INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team ADEPT

Algorithms for Dynamic Dependable Systems

Rennes - Bretagne Atlantique

THEME COM

Activity Report

2007

Table of contents

1. Team

ADEPT *is a joint project-team with CNRS and the University of Rennes I.*

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

Administrative assistants
Lydie Mabil [TR INRIA]
Laétitia Rihet [ACET INRIA, since July 2007]

Research scientist Cnrs
Emmanuelle Anceaume [Research Associate (CR)]

Research scientist University of Rennes I
Frédéric Tronel [Associate Professor]

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

PhD students
Florent Claerhout [Fellowship MENRT until October 2007]
Aina Ravoaja [Fellowship INRIA and Brittany Region, ATER ENS]
Heverson Ribeiro [Fellowship INRIA since October 2007]

Student intern
Jean-Baptiste Billaux [Master Student from February 2007 until July 2007]

Technical staff
Romaric Ludinard [Engineer until July 2007]

Visiting scientists
Francisco Brasileiro [Associate Professor, Federal University of Campina Grande - Brazil, November 2007]
Fabíola Greve [Associate Professor, Federal University of Bahia - Brazil, December 2007]
Carlos Maziero [Associate Professor, Pontifical Catholic University of Parana State - Brazil, December 2007]
Josef Widder [Associate Professor, Vienna University of Technology - Austria, March/April 2007]

2. Overall Objectives

2.1. Introduction

The field of information technologies in general, and distributed computing in particular, is continuously evolving and maturing at a very high pace. The characteristics of networks and connected entities have progressed so much that these improvements have induced radical changes in the very nature of applications that can be built upon these distributed systems. Many applications are now executed on large scale systems using a huge number of computer resources spread all over the world.

Whatever the size of the distributed system, economic activities and sometimes human lives are heavily dependent on the distributed applications executed in such systems. When computing resources and stored data can be affected by occurrences of failures, dependability and consistency issues are of prime importance. While computer-based systems are becoming more and more open and complex (number of interacting nodes, dynamicity, heterogeneity of the hardware and software components, mixing of various standards, use of unreliable components, presence of malicious individuals, ...), the main attributes of dependability (namely reliability, availability, integrity, and confidentiality) are also more and more difficult to guarantee. Although occurrences of accidental and intentional failures tend to be more frequent and severe, most applications require to inhibit (or at least to limit) the possible consequences of these failures.

The design of dependable mechanisms mainly depends on two characteristics of the system, namely its level of dynamicity (and size) and the types of faults that might occur during the computation. Within this document, we will use these two criteria to structure our presentation. Regarding the faults, benign faults (crash, omission, ...) are distinguished from the arbitrary faults (Byzantine faults). In the former case, processes behave according to their specification but after some time they may omit some (or all) computation steps. In the latter case, processes involved in the computation may arbitrarily deviate from their specification. Such faults can be the consequence of malicious intents of individuals.

In distributed systems that have a small up to medium size, communication services can be used to efficiently address dependability issues. In this particular context, solutions to agreement problems (such as the consensus problem) can be 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 faults within the group.

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 power supplier, 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.

In large scale distributed systems, a node only observes a part of the system called its neighborhood. For multiple reasons, the set of neighbors associated to a node has to be changed frequently. 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 investigate.
Applications executed in dynamic systems do 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 towards 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 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 (small distributed systems), 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 sometimes difficult to design. On the contrary, in large scale distributed systems, 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) require new solutions that address theses challenges in the new settings.

2.2. Highlights of the year

With the start of the P2Pim@ges project in 2007, the algorithmic solutions we propose to limit the impact of malicious peers in a peer-to-peer system will be validated in a second practical context (the first one is defined in the RIAM sollipsis project started a few months before).

In 2007, the cooperation project CAPES-COFECUB that involves two brazilian universities and four french research teams has been prolonged for two additional years.

3. Scientific Foundations

3.1. Introduction

Our scientific contributions aim at reaching a deeper understanding of some fundamental problems that arise in unreliable 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 propose 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 experimental evaluations by designing and implementing 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 feed-back 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.
- **Small scale distributed systems.** In such systems we aim at considering both accidental and intentional faults and at designing algorithms and methods to detect or to mask such faults which are sometimes transient (another dynamic aspect). An important part of our activity is dedicated to the study of agreement problems and to their use in group communication services.
- **Large scale distributed systems.** We consider different types of large scale systems and study the main dependability issues that are associated. We consider the problem of task allocation and group management in a Grid. By adopting a hierarchical approach, we show that group communication services design for smaller systems can be used. In peer-to-peer systems, we address the reputation problem: the proposed mechanisms aim at evaluating the trust level of each entity in the system. In sensor networks where faults are frequent, the problem of trajectory tracking leads us to study the problem of reconfiguring overlays to ensure a reliable dissemination of information. Our last challenge is to identify and formally define the consistency criteria required by different applications executed on top of dynamic scalable systems.

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 [44] “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, possibly with different speeds at different nodes. Such nondeterminism makes it impossible to understand exactly what a distributed algorithm will do when it executes. Instead, one generally has to settle for understanding properties of executions, 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 other fields of science) has seen the rise of two main 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 criterion to evaluate this adequation, even though other 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 [43], 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 [35], [28], [27].

A lot of proving techniques have been popularized based on this model like indistinguishable states [35] and state valency [35], [28], [27]. However, by its very nature, this kind of model seems to be not well equipped to determine the frontier between possible and impossible problems with respect to fault-tolerance, when more than one process may 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/automata 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 [24], 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 [47], [40]. These results have however required a paradigm shift which is described in the following section.

3.2.3. Geometric Models for Concurrency

In [41], the authors have been able to extend the result of [24] 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 means of a shared memory. The starting point of all these new results can certainly be dated by the study of k-set agreement by Chaudhuri [29]. 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 Chaudhuri 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 represented 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 modeled 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 algebraic topology are commonly used. This field of mathematics offers powerful techniques that are able to tackle the combinatorial explosion that occurs when using graph theoretical approach. This has been brilliantly demonstrated by [41] 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.

Note that models such as the "thermodynamic models" can also be adopted 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 also 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 include thousands, and may be even millions of nodes, we are convinced that the study of all the explicit interactions between neighbor nodes is 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.

3.3. Small Distributed Systems

Keywords: *agreement problem*, *arbitrary fault*, *atomic broadcast*, *availability*, *benign fault*, *consensus problem*, *dependability*, *distributed computing*, *erroneous suspicions*, *failure model*, *fault-tolerance*, *group communication*, *intrusion detection*, *reconfiguration*, *reliability*, *reliable broadcast*, *security*, *unreliable failure detector*.

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

We target small distributed systems that contain less than a few tens of processors. 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.

In what follows, the description of this general activity is subdivided into three sections devoted respectively to (1) the study of group communication in the presence of benign faults (2) arbitrary behaviors caused by some transient accidental faults, and (3) the detection of arbitrary behaviors due to external attacks. For all the problems described within this section, our research activities takes 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. Group Communication Services

Providing group communication services within a system is essential. Broadcast primitives allow an entity to send information toward all the entities of the system. The properties guaranteed by these services essentially depend on the size of the system. In the case of small systems, broadcast services can guarantee strong properties regarding the delivery of the messages to the recipients and the order in which these messages are delivered [38]. 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 ensured [46]. For instance, to increase the overall reliability of the system, critical data and functionalities have to be replicated on a group of nodes. Ensuring consistency within such a set of 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. 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).

We focus on a particular fundamental problem namely the consensus problem and design solutions to all the other agreement problems that are based on this basic building block.

3.3.2. Transient Arbitrary 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.

We 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 [17].

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 [25], [26]. 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 [19] are based on similar assumptions.

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 used and adapted to achieve this objective.

3.4. Large Scale Systems

Keywords: *availability*, *consistency*, *distributed computing*, *dynamic system*, *free rider*, *grid computing*, *publish/subscribe*, *replication*, *reputation*, *self-stabilization*, *sensor networks*, *shared data*, *task allocation*, *tracking service*.

3.4.1. P2P Systems: Free-Riders and Reputation

Among the important issues raised by the emerging peer-to-peer systems, Free-Riding is the one introduced by non-cooperative nodes. Indeed, it is more or less straightforward that rationality exists in peer-to-peer 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 peer-to-peer 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.

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 peer-to-peer 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 peer-to-peer 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.

3.4.2. Grid Computing: Reconfiguration

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 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 in the hierarchy of the interacting entities. 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.

3.4.3. Sensor Networks: 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).

In this context, we are interested by a particular problem called tracking [32], [33], [39], [51]. It 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. The problem of trajectory tracking consists of 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.).

In [30], we provided a precise definition of the tracking problem. Similarities with the publish/subscribe problems have been identified: when a sensor detects the target it publishes the position data and interested nodes have to be notified. The publish/subscribe paradigm has emerged recently as an effective technique for building distributed applications in which information has to be disseminated from *publishers* (event producers) to *subscribers* (event consumers). The decoupled nature of publish/subscribe interactions makes these architectures particularly suitable for building large-scale applications where scalability plays a key role. Defining the tracking problem as an extension of the publish/subscribe problem allows 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 order in which the notifications are seen has to be consistent with the trajectory of the targets, interesting issues related to clock synchronization and causal dependencies have also to be investigated.

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 overlays (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. We are currently working on the evaluation of these algorithmic solutions.

3.4.4. Consistency Issues in Large Scale Systems

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 [23] 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 [50]. 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 [45] 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 [34]: (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.

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 inexorably weaker) becomes questionable. In particular, regarding **fault tolerance**, a large number of entities (software 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 peer-to-peer 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. The basic elements of Prometeus are Eva, a component-based framework and, on top of Eva, Adam that is a library of agreement components for use by applications.

PARADIS is a middleware for a Grid dedicated to genomic applications. Paradis makes use of the ADAM library to reliably allocate resources to tasks to be executed on the Grid.

In the context of the DADDI project, we have also developed a software system that enhance the integrity and availability of an Intrusion Detection Architecture targeted to a Web Server that delivers dynamic contents.

5.1.1. Eva

EVA is an implementation of a component model that aims at supporting the development of distributed abstractions and high-level communication protocols. EVA implements a publish/subscribe communication environment to structure components composing high level protocols. In the EVA model, protocols are regarded as a number of cooperating components that communicate via an event channel. Communication is achieved via the production of events (output data) by supplier components, and the consumption of these events (input data) by consumer components. A supplier component uses the service of an event channel to route the events it produces to any consumer component that has registered with the event channel its interest in consuming that particular type of event. The event channel decouples suppliers from consumers yielding an interesting flexibility. Synchronous interactions between components is also supported in EVA. Special attention has been devoted to optimize the implementation. For example, potential sources of overheads (in the management or transmission of events) have been limited or eliminated in the design and implementation of EVA.

5.1.2. Adam

ADAM is a library of agreement components, based on the component model implemented by eva. The central element of the ADAM library is gac (generic agreement component). It implements a generic and adaptive fault-tolerant consensus algorithm that can be customized to cope with the characteristics of the environment. Moreover, thanks to a set of versatile methods, its behavior can be tuned to fit the exact needs of a specific agreement problem. A range of fundamental ADAM components are implemented as specializations of this gac component. The ADAM library currently includes the most important components for reliable distributed programming: Group Membership, Atomic Broadcast, Leader Election, etc.

5.1.3. 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.

5.1.4. 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.

Our system has been deployed on a intrusion detection platform that is based on a set of diversified Web servers running on top of three different operating systems (Windows, Linux, Mac OS X).

6. New Results

6.1. Small Distributed Systems

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

Participants: Emmanuelle Anceaume, Jean-Baptiste Billaux, Michel Hurfin, Jean-Pierre Le Narzul, Frédéric Tronel.

6.1.1. *Using Algebraic Topology to Obtain Automatically Impossibility Results in Wait-Free Systems*

In the context of asynchronous distributed systems where processes are prone to crash, lower bounds and impossibility results help to understand the inherent limitations of such systems. For example, the proof of impossibility for consensus in the presence of a single failure is a major result in the field of fault tolerance. The complexity of the first proof provided by Ficher, Lynch, and Paterson in 1985 also highlights the need for adequate formal models. Similarly, the new complexities faced by the research community in large scale dynamic distributed computing also motivate the definition of new appropriate 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 abstract representations if one cannot manage inherent complexity induced by a too great level of details.

Following this trend, our current studies are intended to analyze the mathematical modeling of wait-free systems. In this computation model, processes synchronize and interact by the mean of a totally asynchronous distributed memory, and all of them but one can fail (we only consider fail-stop failures). This computational model is called wait-free, because no process can ever wait for a message coming from any other process, because this process may have crashed. Waiting for a process to make some progress in a computation is clearly a potential deadlock. There has been an intensive research work in this domain, and many deep results have been obtained. For example, thanks to Herlihy and Shavit, we know a precise characterization of the distributed tasks that can be solved in the wait-free computation model. Borowsky and Gafni have also introduced a new communication primitive called *immediate snapshot* whose semantics is specifically designed so that computability in this new model can be easily established (by simple geometrical considerations). The necessary and sufficient set of conditions established by Borowsky and Gafni for this model of computation is exactly the same as the one established by Herlihy and Shavit. Then, Borowsky and Gafni have exhibited two distributed algorithms that respectively implement the immediate snapshot communication primitive in the more crude shared memory model, and conversely. This precisely shows that both are equivalent with respect to computation power.

A restriction of the Herlihy-Shavit result relies in the fact it mixes tools borrowed from the algebraic topology domain with more classical tools such as relations. In particular, the specification of a distributed problem is given as a relation between inputs and outputs. Unfortunately these two worlds do not cooperate smoothly. More recently, Havlicek has reworked the theorem of Herlihy and Shavit to express the whole result in purely algebraic terms, by proposing a way to express the specification of a distributed problem in terms geometrical objects. Using this new way to state the Herlihy-Shavit result, Havlicek has been able to propose a set of necessary conditions for a distributed problem to have a solution in a wait-free systems.

Our research activity focuses on two points. On the one hand, we try to determine computable characterizations of tasks that make them unsolvable in the wait-free model. On the other hand, for tasks that are solvable, we try to find an algorithm that, given a formal specification of such a task, is able to synthesize a wait-free protocol that solves this task. We have extended the result of Havlicek by showing that all the algebraic objects involved in the Havlicek theorem can be automatically computed (chromatic complexes, homology group, etc) [15]. We have been able to show that it is possible for a computer program to derive impossibility results when the necessary conditions exhibited by Havlicek are not satisfied by a given problem. Using this approach we have been able to reestablished well-known impossibility results of the domain.

6.1.2. *Group Communication Services*

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 [31] (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 [10], 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.

Consensus services have been recognized as fundamental building blocks for fault-tolerant distributed systems. Many different protocols to implement such a service have been proposed, however, not a lot of effort has been placed in evaluating their performance. In particular, the performance of round-based consensus protocols for asynchronous systems augmented with failure detectors can be influenced by at least two factors: (i) the impact of the quality of service (QoS) of the failure detector used; and, (ii) the protocol's decision pattern, i.e. how many processes are allowed to autonomously decide in a particular round. There has been some work on evaluating how the QoS of the failure detector impacts the performance of consensus protocols, as well as on the trade-off between having faster decentralized decision - more than one process being able to autonomously decide - at the expenses of generating more network load. These studies, however, focus on protocols that have no mechanism to deal with an eventual bad QoS provided by the failure detector, and have a decision pattern that is either completely centralized - only one process being able to autonomously decide - or completely decentralized - all processes being able to autonomously decide. In [14], we conduct a thorough evaluation of the performance 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. The results, obtained by means of simulation, highlight the advantages and limitations of the two mechanisms and allow one to understand in a comprehensive framework how the protocol's parameters should be set, such that the best performance is achieved depending on the application's requirements.

6.1.3. Transient Arbitrary Failures

Tightly synchronized and accurate clocks among the members of a distributed system is a fundamental service as it allows to perform synchronized actions or to estimate the behavior of the environment in a control system. One way to ensure reliable and tight synchronization among local clocks is the use of a clock synchronization algorithm. Essentially such an algorithm overcomes clock drift, variations of transmission delays and failures. It guarantees that the maximum deviation between (correct) local clocks is bounded (precision) and that these clocks are within a linear envelope of real-time (accuracy).

There is considerable literature devoted to the design and implementation of clock synchronization algorithms; Some algorithms are specified for environments in which processes may crash, may suffer timing failures, or may execute arbitrarily bad operations. The last type of behavior, called Byzantine, is the most severe type of process failures. It captures all causes of failures, ranging from accidental memory bit flips to malicious attacks on a system. Therefore this model seems appropriate for a large range of distributed applications.

Another kind of fault tolerance is self-stabilization. Here it is assumed that the system behaves arbitrarily (including, e.g., that the assumed threshold of faults is temporarily violated) but if eventually all processes behave according to the algorithm the system stabilizes to a good state (where in our case the clock synchronization properties hold). A highly interesting work consist in joining the two approaches: Self-stabilizing clock synchronization algorithms that work in the presence of permanent Byzantine failures. However, these solutions share some properties which seem inherent to the problem of fault-tolerant self-stabilization: First, even processes that always follow their algorithm are not guaranteed to remain synchronized to each other and second, resynchronization of recovered processes takes O(f) time. We propose a model based on the idea, that permanent failures are too optimistic for certain applications, while fault-tolerant self-stabilization might be too pessimistic, or the provided properties too weak [18]. We therefore explore under which conditions clock properties can be provided permanently in the presence of transient and dynamic Byzantine faults, where processes recover from "bad periods" with an arbitrary state and just start following their algorithm. We, however, limit the number of components which may suffer from faults simultaneously. In this model we propose a clock synchronization algorithm tolerant to moving Byzantine failures [11]. In particular our algorithm guarantees that in presence of up to f "moving" and concurrent Byzantine failures, correctly behaving processes (that is at least $n \geq f$ processes, with $n \geq 3f+1$, and n the number of processors in the system) have synchronized logical clocks. Our algorithm is a variation of Srikanth and Toueg's clock synchronization algorithm [48], in which the classic notion of "correct process" is assumed. The challenge of the present work is the guarantee that correctly behaving processes are never corrupted by recovering processes, and that clocks of recovering processes get quickly tightly synchronized with those of correctly behaving processes. We provide an expression for the recovery time (i.e., the period of time after which a recovered process is synchronized with the other processes). This bound is independent of f, and is roughly equal to the time needed to execute two resynchronizations. We derive an expression for the failure turn-over rate, i.e., the maximal allowable frequency at which processes may enter (and leave) faulty periods. This rate is also independent of f, and is roughly equal to the time needed to execute three resynchronizations.

6.1.4. Intentional Faults and Security Aspects: New Attacks

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 returned 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 [42] 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 [49], 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 [16].

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. Large Scale Systems

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

Participants: Emmanuelle Anceaume, Florent Claerhout, Michel Hurfin, Jean-Pierre Le Narzul, Aina Ravoaja, Heverson Ribeiro, Frédéric Tronel.

6.2.1. P2P Systems: Free-Riders and Reputation

In a prior work [22], we proposed a solution to the robust reputation problem. Essentially this problem aims at evaluating peers reputation despite free-riding and dishonest behaviors. In that work, robustness to attack is accomplished through an aggregation technique in which collected first-hand feedback is weighted by a credibility factor locally computed by the requesting peer, while incentive for participation is implemented through a fair differential service mechanism relying on peer's level of participation and peer's credibility. We showed within modest churn the presence of a high fraction of malicious peers does not prevent a correct peer from accurately evaluating the reputation value of a target peer.

However, this is achieved at the expense of a non negligible number of messages mainly due to the way feedback are collected. Indeed, to face malicious peers, feedback is gathered from peers randomly selected within the system which clearly makes the efficiency of this crawling technique highly reliant on the way the graph of witnesses is constructed. Finding the right set of witnesses in an efficient way is a challenging issue since the reputation value depends on the feedback provided by these peers.

In [13], we present a cost effective solution to this problem by relying on two schemes: first, to increase the likelihood of collecting a large amount of feedback from the right set of witnesses, secure gathering techniques are identified. Briefly, these techniques aim at guaranteeing that feedback is collected only from a cluster of peers sharing a similar interest for the target service provider, and that bribes and collusion do not interfere by constraining the gathering scheme. STORM is designed to gather information on the reputation of a peer in $O(log_2N)$ steps, with N the number of peers. Complementary to this scheme, secure routing tables maintenance and secure routing techniques are provided. These techniques ensure first that routing tables cannot be attacked by malicious peers by keeping the distribution of malicious peers uniform. This is achieved by a *randomized decision* algorithm, which prevents malicious peers from strategizing to get inserted in routing tables. We show that this algorithm eventually minimizes the expected number of malicious peers in routing tables. Second, to prevent malicious peers from dropping or delivering a feedback request to their collusion group instead of a legitimate peer, STORM uses *constrained redundant routing*. This scheme guarantees that each request is successful with high probability if the fraction of colluders in a cluster does not exceed $1 - \left(\frac{1}{2(1-c')^{(\log_2 N)/2}}\right)^{\frac{2}{\log_2 N'}}$, with N the size of the system, N' the size of the cluster, and c', the expected number of malicious peers in routing tables. To summarize, the contribution of this work is a scalable protocol for efficient and secure information gathering in a system with high churn and presence of colluding peers.

6.2.2. 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 [9], 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 [46]. 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 determine the right adaptation to react to the observed changes.

6.2.3. Consistency Issues in Large Scale Systems

We have proposed an architecture for *self-adjusting and self-healing atomic memory* in highly dynamic systems exploiting peer-to-peer techniques. Our approach, named *SAM* [20], [21], brings together new and old research areas such as peer-to-peer 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 [12].

7. Other Grants and Activities

7.1. National Project

7.1.1. ACI Daddi (2004-2008)

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.

7.1.2. 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), Tristan Févat (University of Provence), Rémi Morin (University of Provence), Luigi Santocanale (University of Provence), Eric Goubault and Emmanuel Haucourt (CEA), Matthieu Roy (LAAS - CNRS) and 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.

7.1.3. University Research Project (BQR Grant): Perecap (2007-2009)

Participants: Florent Claerhout, Michel Hurfin.

This project entitled "An experimental sensor network platform" aims to bring together researchers from different laboratories located in the city of Rennes: IRISA, IETR and LTSI. The project is coordinated by the INRIA project team Pilgrim (François Charot). The platform that will be deployed progressively will be used by the different participants to conduct experimentations. In this context, the ADEPT's objective is to validate and evaluate some algorithmic solutions proposed to solve the trajectory tracking problem in wireless sensor networks. In particular, we aim to compare results obtained by experiments and by simulations. This work may enable us to reach a deeper understanding of the dynamic behavior of these self-organizing fault-tolerant protocols whose performances are currently unpredictable.

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

Participants: Emmanuelle Anceaume, Aina Ravoaja.

Solipsis is a R&D project (ANR-RIAM) leaded by Orange lab, and supported by ANR and Media and Networks cluster of Brittany. It was launched in january 2007 and is based on prior works stated in 1998. Five partners are involved: IRISA, Archivideo, Artefacto, University of Rennes 2 , and Orange Labs.

Solipsis is an open source system for a massively multi-participant shared 3D virtual world. Briefly, the challenge of this project is to offer the opportunity to the users to imply themselves in the collective creation of a public virtual space. Nowadays, virtual worlds are the property of companies or association which manage centralized servers. Even the most open systems are not freed from a central and responsible authority. By considering an open system based on a peer-to-peer architecture, Solipsis hopes for a rich ecosystem to develop. This architectural choice guarantees that the virtual world can be populated by an unlimited number of users, without cost of deployment.

ADEPT's role in this project is to make the proposed peer-to-peer overlay robust against undesirable behaviors. By not relying on dedicated servers that control consistency and legitimacy of peers behaviors and interactions, we have to face several challenges such as Eclipse attacks that aim at populating honest peers neighborhood by malicious peers, Sybil attacks whose goal is to out vote honest peers in collaborative tasks by the creation of an unbounded number of fake identities. Facing such attacks is worsened by the fact that peers interactions involves huge amount of data (3-D objects) thus making each single maintenance operation critical.

7.1.5. DGE Project: P2Pim@ges (2007-2009)

Participants: Emmanuelle Anceaume, Jean-Pierre Le Narzul, Romaric Ludinard.

The P2Pim@ge project is supported by the Direction Generale des Entreprises. This project aims at studying, prototyping and testing legal advanced streaming technology on peer-to-peer systems. Different applications will be addressed such as video on demand, immediate or differed download, access to scare content, etc. Partners of the project are Thomson R&D, Thomson Broadcast & Multimedia, Mitsubishi Electric ITE/TCL, Devoteam, France Telecom, ENST Bretagne, Marsouin, IRISA, IPdiva, and TMG.

Adept's role in this project is to address failure prone behavior. In such large-scale dynamic systems, users may have a strategic behavior that is neither obedient nor malicious, but just rational. Tracking such behavior is complex since it requires taking into account a large set of features: large population, asymmetry of interest, collusion, "zero-cost identity", high turnover, and rationality. Techniques from the security domain (e.g. intrusion detection), and new fault tolerant distributed algorithms inspired from social theories will be investigated to deal with these undesirable behaviors.

7.2. International Cooperations

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

Participants: Emmanuelle Anceaume, Michel Hurfin, Jean-Pierre Le Narzul, Frederic Tronel.

A cooperation project with the Federal University of Bahia, the Federal University of Paraiba, 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 four years (2005-2008). Michel Hurfin is the French coordinator of this project which focuses on distributed computing and Grid computing. In 2007, Fabiola Greve (Federal University of Bahia) and Francisco Brasileiro (Federal University of Campina Grande) have visited some of the French Laboratories. Emmanuelle Anceaume has visited the Federal University of Campina Grande in July 2007 and has started a joint work on reputation mechanisms. Frédéric Tronel has visited the Federal University of Bahia in October 2007: he worked during his stay on the Consensus problem in asynchronous distributed systems. In 2007, this cooperation has lead to two joint publications both published at DSN 2007 [14] and [37], [36].

8. Dissemination

8.1. 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 gave lectures on fault tolerance and distributed computing to students of two engineering schools: ENST Bretagne (Rennes, 6 hours) and Supelec (Rennes, 8 hours). He also gave lectures on dependability issues and sensor networks at Southeast university (Nanjing, China, 6 hours).

8.2. 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.
- Michel Hurfin has been invited to present his work at Southeast university (Nanjing, China, June 2007) and at the Pontifical Catholic University of Parana State (Curitiba, Brazil, November 2007).

8.3. Integration within the Scientific Community

- Emmanuelle Anceaume :
  - 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 International Workshop on Service, Security and its Data management for Ubiquitous Computing (SSDU 2007), May 2007, Nanjing, China.
  - acts as a program committee member of the 3rd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2007), July 2007, Istanbul, Turkey.
  - acts as a program committee member of the 4th International Workshop on Dependable Embedded Systems (WDES 2007), October 2007, Beijing, China.
  - acts as a program committee member of the 2nd International Symposium on Information Security (IS'07), November 2007, Vilamoura, Algarve, Portugal.
  - acts as a program co-chair of the Third International Symposium in Ubiquitous Computing (Secubiq-2007), December 2007, Taipei, Taiwan.
- Michel Hurfin :
  - acts as a program committee member of the IEEE International Symposium on Ubisafe Computing (UbiSafe-07), May 2007, Niagara Falls, Ontario, Canada.
  - as a program committee member of the Latin America Grid workshop (LAGrid 2007) , May 2007, Rio de Janeiro, Brazil.
  - acts as a program committee member of the 5th International Conference on Autonomic and Trusted Computing (ATC-08), June 2008, Oslo, Norway.
  - acts as a lecture committee member of the 9th African Conference on Research in Computer Science and Applied Mathematics (CARI'2008), October 2008, Rabat, Maroc.
- Frédéric Tronel :
  - acts as a program committee member of the workshop on Foundations of Fault-tolerant Distributed Computing (FOFDC 2007), April 2007, Vienna, Austria.

9. 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] 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.

[5] 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.

[6] 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.

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

[8] Y. Wang, E. Anceaume, F. Brasileiro, F. Greve, M. Hurfin. *Solving the Group Priority Inversion Problem in a Timed Asynchronous System*, in "IEEE Transactions on Computers. Special Issue on Asynchronous Real-Time Disttributed Systems", vol. 51, n^o 8, August 2002, p. 900–915.

Year Publications

Doctoral dissertations and Habilitation theses

[9] J. Pley. *Protocoles d'accord pour la gestion d'une grille de calcul dynamique*, Thèse de doctorat, Université de Rennes I (école doctorale Matisse), Dec 2007.

Articles in refereed journals and book chapters

[10] E. Anceaume, R. Friedman, M. Gradinariu. *Managed Agreement: Generalizing two Fundamental Distributed Agreement Problems*, in "Information Processing Letters", vol. 101, n^o 5, 2007, p. 190-198.

Publications in Conferences and Workshops

[11] E. ANCEAUME, C. DELPORTE-GALLET, H. FAUCONNIER, M. HURFIN, J. WIDDER. *Clock Synchronization in the Byzantine Recovery Failure Model*, in "Proceedings of the 11th International Conference On Principles Of Distributed Systems (OPODIS 2007)", Dec 2007.

[12] V. GRAMOLI, E. ANCEAUME, A. VIRGILLITO. *SQUARE: Scalable Quorum-Based Atomic Memory with Local Reconfiguration*, in "Proceedings of the 22nd ACM Symposium on Applied Computing (SAC 2007), Seoul, Korea", ACM Press, Mar 2007, p. 574-579.

[13] A. RAVOAJA, E. ANCEAUME. *Storm: A Secure Overlay for P2P Reputation Management*, in "First IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007), Boston, USA", Jul 2007.

[14] L. SAMPAIO, M. HURFIN, F. BRASILEIRO, F. GREVE. *Evaluating the Impact of Simultaneous Round Participation and Decentralized Decision on the Performance of Consensus*, in "37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2007), Edinburgh, UK", Jun 2007, p. 625–634.

Miscellaneous

[15] J.-B. BILLAUX. *Décision automatique de résultat d'impossibilité pour une classe de problèmes distribués asynchrones sans attente*, June 2007.

[16] M. HURFIN, R. LUDINARD, F. TRONEL, J.-P. LE NARZUL, L. MÉ, E. TOTEL, F. MAJORCZYCK. *Daddi project - Deliverable 4.1: Dependability*, February 2007, http://www.rennes.supelec.fr/DADDi/RAPPORTS/L41.pdf.

References in notes

[17] 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.

[18] 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", vol. 0, n^o 0, IEEE, jul 2004, p. 262–269.

[19] 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, jul 2004, p. 262–269.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] F. GREVE, S. TIXEUIL. *Connaissance vs. Synchronie pour l'Accord Tolérant aux Pannes dans les Réseaux Inconnus*, in "Proceedings of Algotel 2007, Oléron", May 2007.

[37] F. GREVE, S. TIXEUIL. *Knowledge Connectivity vs. Synchrony Requirements for Fault-Tolerant Agreement in Unknown Networks*, in "Proceedings of IEEE International Conference on Dependable Systems and networks (DSN 2007)", June 2007, p. 82-91.

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

[39] 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.

[40] 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.

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

[42] M. HURFIN, J.-P. LE NARZUL, F. MAJORCZYCK, 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.

[43] 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.

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

[45] 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.

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

[47] 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.

[48] T. K. SRIKANTH, S. TOUEG. *Optimal Clock Synchronization*, in "Journal of the ACM", vol. 34, n^{o} 3, 1987, p. 626-645.

[49] 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.

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

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