INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team ADEPT

Algorithms for Dynamic Dependable Systems

Rennes

THEME COM

Table of contents

1. Team

Head of project-team
Michel Hurfin [CR INRIA]

Administrative assistants
Lydie Mabil [TR INRIA, January-May and October-December 2005]
Florence Santoro [until September 2005]
Céline Ammoniaux [TR CNRS, since September 2005]

Staff member Cnrs
Emmanuelle Anceaume [CR]

Staff members University of Rennes I
Maria Gradinariu [Associate Professor, Research Scientist INRIA]
Philippe Raïpin Parvédy [Lecturer]
Frédéric Tronel [Associate Professor, since February 2005]

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

Ph.D. students
Florent Claerhout [Fellowship MENRT, since October 2005]
Vincent Gramoli [Fellowship MENRT]
Julien Pley [Fellowship MENRT]
Aina Ravoaja [Fellowship INRIA and Brittany Region]

Postdoctoral fellow (Inria)
Antonino Virgillito [until September 2005]

Visiting scientists
Fabíola Greve [Associate Professor, Federal University of Bahia - Brazil, August 2005]
Livia Sampaio [PhD Student, Federal University of Campina Grande - Brazil, April-September 2005]

2. Overall Objectives

2.1. Overall Objectives

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

In a recent past, the focus of the research activities conducted in the ADEPT team has been on the study of dependability (mainly fault-tolerance) and data consistency within small sets of servers called groups. In this particular context, solutions to agreement problems (such as the consensus problem) have been proposed and used as basic building blocks for designing solutions to higher level protocols that are in charge of maintaining global properties at the group level despite the occurrence of crashes within the group. Such solutions have proved to be particularly suited to the case of small to medium systems. While maintaining a research activity in this particular field is important since several problems remain open in small groups of processors, it is obvious that the solutions proposed in the past are not directly applicable to large scale dynamic systems. Fundamental and applied research on models, algorithms and tools enabling to build distributed services and applications has now to face a new challenge, namely the high dynamicity that characterizes many recent distributed systems (in particular world-wide community of processors). In these new settings,

various dynamic changes can affect a distributed computation and therefore need to be addressed at run time. For example, the load (in terms of messages or in terms of computing activities), the topology, the energetic resources, the accessibility of stored data, the set of processes involved in the computation or even the behavior of a participant may change at any time. Assuming that such variations do not occur very often, adaptive algorithms can be proposed to detect a modification of the whole execution context and react globally to this modification (reconfiguration, execution of another code, ...). However, when the system has a very high level of dynamicity, implementing a global observation system that allows to reconfigure the whole system in a single step is no more realistic. Only local observations and progressive adaptations to changes can be performed on cohesive subsets of nodes. Such a radical gap on the scale and dynamicity of systems militates in favor of a paradigm shift for designing solutions to the problems raised by these new systems. The challenge addressed now by the ADEPT team consists in maintaining global properties (in particular dependability and data consistency properties) over the set of entities that belong to a dynamic system.

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

Apprehending dynamic systems goes through an analysis of the principles that govern them. The first principle concerns the exchange of information or resources with the environment (for example, in grid or P2P systems, components are possibly capable of infinitely often retrieving new information/resources from components around them). The second one is the dynamics of these systems (for example, in ad-hoc networks, components have the ability to move around, to leave or to join these systems freely). Finally, the third principle is related to the specificity of the components: among all components of the system, some have huge computation resources, some have large memory space, some are highly dynamic, some have broad centers of interest. Each node may have its own data and its own strategy when it interacts with others: applications are context-sensitive and thus the behavior of a node partially depends on its own (complete or partial) view of the system. Looking at these systems as a mass of components can be a mistake because it abstracts away the differences that might exist between individual components, differences which make the richness of these systems.

All these tenets have as common seed the locality principle, that is some range of effect, both in terms of interaction and knowledge. A node only observes a part of the system called its neighborhood. By definition, the neighborhood of a node n is a subset of the nodes of the system from which n can receive information. To define this set of nodes, a node takes into account both physical and semantical constraints. For example, in P2P systems, a node n' will not be a neighbor of a node n if there is no communication path between them or if they share no common interest. This set evolves dynamically during the computation. Neighbors are removed, added or replaced. For example, if a new node n'' can provide more significant informations than a neighbor n', a node n will replace n' by n'' in its own set of neighbors. The replacement strategy allows to avoid an uncontrolled increase of the size of the set of neighbors. Of course the significance of the information is application dependent. Similarly, in sensor and ad-hoc networks the neighborhood is defined in terms of communication range. For multiple reasons, the communication range of a node can increase or decrease. In

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

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

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

been able to master the difficulties raised by small to medium systems we must reveal the challenge of scaling up our solutions and adapt them to handle dynamicity.

3. Scientific Foundations

3.1. Introduction

Our scientific contributions aim at reaching a deeper understanding of all the fundamental problems that arise in dynamic distributed systems. During the study of a particular problem, our approach consists in identifying for a particular execution environment (characterized by a set of assumptions on the computation model, the failure model, the dynamicity, and scalability, ...), the set of elementary services needed to build a given application, and for each of them, to exhibit solutions as efficient as possible, optimal if possible, and generic. If no solution exists, we aim at exhibiting impossibility results. To validate and to promote the use of these algorithmic solutions, we conduct in parallel experimental evaluations by developing flexible and adaptive middleware services that integrate our know-how and experience in distributed computing. This prototyping activity leads us to consider technical and operational problems as well as methodological issues. The 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.
- **Accidental and intentional faults.** Economic activities and human lives are now heavily dependent on distributed systems and applications. When computing resources and stored data can be affected by the occurrence of failures, dependability becomes a crucial issue. We aim to consider both accidental and intentional faults and to design algorithms and methods to detect or to mask such faults which are sometimes transient (another dynamic aspect).
- **Data consistency.** One of the challenges is to identify and formally define the consistency criteria required by different applications executed on top of dynamic scalable systems.
- **Dissemination of information.** Communicating in dynamic scalable and dependable systems faces different issues: firstly, large scale requires new communication primitives. One of these primitives, the semantic-based primitive (e.g., data-based), enables to transmit anonymously information within the network. That is, the set of recipients is not known in advance by the sender of the information. Recipients are identified only on semantical bases (the match between their own interest and the information content in the message). Secondly, dynamicity requires reactive-based communication primitives. We study these new communication services.

3.2. Models for Dynamic, Scalable and Dependable Systems

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

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

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

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

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

In the sequel, we will discuss the respective benefits and costs of both models, and their adequation to the study of dynamic dependable systems. This adequation could be discussed through a lot of facets (expressiveness, performance analysis ability, etc). We will restrain us to expressiveness, however.

3.2.1. Automata Models

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

- They were natural extensions of well-established models used in sequential computing, making them appealing and easy to use.
- Their popularity has been naturally reinforced by their successful applications in the proof of a large number of seminal papers in the field [52], [43], [42].

A lot of proving techniques have been popularized based on this model like indistinguishable states [52], state valency [52], [43], [42]. All these qualities could have been sufficient for adopting this model in the distributed computing field. However, by its very nature it seems that this kind of model is not well equipped to determine the frontier between possible and impossible problems with respect to fault-tolerance, when more than one process can crash in a system. Although this does not constitute a proof, one can remark that the proof of impossibility for consensus in the presence of a single failure while being relatively old (1985), has not been followed by any others also fundamental results afterward. This can be a posteriori analyzed as an evidence that graph/automatas models are not able 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 [35], 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 [72], [56]. These results have however required a paradigm shift which is the object of the following section.

3.2.2. Geometric Models for Concurrency

In [57], the authors have been able to extend the result of [35] to the case where an unbounded number of processes may fail. They gave a complete characterization of wait-free decision tasks[1] in asynchronous systems where processes communicate by the mean of a shared memory. The starting point of all these new results can certainly be dated by the study of k-set agreement by [44]. This new problem can be seen as an extension of the classical consensus problem. In the latter one, all correct processes have to reach a common agreement, while in the former one, up to k different values may be decided. In this article Chaudhury conjectures that k-set agreement cannot be solved in the presence of k failures (when the set of possible initial values is larger than k). This new definition that covers the one of the classical consensus ($k = 1$) was the missing piece of the puzzle. Researchers had now a new challenge to reveal that was pointing out in the right direction.

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

To study the complex geometrical objects induced by this model, mathematical tools originating from of algebraic topology are commonly used. This field of mathematics offers powerful techniques that are able to tackle the combinatorial explosion that occurred when using graph theoretical approach. This has been brilliantly demonstrated by [57] who have been able to precisely characterize the set of decision tasks that can be solved in wait-free systems.

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

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

From our point of view, both goals are still widely opened 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.

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

3.3. Accidental and Intentional Faults

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

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

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

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

3.3.1. Accidental Faults Resulting in Transient Failures

Accidental faults include among others physical faults that generally result from aging, shocks and physical phenomena (temperature, hydrometry, radiation, ...) that may affect communication hardware, computing hardware or memory hardware. In the worst case, accidental faults may lead to arbitrary behaviors. For example, in the particular case of a spacecraft, radiations (such as alpha particles and cosmic rays) may generate an undesired bit-flip effect that changes the content of a storage element (memory, register or instruction counter). Consequently, the next executed instruction is not necessarily the right one. In the particular case of accidental faults, our aim is to build reliable systems from unreliable components. To tolerate failures (crash or arbitrary behaviors) physical redundancy is mandatory to ensure that faults are masked and will not perturb the computation. Replication of critical data and functionalities on a group of nodes allows to increase the overall reliability of the system [67]. To be able to coordinate the activities of the processors, a significant body of work on replication techniques and agreement problems has been done. Many services that have to be provided when using the concept of group can be classified as agreement problems (membership, total order broadcast, consensus, leader election, atomic validation, ...). As these services are used very often, efficiency is a key issue when designing solutions to such agreement problems. Our first goal is to have an even better understanding of these problems while considering various levels of adversity (various computational models ranging from the purely synchronous one to the purely asynchronous one, various failure models, ...). In particular, the analysis of the amount of time/activity necessary to converge to an agreement may require to use probabilistic models that will characterize the environment and the arrival law of failures occurrences. Such an analysis can also be used to define a good trade off between the price to pay (redundancy, additional activities,...) and the obtained level of dependability (evaluation of the coverage).

Our second goal is to consider that a process has the capability to recover and so should not be excluded or precluded from participating to the forthcoming computations under the pretext that it has suffered a transient

failure (even in the particular case of transient byzantine failures). Classically, the proposed solutions assume that all the nodes involved in the computation are classified into two categories (the *correct* nodes and the *faulty* ones). This classification into two distinct subsets remains unchanged during all the computation. A node is correct if it behaves according to its specification until the completion of the computation; otherwise it is faulty. Within this approach, the design of fault tolerant applications relies on the fact that a correct node never fails. Consequently, once all the faulty nodes have failed, no new failure can occur. Assuming that only a subset of the nodes can fail during the whole computation fits the requirements of many common distributed applications. But when running times of the applications are extremely long, this assumption becomes unrealistic (for example, in the space domain, the useful orbital life of a satellite is expected to last several years): any node may fail before the whole computation finishes. Yet as most of the faults are transient, a node is often able to recover to a consistent state (after having experienced failures). Thus, it is of interest to consider that any node can alternate correct and faulty periods (no node is correct according to the previous definition of correct/faulty nodes). Of course, the complexity of this new approach depends on the type of failures we consider. In the particular case of the crash failure model, the problem is quite simple: a node is aware that it is currently in the recovery phase. This information can also be known and trusted by the other nodes. Additionally, in a crash failure model, a running process always behaves according to its specification. Consequently, any data saved in its permanent storage or logged by another process can be used to recover to a safe state. Thanks to all these strong properties, it is possible to cope with both permanent and transient crashes [27].

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 [37], [40]. We aim to study these relationships. In a pro-active security system, any node can experience arbitrary failures but during a fixed period of time, no more than t nodes can be faulty. To ensure security requirements, algorithms perform periodic computations of critical data (like for example secret keys). The type of solutions we are proposing [30] are based on similar assumptions.

3.3.2. Accidental Faults and Reconfiguration Problems

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

- Reconfiguration in Grid Computing
  The major aim of a Grid is to federate powerful distributed resources within a single virtual entity which can be accessed transparently and efficiently by external users. Since a Grid is a distributed and unreliable system involving heterogeneous resources located in different geographical domains, fault-tolerant resource allocation services have to be provided. In particular, when crashes occur, tasks have to be reallocated quickly and automatically in a completely transparent way from the user's point of view.
  Our goal is not to design a complete range of services for Grid systems. We focus only on two specific issues: resource allocation and dependability. Moreover we restrict the scope of our research to time-consuming applications that can be decomposed into a huge number of independent tasks. As all the

tasks generated during the execution of such an application are independent, they can be allocated independently on the different resources of a Grid. The above assumptions are not unrealistic: a grid dedicated to the execution of genomic applications satisfies the above assumptions (See the description of the ACI GénoGRID in section 8.1.1 and the description of the software Paradis in section 5.2).

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

We assume that the architecture of a grid is made of several hierarchical levels. All the resources belonging to a same domain (for example, a research institute) are connected through a local area network which is synchronous. Ressources 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.

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

3.3.3. Intentional Faults and Security Aspects: New Attacks

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

3.3.4. Intentional Faults and Fairness Aspects: Free-Riders

Among the important issues raised by the emerging Peer-to-peer systems, is the one introduced by non-cooperative nodes. Indeed, it is more or less straightforward that rationality exists in P2P systems. By looking at traces showing nodes behaviors, it is clear that many nodes act in a way that is not desirable. The reason is that nodes have a priori no motivation for cooperation. This a priori non-cooperation between nodes inevitably leads to poor system performance. Different approaches may be adopted to face rationality in P2P systems. Among them are *i)* to ignore rationality and to expect that the system will do its best despite self-interested nodes, *ii)* to limit the effect that a rational node can have on the system by using trusted mechanisms, or *iii)* to adopt the fault tolerance techniques. However, none of these approaches benefit from resources that may be potentially offered by these self-interested nodes. We claim that the system must provide incentives to nodes to participate to the given protocols. Solutions may come from economics and more precisely from the mechanism design theory. The scope of this theory is to provide tools and methods to design protocols for self-interested parties. In other words, this theory deals with designing the rules of the game so that a good system-wide outcome will be achieved despite the fact that parties can act on self-interest. Yet, the mechanism design theory advocates the existence of one central mechanism that is used for the whole system and relies on the capability of all the parties to agree on a set of intentions, which contradicts the large scale

and the nature of open systems we consider. So, we want to relax the goals of the mechanism design theory to consider more realistic mechanisms exhibiting a limited scope to keep the computational and informational aspects reasonable, and to reflect the limited range of influence each party can have. We formalize this locality aspect by relying on the self-organization model. However despite rational behavior, experiences shows that some nodes are altruists. For example, in Gnutella, 10% of the nodes are found to serve about 90% of the total download requests. We intend to exploit these nodes to heal the network from selfish nodes and thus to improve the overall behavior of the system.

3.4. Availability and Consistency of Data in Dynamic Systems

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

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

The literature offers a large class of consistency criteria which could be classified into "strong" and "weak" categories depending on the values retrieved by the invoked read operations. The definitions of different consistency criteria [32] 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 [76]. 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. Our main challenge is to design a persistent storage service for dynamic systems able to provide efficient solutions to the above mentioned problems. The final goal of our work is to combine a persistent storage service with incentive mechanisms to cope with non-cooperative nodes in order to design a general solution for a "highly available distributed data service for dynamic systems". In the following we detail the main problems and challenges in implementing this service.

3.4.1. Replication and Placement Policies

The main challenge here is to specify and implement efficient policies for data replication in large scale dynamic systems. The two main issues in any replicated system are placement and lookup. The former deals with deciding which nodes should act as replicas and the latter solves the issue of how to locate a replica.

Replicating data in P2P systems helps users to substantially reduce the latency of their requests. Moreover, efficient replica placement noticeably reduces the routing costs for the operators (in particular when the lookup should traverse several operators). Replication in P2P systems was merely studied in DHT-based systems [69], [74]. In these systems two potential users of the same replica may be far apart from each other hence the global access cost for a replica is in the best case O(log(n)). We would like to design replicated systems where the replica placement exploits the underlying semantic organization of the network. Generally, when a node needs to access an object then, with high probability, similar nodes will eventually need to access the same object. Hence, the interest to replicate the object in each semantic neighborhood. Based on this strategy the lookup costs for an item already replicated in a neighborhood is O(1).

Caching (replicating) Internet based services is a potentially important application for Mobile Ad hoc NETworks (MANETs): it can improve mobile users' perceived quality of service, reduce their energy consumption, and lower their air-time costs. In [53] we proposed strategies for proxy lookup in MANETs. An interesting future direction is to find optimal strategies for proxy placement. Obviously, choosing the right

proxy can have a benefical impact on the performance of the system. For example, the number of proxies for each service in a MANET should be balanced in order to minimize external communication, minimize internal communication, and maximize the number of services that can be cached within the network within a given period of time. This set of proxies needs to be updated continuously (and with a minimal overhead) to adapt itself to topological changes and the partitioning and merges of the MANET caused by mobility.

Another problem faced by designers of replicated systems in dynamic networks (especially in P2P and MANETs) is the rationality of participants. Some nodes in the system may refuse to host replicas, hence additional incentive mechanisms should be deployed. In this context our main challenge is the design of appropriate incentives for helping replication.

3.4.2. Data Consistency

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 [66] 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 [48]: (1) an integrity requirement which states that a new processor that joins the system, and a processor that leaves the system, should change the quorum sets and (2) a scalability requirement which says that the number of elements in the quorum system may increase over time. The increase in the size of the system should maintain its good qualities, i.e. it should decrease the load on each processor and increase the availability of the system. It is important that when the system scales the complexity of the algorithmic remains low. Finally the Join and Leave operation should be applied with low time and message complexity.

3.4.3. New Consistency Criteria

One of the challenges is to identify, formally define and implement different consistency criteria required by new emergent applications executed on top of dynamic scalable systems.

In [31] we identified three applications in P2P networks (distributed calendar, e-buy and e-flows) that need linearizability (the strongest consistency criteria). Moreover, we studied the building blocks necessary and sufficient to design linearizable dynamic systems. The main contribution was to provide, for the first time, an architecture that brings together several seemingly distinct research areas, namely distributed consensus, group membership, notification services (publish/subscribe), scalable conflict detection (or locking), and scalable

persistent storage. All these components are orchestrated together in order to obtain (strong) consistency on an *a priori* unsafe system.

We also promoted the use of *oracles* as a design principle in implementing the respective components of the architecture. Specifically, each of the modules and services are further decomposed into a "benign" part and an "oracle" part, which are specified in a functional manner. This makes the principles of our proposed solution independent of specific implementations and environment assumptions (e.g., it does not depend on any specific distributed hash tables or specific network timing assumptions, etc).

In P2P systems, applications like e-games, e-auctions, distributed file systems further need to be analyzed in order to identify the needed consistency criteria. Moreover in other dynamic systems like grid networks or adhoc networks, applications like tasks execution or data retrieving from caches need strong consistency guarantees but execution environment imposes several restrictions. For example, in adhoc networks nodes are mobile, so the probability that some parts of the network remain disconnected for long periods of time is higher than in conventional networks. In sensor networks the classical theory of consistency faces some problems related to the physical design of sensors. The classical read and write operations need to be replaced here by their aggregated versions. So, the classical definitions and hierarchy of consistency criteria should be revised in order to cope with approximate operations (aggregate read and writes).

One of our goals is to adapt the architecture and algorithms proposed in [31] in order to cope with new specific consistency criteria and particular execution environments (grids, adhoc network or sensor networks).

3.5. Dissemination of Information

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

3.5.1. Broadcast Services

The provision of communication services within a system is essential to allow an entity to send information toward another entity or to broadcast it to all the entities of the system. The properties guaranteed by these services essentially depend on the size of the system. For relatively small systems (less than 100 nodes), broadcast services guarantee strong properties regarding the delivery of the messages to the recipients and the order in which these messages are delivered [54]. For example, a reliable broadcast primitive guarantees that any message sent to a set of entities is delivered either by all of them or by none of them. An atomic broadcast primitive requires that the order in which messages are delivered to the recipients is the same for all of them. These communication primitives facilitate the task of an application designer since strong properties are guaranteed [67]. For instance, ensuring the consistency of multiple replicated copies becomes trivial whenever an atomic broadcast service is available. Designing an efficient implementation of such primitives is still a challenge that we intend to take up.

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

- The first one, that we could call recipients-based broadcast, has to handle the system size issue. A message has to be received by all the identified recipients (in the worst case, by all the nodes). Clearly, existing solutions for small systems are no more applicable because of their message complexity (from O(n) messages to O(n^2) messages are needed to broadcast (resp. to reliably broadcast) an information within the network [50]). Furthermore, in a dynamic setting where an unbounded number of nodes may join, leave voluntarily, or fail (by stopping) during the course of a computation, it is not feasible to identify the set of recipients of a broadcast off-line. Lynch [34] proposes a solution in which recipients declare their intention to participate during the execution of the broadcast primitive. Their solution assumes a synchronous crash failure model. The challenge was to guarantee consistency among the sequences of messages delivered to different recipients, while achieving timely delivery even in the presence of joins and leaves. We intend to extend their work by studying solutions in a partially asynchronous model (which crash failures) which seems to

closer fit the temporal behavior of large scale networks since no a priori knowledge on transmission delays bounds is available in such networks .
Epidemic broadcast, in which each entity propagates the information by broadcasting it to a subset of its neighborhood, is an emerging technique [33], [59]. It turns out that the efficiency of this broadcast technique with respect to the cover of the system strongly relies on the choice of the relays, together with the number of times information is forwarded. Clearly, such constraints can only be estimated because of the dynamicity of the system and the a priori absence of knowledge of the size of the system. The study of the communication graph between entities is a fundamental aspect that we want to address to evaluate the coverage of these assumptions.
Another technique for implementing efficient multicast/unicast primitives has been proposed by Castro et al [39]. Their primitives, built on top of Pastry, a scalable, proximity-aware peer-to-peer overlay, are well adapted for large and highly dynamic groups.

- The second kind of communication schema is semantic-based oriented. This schema is appropriate to face the dynamicity and the size of dynamic and large scale systems. It allows to distribute information to all interested nodes within the system, where the latter are determined according to their known interests and the information contained in the message. This communication schema is captured by the publish/subscribe paradigm. The publish/subscribe paradigm has emerged in the recent years as an effective technique for building distributed applications in which information has to be disseminated from *publishers* (event producers) to *subscribers* (event consumers) [71], [28]. The decoupled nature of publish/subscribe interaction makes these architectures particularly suitable for building large-scale applications where scalability plays a key role. In publish/subscribe systems, users express their interests in receiving certain types of events by submitting a predicate defined on the event contents. The predicate is called the user *subscription*. When a new event is generated and *published* to the system, the publish/subscribe infrastructure is responsible for checking the event against all current subscriptions and delivering it efficiently and reliably to all users whose subscriptions match the event. The challenges that need to be solved are the following ones: 1) storing efficiently and reliably all subscriptions associated with the respective subscribers. 2) Receiving all relevant events from publishers. 3) Dispatching all published events to the correct subscribers.
  Combining expressiveness of subscription language and scalability of the infrastructure poses an interesting challenge that has inspired many researchers to explore this topic further. Despite these promising features, actual deployment of such architectures in real, large-scale systems is currently limited by their lack of self-organization capabilities. For instance, both Siena and Gryphon require connections among brokers to be set up manually; Gryphon also requires all the paths for events to be updated by users upon subscription change; in Kyra [38], when a node joins or leaves the system, the whole subscription partitioning established among all the brokers has to be moved from one broker to another one. We intend to break these limitations with the DPS architecture [29].

3.5.2. Data Aggregation

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

This assumption is not general. Typically, in sensor networks the quantity of potentially available information is large but it is often considered as incomplete or partial, or even unreliable. Thus, directly disseminating raw data within the network is unacceptable first because of the quantity of information that could be potentially sent, and second because part of it may be useless either because it is irrelevant, or it is unreliable. Thus, to obtain an efficient and, in some extent, reliable dissemination of information, data aggregation is needed. By gathering and/or summarizing raw data either from multiple sensors, or from a single one but during a given period of time, one may extract through global functions a limited set of valid information which can

then be disseminated to the application. Data aggregation is a powerful tool to reduce the space complexity of raw information and also to enhance its quality both in terms of reliability and consistency [51], [41], [68].

The latest generation of peer-to-peer (P2P) networks are typically self-organizing and fully distributed systems. A dynamically changing overlay network is maintained, where nodes appear and disappear continuously, and dynamic cooperation among nodes might be created and removed based on the requirements of the particular application. However, such fully distributed platforms have certain drawbacks as well. Control and monitoring is difficult and performing computations poses the challenge of orchestrating the potentially huge number of participating nodes. Hence the need for an aggregation service to provide users or participants of the network with important information like the number of nodes connected to the network, the identity of the most powerful peer, or the total amount of free space in a distributed storage. Interesting results have emerged so far. Astrolabe [70], [75], an aggregation infrastructure for large scale and dynamic network, has recently developed by Van Renesse et al at Cornell, which is an aggregation infrastructure for large scale and dynamic network. Astrolabe provides a good range of functionalities for aggregation (voting, resource location, load balancing, ...). However there are still shortcomings regarding the consistency of the aggregated data in case of failures, the high latency needed for data aggregation, or even the way data is aggregated (in Astrolabe, aggregation is based on a hierarchical structure). Furthermore, simple problems such as computing the size of the network can still be only estimated if local knowledge is used [58].

In this context, we intend to address these shortcoming by designing an aggregation infrastructure satisfying some accuracy, scalability, adaptive and robustness criteria. In terms of accuracy, we want the scheme to be able to provide aggregate information with a certain accuracy in a dynamic large scale environment (where nodes frequently join and leave). With respect to scalability, we want to minimize message passing and avoid flooding schemes that generate excessive redundant messages, to make the scheme scalable to a large network. We also want to ensure that there is good load-balancing, in the sense that no node in the network should be responsible for forwarding a disproportionate amount of network traffic. In many cases it would be very useful if all nodes knew continuously the value of some aggregate, in an adaptive fashion. Adaptivity means that if the aggregate changes due to network dynamism or variations in the values to be aggregated, the output of the aggregation protocol should follow this change reasonably quickly. Potential beneficiaries of an efficient implementation of this functionality include protocols responsible for self-organization in large-scale systems and collaborative environments. In terms of robustness, a scheme should be resilient to the dynamics of nodes joining, leaving, and failing arbitrarily and independently in a P2P network [65]. We plan to develop such a data aggregation framework on statistical learning, including information on link loss, and node faults (data corruption on the node). The main building blocks in this framework should include aggregation tree constructor, statistical sampler, distribution estimator and data predictor. An ultimate objective in this domain is to make aggregation a standard service as it appears in databases through requests languages.

3.5.3. Tracking Service

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

The two main actors in a tracking service are the information producers (the monitored objects) and the information consumers (the information collectors). Therefore tracking has similar flavor with publish/subscribe. More precisely, it can be viewed as the extension of the publish/subscribe service with real-time constrains. That is, due to the target mobility the information collected and spread in the network is quickly obsolete. Hence, the service should ensure at any moment the collectors are notified with the most recent location/position of the monitored objects. The service will also need to handle a large number of moving objects at once. We postulate that large-scale movement patterns are not likely to be uniform, because large-scale real-world environments usually have inherent structure that makes this infeasible. For example, a downtown area of a city may consists of a street grid and buildings that prevent pedestrians from moving around arbitrarily.

Further, it is uncommon for a city-dweller to roam around uniformly at random; a more likely trajectory consists of spending time within some local region, and occasionally making long-range transitions. As a second example, consider a large-scale disaster recovery or battlefield scenario. It may be unusual for every participant to move uniformly across the entire field; rather, due to organizational structure, the movement may be constrained so that there is little traffic between different units, while movement within units is quite high. The common characteristic of the above examples is the movement locality. This observation provides support for a design of tracking service exclusively based on the locality principle.

Interestingly, the tracking problem has so far no formal specification. However, solutions for tracking using a broad class of hypothesis are proposed in [64], [36], [46], [47], [55], [77]. These solutions are based on networked sensors, that have the ability to cover large regions of interest by using many sensors with small detection range. However, some of the previously mentioned solutions are not fault tolerant or do not use local algorithms or use only simple heuristics.

We are interested in formally specifying the tracking problem and designing an efficient self-organizing and fault tolerant tracking service for dynamic large scale systems. This service will be merely based on the adequate models of behavior of the monitored entities.We believe that the fuzzy set theory [78] coupled with the games theory [49] can be used to this end.

Note that anticipating the mobility of monitored objects and aggregating the data processed by the system are important factors in reducing the global cost of the service. Hence, our tracking service will also include statistical prevision tools [45] as well as aggregation tools.

4. Application Domains

4.1. Software for Telecommunication and Space Industries

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

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

In these context, the activity of the ADEPT research team aims at contributing to the emergence of new fundamental services which can be integrated within systems and middlewares. More precisely, we put the stress on the following user's requirements:

Fault-tolerance: In many application domains, distributed entities have to interact with each other in a robust manner. In that case, fault tolerance is a key requirement. A broad range of failures (from benign failures up to malicious failures) have to be tolerated.

Flexibility and adaptativity: The new generation of distributed services has to be adaptive. To achieve these goal, algorithmic solutions have to benefit from the recent advances in software engineering (componentware approach, ...) .

Real-Time: The way timeliness properties are addressed is far from being optimal. New strategies have to be defined to improve the performance of the solutions, to simplify their development and to facilitate the combination of timing requirements with other non-functional requirements such as fault-tolerance.

Large scale dynamic systems: In large systems, applications involving dynamic and mobile sets of participants start only to emerge. Algorithmic solutions developed in a more static context cannot be simply adapted. New approaches and new abstractions are necessary.

Prototyping activities performed during 2005 aims at developing and experimenting a fault tolerant group communication service that is used on one side to implement an active replication service and on the other

side to provide a task allocation mechanism in a GRID. Our goal is not to focus on a particular application but rather to consider generic services that have an universal dimension.

5. Software

5.1. Eden: a Group Communication Service

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

Participants: Michel Hurfin, Jean-Pierre Le Narzul.

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

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

5.2. Paradis: Resource allocation in a Grid

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

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

PARADIS is an "middelware" for a Grid dedicated to genomic applications. Genomic applications are time-consuming applications that can usually be splitted into a huge number of independent tasks. PARADIS is used to reliably allocate resources to these tasks.

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

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

6. New Results

6.1. Models for Dynamic, Scalable and Dependable Systems

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

Participants: Emmanuelle Anceaume, Maria Gradinariu, Frédéric Tronel.

Our work aims at providing a rigorous definition of *self-organization*, one of the most desired properties for dynamic systems, such as peer-to-peer systems, sensor networks, cooperative robotics, or ad-hoc networks. We propose a framework in order to prove the self-organization of dynamic systems with respect to generic criteria (e.g., similarity, load balancing, geographical neighborhood, battery level) that can be composed in order to construct more complex criteria [10], [11].

We also provide a generic algorithm that ensures the self-organization of a system with respect to a given input criterion. Our algorithm is based on the greedy technique, and relies solely on the local knowledge provided by the direct neighborhood of each process. This algorithm can be used as building-block in the construction of any self-organized DHT-based or unstructured peer-to-peer system. We illustrate our theory with a case study that consists in proving the self-organization of CAN, a representative peer-to-peer system.

Several problems are left open for future investigation. The first one is the design of a probabilistic extension to our model. This study is motivated by the fact that connection/disconnection actions are well-modeled by probabilistic laws. Essentially, the liveness property could be redefined using the Markov chains model for probabilistic dynamic I/O automata. Moreover, since our generic algorithm for self-organization uses a greedy deterministic strategy, it may reach just a local maximum for the global criterion. Adding randomized choices could be a way to converge (with high probability) to a global maximum.

Another interesting research direction is to prove or refute our conjecture that the selfish self-organizing generic strategy (Algorithm LSA) is optimal among all the self-organizing local strategies. Games and economic mechanisms theories are rich in tools adequate to this study.

We also intend to extend our framework towards a unified theory of the self* (self-healing, self-configuration, self-reconfiguration, self-repairing) properties of dynamic systems. To this end, we need to extend our case study to other dynamic systems like robots networks and large scale ad-hoc or sensor networks, that may offer complementary insides for understanding the rules that govern complex adaptive systems.

6.2. Accidental and Intentional Faults

Keywords: *consensus problem, distributed computing, erroneous suspicions, free rider, group communication, performance analysis, unreliable failure detector*.

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

6.2.1. Consensus Problem

The *consensus* problem is a central paradigm of fault-tolerant distributed computing informally stated as follows. Each process of a set of n processes proposes a value, and each non-faulty process has to decide a value (termination) in such a way that a decided value is a proposed value (validity) and the non-faulty processes decide the same value (agreement). *Uniform consensus* is a stronger problem in the sense that it requires that no two processes (correct or not) decide distinct values (uniform agreement). So, uniform consensus prevents a faulty process from deciding differently from the non-faulty processes. Consensus is important for several reasons. Consensus is an abstraction of the agreement part of several fault-tolerant distributed computing problems. It constitutes a basic building block on top of which solutions to those problems can be designed. Consensus is also important because of its relation to problem solvability. It has been shown that some distributed agreement problems can be solved in some system (with a given fault model) if, and only if, consensus can be solved in that system (e.g., atomic broadcast and consensus are equivalent problems in asynchronous distributed systems prone to process crashes). Yet another justification of the importance of consensus lies in the principles and mechanisms that underlie the design of most consensus protocols. Understanding those basic principles and mechanisms can provide system designers who have to cope with unreliable underlying distributed systems with a better understanding on the way the unpredictability and uncertainty inherent to those systems can be mastered.

Asynchronous distributed systems are characterized by the absence of bounds on processing time and message transfer delay. Differently from synchronous systems, consensus (hence, also uniform consensus)

cannot be solved in asynchronous systems prone to even a single crash failure [52]. Several approaches have been proposed to allow for the design of deterministic protocols that circumvent this impossibility result. One approach consists in enriching the underlying asynchronous system with an additional mechanism such as a leader oracle (e.g., Paxos) or a failure detector as defined by Chandra and Toueg so that consensus can be solved in the resulting augmented system. Unreliable failure detectors provide information on process failures. On the one hand, failure detectors allow to state the minimal requirements on process failures that allow to solve problems that cannot be solved in purely asynchronous systems. But, on the other hand, they cannot be implemented in such systems: their implementation requires that the underlying distributed system be enriched with additional assumptions. Classic failure detector implementations rely on additional synchrony assumptions such as partial synchrony.

We have revisited four $\diamond\mathcal{S}$-Consensus protocols proposed by (1) Chandra and Toueg, (2) Schiper, (3) Hurfin and Raynal, and (4) Mostéfaoui and Raynal, respectively. We present the protocols following a generic and unified formalism, based on the roles that processes can play during a round of the protocol. When studying the protocols, we focus on two criteria: (1) their communication pattern, and (2) their insensitivity to erroneous suspicions (a concept that we define in the paper). Chandra-Toueg's protocol follows a centralized communication pattern, and is very sensitive to even a single erroneous suspicion. The other protocols follow a decentralized communication pattern, and have mechanisms that make them relatively insensitive to erroneous suspicions.

We have also analyzed the performance of consensus protocols in asynchronous systems augmented with failure detectors. We show that the performance can be influenced by two parameters: (i) the impact of the quality of service (QoS) of the failure detector used; and, (ii) the impact of having a decentralized decision pattern. While there has been some work on how the QoS of the failure detector impacts the performance of the consensus protocol, very little attention has been dedicated to the analysis of the trade-off between having faster decentralized decision at the expenses of generating more network load. We evaluate the performance of a consensus protocol that uses a configurable decentralized decision pattern to speed up decision and a sliding round window mechanism to tolerate bad QoS delivered by the failure detector. The performance analysis of the consensus protocol followed a quantitative approach based on timed-metrics (decision latency). The results were obtained by means of simulations in different scenarios for the QoS of the failure detector and the number of processes that can autonomously decide at a given round of the consensus execution. Our results show that a judicious combination of the two mechanisms is required to achieve good performance. For instance, if for one side the sliding round window is indeed an effective mechanism to deal with bad QoS in the detection of failures, the use of a decentralized decision pattern is not always a good choice.

The k-set agreement problem is a generalization of the consensus problem: each process proposes a value, and each non-faulty process has to decide a value such that a decided value is a proposed value, and no more than k different values are decided. In [20], [19], we focuse on k-set agreement in the context of synchronous systems where up to $t < n$ processes can experience crash or send omission failures. The paper presents a k-set agreement protocol for this failure model which has two main outstanding features. (1) It is *time-optimal*: no process decides or halts after the round $\min(\lfloor f/k \rfloor + 2, \lfloor t/k \rfloor + 1)$ where f is the number of actual crashes ($0 \leq f \leq t$). (2) It is *decision-optimal*. This new optimality criterion, suited to the omission failure model, concerns the number of processes that decide, namely, the protocol forces all the processes that do not crash to decide (whether they commit omission faults or not). It is noteworthy that each of these properties (time-optimality vs decision-optimality) is not obtained at the detriment of the other. Last but not least, the protocol enjoys another first-class property, namely, simplicity.

6.2.2. Atomic Commitment Problem

In the context of transactional systems, we have proposed in [18] a highly modular derivation of fast non-blocking atomic commit protocols.

Our first contribution is the proposal of an elegant approach to design modular and efficient non-blocking atomic commit protocols. In this approach, an atomic commit protocol relies on an hyperfast consensus protocol that decides in one communication step in good scenarios. When good scenarios do not apply, the

hyperfast consensus protocol makes use of an underlying consensus that allow to terminate. In this general schema, consensus is used as a termination protocol for the AC protocol only when necessary, in case of failures or erroneous suspicions. The main advantage of this approach is that, when certain good but realistic conditions are satisfied, an efficient solution to the consensus problem directly leads to fast atomic commit protocols. By *fast*, we mean algorithms that decide a transaction (abort or commit) as soon as possible. This happens when some process votes NO (resulting in abort) or all processes vote YES and there are no failures or erroneous suspicions (resulting in commit).

From the proposed schema, we have derived two atomic commit protocols, namely *AC-Set* and *AC-Value*. *AC-Set* (respectively *AC-Value*) relies on a hyperfast consensus protocol, called *Set-Consensus* (respectively *Val-Consensus*). These two consensus introduce new assumptions giving rise to *hyperfast learning* algorithms, that allow the learning of decided values within one communication step. This happens when a sufficient number of processes propose the same value for consensus. Recently in [60], Lamport has pointed out the importance in studying and applying new pertinent consensus definitions as a way for breaking the limit on message delays for agreement problems. Our proposal contributes to investigate this approach by the derivation of high-performance atomic commit protocols. Besides, it shows that the design of well structured protocols is compatible with high performance.

Our second contribution is the proposal of atomic commit protocols that exhibit performances that equal or overcome those of ad-hoc protocols proposed so far. Both proposed protocols are as efficient as the (2PC) in terms of latency. They terminate after *three* communication steps in the absence of failures. Protocol *AC-Set* is more efficient than any other failure-detector based AC protocol published so far. It requires $(2nf + 3n)$ messages (without a broadcast network) or $(n + f + 2)$ messages (with a broadcast network) to commit. Thus, in presence of a low resiliency rate ($f < 2$), it is as efficient as 2PC and 3PC. Protocol *AC-Value* requires $(n(2n + 1))$ messages (without a broadcast network) or $(2n + 1)$ messages (with a broadcast network). Compared to other protocols, our protocol requires weaker conditions to early decide. In the case of *AC-Value*, decisions are speed up if at least $(f + 1)$ propositions are for COMMIT in the consensus phase.

6.2.3. Group Communication

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

Many protocols have been designed to cope with permanent crash failures. But such failures constitute just one particular type of failures. Considering more severe failures (like omission or malicious failures) is a major challenge. Moreover, a failure can be permanent or temporary. In some particular contexts (for example, space missions), the repair or the replacement of hardware is impossible or very expensive. Therefore, it should be possible to remove temporally a failed component from the set of operational components and to insert it again latter when the component has recovered.

Part of our current research consists in designing flexible modular and adaptive group communication services that can tolerate various kind of failures. With this goal in mind, we consider component-ware approach and try to benefit from the new advances in software engineering when we design adaptive algorithmic solutions. Componentware is a promising paradigm for helping developers to build distributed applications. The idea behind this paradigm is to allow a developer to configure and assemble components rather than writing complex, specific and somewhat obscure code that uses system services to allow objects to inter-operate. Based on this paradigm, we have designed ADAM, a library of agreement components for reliable distributed programming. ADAM is based on a generic and adaptive solution to the consensus problem that can be customized to cope with the characteristics of the environment as well as the properties of the

reliable distributed abstractions that have to be ensured. In 2005, we have started to extend this library to offer new features. Our aim is to provide group services that can be used to manage a set of replicated IDS (intrusion detection mechanisms).

The major aim of a Grid is to federate several powerful distributed resources within a single virtual entity which can be accessed transparently and efficiently by external users. As a Grid is a distributed and unreliable system involving heterogeneous resources located in different geographical domains, fault-tolerant resource allocation services have to be provided. In particular, when crashes occur, tasks have to be reallocated quickly and automatically, in a completely transparent way from the users' point of view. In 2005 we have continued to design and develop PARADIS, a system based on the consensus building block of Adam and implemented in a Grid dedicated to genomic applications. These time-consuming applications can be split up into a huge number of independent tasks which can be allocated independently on different domains. Load strategies can be plugged to PARADIS thanks to a bid mechanism used for the task allocation.

6.2.4. Free-Riding

In peer-to-peer (P2P) systems resources are highly distributed and stored by individual peers. In such environments where each peer may present a different self-interested entity, designing algorithms for sharing resources becomes a difficult task. Traditional distributed algorithms design faces only two types of behaviors: *obedient* (i.e. to follow the algorithm) or *malicious* (i.e. to "play" against the others). In contrast, in P2P systems users may have a strategic behavior which may be neither obedient nor malicious, just rational. It was established for example that in the absence of incentives a large fraction of peers have *a priori* no motivation to share while they are naturally incited to use system resources. The effects of "free-riding" (consuming resources without contributing) was extensively studied in the last years. This behavior can cause severe degradation of the system performances or even may lead to the system collapse [73]. As emphasized by Shneidman et al., different approaches may be adopted to face rationality: *i)* to ignore it and to expect that system will do its best despite self-interested peers, *ii)* to limit the effect that a rational peer can have on the system by using trusted mechanisms, or *iii)* to adopt the fault tolerance techniques. However, none of these approaches benefits from resources that may be potentially offered by these rational and self-interested peers. Thus the system must motivate each peer to behave rationally in a system efficient way. Solutions come from economics and more precisely from the mechanism design theory that always stressed for algorithms that work correctly in the presence of predictable selfish behavior.

In [13] we address the Fair Resource Sharing problem. Essentially, this problem aims at optimizing the resource sharing in P2P systems where peers exhibit rational behavior. We propose a mechanism to solve this problem that comprises a fair differential service incentive for motivating peers to cooperate and an algorithmic ingredient encapsulated into a distributed middleware layer. The incentive ingredient eventually harnesses the resources of all the peers by motivating free-riders to change their strategy (peers that contribute more get better quality of services) while encouraging greedy peers to share their requests with less solicited ones. The middleware layer is composed of four services: *i)* The Registration service, which is used by each new peer to advertise the system of its resource potential; *ii)* the Semantic Group Membership service, which self-organizes peers into semantics clusters to maximize a fair distribution of requests among peers having similar resources but dissimilar popularity; *iii)* the Tracking service whose role is to track free-riders within a given semantic group, and *iv)*, the Aggregation service which combines the actions of each peer to evaluate his behavioral functions. All these services, bringing together for the first time distinct research areas (i.e. semantic group membership, aggregation and tracking) are distributed and implemented using peer-to-peer tools. The engineering novelty of our mechanism design is twofold. First, services are implemented within a middleware layer, limiting the impact of malicious peers, and second, the modular conception of our mechanism, that abstracts the incentives from the services needed to implement these incentives can be further exploited in the design of a generic middleware for incentive implementations in P2P systems.

6.3. Availability and Consistency of Data in Dynamic Systems

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

Participants: Emmanuelle Anceaume, Maria Gradinariu, Vincent Gramoli, Antonino Virgillito.

6.3.1. Data Consistency

In [17], the authors investigate operation liveness and gossip management of a dynamic distributed atomic data service. The approach proposed by Lynch and Shvartsman uses all-to-all gossip messaging to guarantee view consistency. Moreover, we point out a problem where under certain scenarios read/write operations may become delayed or blocked. We introduce a more practical algorithm providing two refinements. First, in order to cope with congestion due to limited network bandwidth, we reduce all-to-all communication pattern by identifying configuration owner nodes and by restricting gossip across them. Second, we present a solution that allows blocked (or delayed) operations to resume processing and complete successfully. We show improvements of our approach by bounding the operation latency under reasonable assumptions and by providing experimental results of a distributed implementation of this service.

In [14], the authors introduce the RDS (Reconfigurable Distributed Storage) algorithm for implementing a reconfigurable distributed shared memory in an asynchronous dynamic network. RDS achieves the design goals of: (i) allowing read and write operations to complete rapidly, and (ii) providing long-term fault tolerance through reconfiguration, a process that evolves the quorum configurations used by the read and write operations. For the first of these purposes, RDS uses a single round-trip read operation and imposes no dependences between operations and ongoing reconfigurations, while the later purpose is achieved by combining three techniques. First, reconfiguration does not constrain content of subsequent quorum configurations. Second, the use of the "Fast Paxos" consensus algorithm allows the protocol for choosing the next configuration to complete rapidly. Last but not least, we overlap this protocol with the protocol for removing obsolete configurations, that is the reconfiguration delay is seemingly optimal when the system stabilizes. To conclude, RDS improves on previously developed alternatives by using a more efficient reconfiguration protocol, thus guaranteeing better fault tolerance and faster recovery from network instability. This paper presents RDS, a formal proof of correctness, conditional performance analysis, and experimental results.

We propose an architecture for *self-adjusting and self-healing atomic memory* in highly dynamic systems exploiting peer-to-peer (p2p) techniques. Our approach, named *SAM*, brings together new and old research areas such as p2p overlays, dynamic quorums and replica control. In SAM, nodes form a connected overlay. To emulate the behavior of an atomic memory we use intersected sets of nodes, namely *quorums*, where each node hosts a replica of an object. In our approach, a quorum set is obtained by performing a deterministic traversal of the overlay. The SAM overlay features self-* capabilities: that is, the overlay self-heals on the fly when nodes hosting replicas leave the system and the number of active replicas in the overlay dynamically self-adjusts with respect to the object load. In particular, SAM pushes requests from loaded replicas to less solicited replicas. If such replicas do not exist, the replicas overlay self-adjusts to absorb the extra load without breaking the atomicity. We propose a distributed implementation of SAM where nodes exploit only a restricted local view of the system, for the sake of scalability [12]. Proofs of correctness of SAM are shown in [21].

6.3.2. Mobile Philosophers

Research in resource sharing problems (both in discovering new and improved solutions, and introducing new problems) has been active for more than three decades. Mutual exclusion, group mutual exclusion, local mutual exclusion, dining philosophers, drinking philosophers, and readers-writers are some of them. All the above problems assume knowledge of one or more of the following: the number of processes, the number of resources, the layout of the resources, and the degree of synchronization.

In [9] our main motivation is to consider a *dynamic network* in the following sense: (i) Both the number of resources and number of processes can vary. (ii) Both resources and processes can join/leave the network at unknown times. (iii) The processes are mobile. They can connect/disconnect to/from any point in the network. (iv) The processes may request an arbitrary number of resources. We have proposed a new specification for the dining philosopher problem in the context of overlay networks. We also provided a solution to this problem for systems where philosophers can hold simultaneously at most two consecutive resources. The solution has several noteworthy properties: (i) It is asynchronous (a philosopher can ask a resource at any time without any

synchronization). (ii) It is local (no global information is needed). (iii) It works for networks where both the number of resources and number of philosophers are unknown. (iv) It is self-stabilizing (it copes with nodes mobility and local data corruption).

The proposed solution to the mobile philosopher problem works for a simple access pattern (where philosophers can hold two consecutive resources). Self-stabilizing solutions for more general access patterns (e.g., where a philosopher will be allowed to use more than two resources, access two non-consecutive resources, or both) are future research topics. We think that the main problem that needs to be solved to make our solution more general is to design an efficient deadlock detection strategy which will detect the possible cycles of philosophers. Another open issue is to find the minimal requirements to provide fault-tolerant and self-stabilizing solutions (ftss) to the mobile philosopher problem.

6.4. Dissemination of Information

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

Participants: Emmanuelle Anceaume, Florent Claerhout, Maria Gradinariu, Vincent Gramoli, Michel Hurfin, Philippe Raïpin Parvédy, Antonino Virgillito.

6.4.1. Publish/Subscribe Paradigm

The publish/subscribe paradigm has emerged in the recent years as an effective technique for building distributed applications in which information has to be disseminated from publishers (event producers) to subscribers (event consumers). Users express their interests in receiving certain types of events by submitting a filter on the event contents, called a subscription. When a new event is generated and published to the system, the publish/subscribe infrastructure is responsible for checking the event against all current subscriptions and delivering it to all users whose subscriptions match the event. Content-based publish/subscribe systems enable complex filters on the event content, enabling the use of constraints such as ranges, prefixes, and subfixes. Combining expressiveness of subscription language and scalability of the infrastructure poses an interesting challenge that has inspired many researchers to explore this topic further. Typical implementations of publish/subscribe systems rely on a network of dedicated servers (usually called brokers). However, in dynamic decentralized scenarios where it may be necessary to add or remove brokers very often for example to adapt to change of load conditions or failures.

In this context, we propose to build DPS (Dynamic Publish/Subscribe System), a reliable content-based pub/sub architecture with self-$*$ properties, allowing an easier deployment and a more flexible adaptation in a larger spectrum of applications. In this work, self-$*$ capabilities encompass 1) *self-organization* — the capability of the system to reduce the entropy of the system [10], [11]; 2) *self-configuration* — the capability of the processes to set-up their structural relationships; and 3) *self-healing* — the capability of the processes to preserve their structural relationships despite the dynamicity of the system (joins, departures, or failures).

DPS is not based on a network of brokers; rather, subscribers interact on a peer-to-peer basis coordinating among themselves to construct optimized event diffusion paths without any human intervention. More precisely, we propose a subscription-driven overlay in which subscribers selforganize according to similarity relations among their subscriptions. In DPS, two subscribers are considered similar when they share a common attribute of a subscription. Similar subscribers are logically connected into one group. Groups of subscribers self-configure to form tree structures such that only one tree is built per attribute. The DPS overlay is generic in the sense that different algorithms can be used for overlay construction and event publication according to the specific application requirements.

In this context the main contribution we have done in this work is two-fold. Firstly, after giving a formal characterization of the semantics of a pub/sub system, we introduce the DPS overlay and the correctness conditions required for it to properly work as a pub/sub system. Correctness conditions are defined in terms of the similarity relations, stating that DPS satisfies the pub/sub semantics if and only if the graph derived by the similarity relations is completely connected. As a second contribution, we present a specific protocol for building and maintaining an overlay, formally proving that the connectivity condition is always satisfied,

despite nodes joining, leaving, or failing. More precisely, this protocol is able to tolerate a large number of concurrent failures, half of the number of neighbors of a node. At the best of our knowledge, providing a self-healing pub/sub infrastructure and proving its correctness is a novel contribution in the pub/sub area, where formal studies have been devoted mainly to the correctness of event routing. A first version of these contributions is presented in [29].

6.4.2. *Self-∗ Query Region Covering in Sensor Networks*

In sensor networks, *queries* are sent from some devices (could be a PDA, laptop, or any computer) to sense some data/events over some time period and a geographical region, called *query region*. The query region is usually a subset of the total region covered by all the sensors in the network. Since the sensors are usually densely deployed, there are usually a lot more sensors than required to process a given query. One possible way to minimize usage of energy is not to keep all sensor nodes fully active all the time. Some of them can be put to passive mode some times while others are active in sensing the data in the environment. The above scenario is modeled as an optimization problem in sensor networks, called the *connected sensor cover* problem. Given a query over a sensor network, select a minimum or nearly minimum set of sensors, called connected sensor cover, such that the selected sensors cover the query region, and form a connected network among themselves. In its general form, this problem is known to be NP-hard.

In our papers [15] and [16], we design the *fully distributed, strictly localized, scalable,* and *self-∗* solutions to the connected sensor cover problem. The *Self-∗* concept has been used to include many fault-tolerant properties like *self-configuring*, *self-reconfiguring/self-healing*, etc. We present a *self-stabilizing* solutions and show that these solutions are both self-configuring and self-healing. In a self-stabilizing system, every computation, starting from an arbitrary state, eventually reaches a state which satisfies the specification. Nodes achieve the global objective by using only local computations. Local algorithm based sensor networks are more robust, and scale better. The proposed solutions are space optimal in terms of number of states used per node. Another feature of the proposed algorithms is that the faults are contained only within the neighborhood of the faulty nodes.

6.4.3. *Trajectory tracking in sensor networks*

In [22], we focus on the *trajectory tracking* that deals with gathering coherent information on the past behavior of a target. This information is necessary in complex computations (e.g., forecasting) that are critical in the optimization of a wide range of applications (e.g., natural disaster monitoring, pursuer/evader, etc.). We formally specify an architecture, and a fault-tolerant and self* solution for the trajectory tracking in sensor networks. Our architecture elegantly integrates three abstractions that encapsulate the necessary conditions for implementing the trajectory tracking specification. We give the first formal specification of the trajectory tracking problem in sensor networks. One additional merit of our architecture is in blending together for the first time in the context of tracking applications three research areas: temporal correlated data, causal correlated (content related) data, and self* overlays. Moreover, for each of the three abstractions, we propose novel fault-tolerant algorithms with self* (self-organizing and self-healing) capabilities. In particular, the proposed algorithms cope with node additions/failures, transient faults, and crashes. The proposed architecture can be tuned with no extra cost to solve the multi-trajectory tracking problem (assuming the existence of a filtering service) and to forecast velocity or trajectory that can be further used for optimizing many applications including pursuer/evader.

7. Contracts and Grants with Industry

7.1. Speeral Contract (2002-2005)

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

Participants: Emmanuelle Anceaume, Maria Gradinariu, Vincent Gramoli, Aina Ravoaja.

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

All these questions are studied in this collaboration with FT R&D. This study started at the beginning of 2003 and has finished in 2005. In 2005, two deliverables have been provided [24], [23].

7.2. Assert Contract (2004-2006)

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

Participants: Emmanuelle Anceaume, Michel Hurfin.

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

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

The contribution of the ADEPT project focuses on the designed of fault-tolerant middleware services. In 2005, two deliverables co-authored with people from INRIA Rocquencourt and people from the Vienna University of Technology have been provided [25], [26].

8. Other Grants and Activities

8.1. National Project

8.1.1. ACI GénoGRID (2003-2005)

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

In the context of the ACI GRID (Actions Concertées Incitatives - Globalisation des Ressources Informatiques et des Données) program, supported by the French ministry of Research, the GénoGRID project has started in December 2001 and stopped at the beginning of 2005. Two Irisa teams were involved in this project: ADEPT and SYMBIOSE.

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

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

8.1.2. ACI Daddi (2004-2006)

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

Each day, the databases maintaining information about system vulnerabilities are growing with new cases. In addition to the classical prevention security tools, intrusion detection systems (IDS) are nowadays widely used by security administrators to detect attack occurrences against their systems. Anomaly detection is often viewed as the only approach to detect new forms of attack. The main principle of this approach consists in building a reference model of the behavior for a given entity (user, machine, service, or application) in order to compare it with the current observed behavior. If the observed behavior diverges from the model, an alert is raised to report the anomaly. Rather than defining an explicit model, we suggest to consider an implicit one. Design diversity will be used to identify dynamically the reference model. In our approach, any request is forwarded to different modules implementing the same functionality but through diverse designs. Any difference between results that are returned can be interpreted as a possible corruption of one or several modules. The task of the ADEPT project is to provide secure group communication mechanisms that allow to managed the group of modules.

8.1.3. ACI TAGADA (2004-2006)

Participant: Frédéric Tronel.

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

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

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

In the TAGADA project, we want to study and try to unify these results. Until now, we have met twice :

1. May 11 2005 : First TAGADA meeting in Paris.
2. November 3-4 2005 Second TAGADA meeting in Marseille.

8.2. International Cooperations

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

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

In August 2005, Fabiola Greve who is professor at the Federal University of Bahia has spent one month in the ADEPT research team and has worked with members of the ADEPT team on agreements problems and the use of a component model for building a library of fault-tolerant abstractions. A cooperation project with her university, the Federal University of Paraiba (Prof. Francisco Brasileiro) and several French laboratories (ADEPT Team, GRAND LARGE Team, LIP6) is supported by Capes/Cofecub (projet 497/05) during a period of two years (2005-2006). Michel Hurfin is the French coordinator of this project which focuses on distributed computing and Grid computing. In 2005, Francisco Brasileiro (Federal University of Campina Grande) , and Christina Chavez (Federal University of Bahia) have visited IRISA in August and December respectively. In 2005, two French researchers Jean-Pierre Le Narzul (ADEPT) and Sebastien Tixeuil (GRAND LARGE) have visited the Federal University of Bahia. Joint research works have lead to common publications.

8.2.2. USA (University of Nevada)

Participant: Maria Gradinariu.

Ajoy Datta, professor at the University of Nevada, has visited us during one month in June 2004 and will visit us again during one month in summer 2006. Joint works on sensor networks have been initiated. Maria Gradinariu is the co-supervisor of master students of the university of Nevada and has spent a few weeks in the University of Nevada in October 2005.

8.2.3. Japan (JAIST)

Participants: Emmanuelle Anceaume, Maria Gradinariu, Michel Hurfin.

Xavier Defago has spent two weeks in August 2005 at IRISA. During his visit he worked with members of the ADEPT research team [10], [11]. We also submitted a cooperation project focusing on the definition of model and the study of fault-tolerant aspects in the particular case of networks of mobile sensors.

9. Dissemination

9.1. Teaching Activities

- Some members of the ADEPT research team belong to the University of Rennes I or to ENST Bretagne (a telecommunication engineering school). Therefore, an important part of their time is devoted to teaching to engineers and master students.
- Jean-Pierre Le Narzul has the responsibility for organizing several teaching units at ENST Bretagne (RSM Department). He gives lectures on both distributed computing and object-oriented language. He is also involved in the setting of programs for continuous training.
- Frédéric Tronel has the responsibility of managing the master in computer science devoted to security aspects (University of Rennes I, Ifsic).
- Emmanuelle Anceaume belongs to the specialist commission (university of Rennes I, section 27).
- Maria Gradinariu is the co-supervisor (with Prof. Ajoy Datta) of two master students of the university of Nevada, USA.
- Michel Hurfin gives lectures on fault tolerance and distributed computing to students of two engineering schools: ENST Bretagne (Rennes, 10 hours) and Supelec (Rennes, 12 hours).

9.2. Presentations of Research Works

- Emmanuelle Anceaume is the main organizer of the seminars entitled "Networks and Systems" that are periodically held in our institute.
- In September 2005, Emmanuelle Anceaume and Maria Gradinariu have organized a journey entitled “Outils pour systèmes grande échelle dynamiques ”. This event took place at IRISA and was an opportunity for researchers from different communities to meet.
- Maria Gradinariu has presented her research activities
  - In March 2005, during a Tarot (Techniques Algorithmiques, Réseaux et d’Optimisation pour les Télécommunications) meeting organized in Paris.
  - In September 2005, during a meeting organized by the "ACI Sécurité FRAGILE" in Paris.
  - In October 2005, during a visit at the University of Nevada.
- Jean-Pierre Le Narzul has presented his research activities during a visit in the Federal University off Bahia.
- Frédéric Tronel has made a presentation entitled "A brief introduction to algebraic topology tools used in the study of distributed systems" during a TAGADA meeting in may 2005.
- Members of the ADEPT research team have attended several conferences and workshops dealing with distributed computing (the reader is encouraged to refer to the bibliographic references for additional information).

9.3. Integration within the Scientific Community

- Emmanuelle Anceaume
  - belongs to the organization committee of the *8ème Rencontre Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel 2006)*, May 2006, Trégastel, France.
  - acts as a publicity chair of the *3rd International Conference on Autonomic and Trusted Computing (ATC 2006)*, September 2006, China.
- Maria Gradinariu
  - was member of the program committee of the *7th International Symposium on Self Stabilizing Systems (SSS 2005)*, LNCS, October 2005, Spain.
  - belongs to the organization committee of the *8ème Rencontre Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel 2006)*, May 2006, Trégastel, France.
  - is the co-chair of the Symposium on Self Stabilizing Systems that will be organized in 2006 (SSS 2006).
  - has received, in January 2005, the "Wilkes Award second best paper" for the paper entitled "Self-stabilizing mutual exclusion under arbitrary scheduler" published in the Computer Journal (co-authors: Sébastien Tixeuil and Ajoy Datta).
- Michel Hurfin

- was member of the program committee of the IEEE *19th International Conference on Advanced Information Networking and Applications (AINA 2005)* that was organized in March 2005, Taiwan.
- was member of the program committee of the IEEE *11th International Conference on Parallel and Distributed Systems (ICPADS)*, July 2005, Japan.
- was member of the program committee of the *2nd Latin-American Symposium on Dependable Computing (LADC)*, LNCS, October 2005, Brazil.

10. Bibliography

Major publications by the team in recent years

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

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

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

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

[5] M. HURFIN, A. MOSTÉFAOUI, M. RAYNAL. *A Versatile Family of Consensus Protocols Based on Chandra-Toueg's Unreliable Failure Detectors*, in "IEEE Transactions on Computers", vol. 51, n° 4, April 2002, p. 395-408.

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

[7] M. HURFIN, M. RAYNAL. *A simple and Fast Asynchronous Consensus Protocol Based on a Weak Failure Detector*, in "Distributed Computing", vol. 4, n° 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° 8, Aug 2002, p. 900–915.

Articles in refereed journals and book chapters

[9] A. DATTA, M. GRADINARIU, M. RAYNAL. *Stabilizing Mobile philosophers*, in "Information and Processing Letters", IRISA technical report number 1666, vol. 95, 2005, p. 299-306.

Publications in Conferences and Workshops

[10] E. ANCEAUME, X. DEFAGO, M. GRADINARIU, M. ROY. *Toward a theory of self-organization*, in "Proc. of the International Symposium on DIStributed Computing (DISC), short paper, Cracow, Poland", September 2005, p. 505–506.

[11] E. ANCEAUME, X. DEFAGO, M. GRADINARIU, M. ROY. *Toward a theory of self-organization*, in "Proc. of the 9th International Conference on Principles of Distributed Systems (OPODIS)", 2005.

[12] 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.

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

[14] G. CHOCKLER, S. GILBERT, V. GRAMOLI, P. M. MUSIAL, A. A. SHVARTSMAN. *Reconfigurable Distributed Storage for Dynamic Networks*, in "Proc. of the 9th International Conference on Principles of Distributed Systems (OPODIS)", December 2005.

[15] A. K. DATTA, M. GRADINARIU, P. LINGA, P. R. PARVÉDY. *Self-$*$ Distributed Query Region Covering in Sensor Networks*, in "IEEE Symposium on Reliable Distributed Systems (SRDS 2005)", 2005.

[16] A. DATTA, M. GRADINARIU, R. PATEL. *Distributed Self-* Minimum Connected Covering of a Query Region in Sensor Networks*, in "Procs. of the ACM-sigact International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)", 2005.

[17] V. GRAMOLI, P. M. MUSIAL, A. A. SHVARTSMAN. *Operation Liveness and Gossip Management in a Dynamic Distributed Atomic Data Service*, in "Proc. of the 18th International Conference on Parallel and Distributed Computing Systems", September 2005.

[18] F. G. P. GREVE, J.-P. L. NARZUL. *Generating Fast Atomic Commit from Hyperfast Consensus*, in "Proc. of the Second Latin-American Symposium on Dependable Computing (LADC 2005), Salvador (Brazil)", Lecture Notes on Computer Science, Springer-Verlag, October 2005, p. 226–244.

[19] P. R. PARVÉDY, M. RAYNAL, C. TRAVERS. *Decision Optimal Early-Stopping k-set Agreement in Synchronous Systems Prone to Send Omission Failures*, in "Proc. of the IEEE 11th Pacific Rim International Symposium on Dependable Computing (PRDC11)", 2005.

[20] P. R. PARVÉDY, M. RAYNAL, C. TRAVERS. *Early-Stopping k-set Agreement in Synchronous Systems Prone to any Number of Process Crashes*, in "Proc. of the International Conference on Parallel Architectures and Compilation Techniques (PaCT 2005)", 2005.

Internal Reports

[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] F. CLAERHOUT, A. DATTA, M. GRADINARIU, M. HURFIN. *Self* Trajectory Tracking in Wireless Sensor Networks*, Technical report, n° 1772, IRISA, 2005.

Miscellaneous

[23] E. ANCEAUME, M. GRADINARIU, V. GRAMOLI. *Deliverable 5 : SAM: Self* Atomic Memory for Peer-to-Peer Systems*, contract number 42338322, July 2005.

[24] E. ANCEAUME, M. GRADINARIU, A. RAVOAJA. *Deliverable 4 : Incentive for Peer-to-peer Fair Resource Sharing*, contract number 42338322, April 2005.

[25] E. ANCEAUME, M. HURFIN, ALL PBSE ASSERT MEMBERS. *Deliverable D1.1-1, Specifications of Computer Based System Requirements for System Families*, 2005.

[26] E. ANCEAUME, M. HURFIN, ALL PBSE ASSERT MEMBERS. *Deliverable D1.1-3, Spécifications of Generic CBBs for Distribution and Dependability*, 2005.

Bibliography in notes

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

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

[29] E. ANCEAUME, A. DATTA, M. GRADINARIU, G. SIMON, A. VIRGILLITO. *DPS: Self-* dynamic Reliable Content-based Publish/Subscribe System*, Research Report, n° PI-1665, IRISA, dec 2004.

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

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

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

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

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

[35] 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.

[36] R. BROOKS, P. RAMANATHAN, A. SAYEED. *Distributed Target Classification and Tracking in Sensor Networks*, in "Proceedings of the IEEE, Special issue on sensor networks", vol. 91, n° 8, 2003.

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

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

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

[40] 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.

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

[42] 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.

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

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

[45] N. A. C. CRESSIE. *Statistics for Spatial Data*, Wiley Series in Probability and mathematical statistics, 1993.

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

[47] 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.

[48] 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.

[49] P. K. DUTTA. *Strategies and Games: Theory and Practice*, Hardcover, 1999.

[50] X. DÉFAGO, A. SCHIPER, P. URBAN. *Totally ordered broadcast and Multicast algorithms: A comprehen-sivesurvey*, in "ACM Computing Surveys", vol. 36, n° 4, 2004, p. 372–421.

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

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

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

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

[55] 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.

[56] 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.

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

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

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

[60] L. LAMPORT. *Lower Bounds for Asynchronous Consensus*, Research Report, n° MSR-TR-2004-72, Microsoft Corporation, Jul 2004.

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

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

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

[64] K. MECHITOV, S. SUNDRESH, Y. KWON. *Cooperative Tracking with Binary-Detection Sensor Networks*, in "Proceedings of the 1st international conference on Embedded networked sensor systems (Sensys'03)", 2003.

[65] A. MONTRESOR, M. JELASITY, O. BABAOGLU. *Robust Aggregation Protocols for Large-Scale Overlay*

Networks, in "In Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN'04)", IEEE Computer Society, 2004, p. 19–28.

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

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

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

[69] S. RATNASAMY, P. FRANCIS, M. HANDLEY, R. KARP, S. SHENKER. *A Scalable Content Addressable Network*, in "In Proc. SIGCOMM", ACM, 2001.

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

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

[72] 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.

[73] J. SHNEIDMAN, D. PARKES. *Rationality and Self-Interest in Peer to Peer Networks*, in "Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03)", 2003.

[74] I. STOICA, R. MORRIS, D. KARGER, M. F. KAASHOEK, H. BALAKRISHNAN. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*, in "In Proc. SIGCOMM", ACM, 2001.

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

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

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

[78] H. J. ZIMMERMANN. *Fuzzy Ser Theory and its Applications*, Hardcover, 2004.