INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team ADEPT

Algorithms for Dynamic Dependable Systems

Rennes

THEME COM

Activity Report

2006

Table of contents

1. Team

Head of the team
Michel Hurfin [Research Associate (CR) INRIA, HdR]

Administrative assistants
Lydie Mabil [TR INRIA]
Céline Ammoniaux [TR CNRS, until April 2006]

Staff member Cnrs
Emmanuelle Anceaume [Research Associate (CR)]

Staff member Inria
Maria Gradinariu [Associate Professor, Research Scientist INRIA until August 2006]

Staff members University of Rennes I
Frédéric Tronel [Associate Professor]
Philippe Raïpin Parvédy [ATER until January 2006]
François Bonnet [Master Student from February 2006 until June 2006]
Emilien Dutang [Master Student from February 2006 until June 2006]

Research Scientists (External)
Jean-Pierre Le Narzul [Associate Professor, GET/ENST Bretagne]

Ph.D. students
Florent Claerhout [Fellowship MENRT]
Vincent Gramoli [Fellowship MENRT until April 2006]
Julien Pley [Fellowship MENRT until February 2006]
Aina Ravoaja [Fellowship INRIA and Brittany Region]

Engineer
Romaric Ludinard [Engineer since October 2006]

Visiting scientists
Fabíola Greve [Associate Professor, Federal University of Bahia - Brazil, February and November 2006]

2. Overall Objectives

2.1. Overall Objectives

The field of information technologies in general, and distributed computing in particular, is continuously evolving and maturing at a very high pace. Following the technological innovations, new announcements about the deployment of better (technically speaking) network technologies or infrastructures are now occurring monthly. These new environments are characterized by their higher bandwidth, lower latencies and sometimes wireless nature. All these evolutions in the properties of networks and entities they connect, induce radical changes in the very nature of applications that can be built upon these systems.

In a recent past, the focus of the research activities conducted in the ADEPT team has been on the study of dependability (mainly fault-tolerance) and data consistency within small sets of servers called groups. In this particular context, solutions to agreement problems (such as the consensus problem) have been proposed and used as basic building blocks for designing solutions to higher level protocols that are in charge of maintaining global properties at the group level despite the occurrence of crashes within the group. Such solutions have proved to be particularly suited to the case of small to medium systems. While maintaining a research activity in this particular field is important since several problems remain open in small groups of processors, it is obvious that the solutions proposed in the past are not directly applicable to large scale dynamic systems. Fundamental and applied research on models, algorithms and tools enabling to build distributed services and applications has now to face a new challenge, namely the high dynamicity that characterizes many recent distributed systems (in particular world-wide community of processors). In these new settings, various dynamic changes can affect a distributed computation and therefore need to be addressed at run time. For example, the load (in terms of messages or in terms of computing activities), the topology, the energetic resources, the accessibility of stored data, the set of processes involved in the computation or even the behavior of a participant may change at any time. Assuming that such variations do not occur very often, adaptive algorithms can be proposed to detect a modification of the whole execution context and react globally to this modification (reconfiguration, execution of another code, ...). However, when the system has a very high level of dynamicity, implementing a global observation system that allows to reconfigure the whole system in a single step is no more realistic. Only local observations and progressive adaptations to changes can be performed on cohesive subsets of nodes. Such a radical gap on the scale and dynamicity of systems militates in favor of a paradigm shift for designing solutions to the problems raised by these new systems. The challenge addressed now by the ADEPT team consists in maintaining global properties (in particular dependability and data consistency properties) over the set of entities that belong to a dynamic system.

To illustrate the new problems that arise, consider one particular aspect, namely the set of participants to a computation. For some applications (especially the world-wide applications executed on the Internet), it is not realistic to assume that the set of processes involved in a given computation is static for the whole computation. Nodes are continuously joining and leaving the system (for example if we look at P2P systems, more than 100,000 simultaneous nodes may populate the system with an average life-time of the order of minutes). Failures and recoveries are frequent. When the composition of this set evolves at a slow rate, a classical approach consists in computing a new global view each time an event that may impact on the membership of the system is observed (upon receipt of a join request, upon receipt of a leave request or upon detection of a failure). The aim of a group membership service is to provide all the participants with a unique view of the current set of participants. However when the stay of a process within the system is very short or when the number of processes is very large, this strategy can be impossible to manage or at least will create a time-consuming activity dedicated to the membership computation. No central entity can efficiently manage the organization and control of the whole system. It is also impossible to consider the existence of a static, or even semi-static, set of servers that conduct the entire computation for all other nodes, as it is common in the classical client-server model.

Apprehending dynamic systems goes through an analysis of the principles that govern them. The first principle concerns the exchange of information or resources with the environment (for example, in grid or P2P systems, components are possibly capable of infinitely often retrieving new information/resources from components around them). The second one is the dynamics of these systems (for example, in ad-hoc networks, components have the ability to move around, to leave or to join these systems freely). Finally, the third principle is related to 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. Each node may have its own data and its own strategy when it interacts with others: applications are context-sensitive and thus the behavior of a node partially depends on its own (complete or partial) view of the system. Looking at these systems as a mass of components can be a mistake because it abstracts away the 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. A node only observes a part of the system called its neighborhood. By definition, the neighborhood of a node n is a subset of the nodes of the system from which n can receive information. To define this set of nodes, a node takes into account both physical and semantical constraints. For example, in P2P systems, a node n' will not be a neighbor of a node n if there is no communication path between them or if they share no common interest. This set evolves dynamically during the computation. Neighbors are removed, added or replaced. For example, if a new node n'' can provide more significant informations than a neighbor n', a node n will replace n' by n'' in its own set of neighbors. The replacement strategy allows to avoid an uncontrolled increase of the size of the set of neighbors. Of course the significance of the information is application dependent. Similarly, in sensor and ad-hoc networks the neighborhood is defined in terms of communication range. For multiple reasons, the communication range of a node can increase or decrease. In that case, the set of neighbors associated to a node has to be changed accordingly. This ability to react to

external changes by adapting locally the set of neighbors is at the core of the self-organization mechanisms we aim to promote.
In a very dynamic system, no coordination between the nodes can be implemented to ensure consistency properties between the neighborhoods of the different nodes. Each node manages the creation and the evolution of its own neighborhood in an autonomous and independent way. Consequently, a node n' can be in the neighborhood of a node n while n is not in the neighborhood of node n'. Allowing this lack of consistency reduces the cost of the adaptation to the dynamic changes. When it is possible (*i.e.* when the level of dynamicity is rather low) and useful, additional cooperation between nodes can be proposed to ensure stronger properties (symmetry, transitivity, unique view shared by all the members of a neighborhood, ...). Ideally, these properties should be as strong as those maintained in classical group (group membership, view synchrony,...). Unfortunately, when the level of dynamicity is quite high, a trade-off has to be found because such strong properties cannot be guaranteed. The specification of a self-organization mechanism aims, among others, at defining the rules that govern the evolution of the neighborhoods.
Applications executed in dynamic systems try their best to ensure global properties. To achieve this goal, a process may have to interact directly with its neighbors and indirectly with the neighbors of its neighbors (and so on) in order to gather step by step knowledge about the entire system. The goal of an aggregation mechanism that will run in parallel with the self-organization mechanism is to define the rules applied to aggregate data. By aggregating local properties and knowledge, the system should be able to converge toward desirable global properties. Clearly, the speed of the convergence mainly depends on the quality of the interactions within a particular neighborhood and between neighborhoods, on the capacity for a node to spontaneously adapt itself to changes in the environment (self-organization techniques), and on the properties offered by the aggregation techniques.

The capacity of a system to spontaneously adapt itself to changes does not exist in classical distributed systems (since any change in these systems has to be seen, acknowledged or validated by all the nodes of the system). In scalable and dynamic systems this ability guarantees that even in case of unpredictable multiple changes, data structures used to keep a view of the outside world are updated independently by each node. Assuming that no new change occurs, the self-organization and aggregation techniques used for solving a particular application problem have to ensure that the system will stabilize at a particular state such that each node has captured a consistent global knowledge. Clearly an approach based on self-organization and aggregation techniques allows each node to compute updated observations of the system without any risk of deadlocks and without requiring neither an explicit agreement, nor the deployment of a predefined configuration of the system which reveals to be impossible or at least very costly.

To conclude, the field of practical distributed computing is at an exciting point right now, because of the current fast pace of technological innovation in the Internet, in the Web, and in mobile computing systems. The study of models and algorithms that allow to cope with dynamic changes whatever their causes and their level of dynamicity is fundamental. When a low level of dynamicity is assumed, an observation of the whole system is still possible and strong properties can be stated to coordinate the local observations done by each participant. In such a context, designing algorithmic solutions to fundamental problems (related to observation and synchronization issues) remains an important challenge. In particular, due to the adversity of the system (asynchrony and failures), efficient solutions are sometime difficult to design. On the contrary, a high level of dynamicity leads to manage several partial and inconsistent views of the system (each participant may have its own view). All classical distributed computing problems (for example, dependability issues, communication problems, resource allocation, and data management) will require new solutions that address theses challenges in the new settings. More generally, due to high dynamicity, new problems also arise and require the design of specific algorithms (in particular to cope efficiently with frequent changes of the set of participating processes). Among all the issues, dependability and consistency issues are of prime importance. As the computer-based systems are becoming more and more open and complex (number of interacting entities, heterogeneity of the hardware and software components, mixing of various standards, ...), the main attributes of dependability (namely reliability, availability, safety, and security) are also more and more difficult to guarantee. After having been able to master the difficulties raised by small to medium systems we must reveal the challenge of scaling up our solutions and adapt them to handle dynamicity.

3. Scientific Foundations

3.1. Introduction

Our scientific contributions aim at reaching a deeper understanding of all the fundamental problems that arise in dynamic distributed systems. During the study of a particular problem, our approach consists in identifying for a particular execution environment (characterized by a set of assumptions on the computation model, the failure model, the dynamicity, and scalability, ...), the set of elementary services needed to build a given application, and for each of them, to exhibit solutions as efficient as possible, optimal if possible, and generic. If no solution exists, we aim at exhibiting impossibility results. To validate and to promote the use of these algorithmic solutions, we conduct in parallel experimental evaluations by developing flexible and adaptive 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 aim of the ADEPT project is first to propose models for dynamic, scalable and dependable systems, then to identify, specify and design a set of generic elementary services needed to build applications for these systems. More precisely, our contribution focuses on the following themes:

- **Models for dynamic, scalable and dependable systems.** The new complexities faced by the research community in distributed computing (i.e. large scale, dynamicity, dependability) need adequate formal models. These models should include new abstractions for the communication and frameworks for the system execution.
- **Accidental and intentional faults.** Economic activities and human lives are now heavily dependent on distributed systems and applications. When computing resources and stored data can be affected by the occurrence of failures, dependability becomes a crucial issue. We aim to consider both accidental and intentional faults and to design algorithms and methods to detect or to mask such faults which are sometimes transient (another dynamic aspect).
- **Data consistency.** One of the challenges is to identify and formally define the consistency criteria required by different applications executed on top of dynamic scalable systems.
- **Dissemination of information.** Communicating in dynamic scalable and dependable systems faces different issues: firstly, large scale requires new communication primitives. One of these primitives, the semantic-based primitive (e.g., data-based), enables to transmit anonymously information within the network. That is, the set of recipients is not known in advance by the sender of the information. Recipients are identified only on semantical bases (the match between their own interest and the information content in the message). Secondly, dynamicity requires reactive-based communication primitives. We study these new communication services.

3.2. Models for Dynamic, Scalable and Dependable Systems

Keywords: *ad-hoc networks*, *automata model*, *concurrency*, *distributed computing*, *formal model*, *geometric model*, *large scale systems*, *mobility*, *peer-to-peer systems*, *self-organization*, *self-stabilization*, *sensor networks*, *thermodynamic model*.

3.2.1. Why Do We Need Models ?

As stated by Nancy Lynch [63] “Defining formal models for distributed systems has been an integral part of distributed computing theory from the very beginning. Formal models are more critical for distributed algorithms than they are for sequential algorithms, because distributed algorithms are generally much harder to understand since a single piece of distributed code is usually executed concurrently at many system nodes. Such nondeterminism makes it impossible to understand exactly what a distributed will do when it executes. Instead, one generally has to settle for understanding properties of execution, for example, invariants or progress properties. Defining these properties and showing that they hold require formal models“

Defining models for distributed computing is a matter of compromise. One needs to define properties that are sufficiently precise to capture the expected behavior of systems under study, while at the same time maintaining consistency and tractability of the model. There is no point in defining extremely detailed rules if one cannot manage inherent complexity induced by a too great level of details.

The relatively short history of computer science in general and distributed computing in particular (when compared to others field of science) has seen the rise of two principles models for studying distributed systems and proving theorems about them :

- The first one, that one can call the classical one, is an extension of automata for distributed computing.
- The second one which is more recent, is based on rather involved mathematical developments from the field of algebraic topology, which can be qualified as "geometrical" model.

In the sequel, we will discuss the respective benefits and costs of both models, and their adequacy to the study of dynamic dependable systems. This adequation could be discussed through a lot of facets (expressiveness, performance analysis ability, etc). Expressiveness is the main criteria to evaluate this adequation, even though others criteria could also be evaluated.

3.2.2. Automata Models

Since the very first developments of the distributed computing field, at least thirty years ago [61], up to very recent articles, classical models based on input/output automata, and graph theory have prevailed. This can be easily explained by the fact that :

- They were natural extensions of well-established models used in sequential computing, making them appealing and easy to use.
- Their popularity has been naturally reinforced by their successful applications in the proof of a large number of seminal papers in the field [54], [46], [45].

A lot of proving techniques have been popularized based on this model like indistinguishable states [54], state valency [54], [46], [45]. All these qualities could have been sufficient for adopting this model in the distributed computing field. However, by its very nature it seems that this kind of model is not well equipped to determine the frontier between possible and impossible problems with respect to fault-tolerance, when more than one process can crash in a system. Although this does not constitute a proof, one can remark that the proof of impossibility for consensus in the presence of a single failure while being relatively old (1985), has not been followed by any others also fundamental results afterward. This can be a posteriori analyzed as an indication that graph/automatas models do not allow to master the complexity of systems where an arbitrary number of processes may silently fail. The decade that has followed the FLP result can certainly be considered as the "consensus" decade for dependable distributed systems.

In [38], the authors have been able to precisely determine the conditions under which a decision problem has a solution in a totally asynchronous system where at most one process can fail, but they were not able to extend this result for a larger number of failures. This remarkable piece of work has given rise to a tremendous amount of curiosity and effort so as to extend it to a more general framework. This work, although not yet achieved, has been at least partially attained concurrently by [70], [57]. These results have however required a paradigm shift which is the object of the following section.

3.2.3. Geometric Models for Concurrency

In [58], the authors have been able to extend the result of [38] to the case where an unbounded number of processes may fail. They gave a complete characterization of wait-free decision tasks[1] in asynchronous systems where processes communicate by the mean of a shared memory. The starting point of all these

[1] A wait-free decision tasks is a decision problem that can be solved despite the failures of an arbitrary number of process (except one). They are characterized by the fact that no synchronization based on the wait for the occurrence of a message from any other processes in the systems, can ever be used. So any implementation for such a problem must be "wait-free".

new results can certainly be dated by the study of k-set agreement by [47]. This new problem can be seen as an extension of the classical consensus problem. In the latter one, all correct processes have to reach a common agreement, while in the former one, up to k different values may be decided. In this article Chaudhury conjectures that k-set agreement cannot be solved in the presence of k failures (when the set of possible initial values is larger than k). This new definition that covers the one of the classical consensus ($k = 1$) was the missing piece of the puzzle. Researchers had now a new challenge to reveal that was pointing out in the right direction.

However to study the inherent combinatorial complexity induced by this new problem, graphs and automata were not the right tools. Rather than this planar representation of systems, one need objects that span over high dimensional spaces. To keep it simple we consider the case of a system composed of only three processes. In the new geometrical model, a global state of the system is figured by a triangle (3-simplex in the vocabulary associated with this model) where each vertex represents the local state of a single process. A computing step can then be figured by two triangles sharing a common face. More precisely, the two vertexes of this common face, represent the two local states of the two processes that are not involved in the computing step. While the process that has issued the step, has seen its local state modified and is thus modeled by two different vertexes.

To study the complex geometrical objects induced by this model, mathematical tools originating from of algebraic topology are commonly used. This field of mathematics offers powerful techniques that are able to tackle the combinatorial explosion that occurred when using graph theoretical approach. This has been brilliantly demonstrated by [58] who have been able to precisely characterize the set of decision tasks that can be solved in wait-free systems.

The ADEPT team is involved in this domain by its participation to the TAGADA ACI project. The goal of this project is to federate french researchers whose research topics are related to the study of higher dimensional objects (like HDAs) for concurrent systems models.

3.2.4. Thermodynamic Models

These two last decades of distributed computing have shown that the path toward fundamental theoretical advances seems to be highly coupled to :

1. The definition of an emblematic problem that fully captures the inherent complexity of the systems to be studied.
2. The choice of a pertinent mathematical model for mastering this complexity.

From our point of view, both goals are still widely open when one considers dependable dynamic systems. The two traditional models that have been previously illustrated do not seem to completely fulfill the requirements of dynamicity imposed by the new kind of systems we want to study.

To overcome these limitations, we plan to investigate a promising way of studying such complex systems, inspired by some technics of statistical physics. More precisely, since we now consider systems that includes thousands, and may be even millions of nodes, we are convinced that the study of all the explicit interactions between neighbor nodes is almost unfeasible. We strongly advocate for the study of such systems as a whole, without looking at all possible local interactions. This is directly inspired by the way physicists are looking at complex thermodynamic systems. While they do not consider every single interactions between microscopic particles that compose these kind of systems, they can still give a good description of its macroscopic behavior in terms of macroscopic quantities such as its temperature, its free energy, enthalpy and so on. This is exactly what we want: obtaining emerging global properties, without looking at all the local interactions in details.

The idea is to mimic the way physicists study thermodynamic systems. We plan to first translate the vocabulary and notions they employ for physical systems to the world of dynamic distributed systems. Then we will try to describe the behavior of dynamic distributed systems (like information dissemination) in terms of phase transition in the system. Percolation theory, and magnetism theory in physical systems may well be applied to complex dynamic distributed systems. Of course this is a widely open research domain, that we plan to explore in the near future.

3.3. Accidental and Intentional Faults

Keywords: *accidental fault*, *agreement problem*, *availability*, *consensus problem*, *dependability*, *distributed computing*, *failure detector*, *failure model*, *free rider*, *group*, *malicious fault*, *reconfiguration*, *reliability*, *reputation*, *security*.

Economic activities and human lives are now heavily dependent on distributed systems and applications. When computing resources and stored data can be affected by the occurrence of failures, dependability becomes a crucial issue. As the computer-based systems are becoming more and more open and complex (number of interacting entities, heterogeneity of the hardware and software components, mixing of various standards, ...), the main attributes of dependability [62] (namely reliability, availability, safety, and security) are also more and more difficult to guarantee.

In such a context, we aim at specifying and designing methods to cope with the different attributes of dependability when either accidental faults or intentional faults may occur during operation. Design and implementation faults are out of the scope of our research activities: by assumption, any software is supposed to be correct with respect to its specification. The major part of our research activity on both accidental and intentional faults will focus on dependability problems due to arbitrary or malicious behaviors of some nodes. In what follows, the description of this general activity is subdivided into four sections devoted respectively to (1) the arbitrary behaviors caused by some transient accidental faults, (2) the reconfiguration of a system after the occurrence of failures due to accidental faults (3) the malicious behaviors in the presence of external attackers, and (4) the malicious behaviors in the presence of internal free riders (or nodes with bad a reputation). This last category of malicious behaviors is more specific to high dynamic systems such as peer-to-peer networks: some users called free-riders consume network resources without providing their own resources and thus without contributing to their network.

For all the problems described within this section, our research activities will take advantage of the complementary knowledge of the team members who have worked in the field of distributed computing on fault tolerance issues, observation issues, synchronization issues, agreement problems and self-stabilization.

3.3.1. Accidental Faults Resulting in Transient Failures

Accidental faults include among others physical faults that generally result from aging, shocks and physical phenomena (temperature, hydrometry, radiation, ...) that may affect communication hardware, computing hardware or memory hardware. In the worst case, accidental faults may lead to arbitrary behaviors. For example, in the particular case of a spacecraft, radiations (such as alpha particles and cosmic rays) may generate an undesired bit-flip effect that changes the content of a storage element (memory, register or instruction counter). Consequently, the next executed instruction is not necessarily the right one. In the particular case of accidental faults, our aim is to build reliable systems from unreliable components. To tolerate failures (crash or arbitrary behaviors) physical redundancy is mandatory to ensure that faults are masked and will not perturb the computation. Replication of critical data and functionalities on a group of nodes allows to increase the overall reliability of the system [66]. To be able to coordinate the activities of the processors, a significant body of work on replication techniques and agreement problems has been done. Many services that have to be provided when using the concept of group can be classified as agreement problems (membership, total order broadcast, consensus, leader election, atomic validation, ...). As these services are used very often, efficiency is a key issue when designing solutions to such agreement problems. Our first goal is to have an even better understanding of these problems while considering various levels of adversity (various computational models ranging from the purely synchronous one to the purely asynchronous one, various failure models, ...). In particular, the analysis of the amount of time/activity necessary to converge to an agreement may require to use probabilistic models that will characterize the environment and the arrival law of failures occurrences. Such an analysis can also be used to define a good trade off between the price to pay (redundancy, additional activities,...) and the obtained level of dependability (evaluation of the coverage).

Our second goal is to consider that a process has the capability to recover and so should not be excluded or precluded from participating to the forthcoming computations under the pretext that it has suffered a transient failure (even in the particular case of transient byzantine failures). Classically, the proposed solutions assume that all the nodes involved in the computation are classified into two categories (the *correct* nodes and the *faulty* ones). This classification into two distinct subsets remains unchanged during all the computation. A node is correct if it behaves according to its specification until the completion of the computation; otherwise it is faulty. Within this approach, the design of fault tolerant applications relies on the fact that a correct node never fails. Consequently, once all the faulty nodes have failed, no new failure can occur. Assuming that only a subset of the nodes can fail during the whole computation fits the requirements of many common distributed applications. But when running times of the applications are extremely long, this assumption becomes unrealistic (for example, in the space domain, the useful orbital life of a satellite is expected to last several years): any node may fail before the whole computation finishes. Yet as most of the faults are transient, a node is often able to recover to a consistent state (after having experienced failures). Thus, it is of interest to consider that any node can alternate correct and faulty periods (no node is correct according to the previous definition of correct/faulty nodes). Of course, the complexity of this new approach depends on the type of failures we consider. In the particular case of the crash failure model, the problem is quite simple: a node is aware that it is currently in the recovery phase. This information can also be known and trusted by the other nodes. Additionally, in a crash failure model, a running process always behaves according to its specification. Consequently, any data saved in its permanent storage or logged by another process can be used to recover to a safe state. Thanks to all these strong properties, it is possible to cope with both permanent and transient crashes [28].

Unfortunately, all these assumptions are no more true when one considers arbitrary failures. As mentioned previously, accidental faults may lead to arbitrary failures: in this failure model, a faulty node is not always aware that its local state has been altered. For some specific fundamental services (for example, clock synchronization and broadcast primitives), we aim at proposing algorithmic solutions that will address two different issues. The first one consists in ensuring that a process can converge in a bounded time to a consistent state each time it succeeds to come back in an operational state after a transient failure. Self-stabilizing techniques can be used to obtain this convergence. The second issue concerns the service availability. To ensure this property, constraints have to be put on the number of concurrent failures. Any node can experience a failure but at a given time, at most t are not in a "correct" period. To cope with arbitrary behaviors, we assume that $n > 3t$. This kind of study has strong connections with works done on pro-active security [40], [43]. We aim to study these relationships. In a pro-active security system, any node can experience arbitrary failures but during a fixed period of time, no more than t nodes can be faulty. To ensure security requirements, algorithms perform periodic computations of critical data (like for example secret keys). The type of solutions we are proposing [30] are based on similar assumptions.

3.3.2. Accidental Faults and Reconfiguration Problems

In many cases, a reconfiguration of the system is necessary to accommodate failures. In dynamic large scale systems, redundant resources are available and their use can be adapted to cope with failure scenarios in a completely transparent way. We study this problem in a particular context, namely Grid computing.

The major aim of a Grid is to federate 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 user's point of view.

Our goal is not to design a complete range of services for Grid systems. We focus only on two specific issues: resource allocation and dependability. Moreover we restrict the scope of our research to time-consuming applications that can be decomposed into a huge number of independent tasks. As all the tasks generated during the execution of such an application are independent, they can be allocated independently on the different resources of a Grid. The above assumptions are not unrealistic: a grid dedicated to the execution of genomic applications satisfies the above assumptions (See the description of the ACI GénoGRID and the description of the software Paradis).

With respect to these research topics (resource allocation and dependability), our contributions aim to promote two major complementary ideas. First, we suggest that the architecture of a Grid follows a hierarchical structure. Second we claim that interactions within the grid can be reduced to either a classical master-slave scheme or to a sequence of unanimous decisions depending on the level of the interacting entities within the Grid. This strategy allows us to benefit from the existence of synchronous networks where upper temporal bounds (on message transfer delays and also on the time required to execute a computation step) exist and can be known. On the contrary, more complex agreement protocols are used to share a consistent view of the global state of the Grid between unreliable entities linked through an asynchronous networks.

We assume that the architecture of a grid is made of several hierarchical levels. All the resources belonging to a same domain (for example, a research institute) are connected through a local area network which is synchronous. Resources are the lowest level in the hierarchy (level 1). Domains constitute the upper level (level 2). In each domain, at least one node acts as a proxy which is able to communicate with other proxies located outside the domain. In addition to this communication function, the proxy acts also as a coordinator within its own domain. More precisely, interactions between a proxy and its resources are based on the master/slaves schema. Within a domain, tasks can be allocated to the resources by the proxy which has also to react to the failures of resources. We consider that resources fail only by crashing. As a domain is assumed to be a synchronous sub-system, dependability and task allocation issues can be solved assuming that bounds on the time to execute a computation step and bounds on the time required to transfer a message exist. Depending on the number of levels in the hierarchy, the whole grid is either a group of domains (three levels) or a group of groups of domains (four levels) or an even more complex structure. A group of domains is an asynchronous sub-system. At this level, cooperation algorithms between the proxies have to be defined to cope with the dynamic evolution of the group. Like the composition of a domain, the composition of the networks of domains is also dynamic. Through invocations of the join and leave operations, the administrator of a domain can decide to add or remove his own domain from the Grid whenever he wants (maintenance and repair, alternating periods of private and public use, ...). A domain is unavailable if there is no node able to act as a proxy/master or if the domain has been disconnected from the Grid. A group membership service ensures that all proxies that are currently members of the group are consistent with the past history of the group, namely the join and leave operations already executed and the failures suspected to have occurred. When tasks have to be allocated, proxies (representing the domains) participate to an agreement protocol to determine the identity of the domain which seems to be the most appropriate to execute the task.

Based on these general ideas, we propose different basic agreement services to solve dependability and resource allocation issues. Our objective is now to use a similar set of services to cope with two different sources of non-determinism. First, the duration of a task is not known exactly but just estimated through previous benchmarks. Second, the resources are not exclusively dedicated to the Grid's users. Additional tasks are executed by the resources without being planned by the Grid's allocation mechanism. To cope with these two types of uncertainty, we propose to use agreement solutions to undo previous tasks allocation when it is necessary. Reconfiguration will operate either when a failure occurs or when inefficient decisions have been taken in the past. Our aim is to show (through analysis and experiments) that the different parameters can be tuned to obtained a realistic and efficient reallocation strategy.

3.3.3. *Intentional Faults and Security Aspects: New Attacks*

Intentional faults are produced by malicious attackers who try to take advantage of residual vulnerabilities that always exist in a complex system. When considering intentional faults, our aim is not to propose preventive measures (access controls, encryption, firewalls,...). Assuming that an intrusion can succeed, we want to be able to detect it, to confine damage and to clean and recover corrupted entities from errors. To achieve this goal, the concept of design diversity can be used. In the particular context of the ACI Security called "Daddi", we aim to secure a web access to a set of data. These data are replicated and accessible through different systems that may have residual vulnerabilities (but hopefully not necessarily the same ones). A new attack will take advantage of a particular vulnerability of a particular system. Consequently, the attack can succeed on a particular copy but not on all the copies. By checking the values returned by the different copies to the malicious attacker we can identify differences and detect anomalies. Of course, the difficult part is to provide replication and detection mechanisms that are safe and will not become an even more simple target for the attacker. Our aim is to study how group services can be reemployed and adapted to achieve this objective.

3.3.4. Intentional Faults and Fairness Aspects: Free-Riders and Reputation

Among the important issues raised by the emerging Peer-to-peer systems, is the one introduced by non-cooperative nodes. Indeed, it is more or less straightforward that rationality exists in P2P systems. By looking at traces showing nodes behaviors, it is clear that many nodes act in a way that is not desirable. The reason is that nodes have a priori no motivation for cooperation. This a priori non-cooperation between nodes inevitably leads to poor system performance. Different approaches may be adopted to face rationality in P2P systems. Among them are *i)* to ignore rationality and to expect that the system will do its best despite self-interested nodes, *ii)* to limit the effect that a rational node can have on the system by using trusted mechanisms, or *iii)* to adopt the fault tolerance techniques. However, none of these approaches benefit from resources that may be potentially offered by these self-interested nodes. We claim that the system must provide incentives to nodes to participate to the given protocols. Solutions may come from economics and more precisely from the mechanism design theory. The scope of this theory is to provide tools and methods to design protocols for self-interested parties. In other words, this theory deals with designing the rules of the game so that a good system-wide outcome will be achieved despite the fact that parties can act on self-interest. Yet, the mechanism design theory advocates the existence of one central mechanism that is used for the whole system and relies on the capability of all the parties to agree on a set of intentions, which contradicts the large scale and the nature of open systems we consider. So, we want to relax the goals of the mechanism design theory to consider more realistic mechanisms exhibiting a limited scope to keep the computational and informational aspects reasonable, and to reflect the limited range of influence each party can have. We formalize this locality aspect by relying on the self-organization model. However despite rational behavior, experiences shows that some nodes are altruists. For example, in Gnutella, 10% of the nodes are found to serve about 90% of the total download requests. We intend to exploit these nodes to heal the network from selfish nodes and thus to improve the overall behavior of the system.

3.4. Availability and Consistency of Data in Dynamic Systems

Keywords: *availability*, *consistency*, *distributed computing*, *dynamic system*, *replication*, *shared data*.

Some interactions in distributed applications can be modeled by the modifications of shared data (concurrent objects shared by several processes). The availability of data is generally ensured via replication. One of the main issues in replicated systems is to keep the various copies consistent. A consistency criteria basically defines what guarantees are provided by the system, more precisely the values which have to be returned when an operation which spans one or more shared objects is invoked by a process.

The literature offers a large class of consistency criteria which could be classified into "strong" and "weak" categories depending on the values retrieved by the invoked read operations. The definitions of different consistency criteria [35] do not depend on the system characteristics. Moreover, for some consistency conditions once the condition is preserved independently by each replicated component implementation, it is preserved by the system as a whole without any other coordination. This highly desired property of consistency conditions, is called locality and was analyzed in [73]. The authors study the locality of the most common consistency criteria and specify new ways of constructing local consistency criteria.

Despite the huge work in the area of replication and consistency criteria, managing shared data in large scale dynamic systems still raises specific difficulties like ensuring the coherent data persistency, defining efficient placement policies, or specifying consistency criteria for new emergent applications.

A crucial challenge in a replicated system is consistency. Quorum systems are a valuable tool for building consistent systems. Quorum systems have been used in the study of a broad distributed control and management problems. In many applications of quorum systems the underlying universe is associated with a network of processors, and a quorum is employed for accessing each of its elements.

In a typical application of data replication, the quorum sets are divided into reading quorums and writing quorums where each reading quorum intersects each writing quorum. When a data item is added to the system, it is written into all the members of a writing quorum. A data item is searched by querying all the members of a reading quorum. The intersection property guarantees the effectiveness of the search.

Clearly, in dynamic systems, the settings in which operates a quorum system should accommodate dynamic changes. Addressing the problem of designing a quorum system that fits a scalable and dynamic environment where processors leave and join at will is one of the aspects we want to investigate.

More precisely, Naor and Wool [65] have analyzed the metrics that measure the quality of a dynamic quorum system. These metrics (load, availability, and complexity) relate both the combinatorial structure of the quorum and its capability of being implemented in a distributed network. Briefly, the load measures the quality of the quorum system in the following sense: if the load is low, then each element is accessed rarely and thus is free to perform other unrelated tasks. Assuming that each element fails with probability p, the availability is the probability Fp, that the surviving elements contain a quorum. This measures how resilient the system is, and clearly we would like Fp to be as close to 1 as possible. The notion of availability is important when dealing with temporary faults. The most common strategy to deal with faults is to bypass them; i.e. to find a quorum set for which all processors are alive. The complexity of the algorithms for finding a quorum should be low. In particular, if all processors are alive the network should allow easy access to elements of the same quorum system. In case some elements fail, finding a live quorum set can be a difficult algorithmic task.

Clearly, the introduction of a dynamic environment requires additional demands [51]: (1) an integrity requirement which states that a new processor that joins the system, and a processor that leaves the system, should change the quorum sets and (2) a scalability requirement which says that the number of elements in the quorum system may increase over time. The increase in the size of the system should maintain its good qualities, i.e. it should decrease the load on each processor and increase the availability of the system. It is important that when the system scales the complexity of the algorithmic remains low. Finally the Join and Leave operation should be applied with low time and message complexity.

3.5. Dissemination of Information

Keywords: *broadcast service*, *data aggregation*, *distributed computing*, *publish/subscribe*, *recipient-based broadcast*, *reliable broadcast*, *semantic-based broadcast*, *tracking service*.

3.5.1. Broadcast Services

The provision of communication services within a system is essential to allow an entity to send information toward another entity or to broadcast it to all the entities of the system. The properties guaranteed by these services essentially depend on the size of the system. For relatively small systems (less than 100 nodes), broadcast services guarantee strong properties regarding the delivery of the messages to the recipients and the order in which these messages are delivered [55]. For example, a reliable broadcast primitive guarantees that any message sent to a set of entities is delivered either by all of them or by none of them. An atomic broadcast primitive requires that the order in which messages are delivered to the recipients is the same for all of them. These communication primitives facilitate the task of an application designer since strong properties are guaranteed [66]. For instance, ensuring the consistency of multiple replicated copies becomes trivial whenever an atomic broadcast service is available. Designing an efficient implementation of such primitives is still a challenge that we intend to take up.

Regarding scalable systems, the specification of a communication service poses new challenges. Broadly, two communication schemes exist: recipients-based broadcast and semantic-based broadcast.

- The first one, that we could call recipients-based broadcast, has to handle the system size issue. A message has to be received by all the identified recipients (in the worst case, by all the nodes). Clearly, existing solutions for small systems are no more applicable because of their message complexity (from O(n) messages to O(n^2) messages are needed to broadcast (resp. to reliably broadcast) an information within the network [52]). Furthermore, in a dynamic setting where an unbounded number of nodes may join, leave voluntarily, or fail (by stopping) during the course of a computation, it is not feasible to identify the set of recipients of a broadcast off-line. Lynch [37] proposes a solution in which recipients declare their intention to participate during the execution of the broadcast primitive. Their solution assumes a synchronous crash failure model. The challenge was to guarantee consistency among the sequences of messages delivered to different recipients, while achieving timely delivery even in the presence of joins and leaves. We intend to extend their work by studying solutions in a partially asynchronous model (which crash failures) which seems to closer fit the temporal behavior of large scale networks since no a priori knowledge on transmission delays bounds is available in such networks .

Epidemic broadcast, in which each entity propagates the information by broadcasting it to a subset of its neighborhood, is an emerging technique [36], [60]. It turns out that the efficiency of this broadcast technique with respect to the cover of the system strongly relies on the choice of the relays, together with the number of times information is forwarded. Clearly, such constraints can only be estimated because of the dynamicity of the system and the a priori absence of knowledge of the size of the system. The study of the communication graph between entities is a fundamental aspect that we want to address to evaluate the coverage of these assumptions.

Another technique for implementing efficient multicast/unicast primitives has been proposed by Castro et al [42]. Their primitives, built on top of Pastry, a scalable, proximity-aware peer-to-peer overlay, are well adapted for large and highly dynamic groups.

- The second kind of communication schema is semantic-based oriented. This schema is appropriate to face the dynamicity and the size of dynamic and large scale systems. It allows to distribute information to all interested nodes within the system, where the latter are determined according to their known interests and the information contained in the message. This communication schema is captured by the publish/subscribe paradigm. The publish/subscribe paradigm has emerged in the recent years as an effective technique for building distributed applications in which information has to be disseminated from *publishers* (event producers) to *subscribers* (event consumers) [69], [29]. The decoupled nature of publish/subscribe interaction makes these architectures particularly suitable for building large-scale applications where scalability plays a key role. In publish/subscribe systems, users express their interests in receiving certain types of events by submitting a predicate defined on the event contents. The predicate is called the user *subscription*. When a new event is generated and *published* to the system, the publish/subscribe infrastructure is responsible for checking the event against all current subscriptions and delivering it efficiently and reliably to all users whose subscriptions match the event. The challenges that need to be solved are the following ones: 1) storing efficiently and reliably all subscriptions associated with the respective subscribers. 2) Receiving all relevant events from publishers. 3) Dispatching all published events to the correct subscribers.

  Combining expressiveness of subscription language and scalability of the infrastructure poses an interesting challenge that has inspired many researchers to explore this topic further. Despite these promising features, actual deployment of such architectures in real, large-scale systems is currently limited by their lack of self-organization capabilities. For instance, both Siena and Gryphon require connections among brokers to be set up manually; Gryphon also requires all the paths for events to be updated by users upon subscription change; in Kyra [41], when a node joins or leaves the system, the whole subscription partitioning established among all the brokers has to be moved from one broker to another one. We intend to break these limitations with the DPS architecture [31].

3.5.2. Data Aggregation

So far, we have assumed that the information to be disseminated within the network is unique and complete in the sense that recipients do not have to combine received messages to obtain a relevant information.

This assumption is not general. Typically, in sensor networks the quantity of potentially available information is large but it is often considered as incomplete or partial, or even unreliable. Thus, directly disseminating raw data within the network is unacceptable first because of the quantity of information that could be potentially sent, and second because part of it may be useless either because it is irrelevant, or it is unreliable. Thus, to obtain an efficient and, in some extent, reliable dissemination of information, data aggregation is needed. By gathering and/or summarizing raw data either from multiple sensors, or from a single one but during a given period of time, one may extract through global functions a limited set of valid information which can then be disseminated to the application. Data aggregation is a powerful tool to reduce the space complexity of raw information and also to enhance its quality both in terms of reliability and consistency [53], [44], [67].

The latest generation of peer-to-peer (P2P) networks are typically self-organizing and fully distributed systems. A dynamically changing overlay network is maintained, where nodes appear and disappear continuously, and dynamic cooperation among nodes might be created and removed based on the requirements of the particular application. However, such fully distributed platforms have certain drawbacks as well. Control and monitoring is difficult and performing computations poses the challenge of orchestrating the potentially huge number of participating nodes. Hence the need for an aggregation service to provide users or participants of the network with important information like the number of nodes connected to the network, the identity of the most powerful peer, or the total amount of free space in a distributed storage. Interesting results have emerged so far. Astrolabe [68], [72], an aggregation infrastructure for large scale and dynamic network, has recently developed by Van Renesse et al at Cornell, which is an aggregation infrastructure for large scale and dynamic network. Astrolabe provides a good range of functionalities for aggregation (voting, resource location, load balancing, ...). However there are still shortcomings regarding the consistency of the aggregated data in case of failures, the high latency needed for data aggregation, or even the way data is aggregated (in Astrolabe, aggregation is based on a hierarchical structure). Furthermore, simple problems such as computing the size of the network can still be only estimated if local knowledge is used [59].

In this context, we intend to address these shortcoming by designing an aggregation infrastructure satisfying some accuracy, scalability, adaptive and robustness criteria. In terms of accuracy, we want the scheme to be able to provide aggregate information with a certain accuracy in a dynamic large scale environment (where nodes frequently join and leave). With respect to scalability, we want to minimize message passing and avoid flooding schemes that generate excessive redundant messages, to make the scheme scalable to a large network. We also want to ensure that there is good load-balancing, in the sense that no node in the network should be responsible for forwarding a disproportionate amount of network traffic. In many cases it would be very useful if all nodes knew continuously the value of some aggregate, in an adaptive fashion. Adaptability means that if the aggregate changes due to network dynamism or variations in the values to be aggregated, the output of the aggregation protocol should follow this change reasonably quickly. Potential beneficiaries of an efficient implementation of this functionality include protocols responsible for self-organization in large-scale systems and collaborative environments. In terms of robustness, a scheme should be resilient to the dynamics of nodes joining, leaving, and failing arbitrarily and independently in a P2P network [64]. We plan to develop such a data aggregation framework on statistical learning, including information on link loss, and node faults (data corruption on the node). The main building blocks in this framework should include aggregation tree constructor, statistical sampler, distribution estimator and data predictor. An ultimate objective in this domain is to make aggregation a standard service as it appears in databases through requests languages.

3.5.3. Tracking Service

In a wireless sensor network composed of hundreds or even thousands of sensors, accidental faults can occur with high probability (due to the vast number of components involved and the relatively poor quality of this cheap hardware). To achieve reliability, each geographical area is populated with many redundant sensors that are not necessarily activated (in particular to reduce power consumption). When a sensor crashes (either because of battery depletion or due to any accidental faults), neighboring sensors can cover, at least partially, its sensing task. To perform this reconfiguration, information about the available sensors have to be maintained and used to reorganize dynamically the network in the case of sensors removing (and also sensor addition).

Another semantic-based scheme, namely tracking [49], [50], [56], [74], consists of detecting and monitoring locations of real-world objects or individuals, possibly using several types of sensing such as acoustic, seismic, electromagnetic, statistics on the individual behavior, etc. Numerous applications of tracking are currently in use; for example, air traffic control, fleet tracking, habitat and wild animals monitoring, mobile telephony, free-riding monitoring etc.

First we aim to have a precise definition of the tracking problem: a mobile target cross a geographical area covered by sensors that have to cooperate to provide the position of the target to the interested nodes. Similarities with the publish/subscribe problems have to be identified: when a sensor detects the target it publishes the position data and interested nodes have to be notified. Defining the tracking problem as an extension of the publish/subscribe problem will allow us to propose modular solutions. In particular, the definition of efficient self-organizing routing strategies able to resist to the failures of some sensors is a challenge when designing publish/subscribes services. As the notification order has to be consistent with the trajectory of the targets, interesting issues related to clock synchronizations and causal dependencies have also to be investigated.

4. Application Domains

4.1. Space domain applications

Keywords: *Byzantine algorithms*, *modularity*, *soft and hard real-time constraints*.

To cope with more and more complex requirements, this sector of activity shows a growing interest in distributed computing. 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 weaker and weaker) becomes questionable. In particular, regarding **fault tolerance**, a large number of entities (sofware and hardware entities) of the embedded computer-based system interact with each other. To make interaction robust, a broad range of failures (from benign failures up to malicious failures) have to be tolerated. Regarding **Flexibility and adaptability**, the new generation of distributed services has to be adaptive. To achieve this goal, algorithmic solutions have to benefit from the recent advances in software engineering (componentware approach). Finally, **real-time** constraints may span different levels of criticality. Combining these criticality levels with non-functional requirements such as fault tolerance, and distribution requires a provable methodology to specify, design and prove the distributed algorithms needed for this domain.

4.2. Telecommunication applications

Keywords: *broadcast service*, *data aggregation*, *distributed computing*, *publish/subscribe*, *recipient-based broadcast*, *reliable broadcast*, *semantic-based broadcast*, *tracking service*.

Telecommunications domain is very interested in P2P computing. Indeed, for the last few years, more than 90% of the France Telecom traffic has been devoted to peer-to-peer applications. To take into account these new kind of traffic, telecommunication industry needs span from communication functionalities such as publish/subscribe systems for resource discovery and multicast/gossip primitives to disseminate information to geographic aware overlays to limit the cost of non national traffic.

Space and telecommunication applications are being experimented through contracts with the industry and participation to application-oriented research grants.

5. Software

5.1. PROMETEUS: a Group Communication Service

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

Participants: Michel Hurfin, Jean-Pierre Le Narzul, Romaric Ludinard, Frédéric Tronel.

The PROMETEUS project, part of the Inria Gforge, is a software environment for reliable programming developed by the Adept team. It encompasses libraries, frameworks and middleware whose goal is to ease the design and implementation of reliable distributed applications.

5.1.1. Adam

ADAM is a component-based library of agreement abstractions, used to build reliable programming toolkits. ADAM is based on a generic agreement component that can be configured to build many abstractions such as *Atomic Broadcast*, *Membership Management*, *View Synchrony*, *Non-Blocking Atomic Commitment*, etc. Currently, ADAM is used by Paradis, a middleware for Grid applications and Janus, a reliable intrusion detection system.

Regarding the implementation, ADAM is based on EVA [39], an event-based framework for developing distributed abstractions and high-level communication protocols.

5.1.2. Paradis

PARADIS is a middleware for a Grid dedicated to genomic applications. Genomic applications are time-consuming applications that can usually be split 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.

5.1.3. Extensions to cope with Security Aspects

In the context of the DADDi project, we develop a software system that enhance the integrity and availability of an Intrusion Detection Architecture targeted to a Web Server that delivers dynamic contents. Like PARADIS, this system relies on the ADAM library and more specifically on the leader election, membership management and atomic broadcast components. Thanks to these components, our system provides to the intrusion detection architecture, the two following properties: (1) availability through the replication of the IDS (2) integrity of data through the replication of the SQL backend.

6. New Results

6.1. Models for Dynamic, Scalable and Dependable Systems

Keywords: *distributed computing*, *dynamic systems*, *large scale systems*, *models self-organization*, *self-stabilization*.

Participants: Emmanuelle Anceaume, François Bonnet, Michel Hurfin, Frédéric Tronel.

6.1.1. Using Stochastic Models to Analyze the Performances of P2P Algorithms

Massively distributed systems are nowadays of paramount importance. This class of large scale distributed systems requires a paradigm shift when compared to traditional smaller distributed systems. Peer-to-peer algorithms are often advocated as a good solution to tackle the inherent complexity induced by theses systems. In this algorithmic model, no node has a prevalent role over the others nodes in the system. A lot of research has been devoted to the building of network overlays. Network overlays are communication graphs that connect peers. These graphs must exhibit good properties. Indeed, due to the fact that the number of nodes in such systems is expected to be huge, it is virtually impossible for a single to know the complete topology of the system. Hence, it becomes necessary to build network overlays where the number of edges that connect one node to the rest of the system remains small enough to be manageable, while at the same time ensuring that the routing of messages between any two nodes in the system remains efficient (small in number of hops).

Network overlays can be classified as follows :

- hierarchical overlays (Kazaa, eDonkey) : there exists some super-peers, also called servers, that have a complete knowledge of a portion of the overlay.
- structured overlays (Chord, Pastry, CAN) : there is no central node but the network is organized “regularly” with the help of Distributed Hash Tables. Every node receives an unique identify that refers to a special position in a virtual space; the collaboration between nodes is organized in this virtual space.

In [16], we study the performances of a probabilistic algorithm called CYCLON. This algorithm builds an overlay that belongs neither to the hierarchical class of overlays, nor to the structured one. Cyclon is a gossip-based unstructured system: the overlay built by Cyclon do not have an *a priori* structure. Each peer in the network knows a subset of all participating nodes and it exploits randomness to optimize the network with the knowledge of its local view only. We provide a theoretical study of the protocol that gives quantitative measures to characterize the nature of the obtained overlays. We study this algorithm two different models of scheduling. The first model is completely synchronous, while in the second model, we introduce a piece of asynchrony. For both models, we provide simulations and analytical models that coincide very precisely. These analytical results are obtained using classical results coming from the field of Markov chains and generating functions.

6.2. Accidental and Intentional Faults

Keywords: *agreement problem*, *consensus problem*, *distributed computing*, *erroneous suspicions*, *fault-tolerance*, *grid computing*, *group communication*, *intrusion detection*, *mobile robots gathering*, *reputation*, *self-stabilization*, *unreliable failure detector*.

Participants: Emmanuelle Anceaume, Maria Gradinariu, Michel Hurfin, Jean-Pierre Le Narzul, Philippe Raïpin Parvédy, Aina Ravoaja, Julien Pley, Frédéric Tronel.

6.2.1. Failure Detectors

In [25], 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 $\mathcal{S}$.

6.2.2. Agreement Problems

Consensus and Non-Blocking Atomic Commit (NBAC) are two fundamental distributed agreement problems. Intuitively, the specification of these problems assumes that each node in a distributed system starts with a given input value and the goal of each node is to decide on some output value. However, the decided values are restricted such that: all processes that do not crash eventually decide (*termination*), the value decided by all processes is the same (*agreement*), and the value decided must be related to the initial input values (*validity*). The difference between consensus and NBAC is in their validity requirements. Specifically, Consensus only requires that a decided value is also a value that was proposed. In NBAC, on the other hand, it is assumed that the possible initial values are *yes* and *no* and the possible decision values are *commit* and *abort*. If the initial value of at least one node is *no*, then the decision must be *abort*. On the other hand, if the initial values of all nodes are *yes* and there are no crash failures, then the decision value must be *commit*. Despite the similarity in structure of the definitions of consensus and NBAC, in asynchronous distributed systems prone to crash failures, these are two different problems. In particular, neither problem can be solved in a purely asynchronous system. However, it was shown that the minimal synchrony required to solve consensus is strictly weaker than the one required to solve NBAC [48] (we make this statement more precise below). On the other hand, a black-box implementation of NBAC is not sufficient to solve consensus in an otherwise asynchronous environment.

In [11], we propose a family of problems that we call *managed agreement*, which generalizes both NBAC and consensus. Specifically, the definition of managed agreement is based on the notion of *aristocrat* nodes. In managed agreement, there exists a value such that if any of the aristocrats proposes this value, then a corresponding value must be decided. On the other hand, if none of the aristocrats proposed the special value and none of the aristocrats failed, then any possible decision value that corresponds to a value that was proposed can be decided on. Thus, NBAC is a special case of managed agreement when all nodes are aristocrats, whereas consensus is a special case of managed agreement when there are no aristocrats.

6.2.3. Using Agreement Services in Grid Computing

A Grid is a distributed system involving heterogeneous resources located in different geographical domains that are potentially managed by different organizations (companies, laboratories, universities, ...) or individuals. The major purpose of a Grid is to federate multiple powerful distributed resources (computers but also data storage facilities) within a single virtual entity which can be accessed transparently and efficiently by external users. In our work, we consider a Grid composed of resources provided by various institutions. These potential contributors are identified preliminarily and correspond to well-established institutions that agree to share their resources and to trust each other. Yet each institution keeps its independence and freedom. The decision to include or to exclude some (or even all) local resources from the Grid can be taken at any time by the local administrator without any coordination with the others.

In [23] and [24], we specifically address the resource allocation and dependability issues. We define services that allow a Grid user to continuously take full advantage of the computing power offered by the Grid in a simple and completely transparent manner. Whatever the circumstances, a complete transparency and a quick response time are always expected by the customers. To fulfill these two requirements, adaptive control mechanisms have to be proposed, on one hand to cope efficiently with the dynamic changes of the computing capacity of the Grid (even if these changes are unpredictable) and, on the other hand to distribute the tasks among the resources in an efficient way (dynamic load balancing). This leads us to address two major issues that both require a continuous adaptation to the changing computing environment, namely the *Resource allocation* issue and the *dependability* issue. We propose to solve both problems in an homogeneous way using a slightly modified group concept [66]. More precisely, all distant interactions between domains corresponding to distinct organizations are managed by a small group of registered processors (exactly one per domain). Each member of this group acts as a *master* for its own domain and interacts with the other members of the group to build consistent observations (1) of current workloads in each domain and (2) of the current composition of the group. In that sense, we argue that, in a distributed system prone to failures, an agreement service is a key concept to transform several local views into a single global one without opting for a centralized control approach and thus without having a single point of failure. An agreement service allows all the domains to acquire the same set of accurate data describing the current state of the Grid. Based on this unanimous observation, each domain can locally enact the right adaptation to react to the observed changes.

6.2.4. A Dependable Intrusion Detection Architecture Based on Agreement Services

Intrusion detection is one of the numerous techniques that help improving the overall security of a system. It is traditionally based on explicit methods that be can classified as follows :

- scenario-based techniques : one needs to explicitly provide scenarios of previously known attacks. All kinds of probes are placed in the system to protect and the flow of information returns by these probes is analyzed to find possible traces of attacks. Of course, this method is extremely sensible to the quality of scenarios that are provided. Furthermore, it cannot detect previously unknown attacks.
- behavioral techniques : this method requires to provide a reference model which precisely describes the legal states of a system. The system is observed by an external observer which raises alarms when the system deviates from the set of legal states. Of course, providing a good reference that do not raise to many false alarms, and that can detect real attacks is challenging.

In [22] we take a radically different approach called implicit intrusion detection. We show that the use of diversified COTS servers allows to detect intrusions. This approach can detect even previously unknown attacks. Based on a architecture proposed by our partners in the DADDi project [71], we propose an extension that brings the availability and integrity properties to the system. Replication techniques implemented on top of agreement services are used to avoid any single point of failure.

We focus on a particular case study, namely a Web server that delivers dynamic content. This technology traditionally implements the storage of this content in a database backend that receives read/write operations issued by the Web server. An interesting property of this Web server architecture resides in the fact that the whole internal state of the COTS servers is located in the database backend and that any change to the internal state is carried out by the means of SQL queries. We take advantage of this property in order to ensure integrity of the data by introducing proxies located between the Web servers and the database whose goal is to compare the SQL queries submitted by the diverse Web servers to the database. As unexpected SQL queries issued by a corrupted Web server would threaten data integrity, we use a majority voting algorithm to compare queries submitted to the database; this comparison allows us to detect and mask any attempt to data integrity.

6.2.5. Fault-tolerant and Self-stabilizing Mobile Robots Gathering

Gathering is a fundamental coordination problem in cooperative mobile robotics. In short, given a set of robots with arbitrary initial location and no initial agreement on a global coordinate system, gathering requires that all robots, following their algorithm, reach the exact same but not predetermined location. Gathering is particularly challenging in networks where robots are oblivious (i.e., stateless) and the direct communication is replaced by observations on their respective locations. interestingly any algorithm that solves gathering with oblivious robots is inherently self-stabilizing.

In [21], we significantly extend the studies of deterministic gathering feasibility under different assumptions related to synchrony and faults (crash and Byzantine). Unlike prior work, we consider a larger set of scheduling strategies, such as bounded schedulers, and derive interesting lower bounds on these schedulers. in addition, we extend our study to feasibility of probabilistic gathering in both fault-free and fault-prone environments. To the best of our knowledge our work is the first to address the gathering from a probabilistic point of view.

6.2.6. Reputation

With the emergence of e-commerce in open, large-scale distributed marketplaces, reputation systems are becoming attractive for encouraging trust among entities that usually do not know each other. A reputation system collects, distributes, and aggregates feedback about the past behavior of a given entity. The derived reputation score is used to help entities to decide whether a future interaction with that entity is conceivable or not. Without reputation systems, the temptation to act abusively for immediate gain can be stronger than the one of cooperating. In closed environments, reputation systems are controlled and managed by large centralized enforcement institutions. Designing reputation systems in P2P systems has to face the absence of such large and recognizable but costly organizations capable of assessing the trustworthiness of a service provider. The only viable alternative is to rely on informal social mechanisms for encouraging trustworthy behavior. Proposed mechanisms often adopt the principle that "you trust the people that you know best", just like in the word-of-mouth system, and build transitivity trust structures in which credible peers are selected. However such structures rely on the willingness of entities to propagate information. Facing free-riding and more generally under-participation is a well known problem experienced in most open infrastructures. The efficiency and accuracy of a reputation system depends heavily on the amount of feedback it receives from participants. According to a recognized principle in economics, providing rewards is an effective way to improve feedback. However rewarding participation may also increase the incentive for providing false information. Thus there is a trade-off between collecting a sizable set of information and facing unreliable feedback. An additional problem that needs to be faced with P2P systems, is that peers attempt to collectively subvert the system. Peers may collude either to discredit the reputation of a provider to lately benefit from it (bad mouthing), or to advertise the quality of service more than its real value to increase their reputation (ballot stuffing). Lot of proposed mechanisms break down if raters collude.

In [14] we address the robust reputation problem. Essentially this problem aims at motivating peers to send sufficiently honest feedback in P2P systems in which peers may free-ride or be dishonest. This work has been motivated by a previous one in which the proposed architecture is built on top of a supervising overlay made of trusted peers [34]. The mechanism we propose achieves high robustness to attacks (from individual peers or from collusive ones), and provides incentive for participation. This is accomplished by an aggregation technique in which a bounded number of peers randomly selected within the system report directly observed information to requesting peers. Observations are weighted by a credibility factor locally computed. Incentive for participation is implemented through a fair differential service mechanism. It relies on peer's level of participation, a measure of peers' contribution over a fixed period of time, and on the credibility factor, assessing the confidence one has in a peer. Our results are promising: We prove that through sufficient and honest cooperation, peers increase the quality of their reputation mechanism. We show that the reputation estimation efficiently filters out malicious behaviors in an adaptive way. Presence of a high fraction of malicious peers does not prevent a correct peer from computing an accurate reputation value, at the expense of a reasonable convergence time. Furthermore, the trade-off between the sensitivity of the mechanism facing up malicious peers and the duration of the computation is tuned through a single input parameter. These properties, combined with the incentive scheme, makes our mechanism adapted to P2P networks.

6.3. Availability and Consistency of Data in Dynamic Systems

Keywords: *distributed computing*, *dynamic quorum*, *reconfiguration*, *replication*, *shared data*.

Participants: Emmanuelle Anceaume, Maria Gradinariu, Vincent Gramoli.

We have proposed an architecture for *self-adjusting and self-healing atomic memory* in highly dynamic systems exploiting peer-to-peer (p2p) techniques. Our approach, named *SAM* [32], [33], brings together new and old research areas such as p2p overlays, dynamic quorums and replica control. In SAM, nodes form a connected overlay. To emulate the behavior of an atomic memory we use intersected sets of nodes, namely *quorums*, where each node hosts a replica of an object. In our approach, a quorum set is obtained by performing a deterministic traversal of the overlay. The SAM overlay features self-* capabilities: that is, the overlay self-heals on the fly when nodes hosting replicas leave the system and the number of active replicas in the overlay dynamically self-adjusts with respect to the object load. In particular, SAM pushes requests from loaded replicas to less solicited replicas. If such replicas do not exist, the replicas overlay self-adjusts to absorb the extra load without breaking the atomicity. We have proposed a distributed implementation of SAM where nodes exploit only a restricted local view of the system, for the sake of scalability. Specifications, simulations, and some improvements of the SAM solution have been proposed in [26].

6.4. Dissemination of Information

Keywords: *P2P*, *connected sensor cover*, *connectivity*, *distributed computing*, *publish/subscribe*, *sensor networks*, *trajectory tracking*.

Participants: Emmanuelle Anceaume, Florent Claerhout, Maria Gradinariu, Michel Hurfin, Philippe Raïpin Parvédy.

6.4.1. Publish/Subscribe Paradigm

Publish/Subscribe systems provide a useful platform for delivering data (events) from publishers to subscribers in an anonymous fashion in distributed networks. In [13], we promote a novel design principle for self-. dynamic and reliable content-based publish/subscribe systems and perform a comparative analysis of its probabilistic and deterministic implementations. More specifically, in this work, we present a generic content-based publish/subscribe system, called DPS (Dynamic Publish/Subscribe). DPS combines classical content-based filtering with self-. (self-organizing, self-configuring, and self-healing) subscription-driven clustering of subscribers. DPS gracefully adapts to failures and changes in the system while achieving scalable events delivery. DPS includes a variety of fault-tolerant deterministic and probabilistic content-based publication/subscription schemes. These schemes are targeted toward scalability, and aim at reducing and distributing the number of messages exchanged. Reliability and scalability of our system are shown through analytical and experimental evaluation.

6.4.2. Connectivity in Peer-to-Peer Networks

The network connectivity is a basic requirement while implementing fundamental communication and storage abstractions in P2P networks, featuring scalability and fault-tolerance. The quality of services of abstractions like for example multicast, publish/subscribe, group membership or persistent storage is strongly related to the connectivity degree of the underlying overlay. Intuitively, a higher overlay connectivity ensures a reinforced reliability and consequently, the deployment of distributed applications with real-time constraints on top of these overlays becomes feasible even in environments characterized by a high dynamicity, i.e., nodes arriving and departing at a high rate.

In [19], we propose a novel δ-connected DHT-free P2P overlay. This overlay offers strong connectivity guarantees despite the system dynamicity. The construction and the maintenance of our overlay is completely decentralized and handled strictly locally, through deterministic algorithms whose correctness is rigorously proved. The overlay has three main features: (1) its diameter grows logarithmically with the number of nodes in the network; (2) it remains connected after random failures of a linear subset of its nodes and/or edges and (3) partitions occur only if at least δ are concurrently removed.

6.4.3. Self-$*$ Query Region Covering in Sensor Networks

In sensor networks, queries are sent from some devices (a PDA, a laptop, or any computer) to sense some data/events over some time period and a geographical region, called the query region. The query region is usually a subpart of the whole region covered by the sensor network. Since the sensors are usually densely deployed, there are usually much more sensors than necessary to process a given query. To reduce usage of energy, only some sensor nodes have to be fully active. The passive ones contribute neither to the sensing activity nor to the message transmission induced by the query. Determining a set of active sensors is modeled as an optimization problem, called the "connected sensor cover" problem. Given a query over a sensor network, the goal is to select a minimum or nearly minimum set of sensors, called connected sensor cover, such that the selected sensors cover the query region, and form a connected network among themselves. In its general form, this problem is known to be NP-hard.

In [12], we design fully distributed, strictly localized, scalable, and self-$*$ solutions to the connected sensor cover problem. We present 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. Moreover, in the proposed algorithms, the faults are contained only within the neighborhood of the faulty nodes.

6.4.4. Trajectory tracking in sensor networks

In [18] we address the problem of trajectory tracking that deals with gathering coherent information on the past behavior of a mobil target. When a target enters and moves within a region covered by a sensor network, information about this target is generated by any active sensor that detects the target in its monitoring area. These gathered data have to be received by some registered nodes that are in charged of identifying the trajectory of the target to perform latter complex computations at the application level (e.g., trajectory forecasting, pursuer/evader, optimization of the management of natural disasters, etc.). Our first contribution consists in formally specifying the problem: the proposed formal specification is the first one to the best of our knowledge. It extends a classical publish/subscribe specification.

We also propose an original architecture combining three distinct abstractions (target detector, trajectory manager and notification manager) which allows to describe various algorithmic solutions. A target detector is in charge of (1) detecting targets when they cross the monitored area and (2) time-stamping the generated detection reports. The purpose of the trajectory manager is to initiate the publication of a detection event: thanks to a delegation mechanism and an aggregation mechanism, the publication is not necessarily performed by the node which has previously made the detection and the published data does not allays corresponds to a single detection report. Finally, a notification manager (publish/subscribe mechanism) is in charge of dispatching trajectory related events to the interested parties. The proposed architecture allows to implement a wide range of algorithmic solutions (including the simple flooding solution). We suggest a new solution called TRAC which relies on two distinct overlay (a sequence of sensors that maps the trajectory and a tree of sensors such that (1) the root is one node on the trajectory (2) the distance between any node and the root is minimal. The proposed algorithms cope with node additions/failures, transient faults, and crashes.

A simulator has been designed to evaluate the interest of some algorithmic solutions. In [17] , two solutions (namely the basic flooding solution and the Trac solution) are compared using some particular scenarios. When the selected scenario is simple enough, analytical interpretations are provided and compared to simulation results. Our main concern is the number and the size of the messages that are exchanged to provide the detection reports to the interested nodes. The higher these two values are the more important is the energy consumption.

7. Contracts and Grants with Industry

7.1. Assert Contract (2004-2006)

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

Participant: 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. In 2006, one deliverable co-authored with people from INRIA Rocquencourt and people from the Vienna University of Technology have been provided [27].

8. Other Grants and Activities

8.1. National Project

8.1.1. ACI Daddi (2004-2007)

Participants: Michel Hurfin, Jean-Pierre Le Narzul, Romaric Ludinard, Frédéric Tronel.

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 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 for a given entity (user, machine, service, or application) in order to compare it with the current observed behavior. If the observed behavior diverges from 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 results that are returned 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.

8.1.2. ACI TAGADA (2004-2007)

Participant: Frédéric Tronel.

The Adept team is involved in the TAGADA ACI project since early 2005. TADAGA is a three years project funded by the French ministry of education, research and technology within the framework of the program "*ACI Jeunes Chercheurs*". The TAGADA project, which is an acronym for "Topologie Algébrique, Géométrie pour l'Algorithmique Distribuée Asynchrone" focuses on the study of models for distributed systems in order to improve the dependability (mainly fault-tolerance) and security of such systems. The project involves the following people :

- Emmanuel Godard (coordinator, University of Provence)
- Rémi Morin (University of Provence)
- Luigi Santocanale (University of Provence).
- Eric Goubault and Emmanuel Haucourt (CEA).
- Matthieu Roy (LAAS - CNRS)
- Frédéric Tronel (IRISA)

These last years have seen an important research activity related to geometrical models in the domain of distributed systems, as well as the theory of concurrency. In both case, it has been shown that using high dimensional models (like simplicial complexes, or high dimensional automata) can help capturing subtle properties of concurrent systems. They have shown to be superior than more traditional approaches by the fact they do not lose any semantical precision while at the same time being sufficiently concise. However these mathematical objects remain extremely complex. That's why they have to be studied by the mean of mathematical tools (like homological groups) inherited from the field of algebraic topology. The goal of algebraic topology is to state whether two geometrical objects are similar (modulus continuous et reversible transformation). This can be achieved by associating algebraic objects to geometrical objects and comparing these (much simpler) algebraic objects (like groups). Extremely important results have been obtained by several researcher like Maurice Herlihy and Sergio Rajsbaum in the domain of fault-tolerant distributed systems or Eric Goubault in the domain of concurrency. In the TAGADA project, we want to study and try to unify these results.

8.1.3. RIAM Project (ANR): Solipsis (2006-2008)

Participant: Emmanuelle Anceaume.

The project involves the following people :

- France Telecom (prime)
- IRISA (ASap, Adept)
- Archivedeo
- Artefacto
- University of Rennes II (LARES)

The beginning of the 21st century is marked by a surprising social phenomenon: the users evolved from consumers of contents to producers, editors and broadcaster. The popularity of weblogs, picture sharing or peer-to-peer networks takes part of this movement which destabilizes the sector of the digital contents and telecommunications. The industry related to Virtual worlds, especially virtual reality and network games, has not yet been affected by this new paradigm. Two explanations are regularly proposed. First of all, the 3D tools aim almost exclusively computer graphics experts. Then, no service to share "self-produced" creations easily emerged. However, the immersion in virtual worlds is an attractive experience which knows an important interest as current computers allows a great number of users to interact in a rich and varied way among complex virtual environments. A new interesting challenge is to offer the opportunity to the users to imply themselves in the collective creation of a public virtual space. As Neal Stephenson in "Snow crash" depicts it, its prospect is likely to upset the communication between the human ones. This project aims at carrying out a significant step in this direction through three development and complementary research orientations: Peer-to-Peer systems, modelling and vizualisation of 3D contents and social. In the very context of peer-to-peer networks, virtual worlds are currently the property of companies or associations which manage servers unless these environments would not exist. Even the most open systems are not freed from a central and responsible authority. However, this architecture prevent the construction of a free and perennial virtual world. Indeed, beyond the penal problems and the potential abuses of power, a rich ecosystem can probably not develop whereas the system is controlled by a third all-powerful actor. Thus, we consider an open system based on a software peer-to-peer architecture in which the maintenance of the properties of the virtual world is ensured by the coordination of the participants. Moreover, this architectural choice guarantees that the virtual world can be populated by an unlimited number of users, without cost of deployment.

8.2. International Cooperations

8.2.1. Brazil (Federal University of Bahia and Federal University of Campina Grande)

Participants: Michel Hurfin, Jean-Pierre Le Narzul.

A cooperation project with the Federal University of Bahia (Prof. Fabiola Greve), the Federal University of Paraiba (Prof. Francisco Brasileiro) and several French laboratories (ADEPT Team, GRAND LARGE Team, REGAL Team and ENST Bretagne) is supported by Capes/Cofecub (projet 497/05) during a period of two years (2005-2006). Michel Hurfin is the French coordinator of this project which focuses on distributed computing and Grid computing. In 2006, Fabiola Greve (Federal University of Bahia) and Walfredo Cirne (Federal University of Campina Grande) have visited some of the French Laboratories. Pierre Sens (REGAL team) has visited the Federal University of Bahia while Michel Hurfin has visited the Federal University of Campina Grande. During his visit, Michel Hurfin has worked on the evaluation of a consensus protocol that has two unique features. Firstly, it mitigates the problems due to bad QoS delivered by the failure detector by allowing processes to simultaneously participate in multiple rounds. Secondly, it allows its decision pattern to be configured to have different numbers of processors allowed to autonomously decide. We have measured the decision latency of the protocol to conduct the performance analysis.

8.2.2. USA (University of Nevada)

Participant: Maria Gradinariu.

Maria Gradinariu is involved in several joint activity with Ajoy Datta, professor at the University of Nevada. In particular, she is the co-supervisor of master students of the university of Nevada. Four papers that have been published in 2006 are co-authored with Ajoy Datta.

8.2.3. Japan (JAIST)

Participant: Maria Gradinariu.

Maria Gradinariu has spent more than two weeks in 2006 at JAIST. The results of a joint work with Xavier Défago have been published at DISC 2006.

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.
- Frédéric Tronel has the responsibility of managing the master in computer science devoted to security aspects (University of Rennes I, Ifsic).
- Emmanuelle Anceaume belongs to the specialist commission (university of Rennes I, section 27).
- Michel Hurfin gives lectures on fault tolerance and distributed computing to students of two engineering schools: ENST Bretagne (Rennes, 10 hours) and Supelec (Rennes, 12 hours).
- Maria Gradinariu gives lectures at the university of Rennes I and at ENST Bretagne (on P2P systems).

9.2. Presentations of Research Works

- Since january 2000, Emmanuelle Anceaume co-organizes with Bruno Tuffin the seminars entitled "Networks and Systems" that are periodically held in our institute.
- In september 2006, Emmanuelle Anceaume has presented her research activities during a seminar in Dagsthul.
- Michel Hurfin has presented his research activities during a visit in the Federal University of Campina Grande.
- 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

- Emmanuelle Anceaume
  - belongs to the organization committee of the 8ème Rencontre Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel 2006), May 2006, Trégastel, France.
  - acts as the publicity chair of the 3rd International Conference on Autonomic and Trusted Computing (ATC 2006), September 2006, China.
  - acts as the publicity chair of the 2007 IEEE International Symposium on Ubisafe Computing (UbiSafe-07), May 2007, Niagara Falls, Ontario, Canada.

  - acts as a program committee member of the 2nd International Workshop on Trustworthiness, Reliability and services in Ubiquitous and Sensor neTworks (TRUST 2007), December 2007, Taipei, Taiwan.
  - acts as a program committee member of the IEEE 21st International Conference on Advanced Information Networking and Applications (AINA-07), May 2007, Niagara Falls, Canada.
  - acts as a program committee member of the First International Workshop on Application and Security Service in Intelligent Environments'07 (ASSIE'07), May 2007.
  - acts as program co-chair of the Third International Symposium in Ubiquitous Computing (Secubiq-2007), December 2007, Taipei, Taiwan.
- Maria Gradinariu
  - belongs to the organization committee of the *8ème Rencontre Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel 2006)*, May 2006, Trégastel, France.
  - acts as a program committee member of the International Workshop on Reliability in Decentralized Distributed systems (RDDS 2006), october 2006, Montpellier, France.
  - is the program co-chair of the 8th Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2006), November 2006, Dallas, Texas.
- Michel Hurfin
  - acts as a member of the program committee of the *8th International Symposium on Self Stabilizing Systems (SSS 2006)*, November 2006, Dallas, Texas.
  - acts as a member of the program committee of the IEEE International Symposium on Ubisafe Computing (UbiSafe-07), May 2007, Niagara Falls, Ontario, Canada.

10. Bibliography

Major publications by the team in recent years

[1] E. ANCEAUME, A. DATTA, M. GRADINARIU, G. SIMON. *Publish/Subscribe Scheme for Mobile Networks*, in "Proc. of the 2nd ACM International Workshop on Principles of Mobile Computing (POMC), Toulouse, France", October 2002, p. 74–81.

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

[3] 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 3rd International Symposium on Parallel and Distributed Computing (ISPDC), Cork, Ireland", July 2004, p. 262–269.

[4] J. BEAUQUIER, A. DATTA, M. GRADINARIU, F. MAGNIETTE. *Self-stabilizing local mutual exclusion and daemon refinement*, in "Proc. of the 14th International Symposium on Distributed Computing (DISC), Toledo, Spain", LNCS 1914, Springer Verlag, October 2000, p. 223–237.

[5] J. BEAUQUIER, M. GRADINARIU, C. JOHNEN. *Memory space requirements for self-stabilizing leader election protocols*, in "Proc. of the 18th IEEE Symposium on Principles of Distributed Computing (PODC), Atlanta, Georgia", May 1999, p. 199–207.

[6] F. BRASILEIRO, F. GREVE, M. HURFIN, J.-P. LE NARZUL, F. TRONEL. *Eva: an Event-Based Framework for Developing Specialised Communication Protocols*, in "Proc. of the 1st IEEE International Symposium on Network Computing and Applications (NCA), Cambridge, MA", February 2002.

[7] J.-M. HELARY, M. HURFIN, A. MOSTÉFAOUI, M. RAYNAL, F. TRONEL. *Computing Global Functions in Asynchronous Distributed Systems with Process Crashes*, in "Proc. of the 20th International Conference on Distributed Computing Systems (ICDCS)", Best paper award, April 2000, p. 584–591.

[8] 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.

[9] 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.

[10] 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, August 2002, p. 900–915.

Year Publications

Articles in refereed journals and book chapters

[11] E. ANCEAUME, R. FRIEDMAN, M. GRADINARIU. *Managed Agreement: Generalizing Two Fundamental Distributed Agreement Problems*, in "Information Processing Letters", to appear, 2006.

[12] A. DATTA, M. GRADINARIU, P. LINGA, P. RAÏPIN PARVÉDY. *Self* query region covering in sensor networks*, to appear, 2006.

Publications in Conferences and Workshops

[13] E. ANCEAUME, M. GRADINARIU, A. DATTA, G. SIMON, A. VIRGILLITO. *A Semantic Overlay for Self- Peer-to-Peer Publish/Subscribe*, in "Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06), Lisboa, Portugal", July 2006.

[14] E. ANCEAUME, A. RAVOAJA. *Incentive-based Robust Reputation Mechanism for P2P Services*, in "Proc. of the 10th International Conference On Principles Of Distributed Systems (OPODIS 2006), Bordeaux, France", December 2006.

[15] F. BONNET, P. EZHILCHELVAN, E. VOLLSET. *Quiescent Consensus in Mobile Ad-hoc Networks using Eventually Storage-Free Broadcasts*, in "Proc. of the 21st ACM symposium on Applied Computing (SAC 2006), Dijon, France", April 2006, p. 670–674.

[16] F. BONNET, F. TRONEL, S. VOULGARIS. *Brief Announcement: Performance Analysis of Cyclon, an Inexpensive Membership Managerment for Unstructured P2P Overlays*, in "Proc. of the 20th International Symposium on Distributed Computing (DISC 2006), Stockholm, Sweden", LNCS 4167, Springer Verlag, September 2006, p. 560–562.

[17] F. CLAERHOUT. *Evaluation of a Tracking Architecture in Wireless Sensor Networks*, in "Proc. of the 8th International Symposium on Stabilization Safety and Security (SSS 2006), Dallas, Texas", LNCS 4280, Springer Verlag, November 2006, p. 169–183.

[18] F. CLAERHOUT, A. DATTA, M. GRADINARIU, M. HURFIN. *Self-* Architecture for Trajectory Tracking in Wireless Sensor Networks*, in "Proc. of the 5th IEEE International Symposium on Network Computing and Applications (NCA 2006), Cambridge, Massachusetts", July 2006, p. 40–47.

[19] A. DATTA, M. GRADINARIU, A. VIRGILLITO. *Deterministic delta-Connected Overlay for Peer-to-Peer Networks*, in "Proc. of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 2006), Gyeongju, Korea", April 2006, p. 159–165.

[20] V. DRABKIN, R. FRIEDMAN, M. GRADINARIU. *Self-stabilizing Wireless Connected Overlays*, in "Proc. of the 10th International Conference On Principles Of Distributed Systems (OPODIS 2006), Bordeaux, France", December 2006.

[21] X. DÉFAGO, M. GRADINARIU, S. MESSIKA, P. RAÏPIN PARVÉDY. *Fault-tolerant and Self-stabilizing Mobile Robots Gathering*, in "Proc. of the 20th International Symposium on Distributed Computing (DISC 2006), Stockholm, Sweden", LNCS 4167, Springer Verlag, September 2006, p. 46–60.

[22] M. HURFIN, J.-P. LE NARZUL, F. MAJORCZYK, L. MÉ, A. SAIDANE, E. TOTEL, F. TRONEL. *A dependable Intrusion Detection Architecture based on Agreement Services*, in "Proc. of the 8th International Symposium on Stabilization Safety and Security (SSS 2006), Dallas, Texas", LNCS 4280, Springer Verlag, November 2006, p. 378–394.

[23] M. HURFIN, J.-P. LE NARZUL, J. PLEY, P. RAÏPIN PARVÉDY. *PARADIS: an Adaptive Middleware for Dynamic Task Allocation in a Grid*, in "Proc. of the Brazilian Symposium on Computer Networks (SBRC 2006), Curitiba, Brazil", June 2006.

[24] M. HURFIN, J.-P. LE NARZUL, J. PLEY, P. RAÏPIN PARVÉDY. *Using Agreement Services in Grid Computing*, in "Proc. of the International Symposium on Parallel and Distributed Processing and Applications (ISPA 2006, Frontier on High Performance Computing and Networking), Sorrento, Italy", LNCS 4331, Springer Verlag, December 2006, p. 165–174.

[25] M. LARREA, M. HURFIN, A. LAFUENTE. *Implementing Failure Detector Classes Perfect and Strong with Limited Link Synchrony*, in "Proc. of the Sixth IEEE European Dependable Computing Conference (EDCC-6, fast abstract), Coimbra, Portugal", October 2006.

Internal Reports

[26] E. ANCEAUME, V. GRAMOLI, A. VIRGILLITO. *Square: Scalable Quorum-Based Atomic Memory with Local Reconfiguration*, Technical report, n^{o} 1805, IRISA, 2006.

Miscellaneous

[27] B. CHARRON-BOST, J.-F. HERMANT, M. HURFIN, G. LE LANN, A. SCHIPER, M. BIELY, M. HUTLE, U. SCHMID, J. WIDDER. *Deliverable D1.1-3b (Previously labeled D1.1-4): Analysis of preliminary Middleware Solutions for ASSERT Phase 2 Selected Topics*, 103 pages, February 2006.

References in notes

[28] M. AGUILERA, W. CHEN, S. TOUEG. *Failure Detection and Consensus in the Crash-recovery Model*, in "Distributed Computing", vol. 13, n^o 2, 2000, p. 99–125.

[29] M. K. AGUILERA, R. E. STROM, D. C. STURMAN, M. ASTLEY, T. D. CHANDRA. *Matching Events in a Content-Based Subscription System*, in "Symposium on Principles of Distributed Computing", 1999, p. 53-61, http://www.research.ibm.com/distributedmessaging/gryphon.html.

[30] 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.

[31] E. ANCEAUME, M. GRADINARIU, A. DATTA, G. SIMON, A. VIRGILLITO. *A Semantic Overlay for Self- Peer-to-Peer Publish/Subscribe*, in "Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06), Lisboa, Portugal", July 2006.

[32] E. ANCEAUME, M. GRADINARIU, V. GRAMOLI, A. VIRGILLITO. *P2P Architecture for self-* Atomic Memory*, in "Proc. of the ACM-sigact International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)", 2005.

[33] E. ANCEAUME, M. GRADINARIU, V. GRAMOLI, A. VIRGILLITO. *SAM: Self* Atomic Memory for P2P Systems*, Technical report, n^o 1717, IRISA, 2005, http://www.irisa.fr/bibli/publi/pi/2005/1717/1717.html.

[34] E. ANCEAUME, M. GRADINARIU, A. RAVOAJA. *Incentive for P2P fair resource sharing*, in "Proc. of the IEEE P2P Conference", IEEE, 2005.

[35] H. ATTIYA, J. WELCH. *Distributed Computing : Fundamentals, Simulations and Advanced Topics*, 1999.

[36] S. BAEHNI, P. EUGSTER, R. GUERRAOUI. *Data-Aware Multicast*, in "Proc. of the 5th International Conference on Dependable Systems and Networks", IEEE, 2004.

[37] Z. BAR-JOSEPH, I. KEIDAR, N. LYNCH. *Early-delivery dynamic atomic broadcast*, Technical report, n^o MIT-LCS-TR-840, MIT Lab. for Computer Science, 2002.

[38] O. BIRAN, S. MORAN, S. ZAKS. *A combinatorial Characterization of the Distributed Tasks which are Solvable in the Presence of One Faulty Processor*, in "Proc. of the 7th Principles of Distributed Computing", august 1998.

[39] F. BRASILEIRO, F. GREVE, M. HURFIN, J. LE NARZUL, F. TRONEL. *Eva: an Event-Based Framework for Developing Specialised Communication Protocols*, in "Proc. of the 1st Int. Symp. on Network Computing and Applications (NCA2001)", 2002, p. 108–119.

[40] R. CANETTI, R. GENNARO, A. HERZBERG, D. NAOR. *Proactive security: Long-term protection against break-ins*, in "RSA Laboratories' CryptoBytes", vol. 33, n^o 1, 1997, p. 1–8.

[41] F. CAO, J. SINGH. *Efficient event routing in content-based publish-subscribe service networks*, in "Proc. of the 23rd Conference on Computer Communications", 2004.

[42] M. CASTRO, P. DRUSCHEL, A.-M. KERMARREC, A. ROWSTRON. *Scalable Application-level Anycast for Highly Dynamic Groups*, in "Proc. Of NGC 2003", 2003.

[43] M. CASTRO, B. LISKOV. *Proactive Recovery in a Byzantine-Fault-Tolerant System*, in "In Proc of the 4th Symposium on Operating Systems Design and Implementation (OSDI)", 2000, p. 273-287.

[44] A. CERPA, D. ESTRIN. *ASCENT: Adaptive Self-Configuring SEnsor Networks Topologies*, in "IEEE Transactions on Mobile Computing Special Issue on Mission-Oriented Sensor Networks", vol. 3, n^{o} 3, 2004.

[45] T. D. CHANDRA, V. HADZILACOS, S. TOUEG. *The Weakest Failure Detector for Solving Consensus*, in "Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing (PODC'92), Vancouver, BC, Canada", M. HERLIHY (editor). , ACM Press, 1992, p. 147–158, http://citeseer.ist.psu.edu/chandra96weakest.html.

[46] T. D. CHANDRA, S. TOUEG. *Unreliable failure detectors for reliable distributed systems*, in "Journal of the ACM", vol. 43, n^{o} 2, 1996, p. 225–267, http://citeseer.ist.psu.edu/chandra96unreliable.html.

[47] S. CHAUDHURI. *More Choices Allow More Faults: Set Consensus Problems in Totally Asynchronous Systems*, Technical report, n^{o} MIT/LCS/TM-475, 1992, http://citeseer.ist.psu.edu/chaudhuri92more.html.

[48] C. DELPORTE-GALLET, H. FAUCONNIER, R. GUERRAOUI, V. HADZILACOS, P. KOUZNETSOV, S. TOUEG. *The weakest failure detectors to solve certain fundamental problems in distributed computing*, in "Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing (PODC 2004)", 2004, p. 338–346.

[49] M. DEMIRBAS, A. ARORA, M. GOUDA. *A Pursuer-Evader Game for Sensor Networks*, in "Sixth Symposium on Self-Stabilizing Systems (SSS)", 2003.

[50] M. DEMIRBAS, A. ARORA, T. NOLTE, N. LYNCH. *STALK: A Self-Stabilizing Hierarchical Tracking Service for Sensor Networks*, in "Symposium on Principles of Distributed Computing (PODC)", ACM, 2004.

[51] S. DOLEV, S. GILBERT, N. LYNCH, A. SHVARTSMAN, J. WELCH. *Geoquorums: Implementing atomic memory in mobile ad hoc networks*, in "Proc. of the 17th International Symposium on Distributed Computing (DISC 2003)", 2003.

[52] X. DÉFAGO, A. SCHIPER, P. URBAN. *Totally ordered broadcast and Multicast algorithms: A comprehensivesurvey*, in "ACM Computing Surveys", vol. 36, n^{o} 4, 2004, p. 372–421.

[53] J. ELSON, D. ESTRIN. *A bridge to the Physical World*, chap. 1, in Wireless Sensor Networks, Kluwer, 2004.

[54] M. J. FISCHER, N. A. LYNCH, M. S. PATERSON. *Impossibility of distributed consensus with one faulty process*, in "J. ACM", vol. 32, n^{o} 2, 1985, p. 374–382.

[55] V. HADZILACOS, S. TOUEG. *Fault-tolerant broadcasts and related problems*, S. MULLENDER (editor). , chap. Distributed Systems, ACM Press, 1996.

[56] T. HE, S. KRISHNAMURTHY, J. STANKOVIC, T. ABDELZAHER, L. LUO, R. STOLERU, T. YAN, L. GU. *Energy-Efficient Surveillance System Using Wireless Sensor Networks*, in "ACM Press", 2004, p. 270-283.

[57] M. HERLIHY, N. SHAVIT. *The Asynchronous Computability Theorem for t-Resilient Tasks (Preliminary Version)*, in "ACM Symposium on Theory of Computing", 1993, p. 111-120, http://citeseer.ist.psu.edu/herlihy93asynchronous.html.

[58] M. HERLIHY, N. SHAVIT. *The topological structure of asynchronous computability*, in "Journal of the ACM", vol. 46, n^o 6, 1999, p. 858–923.

[59] K. HOROWITZ, D. MALKHI. *Estimating Network Size from Local Information*, in "The Information Processing Letters journal", vol. 88, n^o 5, 2003, p. 237–243.

[60] A. KERMARREC, L. MASSOULIÉ, A. GANESH. *Probabilistic reliable dissemination*, in "IEEE Transactions on Parallel and Distributed Systems", vol. 14, n^o 3, 2003.

[61] L. LAMPORT. *Time, Clocks and the Ordering of Events in a Distributed System*, in "Communications of the ACM", vol. 21, n^o 7, July 1978, p. 558–565.

[62] J. LAPRIE. *Dependability: Basic Concepts and Terminology in English, French, German, Italian, and Japanese.*, in "Dependable Computing and Fault-Tolerant Systems", vol. 5, 1992.

[63] N. LYNCH. *Some Perspective on PODC*, in "Distributed Computing", vol. 16, 2003, p. 71–74.

[64] A. MONTRESOR, M. JELASITY, O. BABAOGLU. *Robust Aggregation Protocols for Large-Scale Overlay Networks*, in "In Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN'04)", IEEE Computer Society, 2004, p. 19–28.

[65] M. NAOR, A. WOOL. *The load, capacity, and availability of quorum systems*, in "SIAM Journal on Computing", vol. 27, n^o 2, 1998, p. 423–447.

[66] D. POWELL. *Group Communication*, in "Communications of the ACM", vol. 39, n^o 4, 1996, p. 50–53.

[67] B. PRZYDATEK, D. SONG, A. PERRING. *SIA: Secure Information Aggregation in Sensor Networks*, in "Proc. of the 2003 Sensys", ACM, 2003.

[68] R. V. RENESSE, K. P. BIRMAN, W. VOGELS. *Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining*, in "ACM Transactions on Computer Systems", vol. 21, n^o 3, 2003.

[69] SIENA WEB SITE. *Siena: A Wide-Area Event Notification Service*, http://www.cs.colorado.edu/users/carzanig/siena.

[70] M. SAKS, F. ZAHAROGLOU. *Wait-free k-set agreement is impossible: the topology of public knowledge*, in "STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, New York, NY, USA", ACM Press, 1993, p. 101–110.

[71] E. TOTEL, F. MAJORCZYK, L. MÉ. *COTS diversity based intrusion detection and application to web servers*, in "Proceedings of 8th International Symposium on Recent Advances in Intrusion Detection (RAID '2005)", 2005, p. 43–62.

[72] R. VAN RENESSE. *The importance of aggregation*, in "Lecture Notes in Computer Science", A. SCHIPER, A. SHVARTSMAN, H. WEATHERSPOON, B. ZHAO (editors). , n^{o} 2584, 2003.

[73] R. VITENBERG, R. FRIEDMAN. *On the Locality of Consistency Conditions.*, in "International Symposium on Distributed Computing (DISC)", 2003, p. 92-105.

[74] W. ZHANG, G. CAO. *Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks*, 2004.