INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Regal

Resource management in large scale distributed systems

Rocquencourt

THEME COM

Table of contents

1. Team

Regal is a joint team between INRIA, CNRS and Paris 6 University, through the "Laboratoire d'Informatique de Paris 6", LIP6 *(UMR 7606).*

Head of team
Pierre Sens [Professor Univeristé Paris 6, HdR]

Vice-head ofteam
Mesaac Makpangou [Research associate (CR) Inria, HdR]

Administrative assistant
Tran Thi-Thanh-Van [Secretary Inria]

Staff member INRIA
Marc Shapiro [Research Director (DR) INRIA]

Staff member LIP6
Luciana Arantes [Associate professor Université Paris 6]
Denis Conan [6 months Sabbatical at Regal - Associate professor INT]
Bertil Folliot [Professor Université Paris 6, HdR]
Olivier Marin [Associate professor Université Paris 6]
Maria Gradinariu [Associate professor Université Paris 6]
Gaël Thomas [Associate professor Université Paris 6]

Ph. D. student
Lamia Benmouffok [Microsoft research grant - Université Paris 6]
Jean-Michel Busca [ATER Université Paris]
Ikram Chabbouh [Université de Tunis]
Charles Clement [Université Paris 6]
Nicolas Geoffray [Université Paris 6]
Corina Ferdean [Université Paris 6]
Nicolas Gibelin [Université Paris 6]
Assia Hachichi [Université Paris 6]
Cyril Martin [Université Paris 6]
Fabio Piconni [Université Paris 6]
Julien Sopena [Université Paris 6]
Pierre Sutra [Université Paris 6]

2. Overall Objectives

2.1. Overall Objectives

Regal is a joint research team between LIP6 and INRIA-Rocquencourt. Regal focuses on management of large scale distributed systems (especially P2P and Grid architectures). A significant axis of the project work is devoted to large scale replication for applications which have strong constraints in terms of dynamicity (multi-agents applications, resource servers on the Web).

3. Scientific Foundations

3.1. Scientific Foundations

Keywords: *Grid computing*, *Peer-to-peer*, *consistency*, *distributed system*, *dynamic adaptation*, *fault tolerance*, *large scale environments*, *replication*.

Scaling to large configurations is one of the major challenges addressed by the distributed system community lately. The basic idea is how to efficiently and transparently use and manage resources of millions of hosts spread over a large network. The problem is complex compared to classical distributed systems where the number of hosts is low (less than a thousand) and the inter-host links are fast and relatively reliable. In such “classical” distributed architectures, it is possible and reasonable to build a single image of the system so as to “easily” control resource allocation.

In large configurations, there is no possibility to establish a global view of the system. The underlying operating system has to make decisions (on resource allocation, scheduling ...) based only on partial and possibly wrongs view of the resources usage.

Scaling introduces the following problems :

- Continuous failures occurrence: as the number of hosts increases, the probability of a host failure converges to one. For instance if we consider a classical host MTBF (Mean Time Between Failure) equals to 13 days, in a middle scale system composed of only 10000 hosts, a failure will occur every 4 minutes. Compared to classical distributed systems, failures are more common and have to be efficiently processed.
- Asynchronous networks: clearly on the Internet network transmission delays are very dynamic and unbounded. The asynchronous nature of the Internet makes it extremely problematic to control the consistency of the system. This problem is mainly due to the Fischer, Lynch, Paterson impossibility which shows that a basic feature such as consensus cannot be deterministically solved in an asynchronous system subject to only one crash failure. The fundamental difficulty is that the system has to compose on possibly wrong views of the system where hosts suspected faulty are correct, or where really faulty host are seen in a correct state.
- Failure models: classical distributed systems usually consider crash or omission failures. In a large scale context like peer-to-peer overlay networks, such an assumption is not reasonable since hosts can not only be affected by failures but can be attacked and consequently become malicious. Clearly the failure model has to be extended to deal with a possibly Byzantine behavior of hosts.

Two architectures in relation with the scaling problem have emerged during the last years :

Grid computing : Grid computing offers a model for solving massive computational problems using large numbers of computers arranged as clusters interconnected by a telecommunications infrastructure as internet, renater or VTHD.

If the number of involved hosts can be high (several thousands), the global environment is relatively controlled and users of such systems are usually considered safe and only submitted to host crash failures (typically, Byzantine failures are not considered).

Peer-to-peer overlay network : Generally, a peer-to-peer (or P2P) computer network is any network that does not rely on dedicated servers for communication but, instead, mostly uses direct connections between clients (peers). A pure peer-to-peer network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both "clients" and "servers" with respect to the other nodes on the network.

This model of network arrangement differs from the client-server model where communication is usually relayed by the server. In a peer-to-peer network, any node is able to initiate or complete any supported transaction with any other node. Peer nodes may differ in local configuration, processing speed, network bandwidth, and storage capacity.

Different peer-to-peer networks have varying P2P overlays. In such systems, no assumption can be made on the behavior of the host and Byzantine behavior has to be considered.

Regal is interested in how to adapt distributed middleware to these large scale configurations. We target Grid and Peer-to-peer configurations. This objective is ambitious and covers a large spectrum. To reduce its spectrum, Regal focuses on fault tolerance, replication management, and dynamic adaptation.

Basically, Regal proposes the use of *reactive replication* to tolerate faults and to reduce the access time to get data, while adapting dynamically to the environmental constraints and the evolution of application behavior.

We concentrate on the following research themes:

Data management: the goal is to be able to deploy and locate effectively data while maintening the required level of consistency between data replicas.

System monitoring and failure detection: we envisage a service providing the follow-up of distributed information. Here, the first difficulty is the management of a potentially enormous flow of information which leads to the design of dynamic filtering techniques. The second difficulty is the asynchronous aspect of the underlying network which introduces a strong uncertainty on the collected information.

Adaptive replication: we design parameterizable techniques of replication aiming to tolerate the faults and to reduce information access times. We focus on the runtime adaptation of the replication scheme by (1) automatically adjusting the internal parameters of the strategies and (2) by choosing the replication protocol more adapted to the current context.

The dynamic adaptation of application execution support: the adaptation is declined here to the level of the execution support (in either of the high level strategies). We thus study the problem of dynamic configuration at runtime of the low support layers.

4. Application Domains

4.1. Application Domains

Keywords: *Internet services*, *data storage*, *data-sharing*, *multi-agent systems*.

As we already mentioned, we focus on two kinds of large scale environments : computational grids and peer-to-peer (P2P) systems. Although both environments have the same final objective of sharing large sets of resources, they initially emerged from different communities with different context assumptions and hence they have been designed differently. Grids provide support for a large number of services needed by scientific communities. They usually target thousands of hosts and hundreds of users. Peer-to-peer environments address millions of hosts with hundreds of thousands of simultaneous users but they offer limited and specialized functionalities (file sharing, parallel computation).

In peer-to-peer configurations we focus on the following applications :

- Internet services such as web caches or content distribution network (CDN) which aim at reducing the access time to data shared by many users,
- Data storage of mutable data. Data storage is a classical peer-to-peer application where users can share documents (audio and video) across the Internet. A challenge for the next generation of data sharing systems is to provide update management in order to develop large cooperative applications.

In Grid configurations we address resource management for two kinds of applications :

- Multi-agent applications which model complex cooperative behaviors.
- Application Service Provider (ASP) environments in cooperation with the DIET project of the GRAAL team.

Our third application domain is based on data sharing. Whereas most work on P2P applications focuses on write-once single-writer multiple-reader applications, we consider the (more demanding) applications that share mutable data in large-scale distributed settings. Some examples are co-operative engineering, collaborative authoring, or entreprise information libraries: for instance co-operative code development tools or decentralised wikis. Such applications involve users working from different locations and at different times, and for long durations. In such settings, each user *optimistically* modifies his private copy, called a replica, of a shared datum. As replicas may diverge, this poses the problem of reconciliation. Our research takes into account a number of issues not addressed by previous work, for instance respecting application semantics, high-level operations, dependence, atomicity and conflict, long session times, etc.

5. Software

5.1. Pastis : A peer-to-peer file system

Participants: Pierre Sens [correspondent], Jean-Michel Busca, Fabio Picconi.

Pastis is a distributed multi-writer file system. It aims at making use of the aggregate storage capacity of hundreds of thousands of PCs connected to the Internet by means of a completely decentralized peer-to-peer (P2P) network. Replication allows persistent storage in spite of a highly transient node population, while cryptographic techniques ensure the authenticity and integrity of file system data.

Routing and data storage in Pastis are handled by the Pastry routing protocol and the PAST distributed hash table (DHT). The good locality properties of Pastry/PAST allow Pastis to minimize network access latencies, thus achieving a good level of performance when using a relaxed consistency model. Moreover, Pastis does not employ heavy protocols such as BFT (Byzantine Fault Tolerance), like other P2P multi-writer file systems do. In Pastis, for a file system update to be valid, the user must provide a certificate signed by the file owner which proves that he has write access to that file.

5.2. LS3 : Large Scale Simulator

Participants: Pierre Sens [correspondent], Jean-Michel Busca.

LS3 is a discrete event simulator originally developped for Pastis, a peer-to-peer file system based on Pastry. LS3 allows to build a network of tenths of thousands nodes on a single computer, and simulate its execution by taking into account message transmission delays. LS3 transparently simulates communication layers between nodes, and executes the same application code (including Pastry, Past and higher layers) as in a real execution of the system.

LS3's modular design consists of three independent layers, allowing the simulator to be reused in areas other than Pastis and Pastry:

- At the kernel level, the system being simulated is described in a generic way in terms of entities triggered by events. Each entity has a current state and a current virtual time, and can be programmed either in synchronous mode (blocking wait of the next event) or in asynchronous mode (activation of an event handler). A multi-threaded event engine delivers events in chronological order to each entity by applying a conservative scheduling policy, based on the analysis of event dependencies.
- At the network level, the system being simulated is modeled in terms of nodes sending and receiving messages, and connected through a network. The transmission delay of a message is derived from the distance between the sending and the receiving nodes in the network, according to the selected topology. Three topologies can be used: local network (all nodes belong to the same local network), two-level hierarchy (nodes are grouped into LANs connected through WANs) and sphere (nodes are located on a sphere). It is possible to set the jitter rate of transmission delays, as well as the rate of message loss in the network.
- The stubs Pastry level interfaces Pastry with LS3: it defines a specialization of Pastry nodes that allows them to interface with standard LS3 nodes. Several parameters and policies that drive the behaviour and the structure of a Pastry network can be set at this level, including: the distribution of node ids, the selection of boostrap nodes, the periodicity of routing tables checks and the rate of node churn. It is also possible to simulate the ping messages that nodes send to supervise each other, and set failure detection thresholds.

Some figures: LS3 can simulate a network of 20 000 Pastry nodes with no application within 512 Mb of RAM, and it takes approximately 12 minutes on a single processor Pentium M 1,7 GHz to build such network. When simulating the Pastis application, event processing speed is about 500 evt/s. The speedup factor depends on the simulated load: as an example, speedup ranges from 20 for a single user to 0,05 for 400 simultaneous users.

6. New Results

6.1. Introduction

In 2006, we focused our research on the following areas :

- distributed algorithms for Grid configuration,
- impact of churn is peer-to-peer data storage,
- dynamic adaptation of virtual machines,
- services management in large scale environments,
- Formal and practical study of optimistic replication, incorporating application semantics.
- Decentralised commitment protocols for semantic optimistic replication.
- dynamic replication in distributed multi-agent systems.
- self* systems

6.2. Distributed algorithms

Participants: Luciana Arantes [correspondent], Mathieu Bouillaguet, Pierre Sens, Julien Sopena.

Our current research focuses on adapting distributed algorithms to large-scale and heterogeneous environments. We target particularly three basic blocks : mutual exclusion algorithms, resource allocation and failure detection.

- In mutual exclusion algorithm, we have been interested in scalability, fault-tolerance and latency tolerance aspects of distributed mutual exclusion algorithms.

  The majority of current distributed mutual exclusion algorithms are not suited for distributed applications on top of large-scale or heterogeneous systems such as Grid or peer-to-peer systems since they do not consider the latency heterogeneity of the platform or scalable fault tolerance mechanisms. Exploiting these points, we have proposed two distributed mutual exclusion algorithms, based on Naimi-Trehel's token-based algorithm: the first on takes into account latency gaps, especially those between local and remote clusters of machines in a Grid environment while the second one provides scalable fault tolerance feature by minimizing the use of broadcast support. It exploits the distributed queue of token requests to offer fault tolerance support. Both algorithms have been published in recent conferences [29], [28] and in a JPDC journal [14].

- Failure detectors are well-known as a basic building block of fault-tolerant distributed systems. A unreliable failure detector can be informally considered as an oracle per process. Such an oracle provides the set of processes that it currently suspects of having crashed. Many applications use a failure detection service in order to solve the consensus problem. Typically, failure detectors use messages sent periodically between hosts. Most of implementations are based on all-to-all communication where each process periodically sends an I-am-alive message to all processes in order to inform them that it is still alive, and thus resulting in a quadratic number of message to be periodically send.

  We previously proposed a scalable implementation of failure detector (called GFD for *Grid Failure Detector*) [4] taking into account the network topology. In [27], consider the problem of failure detection in dynamic network such as MANET or Peer-to-peer overlay. Unreliable failure detectors (FD) are classical mechanisms providing information about process failures. They allow to solve consensus in asynchronous network. However, most of implementations consider a set of known processes fully connected by reliable links. Such an assumption is not applicable in dynamic environments. Furthermore, the majority of current failure detector implementation are timer-based ones while in dynamic networks there is not an upper bound for communication delays.

We propose an asynchronous implementation of a failure detector for dynamic environments. Our implementation is an extension of the query-response algorithm previously proposed by Mostefaoui, Mourgaya and Raynal which does not rely on timers to detect failures. We assume that neither the identity nor the number of nodes are initially known. We also proof that our algorithm can implement failure detectors of class $\diamondsuit S$ when some behavioral properties are satisfied by the underlying system. Simulation results have validated our approach.

6.3. Impact of churn and in peer-to-peer data storage

Participants: Pierre Sens [correspondent], Jean-Michel Busca, Fabio Picconi.

Since 2003, we develop Pastis [5] is a new completely decentralized multi-user read-write peer-to-peer file system. Pastis is based on the FreePastry Distributed Hash Table (DHT) of the Rice University. DHTs provide a means to build a completely decentralized, large-scale persistent storage service from the individual storage capacities contributed by each node of the peer-to-peer overlay

However, persistence can only be achieved if nodes are highly available, that is, if they stay most of the time connected to the overlay. Churn (i.e., nodes connecting and disconnecting from the overlay) in peer-to-peer networks is mainly due to the fact that users have total control on theirs computers, and thus may not see any benefit in keeping its peer-to-peer client running all the time. This is very common in existing peer-to-peer file sharing networks, as many users connect to the overlay to download a particular file, and disconnect soon after the download has finished. Although intermittent connections are not particularly harmful in file sharing networks, this kind of unstable user behavior is undesirable on DHTs. Contrary to file sharing systems, DHTs are designed to guarantee data persistence. This is achieved by replicating data blocks on geographically dispersed nodes, which minimizes the probability of correlated failures, and by regenerating replicas as soon as they leave the network so that the replication factor is kept constant. This reduces the risk of data becoming unavailable if all replicas leave the network, but it also means that as nodes join and leave the network the DHT maintenance algorithm needs to transfer a large number of replicas from one node to another, consuming a lot of bandwidth. Furthermore, DHTs clients lack any flexibility to choose where their data is stored in the overlay.

Since 2005 we study the effect of churn on Pastis and more generally on DHT. We propose a new incentives-based mechanism to increase the availability of DHT nodes, thereby providing better data persistence for DHT users. High availability increases a node's reputation, which translates into access to more DHT resources and a better Quality-ofService. The mechanism required for tracking a node's reputation is completely decentralized, and is based on certificates reporting a node's availability which are generated and signed by the node's neighbors. An audit mechanism deters collusive neighbors from generating fake certificates to take advantage of the system. Prelimary results of this proposal has be published in the HotP2P workshop [26].

Recently, we consider the problem of data durability in wide-area distributed storage systems, such as Distributed Hash Tables (DHTs). Given the limited bandwidth between replicas, these systems suffer from long repair times after a hard disk crash, making them vulnerable to data loss when several replicas fail within a short period of time. Recent work has suggested that the probability of data survival can be predicted by modeling the number of live replicas using a Markov chain. However, the prediction accuracy relies on a realistic estimation of the model parameters, that is, the chain transition rates from one state to another. In [25], we focus on obtaining a good approximation of the Markov chain parameters used to predict data durability. We present a new set of expressions for these parameters based on a theoretical analysis and on empirical data. We argue that these parameters produce a conservative estimation of the probability of object survival. In other words, our expression allows the system designer to choose the appropriate replication factor to guarantee a minimum level of data durability. To the best of our knowledge, this is the first work that focuses on this parameter estimation problem.

6.4. Self* properties of dynamic systems

Participant: Maria Gradinariu [correspondent].

Dynamic systems (eg. P2P networks, sensor networks or robots networks) share several common caracteristics such as mobility, communication constraints, no centralized authority and faulty prone behavior. In P2P networks, nodes join and leave the network at any time while in sensor networks nodes may stop participating to the welfare of the network due to their limited energy or various scenario of faults (eg. crashes while the nodes are deployed, un-friendly environments). Moreover nodes may "join" the network when new nodes are deployed or the batteries of already deployed nodes are recharged. Robots, in robots networks, embed motion capabilities which have an impact on network topology or neighborhood relationship. Therefore when such networks are deployed nodes have to embed self-healing capabilities in order to be able to detect or to repair the network whenever existing nodes leave the system or new nodes join.

In P2P networks, since these networks may have millions of participators it's impossible or innefficient to maintain a direct communication (one hop communication) between all nodes of the system. Therefore nodes dispose of local views of the hole network. These views are further used in order to implement multihop communications. The communication via local views is also present in sensors or robots networks imposed by the physical restrictions such as limited communication ranges. Any application on top of such networks has to maintain coherent local views in a self-organized manner since no centralized authority can be used as reference. Moreover, in order to economize the local resources such as batteries or bandwidth only a subset of nodes are supposed to be active. Therefore, nodes have to self-organize in order to deploy intelligent overlays aimed to economize the system resources or to reduce the communication cost between interested parties.

In dynamic networks the applications have to well function despite the dynamicity of nodes therefore in these networks wait-freedom becomes one of the concepts the most seducing [1] In other words a node(process) in a wait-free system executes its task independently of the behavior or the speed of other nodes in the system. The applications designed for dynamic systems have to be wait-free, self-stabilizing (tolerant to various corruptions of nodes memory or counter program) and tolerant to Byzantine behavior.

Self* Overlays and their applications: Overlays in dynamic systems have various roles: they help in economizing the network ressources by activating only a subset of nodes or they may be dimensioned in order to optimize data agregation or diffusion. In [17] we propose correctness proofs and complexity analysis for the first self-stabilizing constructions of connected overlays for wireless networks (eg. MANETs, WSN) based on the computation of *Connected Dominating Set*(CDS). The basic idea is to construct an overlay that contains a small number of nodes, but still obtain full connectivity of the network while only relying on local exchanges of information and knowledge. We adopt two methodologies of construction: the first methodology consists of two parallel tasks, namely, computing a *maximal independent set* (MIS) and then adding bridge nodes between the MIS nodes. The second methodology computes a connected dominating set using the observation that a dominator is a bridge between nodes that do not share the same neighborhood. The proposed algorithms are fully decentralized and are designed in a self-stabilizing manner in order to cope with transient faults, mobility and nodes join/leave. In particular, they do not need to be (re)initialized after a fault or a physical topology change. That is, whatever the initial configuration is, the algorithms satisfy their specification after a stabilization period. The convergence time of our algorithms is linear in the size of the network and they use only one extra bit of memory.

In [16] we have proposed a stabilizing overlay for P2P networks optimized to reduce the false positives in publish/subscribe systems. Our overlay is based on the extension of R-trees. R-trees are well known indexing structures specially designed to support spatial database queries. We propose a distributed self-stabilizing implementation of an R-tree-like overlay that copes with nodes dynamicity (frequent joins and leaves) and memory and counter program corruptions. The maintenance of this structure is local and no additional memory cost is needed for guaranteeing its stabilization. Additionally, we propose an application of the designed structure to Complex Content-based filtering in Publish/Subscribe systems. These systems provide a useful platform for delivering data (events) from publishers to subscribers in a decoupled fashion in distributed networks. Publish/Subscribe systems have many applications, including web services, stock quotes,

[1] A system of size n is called wait-free if it respects its specification despite the crash of at most $n-1$ nodes.

alerts monitoring, free riding monitoring, and Internet games. Developing efficient publish/subscribe schemes for complex subscriptions (subscriptions spanning multi-dimensional intervals) in dynamic distributed systems is challenging nowadays. The use of dynamic self-stabilizing R-trees in this context offers an efficient balanced structure and provide guaranties in terms of fault-tolerance, response time and storage space. That is, subscribers and publishers can connect disconnect at any time. Moreover, their memory and counter program can be corrupted. Additionally, the worst subscription or publication time is $O(log_m N)$ and the extra storage space needed at each node for maintaining the infrastructure is $O(log(Mlog_m(N)))$ bits where M and m are two parameters defining the minimal respective the maximal degree of a node in the tree.

Robots networks are systems formed of mobile entities that self-organize in order to achieve specific tasks (eg. exploration, carring/pushing boxes). Robots, in order to realize a specific task have to form, in general, a specific geometric structure (eg. cercles, rectangles, lines etc etc). Most of these problems can be easilly solved once robots agree on a common point in the space (this can be further used as the origin of a coordinate system). Agreeing on a common point in the space can be achieved via gathering (i.e. robots meet at a common point in a finite number of steps). The gathering becomes challenging in networks with stateless robots (robots that cannot recall the past). In this context we have studied in [18] the possibility and impossibility of robots gathering in both fault-free and fault-prone environements. The proposed algorithms are also self-stabilizing since robots are stateless.

Managed Agreement: In [13] we introduce a family of agreement problems called *Managed Agreement*, which is parameterized by the number of *aristocrat* nodes in the system; NBAC is a special case of this family when all nodes are aristocrats while Consensus is a special case of this family when there are no aristocrats. The paper also presents a parameterized family of failure detectors $\mathcal{F}(A)$ such that $\mathcal{F}(A)$ is the weakest failure detector class that enables solving Managed Agreement with a set A of aristocrats in an asynchronous environment.

Self-stabilizing Algorithms: In [23], we specify the conflict manager abstraction. Informally, a conflict manager guarantees that any two conflicting nodes cannot enter their critical simultaneously (safety), and that at least one node is able to execute its critical section (progress). The conflict manager problem is strictly weaker than the classical local mutual exclusion problem, where any node that requests to enter its critical section eventually does so (fairness). We argue that conflict managers are a useful mechanism to transform a large class of self-stabilizing algorithms that operate in an essentially sequential model, into self-stabilizing algorithm that operate in a completely asynchronous distributed model. We provide two implementations (one deterministic and one probabilistic) of our abstraction, and provide a composition mechanism to obtain a generic transformer. Our transformers have low overhead: the deterministic transformer requires one memory bit, and guarantees time overhead in order of the network degree, the probabilistic transformer does not require extra memory. While the probabilistic algorithm performs in anonymous networks, it only provides probabilistic stabilization guarantees. In contrast, the deterministic transformer requires initial symmetry breaking but preserves the original algorithm guarantees.

6.5. Virtual virtual machine (VVM)

Participants: Bertil Folliot [correspondent], Charles Clement, Nicolas Goeffray, Assia Hachichi, Cyril Martin, Gaël Thomas.

The VVM group works on flexible execution environments, based on a HAL (Hardware Abstraction Layer) and a flexible dynamic compiler, free of any predefined, imposed abstractions.

In 2006, VVM group focused on the following themes:

Adaptive language for middleware: Today, component oriented middlewares are used to design, develop and deploy distributed applications easily. They ensure the heterogeneity, interoperability, and reuse of software modules. Several standards address this issue: CCM (CORBA Component Model), EJB (Enterprise Java Beans) and .Net. However they offer a limited and fixed number of system

services, and their deployment and configuration mechanisms cannot be used by any language nor API dynamically. As a solution, we propose a generic high-level language to adapt system services dynamically in existing middlewares. This solution is based on a highly adaptable platform which enforces adaptive behaviours, and offers a means to specify and adapt system services dynamically. Experiments show that the language can adapt the OpenCCM platform and the JONAS plateform through a complete prototype of execution engine based on the VVM. A PhD was presented by Assia Hachichi on December on this subject.

Dynamic Bytecode offloading in a network of workstations: We propose a new runtime infrastructure able to transparently distribute computation between interconnected workstations. Application source code is not modified: instead, dynamic aspect weaving within an extended virtual machine allows to monitor and distribute entities dynamically. Applications that benefit from our system can be (i) applications that execute on resource constrained devices, and will offload to nearby computers, or (ii) large scale applications, where dynamic load-balancing during high load will offer a guarantee of service. The platform developed is an extension of a standard and generic Java virtual machine, the JNJVM, that provides abstractions to modify on the fly the behavior of the runtime. Our offloading system is thus loaded only when the application needs it. Existing applications can be used, and the offloading system can be deployed after launching time, without interrupting the application, even if the administrator has not planned to inject this non functional behavior. A complete evaluation of the offloading system on a cluster of 20 hosts shows that our technique can absorb an overload in a web server.

Migration and process image capture: We propose a system to capture the image of a running processes during its execution by using reflexion mechanisms at the flexible dynamic compiler level of the VVM. Our system is generic and reusable for each execution environment built upon the VVM: the JnJVM (our Java Virtual Machine built upon the VVM) uses this new system services to migrate threads. Domains that benefits from the capture system are: (i) process migration by moving the image on a new node of the network, (ii) distribution process check-pointing for fault tolerance by using captured images to restart the application in a consistent state, (iii) memory and performance optimisations by applying optimizers on the image, which is initialized and can't reach all possible state that were possible before initialisation. The process image capture is achieved and we are now working on the applications for the three domains.

6.6. Services management in large-scale environments

Participants: Mesaac Makpangou [correspondent], Ikram Chabbouh, Corina Ferdean, Nicolas Gibelin.

The research topic we are interested in is autonomic replication management systems in the context of the Internet. We target the replication of Web applications and services (e.g. web services). In 2006, we consolidate and extend our previous work on three aspects.

Response time driven replica selection: We consider a replicated service, consisting of a known set of replicas, spread all over the world. In such a context, each replication system must answer the challenging questions: which is the suitable replica to be assigned to each client and how to find that replica efficiently, in a large scale environment? Existing replica selection systems rely on metrics that are correlate with the demands of a particular resource (cpu, disk bandwidth, network latency and bandwidth).

We proposed a response time-driven replica selection approach, which relies on a demand-aware response time estimator. That is, the suitable replica is one for which the estimated response time is either less or equal to a given threshold, or is the smallest [10]. The proposed estimator takes into account the real resource demands and the current resource utilization. The estimator computes independently the CPU service time, disk I/O service time, CPU waiting time, disk I/O waiting time and network transfer time. By using a few baseline measures of the CPU (resp. IO) waiting time, for a few workloads, under a few CPU ((resp. IO)) utilization measures, we are able to estimate the

CPU ((resp. IO)) waiting time for any workload, under any utilization value. The experiments we conducted show good correlation between the estimated response time and the measured one [19].

Flexible consistency management system: A large spectrum of replica consistency models are proposed by existing consistency management systems. Each replica consistency model (i.e. replica consistency management system) captures a trade-off suitable for a certain class of replicated services (e.g. resource allocation, collaborative edition, and web applications). However, for programmers (or operators) in charge of several services that can benefit from different consistency management trade-offs, none of the existing consistency management systems is really satisfying. Such operators have either to learn to use several consistency management systems/middlewares, or to implement a middleware capable of offering the set of trade-offs they need.

We developped a flexible replica consistency management middleware capable to tailor the trade-off enforced for a service to the specifications of its provider. Requestable trade-offs include thoses proposed by existing replica consistency models, plus any sound and novel ones. These trade-offs are offered, thanks to a generic and customizable replica consistency manager attached to each replica of the service. For each invocation requested by a client, the behaviour of the group of consistency managers (attached to the replicas of the replicated service) depends on the trade-off requested by this service provider for this invocation. For that, each consistency manager is configured with a service-specific contract object that is queried, at run-time, to determine the trade-off to satisfy for each access. Services providers specify their requirements in a XML-based specific language. A compiler interprets the specifications, then generates the contract object used to configure the consistency managers.

Autonomic replication of web applications: Content Delivery Networks enable rapid and reliable retrieval of Web pages from any end-user location by serving the content to clients from nearby servers. The main value of a CDN is its ability to serve the dynamic content which cannot be cached. Currently, CDNs serve the dynamic content in two major ways: they either assemble at the edge of the network the fragments of a page that have been generated by the origin servers, or they compute the page on the edge by replicating the application's code and possibly data on the edge servers. Page assembly assumes that temporal locality of requests is high and database changes are infrequent, while edge computing generally assumes that the whole application is replicated on the edge which is not always suitable.

In our latest research we have proposed an hybrid CDN called FRACS which combines both HTML and application fragments caching. First, FRACS transforms the web application code accordingly so as to enable the origin server to generate fragmented Web pages in ESI format and to serve the fragments separately. Individaul HTML fragments (fetched separately) can be cached at the edge servers. Secondly, FRACS determines the fragments of a web application that can be served from the edge and generates replication scripts which are used by the edge servers in order to replicate these application fragments. Finally FRACS provides support to ensure the freshness of cached fragments as well as the consistency of replicated application fragments. Using the TPC-W benchmark we show that FRACS is able to achieve up to 60% savings in bandwidth and more than 80% reduction in response time.

6.7. Optimistic replication, reconciliation and commitment

Participants: Marc Shapiro [correspondent], Lamia Benmouffok, Pierre Sutra, Maria Gradinariu.

In distributed systems, information is replicated. Data replication enables cooperative work, improves access latency to data shared through the network, and improves availability in the presence of failures. When the information is updated, maintaining consistency between replicas is a major challenge.

Previous studies of data replication considered different areas separately, often ignoring the requirements of other areas. For instance, OS researchers often assume updates are independent; CSCW researchers ignore conflicts; algorithms research mostly ignores semantics; peer-to-peer systems often ignore mutable data and hence consistency; none of the above have addressed partial replication.

We study optimistic replication for multi-user collaborative applications such as co-operative engineering (e.g., co-operative code development), collaborative authoring (e.g., a decentralised wikipedia), or entreprise information libraries. We propose a general-purpose approach, subsuming the previous work in different areas. It takes addresses respecting application semantics, high-level operations, dependence, atomicity and conflict, long session times, etc.

Earlier, we proposed the Action-Constraint Framework (ACF) formalism to reify application semantics and to reason about reconciliation. ACF represents user intents and application semantics as constraints i.e., invariants between actions (operations) that each local scheduler must maintain. Consistency is defined abstractly by two properties, mergeability (safety) and eventual decision (liveness). Our IceCube reconciliation engine used ACF to select promising reconciliation combinations and propose them to the user. However, IceCube was not compatible with a peer-to-peer approach, because it assumed that there is a primary reconciliation site. Decentralised reconciliation algorithms exist, but they ignore semantic relations.

Therefore, in 2006, we took a different appraoch. Any number of "proposer" peers make reconciliation proposals (computed using IceCube, some other algorithm, or even manually). Any number of "acceptor" peers receive the proposals, which can be broken into well-formed granules called candidates. Acceptors exchange votes regarding the candidates. Candidates that have certain common characteristics compete in an election. The winner of an election is the candidate with the highest number of votes (not necessarily a majority). Every acceptor eventually unambiguously learns the outcome of each election. The protocol is completely decentralised and asynchronous. In the presence of crashes, it remains safe, and is live as long as a sufficient number of votes remain available. This work was submitted for publication [31].

During 2006, we also we worked on building Telex, a general-purpose peer-to-peer middleware, based on ACF and IceCube and a successor to our previous platform (Joyce). Telex will serve both as an experimental testbed for our research on optimistic replication, and as an implementation platform for co-operative applications. Telex is designed as a general-purpose consistency layer on top of a distributed file system such as Pastis. The main abstraction of Telex is the multilog, which encapsulates both a multicast communication channel between peers that are co-operating over some document, and persistent storage of actions, constraints, and versions. The multilog model offers unprecedented flexibility, allowing each user to tailor a local view of shared information.

We have also started addressing issues such as partial replication and improved fault tolerance. The work described above takes place in the context of several joint projects: Grid4All, Respire and Recall.

6.8. Safe concurrent programming

Participant: Marc Shapiro [correspondent].

Modern computer architectures are increasingly parallel: viz., clusters and multi-core PCs. More and more developers will be seduced into concurrent programming, unprepared for the difficulties of understanding, writing and debugging concurrent programs. Proposed higher-level abstractions (such as lightweight transactions) may provide the illusion that concurrency is easy, but there is a fundamental theoretical issue: threads can interfere with one another in arbitrary ways; the number of cases is combinatorial.

In practice however, reasonable programs have concurrency control disciplines (e.g., locking) that avoid the bad interactions. We propose to formalise this concurrency control and to leverage it, in order to reason in a modular fashion and side-step the combinatorial explosion. To this effect, we use the "rely-guarantee" (R-G) approach [7]. In addition to the standard pre- and post-conditions of sequential Hoare logic, a program is equipped with a non-interference assertions: a *rely* condition limits the interference it may suffer from its environment; a *guarantee* condition specifies what interference it may inflict on its environment. If the rely condition of any particular thread is implied by all other threads' guarantee conditions (and if certain technical conditions are met), then standard sequential reasoning can be used to prove the post-condition.

We describe some extensions to the basic R-G approach to make it practical. As an example, we study a family of implementations for linked lists using fine-grain synchronisation. This approach enables greater concurrency, but correctness is a greater challenge than for classical, coarse-grain synchronisation. Our examples are demonstrative of common design patterns such as lock coupling, optimistic, and lazy synchronisation. Although they are highly concurrent, we prove that they are linearisable, safe, and they correctly implement a high-level abstraction.

With this work, we learned that the extended R-G approach is a powerful and intuitive engineering tool, because (i) it confines correctness reasoning to a single thread or module at a time and (ii) R-G proofs actually explain why a piece of code is correct. We also learned that the crucial difficulty is in extracting appropriate invariants, and not in developing the proof itself.

This work appeared in Principles and Practice of Parallel Programming (PPoPP 2006).

6.9. Dynamic replication in multi-agent systems

Participants: Olivier Marin [correspondent], Pierre Sens.

Distributed agent systems stand out as a powerful tool for designing scalable software. The general outline of distributed agent software consists of computational entities which interact with one another towards a common goal that is beyond their individual capabilities. There are many varying definitions of the notion of software agent. The main common characteristics are: (a) the possession of individual goals, resources and competences, (b) the ability to perceive and to act, to some degree, on the near environment; this includes communications with other agents, (c) the faculty to perform actions with some level of autonomy and/or reactiveness, and eventually to replicate, (d) and the capacity to provide services.

The above-mentioned properties induce that the agent paradigm is also very suitable for building adaptive applications, where interactions amongst the different entities involved may be altered during the course of computation, and where this change must have an impact on the software behaviour. This is all the more important in scalable systems because environment characteristics can vary a lot from one vicinity of the network to another.

However, as the amount of locations and agents within the system increases, so does the probability of failures. As mentioned earlier, multi-agent applications rely on collaboration amongst agents. It follows that the failure of one of the involved agents can bring the whole computation to a dead end. It is therefore crucial to apply dependability protocols when distributing agent systems over large-scale networks.

Replicating agents seems to be the only efficient way to achieve fault tolerance in scalable agent systems. A replicated software component is defined as a software component that possesses a representation on two or more hosts. Consistency between replicas can be maintained following two main strategies: the active one in which all replicas process all input messages concurrently, and the passive one in which only one of the replicas processes all input messages and periodically transmits its current state to the other replicas. Each type of strategy has its advantages and disadvantages. Active replication is dedicated to applications which require full recovery over short delays. Passive replication makes for lower overhead in a failure-free environment but recovery is less efficient. The choice of the most suitable strategy is directly dependent of the environment context, especially the failure rate, the kind of failure that must be tolerated, and the application requirements in terms of recovery delay and overhead.

In practice, replicating every agent over a large-scale network would lead to excessive overheads, both in terms of network load and in terms of computation time. Besides, at any given point of the computation, only a small subset of agents is likely to be crucial to its continuation. Therefore only the specific agents which are temporarily identified as crucial to the application should be replicated, and the applied strategy should be carefully selected.

Our work serves a twofold objective: (1) to provide efficient fault-tolerance to multi-agent systems through selective agent replication, (2) to take advantage of the specificities of multi-agent platforms to develop a suitable architecture for performing adaptive fault-tolerance within distributed applications; such applications would then be liable to operate efficiently over large-scale networks.

Our project has lead to the design and development of DARX, an architecture for fault-tolerant agent computing. As opposed to the main conventional distributed programming architectures, ours offers dynamic properties: software elements can be replicated and unreplicated on the spot and it is possible to change the current replication strategies on the fly. More importantly, DARX allows to automate its fault tolerance support for agent applications with respect to the behaviour of their runtime environment.

The originality of our approach lies in two features: (i) the possibility for applications to automatically choose which computational entities are to be made dependable, to which degree, and at what point of the execution, and (ii) the hierarchic architecture of the middleware which ought to provide suitable support for large-scale applications.

DARX consists of several services. A failure detection service maintains dynamic lists of all the running hosts as well as of the valid software elements which participate to the supported application, and notifies the latter of suspected failure occurrences. A naming and localisation service generates a unique identifier for every agent in the system, and returns the addresses for all agent replicas in response to an agent localisation request. A system observation service monitors the behaviour of the underlying distributed system: it collects low-level data by means of OS-compliant probes and diffuses processed trace information so as to make it available for the adaptive replication control process. An application analysis service builds a global representation of the supported agent application in terms of fault tolerance requirements. A replication service brings all the necessary mechanisms for replicating agents, maintaining the consistency between replicas of a same agent, and automating replication scheme adaptation for every agent according to the data gathered through system monitoring and application analysis. An interfacing service offers wrapper-making solutions for agents, thus rendering the DARX middleware usable by various multi-agent systems and even making it possible to introduce interoperability amongst different systems.

Our current prospective work addresses the following issues: - Scalable monitoring: we are looking into how to extract and aggregate useful information about the computing environment, and how to diffuse this information in a timely manner to the locations that require it. - Fault-tolerance by contract: we are working on the establishment of contracts between DARX and its supported applications that satisfy both the application requirements and the environment constraints. - Reliability scheme validation: we are designing a modeling language which is close to Pi-calculus and yet focuses on communication channels in order to represent replicated applications and introduce failures. This will enable us to test the validity of applied replication schemes over DARX. This work is part of FRAME, a project funded by the LIP6.

7. Other Grants and Activities

7.1. National initiatives

7.1.1. Grid Data Service - (2003–2006)

Members: IRISA (Paris Team), ENS-Lyon (LIP - Remap Team), Regal

Funding: GDS project is funded by ACI MD

Objectives: The goal of this project is to propose an approach where the grid computation is decoupled from data management, by building a data sharing service adapted to the constraints of scientific grid computations. The main goal of this project is to specify, design, implement and evaluate a data sharing service for mutable data and integrate it into the DIET ASP environment developed by the ReMaP team of LIP. This service will be built using the generic JuxMem platform for peer-to-peer data management (currently under development within the PARIS team of IRISA, Rennes). The platform will be used to implement and compare multiple replication and data consistency strategies defined together by the PARIS team (IRISA) and by the REGAL team of LIP6.

7.1.2. Data Grid eXplorer (2003–2006)

Members: IMAG-ID, Laria, LRI, LAAS, LORIA, LIP Ens-Lyon, LIFL, INRIA Sophie Antipolis, LIP6, IBCP, CEA, IRISA INRIA Rocquencourt

Funding: DGX project is funded by ACI MD

Objectives: The goal of Data Grid Explorer is to build an emulation environment to study large scale configurations. Today, it is difficult to evaluate new models for data placement and caching, network content distribution, peer-to-peer systems, etc. Options include writing simulation environments from scratch, employing detailed packet-level simulation environments such as NS, local testing within a controlled cluster setting, or deploying live code across the Internet or a Testbed. Each approach has a number of limitations. Custom simulation environments typically simplify network and failure characteristics. Packet-level simulators add more realism but limit system scalability to a few hundred of simultaneous nodes. Cluster-based deployment adds another level of realism by allowing the evaluation of real code, but unfortunately the network is highly over-provisioned and uniform in its performance characteristics. Finally, live Internet and Testbed deployments provide the most realistic evaluation environment for wide-area distributed services. Unfortunately, there are significant challenges to deploying and evaluating real code running at a significant number of Internet sites. The main benefit of emulation is the ability to reproduce experimental conditions and results.

The project is structured horizontally into transverse working groups: Infrastructure, Emulation, Network, and Applications. The Regal team is leader for the Emulation working group.

7.1.3. Gedeon - ACI MD (2004-2007)

Members: IMAG-ID, IMAG-LSR, IBCP, Regal

Funding: Gedeon project is funded by ACI MD

Objectives: File systems (FS) are commonly used to store data. Especially, they are intensively used in the community of large scientific computing (astronomy, biology, weather prediction) which needs the storage of large amouts of data in a distributed manner. In a GRID context (cluster of clusters), traditional distributed file systems have been adapted to manage a large number of hosts (like the Andrew File System). However, such file systems remain inadequate to manage huge data. They are suited for traditional unix (small) files. Thus, the grain of distribution is typically an entire file and not a piece of file which is essential for large files. Furthermore, the tools for managing data (e.g, interrogation, duplication, consistency) are unsuited for large sizes.

Database Management Systems (DBMS) provides different abstraction layers, high level languages for data interrogation and manipulation etc. However, the imposed data structuration, the low distribution, and the usually monolithic architecture of DBMSs limit their utilisation in the scientific computing context.

The main idea of the Gedeon project is to merge the functions of file systems and DBMS, focusing on structuration of meta-data, duplication and coherency control. Our goal is NOT to build a DBMS describing a set of files. We will study how database management services can be used to improve the efficiency of file access and to increase the functionality provided to scientific programmers.

7.1.4. Respire - (2005–2008)

Members: LIP6, Atlas (IRISA), Paris (IRISA), Regal

Funding: RESPIRE project is funded by ANR (ARA MDSA)

Objectives: The Respire project aims to develop support for sharing information (including either data or meta-data) and services for managing this information in a Peer-to-Peer (P2P) environment. A P2P architecture is potentially more scalable than previous client-server approaches, but raises many interesting scientific issues. Peers are autonomous and may join or leave the network at any time. Peers publish resources for sharing, such as data or services, and may use resources

published by other peers. Users may collaborate without any explicit or implicit hierarchy. We target applications that enable world-wide spread professional communities (such as a group of researchers) to collaborate, or learning scenarios. These applications manipulate heterogenous, semantically rich data. Therefore they require more advanced functionality than existing P2P file systems.

Respire is a collaboration between research teams from different areas, distributed databases and distributed systems, which have until now largely ignored each other. This synergetic approach enables each community to question hidden assumptions, and to take into account new approaches and requirements.

7.1.5. Recall - (2005–2008)

Members: LIP6, Atlas (IRISA), Paris (IRISA), Regal

Funding: Recall is funded by INRIA (Action de Recherche Coopérative)

Objectives: Recall aims to develop optimistic replication algorithms for supporting massive collaboartive editing applications. The goal is to enable classical collaborative applications to scale and to tolerate faults, by deploying them above peer-to-peer networks, and without expensive hardware requirements. This project will show that P2P networks are a viable solution, not only for distributing content, but also for creating and editing it.

7.2. European initiatives

7.2.1. Grant from Microsoft Research Cambridge

Data replication enables cooperative work, improves access latency to data shared through the network, and improves availability in the presence of failures. This grant supports a doctoral student for studying consistency between replicas of mutable, semantically-rich data in a peer-to-peer fashion. This study should enable to engineer distributed systems and applications based on them, supporting cooperative applications in large-scale collaboration networks. It includes a systematic exploration of the solution space, in order to expose the cost vs. performance vs. availability vs. quality trade-offs, and understanding fault tolerance and recovery aspects. This work combines formal approaches, simulation, implementation, and measurement.

7.2.2. Grid4All - (2006-2009)

Members: France Télécom Recherche et Développement, INRIA (Regal, Atlas and Grand-Large), SICS, KTH, ICCS, UPRC, UPC, Redidia.

Funding: European Commission, 6th Framework Programme, STREP (Specific Targeted Research Project)

Objectives: Grid4All embraces the vision of a “democratic” Grid as a ubiquitous utility whereby domestic users, small organizations and enterprises may draw on resources on the Internet without having to individually invest and manage computing and IT resources. This project is funded by the 6th Framework Programme of the European Commission. It involves institutional and industrial partners. Its budget is slightly over 4.8 million euros.

Grid4All has the following objectives:

- To alleviate administration and management of large scale distributed IT infrastructure, by pioneering the application of component based management architectures to self-organizing peer-to-peer overlay services.
- To provide self-management capabilities, to improve scalability, resilience to failures and volatility thus paving the way to mature solutions enabling deployment of Grids on the wide Internet.
- To widen the scope of Grid technologies by enabling on-demand creation and maintenance of dynamically evolving scalable virtual organisations even short lived.

- To apply advanced application frameworks for collaborative data sharing applications executing in dynamic environments.
- To capitalize on Grids as revenue generating sources to implement utility models of computing but using resources on the Internet.

Grid4All will help to bring global computing to the broader society beyond that of academia and large enterprises by providing an opportunity to small organisations and individuals to reap the cost benefit of resource sharing without however the burdens of management, security, and administration.

The consortium will demonstrate this by applying Grid4All in two different application domains: collaborative tools for e-learning targeting schools and digital content processing applications targeting residential users.

7.3. International initiatives

JAIST (Japon). With the group of Prof. Xavier Defago we investigate various aspects of self-organization and fault tolerance in the context of robots networks.

UNLV (SUA) With the group of Prof. Ajoy Datta we collaborate in designing self* solutions for the computations of connected covers of query regions in sensor networks.

Technion (Israel). We collaborate with Prof. Roy Friedman on divers aspects of dynamic systems ranging from the computation of connected covers to the design of agreement problems adequate for P2P networks.

COFECUB (Brazil). With the group of Prof. F. Greve. (Univ. Federal of Bahia), we investigate various aspects of failure detection for dynamic environement such as MANET of P2P systems.

Informal collaboration with INESC. JoÃ£o Barreto, a PhD student at INESC and Instituto Superior Técnico spent six months in Projet Regal, working on ubiquitous information sharing in mobile ad-hoc networks. He took an important role in the design of our decentralised commitment algorithm [31].

8. Dissemination

8.1. Program committees and responsibilities

Luciana Arantes is:

- Member of the “Commission de spécialiste” of the Paris 5 University.

Bertil Folliot is:

- Member of the program committee of The IASTED International Conference on Parallel and Distributed Computing and Networks, PDCN 2006, Innsbruck, Autriche, février 2006.
- Member of the program committee of IFIP Symposium on Computer Architecture and High Performance Computing, Ouro Preto, Brésil, octobre 2006.
- Member of the program committee of 5ème Conférence française sur les systèmes d’exploitation, CFSE-5, Association de l’ACM-SIGOPS de France, Perpignan, octobre 2006.
- Member of the “Commission de spécialistes” of the Paris 6 and Paris 9 Universities.
- Co-chair of the middleware group of GdR ASR

Mesaac Makpangou is:

- Member of the “Commission de spécialistes” of the University of Marne La Vallée.

Olivier Marin is:

- co-editor for IEEE distributed systems on-line in distributed agents topic.

Pierre Sens is:

- Member of the program committee of CFSE-5- *Conférence française sur les systèmes d'Exploitation*. 2006, France.
- Member of the program committee of SSS'2006 - *8th International Symposium on Stabilization, Safety, and Security of Distributed Systems (formerly Symposium on Self-stabilizing Systems) (SSS 2006) November 17th-19th, 2006, Dallas, Texas, USA*
- Member of the program committee of PARCO'2006 - *Special Issue of Parallel Computing Journal on Large Scale Grid*
- Project reviewer for ANR projects (ARA MDSA, Blanche, Securité).
- Member of the committee of the ACM Sigops France prize of the best PhD thesis in operating system.
- Member of the "Commission de spécialistes" of Paris 6 and Lille 1 Universities.
- Member of the national jury for PEDR.
- Elected member of the "Institut de Formation Doctorale" of Paris 6 University.
- Vice-chair of the LIP6 laboratory.

Marc Shapiro is:

- Founder and Chair of EuroSys, the European professional society on Computer Systems (European chapter of ACM SIGOPS).
- Member of the White Paper committee "Fostering Systems Research in Europe."
- Co-chair of Authoring Workshop at EuroSys conference, Leuven, March 2006.
- Member of the Board of ASTI, Fédération des Associations Françaises des Sciences et Technologies de l'Information.
- Member of the programme committee of the Int. W. on High-Perf. Data Mgt. in Grid Environments 2006, Rio de Janeiro, July 2006.
- Member of the programme committee of the VLDB 2006 Workshop on Data Management in Grids, Seoul, Sept. 2006.
- Invited speaker at ICFP (Int. Conf. on Functional Programming), Portland Oregon (USA), Sept. 2006.
- Member of the Programme Committee of OPODIS (Int. Conf. on Principles Of Distributed Systems), Bordeaux, Dec. 2006.
- Member of the Programme Committee of ICDCS (Int. Conf. on Distributed Computing Systems), Lisbon July 2006 and Toronto June 2007.
- Reviewer for NSDI Conference (Networked Systems Design and Implementation), Cambridge MA (USA), April 2007.
- Reviewer for ACM TOCS (Transactions on Computer Systems).
- Reviewer for Distributed Computing journal.
- Reviewer for the journal "Science of Computer Programming."
- Reviewer for the journal "Concurrency and Computation: Practice and Experience."
- Project reviewer for the Swedish Research Council.
- Project reviewer for the Swiss National Science Foundation.
- Tenure reviewer for Università della Svizzera Italiana, Lugano, Switerland.

8.2. PhD reviews

Bertil Folliot was HDR rewiever of :

- Didier Donsez. *Objets, composants et services : intégration de propriétés non fonctionnelles*, Institut National Polytechnique de Grenoble.

Pierre Sens was PhD reviewer of :

- Takoua Abdellatif. *Apport des architectures à composants pour l'administration des intergiciels*. Institut National Polytechnique de Grenoble, Advisor: J. Mossière.
- Erwan Becquet. *Spécification et Prototypage d'une Messagerie Industrielle à Contraintes Temporelles orientée Objets et Composants* , CNAM, Advisor: G. Florin.
- Aimen Bouchhima. *Modélisation du logiciel embarqué à différents niveaux d'abstraction en vue de la validation et la synthèse des systèmes monopuces*. Institut National Polytechnique de Grenoble, Advisor: J-L Petrot.
- Samir Jafar. *Programmation des systèmes parallèles distribués : tolérance aux pannes, résilience et adaptabilité* Institut National Polytechnique de Grenoble, Advisor: J. Mossière. , Advisor: J-L Roch.
- Adrien Lebre. *aIOLi : contrôle, ordonnancement et régulation des accès aux données persistantes dans les environnements multi-applicatifs haute performance*. Institut National Polytechnique de Grenoble, Advisor: B. Plateau.
- Sébastien Monnet. *Gestion des données dans les grilles de calcul : support pour la tolérance aux fautes et la cohérence des données*. Université de Rennes 1. Advisor: L. Bougé.
- Maria Pilar del Villamil. *Service de localisation de données pour les systèmes P2P*. Institut National Polytechnique de Grenoble, Advisor: C. Roncancio.

Marc Shapiro was PhD reviewer of :

- Sapan Bhatia, "Optimistic Compiler Optimizations for Network Systems", U. of Bordeaux, June 2006.
- Corina Ferdean, "Conception et implantation de mécanismes de replication adaptables aux besoins des utilisateurs," U. Paris-6, Dec. 2006.

8.3. Teaching

- Luciana Arantes :
  - Principles of operating systems in Licence d'Informatique, Université Paris 6
  - Operating systems kernel in Maîtrise d'Informatique, Université Paris 6
  - Responsible for projects in operating system, Maîtrise d'Informatique, Université Paris 6
- Bertil Folliot :
  - Principles of operating systems in Licence d'Informatique, Université Paris 6
  - Distributed algorithms and systems in Master Informatique, Université Paris 6
  - Distributed systems and client/serveur in Master Informatique, Université Paris 6
  - Projects in distributed programming in Master Informatique, Université Paris 6
- Mesaac Makpangou
  - Systems and networks, Master, Pôle Universitaire Leonard de Vinci
- Oliver Marin

  - Operating system programming, Master d'Informatique, Univeristé Paris 6
  - Operating system Principles, Licence d'Informatique, Univeristé Paris 6 Université Paris 6
  - Parallel and distributed systems,Master d'Informatique, Université Paris 6
  - Client/server arcitecture in Licence professionelle d'Informatique, Université Paris 6
- Pierre Sens
  - Principles of operating systems in Licence d'Informatique, Université Paris 6
  - Operating systems kernel in Master Informatique, Université Paris 6
  - Distributed systems and algorithms in Master Informatique, Université Paris 6
- Gaël Thomas
  - Principles of operating systems in Licence d'Informatique, Université Paris 6
  - Operating systems kernel in Master Informatique, Université Paris 6
  - Distributed systems and Middleware in Master Informatique, Université Paris 6

9. Bibliography

Major publications by the team in recent years

[1] L. B. ARANTES, D. POITRENAUD, P. SENS, B. FOLLIOT. *The Barrier-Lock Clock: A Scalable Synchronization-Oriented Logical Clock*, in "Parallel Processing Letters", vol. 11, n^{o} 1, 2001, p. 65–76.

[2] M. BERTIER, L. ARANTES, P. SENS. *Hierarchical token based mutual exclusion algorithms*, in "Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid '04), Chicago (USA)", IEEE Society Press, April 2004.

[3] M. BERTIER, O. MARIN, P. SENS. *Implementation and performance of an adaptable failure detector*, in "Proceedings of the International Conference on Dependable Systems and Networks (DSN '02)", June 2002.

[4] M. BERTIER, O. MARIN, P. SENS. *Performance Analysis of Hierarchical Failure Detector*, in "Proceedings of the International Conference on Dependable Systems and Networks (DSN '03), San-Francisco (USA)", IEEE Society Press, June 2003.

[5] J.-M. BUSCA, F. PICCONI, P. SENS. *Pastis: a Highly-Scalabel Multi-User Peer-to-Peer File Systems*, in "Euro-Par'05 - Parallel Processing, Lisboa, Portugal", Lecture Notes in Computer Science, Springer-Verlag, August 2005.

[6] A.-M. KERMARREC, A. ROWSTRON, M. SHAPIRO, P. DRUSCHEL. *The IceCube approach to the reconciliation of divergent replicas*, in "20th Symp. on Principles of Dist. Comp. (PODC), Newport RI (USA)", ACM SIGACT-SIGOPS, August 2001, http://research.microsoft.com/research/camdis/Publis/podc2001.pdf.

[7] N. KRISHNA, M. SHAPIRO, K. BHARGAVAN. *Brief announcement: Exploring the Consistency Problem Space*, in "Symp. on Prin. of Dist. Computing (PODC), Las Vegas, Nevada, USA", ACM SIGACT-SIGOPS, July 2005.

[8] O. MARIN, M. BERTIER, P. SENS. *DARX - A Framework For The Fault-Tolerant Support Of Agent S oftware*, in "Proceedings of the 14th IEEE International Symposium on Sofwatare Reliability Engineering (ISSRE '03), Denver (USA)", IEEE Society Press, November 2003.

[9] F. OGEL, G. THOMAS, A. GALLAND, B. FOLLIOT. *MVV : une Plate-forme à Composants Dynamiquement Reconfigurables — La Machine Virtuelle Virtuelle*, 2004.

Year Publications

Doctoral dissertations and Habilitation theses

[10] C. FERDEAN. *Enforcing Service-Specific Replica Consistency Models and Performance Requirements for Heterogenous Replicated Services*, Ph. D. Thesis, Université Pierre et Marie Curie (Paris 6), 4, place Jussieu, Paris, december 2006.

[11] A. HACHICHI. *Container Virtual Machine : une plate-forme générique pour l'adaptation dynamique des services système dans les intergiciels orientés composants*, Ph. D. Thesis, Université Pierre et Marie Curie (Paris 6), 4, place Jussieu, Paris, december 2006.

[12] F. PICCONI. *Gestion de la persistance et de la volatilité dans le système de fichiers pair-à-pair Pastis*, Ph. D. Thesis, Université Pierre et Marie Curie (Paris 6), 4, place Jussieu, Paris, december 2006.

Articles in refereed journals and book chapters

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

[14] M. BERTIER, L. ARANTES, P. SENS. *Distributed Mutual Exclusion Algorithms for Grid Applications: A Hierarchical Approach*, in "JPDC: Journal of Parallel and Distributed Computing", vol. 66, 2006, p. 128–144.

[15] P. SENS. *Systèmes d'exploitation*, in "Corpus in Encyclopaedia Universalis", Encyclopaedia Universalis, 2006.

Publications in Conferences and Workshops

[16] S. BIANCHI, A. DATTA, P. FELBER, M. GRADINARIU. *Stabilizing Dynamic Spatial Filters*, in "submitted to International Conference on Distributed Systems ICDS 2007", 2006.

[17] V. DRABKIN, R. FRIEDMAN, M. GRADINARIU. *Self-stabilizing Wireless Connected Overlays.*, in "OPODIS", 2006, p. 425-439.

[18] X. DÉFAGO, M. GRADINARIU, S. MESSIKA, P. RAIPIN-PARVEDY. *Fault-tolerant and Self-stabilizing Mobile Robots Gathering*, in "20th International Symposium on Distributed Computing DISC 2006", 2006, p. 46-60.

[19] C. FERDEAN, M. MAKPANGOU. *A Response Time-Driven Server Selection Substrate for Application Replica Hosting Systems*, in "Proceedings of the Symposium on Applications and Internet (SAINT), Phoenix, Arizona, USA", January 2006.

[20] N. GEOFFRAY, G. THOMAS, B. FOLLIOT. *Distribution transparente et dynamique de code pour applications Java.*, in "Actes de la 5ème Conférence Française sur les Systèmes d'Exploitation, CFSE'5, Perpignan, France", october 2006.

[21] N. GEOFFRAY, G. THOMAS, B. FOLLIOT. *Live and Heterogeneous Migration of Execution Environments*, in "The First International Workshop on Pervasive Systems (PerSys'06), Montpellier, France", october 2006.

[22] N. GEOFFRAY, G. THOMAS, B. FOLLIOT. *Transparent and Dynamic Code Offloading for Java Applications*, in "The 8th International Symposium on Distributed Objects and Applications (DOA), Montpellier, France", october 2006.

[23] M. GRADINARIU, S. TIXEUIL. *Conflict Managers for Self-stabilization without Fairness Assumption*, in "submitted to International Conference on Distributed Systems ICDS 2007", 2006.

[24] O. MARIN. *DARX - a Self-Healing Framework For Agents*, in "Proceedings of the 12th Monterey Workshop", LNCS, Springer Verlag, 2006.

[25] F. PICCONI, B. BAYNAT, P. SENS. *Predicting durability in DHTs using Markov chains*, in "submitted to International Conference on Distributed Systems ICDS 2007", 2006.

[26] F. PICCONI, P. SENS. *Using incentives to increase availability in a DHT*, in "Third Intenational Workshop on Hot Topics in P2P Systems (Hot-P2P), In conjunction with: IPDPS IEEE International Parallel and Distributed Processing Symposium, Rhodes Island, Greece", IEEE Computer Society Press, April 2006.

[27] P. SENS, L. ARANTES, M. BOUILLAGUET. *Asynchronous Implementation of Failure Detectors with partial connectivity and unknown participants*, in "submitted to International Conference on Dependable Systems and Networks DSN 2007", 2006.

[28] J. SOPENA, L. ARANTES, P. SENS. .*Un algorithme équitable d'exclusion mutuelle tolérant les fautes*, in "Actes de la 5ème Conférence Française sur les Système d'Exploitation (CFSE'6), Perpignan, France", October 2006.

[29] J. SOPENA, L. ARANTES, P. SENS. *Performance evaluation of a fair fault-tolerant mutual exclusion algorithm*, in "Proceedings of the 25th IEEE International Symposium on Reliable Distributed Systems (SRDS-25), Leeds, UK", IEEE Computer Society Press, September 2006.

[30] O. VALENTIN, F. JOUANOT, L. D'ORAZIO, Y. DENNEULIN, C. RONCANCIO, C. LABBE, C. BLANCHET, P. SENS, C. BONNARD. *GEDEON, un Intergiciel pour Grille de Données*, in "Actes de la 5ème Conférence Française sur les Système d'Exploitation (CFSE'6), Montpellier, France", October 2006.

Internal Reports

[31] P. SUTRA, J. BARRETO, M. SHAPIRO. *An asynchronous, decentralised commitment protocol for semantic optimistic replication*, Rapport de recherche, n^{o} 6069, Institut National de la Recherche en Informatique et Automatique (INRIA), Rocquencourt, France, December 2006, https://hal.inria.fr/inria-00120734.