strategic objective of "Embedded Systems".

The ASSERT main goal is to improve the system-and-software development process for critical embedded real-time systems, in the Aerospace and Transportation domains by (1) identifying and developing proven critical system families’ architecture, using a proof based development process supported by formal notations, component models, and innovative processes and tools and (2) developing associated building blocks that can be composed, tailored and verified in open frameworks that shall be reused and shared by European teams across multi domain projects.

The contribution of the ADEPT project focuses on the designed of fault-tolerant middleware services.

8. Other Grants and Activities

8.1. National Project

8.1.1. ACI GénoGRID (2003-2004)

Participants: Michel Hurfin, Jean-Pierre Le Narzul, 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. ACI Daddi (2004-2006)

Participants: Michel Hurfin, Jean-Pierre Le Narzul.

Each day, the databases maintaining information about system vulnerabilities are growing with new cases. In addition to the classical prevention security tools, intrusion detection systems (IDS) are nowadays widely used by the security administrators to detect attack occurrences against their systems. Anomaly detection is often viewed as the only approach to detect new forms of attack. The main principle of this approach consists in building a reference model of the behavior of a given entity (user, machine, service, or application) in order to compare it with the current observed behavior. If the observed behavior does not match the model, an alert is raised to report the anomaly. Rather than defining an explicit model, we suggest to consider an implicit one. Design diversity will be used to identify dynamically the reference model. In our approach, any request is forwarded to different modules implementing the same functionality but through diverse designs. Any difference between the obtained results can be interpreted as a possible corruption of one or several modules. The task of the ADEPT project is to provide secure group communication mechanisms that allow to managed the group of modules.

8.1.3. 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 has received a support from CNRS during one year.

8.2. International Cooperations

8.2.1. Brazil (Federal University of Bahia)

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

In June 2004, Fabiola Greve who is professor at the Federal University of Bahia stayed one month in the ADEPT research team and worked with members of the ADEPT team on agreements problems and the concept of component. A cooperation project with her university, the Federal University of Paraiba (Prof. Francisco

Brasileiro) and several French laboratories (ADEPT Team, GRAND LARGE Team, LIP6) has been submitted in response to to the Capes/Cofecube call for proposal. Grid computing is the central topic of the proposal.

8.2.2. China (Southeast University)

Participants: Michel Hurfin, Jean-Pierre Le Narzul.

Strong relationships are maintained with Professor Yun Wang of Southeast university (Nanjing). After a stay of one year of Ma Xiaojun in 2003 (Post-doctoral position), we keep on interacting on the development of the Eden platform.

8.2.3. Japan (JAIST)

Participant: Maria Gradinariu.

In November 2004, Maria Gradinariu has spent more than 2 weeks in the Japan Advanced Institute of Science and Technology (JAIST) and has worked with Xavier Defago on the problem of cooperative mobile robotics. The goal is to control a group of robot in a way that they cooperate to perform some collaborative actions. They have also study the problem of self-organization in large scale dynamic open systems.

8.2.4. USA (University of Nevada)

Participants: Maria Gradinariu, Philippe Raïpin Parvédy.

Ajoy Datta, Professor of the University of Nevada, has made a working visit of one month in June 2004. Joint works on sensor networks have been initiated. Maria Gradinariu is the co-supervisor of two master students of the university of Nevada.

9. Dissemination

9.1. Teaching Activities

- Some 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.
- 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.
- Since September 2004, Achour Mostéfaoui is responsible of the engineer diploma (DIIC) of the Department of Computer Science (IFSIC) of the University of Rennes 1.
- Emmanuelle Anceaume and Achour Mostefaoui belong to the specialist commission (university of Rennes I, section 27).
- Maria Gradinariu is the co-supervisor (with Prof. Ajoy Datta) of two master students of the university of Nevada, USA.
- 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 a visit at the University of the Basque country (San Sebastian, Spain), Michel Hurfin has given lectures on fault tolerance in distributed systems (6 hours) to master and PhD students.
- Achour Mostefaoui has given lectures on distributed algorithms and systems (25 hours) to the PhD students of the University of Bougie (Algeria).

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.
- In June 2004, Emmanuelle Anceaume and Maria Gradinariu have organized a journey entitled “Outils pour systèmes dynamiques grande échelle”. This event took place at IRISA and was an opportunity to merge researchers from different communities.
- In November 2004, Maria Gradinariu has spent more than two weeks in Japan in the JAIST institute (Japan Advanced Institute of Science and Technology) in the research team of Xavier Defago. During her stay, she has presented her research activities.
- 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

- Michel Hurfin
  - was member of the program committee of the *4th International Workshop on Distributed Auto-adaptive and Reconfigurable Systems (DARES 2004)* that was organized in March 2004 in Tokyo (Japan) in conjunction with ICDCS 2004.
  - was member of the program committee of the *18th International Conference on Advanced Information Networking and Applications (AINA 2004)* that was organized in March 2004 in Fukuoka (Japan).
  - is member of the program committee of the *19th International Conference on Advanced Information Networking and Applications (AINA 2005)* that will be organized in March 2005 in Taiwan.
  - is member of the program committee of the *11th International Conference on Parallel and Distributed Systems (ICPADS)* that will be organized in July 2005 in Japan.
  - is member of the program committee of the *2nd Latin-American Symposium on Dependable Computing (LADC)* that will be organized in October 2005 in Brazil.
- Jean-Pierre Le Narzul was member of the program committee of the *workshop ADSN (Assurance in Distributed Systems and Networks)* which was held in Tokyo (Japan) in conjunction with ICDCS 2004.
- Achour Mostéfaoui was member of the program committee of the *18th Annual Conference on Distributed Computing (DISC’04)* that was held in Amsterdam in October 2004.

10. Bibliography

Major publications by the team in recent years

[1] E. ANCEAUME, M. HURFIN, P. RAIPIN PARVEDY. *An Efficient Solution to the k-set Agreement Problem*, in "Proc. of the Fourth European Dependable Computing Conference (EDCC-4), Toulouse, France", LNCS 2485, Springer Verlag, Oct 2002, p. 62–78.

[2] J. BEAUQUIER, A.K. DATTA, M. GRADINARIU, F. MAGNIETTE. *Self-stabilizing local mutual exclusion and daemon refinement*, in "Proc. of the DISC 2000, LNCS 1914", 2000, p. 223-237.

[3] J. BEAUQUIER, J. DURAND-LOSE, M. GRADINARIU, C. JOHNEN. *Token based self-stabilizing uniform algorithms*, in "Journal of Parallel and Distributed Computing (JPDC)", vol. 62, n^o 5, 2002, p. 899–921.

[4] J. BEAUQUIER, M. GRADINARIU, C. JOHNEN. *Memory space requierement for self-stabilizing leader election protocols*, in "Proc. of the PODC'99", 1999, p. 199-208.

[5] 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", vol. 51, n^o 4, April 2002, p. 395-408.

[6] 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, San Diego, California", Reprinted in SIGPLAN Notices, vol. 28,12, December 1993, May 1993, p. 32-42.

[7] M. HURFIN, M. RAYNAL. *A simple and Fast Asynchronous Consensus Protocol Based on a Weak Failure Detector*, in "Distributed Computing", vol. 4, n^o 12, 1999, p. 209–223.

[8] Y. WANG, E. ANCEAUME, F. BRASILEIRO, F. GREVE, M. HURFIN. *Solving the Group Priority Inversion Problem in a Timed Asynchronous System*, in "IEEE Transactions on Computers. Special Issue on Asynchronous Real-Time Disttributed Systems", vol. 51, n^o 8, Aug 2002, p. 900–915.

Doctoral dissertations and Habilitation theses

[9] M. HURFIN. *Conception et expérimentation de solutions à des problèmes d'accord*, Habilitation à diriger des recherches, Université de Rennes I (école doctorale Matisse), nov 2004.

[10] P. RAÏPIN PARVÉDY. *Accords tolérant les fautes dans les systèmes répartis synchrones et asynchrones*, Thèse de doctorat, Université de Rennes I (école doctorale Matisse), oct 2004.

[11] G. SIMON. *Conception et réalisation d'un système pour environnement virtuel massivement partagé*, Thèse de doctorat, Université de Rennes I (école doctorale Matisse), dec 2004.

Articles in referred journals and book chapters

[12] 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)", vol. 68, n^o 1, 2004, p. 123-133.

[13] E. ANCEAUME, E. MOURGAYA, P. RAÏPIN PARVÉDY. *Unreliable distributed timing scrutinizers to converge towards Conditions*, in "Studia Universalis Informatica", vol. 3, n^o 1, 2004, p. 17–36.

[14] A. DATTA, M. GRADINARIU, S. TIXEUIL. *Self-stabilizing mutual exclusion using unfair distributed scheduler*, in "The Computer Journal", vol. 47, n^o 3, 2004, p. 289-298.

[15] R. FRIEDMAN, A. MOSTEFAOUI, M. RAYNAL. *A Weakest Failure Detector-Based Asynchronous Consensus Protocol for f<n*, in "Information and Processing Letters", vol. 90, n^o 1, 2004, p. 39-46.

[16] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL, M. ROY. *Condition-Based Consensus Solvability: a Hierarchy of Conditions and Efficient Protocols*, in "Distributed Computing", vol. 17, n^o 2, Apr 2004.

Publications in Conferences and Workshops

[17] E. ANCEAUME, C. DELPORTE-GALLET, H. FAUCONNIER, M. HURFIN, G. LE LANN. *Designing Modular Services in the Scattered Byzantine Failure Model*, in "Proc. of the Third International Symposium on Parallel and Distributed Computing (ISPDC), Cork, Irlande", IEEE (editor)., jul 2004, p. 262–269.

[18] E. ANCEAUME, C. DELPORTE-GALLET, H. FAUCONNIER, M. HURFIN, G. LE LANN. *Modular Services with Byzantine Recovery*, in "Proc. of the International Conference on Dependable Systems and Networks (DSN), Florence, Italy", IEEE (editor)., (fast abstract), jul 2004, p. 46–47.

[19] E. ANCEAUME, R. FRIEDMAN, M. GRADINARIU, M. ROY. *An Architecture for Dynamic Scalable Self-Managed Persistent Objects*, in "Proc. of the International Symposium on Distributed Objects and Applications (DOA), Agia Napa, Cyprus", S. VERLAG (editor)., LNCS, n^o 3291, oct 2004, p. 1445–1462.

[20] R. FRIEDMAN, M. GRADINARIU, G. SIMON. *Locating cache proxies in MANETs*, in "Proc. of the 5th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC), Tokyo, Japan", ACM (editor)., may 2004, p. 175–186.

[21] R. FRIEDMAN, A. MOSTEFAOUI, M. RAYNAL. *Simple and Efficient Oracle-Based Consensus Protocols for Asynchronous Byzantine Systems*, in "Proc. of the 23th IEEE Symposium on Reliable Distributed Systems, (SRDS-04), Florianõpolis, Brazil", IEEE, oct 2004, p. 228-237.

[22] R. FRIEDMAN, A. MOSTEFAOUI, M. RAYNAL. *The Notion of Veto Number and the Respective Power of eventual failure detectors to Solve One-Shot Agreement Problems*, in "Proc. of the 18th International Symposium on Distributed Computing (DISC-04), Amsterdam, NL", LNCS, n^o 3274, Springer-Verlag, oct 2004, p. 41-55.

[23] F. GREVE, M. HURFIN, J. LE NARZUL. *ADAM: une bibliothèque de composants d'accord pour la programmation d'applications fiables*, in "Actes des Journées Composants 2004, Lille, France", mar 2004.

[24] F. GREVE, J. LE NARZUL. *Designing a Configurable Group Service with Agreement Components*, in "Workshop on Fault-Tolerant Computing, in conjunctionwith SBRC 2004: Brazilian Symposium on Computer Networks, Gramado, Brazil", may 2004.

[25] A. MOSTEFAOUI, D. POWELL, M. RAYNAL. *A Hybrid Approach for Building Eventually Accurate Failure Detectors*, in "Proc. of the IEEE Pacific Rim Int. Symposium on Dependable Computing (PRDC-04), Papeete, Tahiti", Mar 2004.

[26] A. MOSTEFAOUI, S. RAJSBAUM, M. RAYNAL. *The Synchronous Condition-Based Consensus Hierarchy*, in "Proc. of the 18th International Symposium on Distributed Computing (DISC-04), Amsterdam, NL", R. GUERRAOUI (editor)., LNCS, n^o 3274, Springer-Verlag, oct 2004, p. 1-15.

[27] P. RAÏPIN PARVÉDY, M. RAYNAL. *Optimal Early Stopping Uniform Consensus in Synchronous System with Process Omission Failures*, in "Proc. of Sixteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'04)", ACM Press, 2004.

Internal Reports

[28] E. ANCEAUME, C. DELPORTE-GALLET, H. FAUCONNIER, M. HURFIN, G. L. LANN. *Rapport phase 2: solution pour le problème AGS*, 96 pages, Deliverable AGS - Architectures Génériques pour le Spatial, IRISA - LIAFA, March 2004.

[29] E. ANCEAUME, M. GRADINARIU, A. RAVOAJA. *Deliverable* SPEERAL *2 - Etat de l'art sur la théorie des jeux et les systèmes de pairs*, Technical report, IRISA, 2004.

[30] E. ANCEAUME, M. GRADINARIU, A. RAVOAJA. *Deliverable* SPEERAL *3 - Théorie des jeux et systèmes de pairs: une solution pour l'incitation à participer*, Technical report, IRISA, 2004.

[31] E. ANCEAUME, M. GRADINARIU, A. RAVOAJA. *Théorie des jeux et systèmes de pairs*, Research Report, n^o 1631, IRISA, Jul 2004.

[32] A. DATTA, M. GRADINARIU, P. LINGA, P. RAÏPIN-PARVÉDY. *Self-* Distributed Query Region Covering in Sensor Networks*, Research Report, n^o 1607, IRISA, Jun 2004.

[33] M. HURFIN, M. LARREA, A. LAFUENTE. *On implementing P and S with weak reliability and synchrony assumptions*, Research Report, n^o 1635, IRISA, Nov 2004.

[34] M. HURFIN, M. LARREA, J. PLEY. *Revisiting Diamond S Consensus protocols: (in)sensitivity to erroneous suspicions*, Research Report, n^o 1634, IRISA, Nov 2004.

Bibliography in notes

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

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

[37] M. FISCHER, N. LYNCH, M. PATERSON. *Impossibility of Distributed Consensus with One Faulty Process*, in "Journal of the ACM", vol. 32, n^o 2, April 1985, p. 374-382.

[38] 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, p. 171–180.

[39] 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, p. 58-67.