Abstraction Layers for Distributed Computing

Abstract

Despite the practical interests of reusable frameworks for implementing specific distributed services, many of these frameworks still lack solid theoretical bases, and only provide partial solutions for a narrow range of services. We argue that this is mainly due to the lack of a generic framework that is able to unify the large body of fundamental knowledge on distributed computation that has been acquired over the last 40 years. The DESCARTES project aims at bridging this gap, by developing a systematic model of distributed computation that organizes the functionalities of a distributed computing system into reusable modular constructs assembled via well-defined mechanisms that maintain sound theoretical guarantees on the resulting system. DESCARTES arises from the strong belief that distributed computing is now mature enough to resolve the tension between the social needs for distributed computing systems, and the lack of a fundamentally sound and systematic way to realize these systems.

Abstractions modulaires pour le calcul distribué

Résumé

Malgré l'intérêt que des environnements réutilisables pour l'implémentation de services distribués spécifiques peuvent avoir en pratique, encore beaucoup de ces environnements manquent de bases théoriques solides et fournissent seulement des solutions partielles pour une gamme de services limitée. Selon nous, la raison principale est qu'il n'existe pas encore de cadre suffisamment générique capable d'unifier l'ensemble conséquent des connaissances fondamentales du calcul distribué acquises pendant ces 40 dernières années. Le projet DESCARTES a pour but de combler ce vide en développant un modèle systématique du calcul distribué capable d'organiser un système distribué en modules réutilisables, assemblés par des mécanismes bien définis permettant de maintenir de solides garanties théoriques sur le système résultant. Nous pensons venu le temps pour un tel projet car nous croyons fortement que le calcul distribué est maintenant assez mature pour résoudre le conflit entre les besoins sociétaux en systèmes distribués et le manque d'une méthodologie systématique aux bases solides permettant de réaliser ces systèmes.

1 Context, position and objectives of the proposal **4**
1.1 Our general ambition . . . 4
1.2 Difficulties and obstacles . . . 5
1.3 State of the art . . . 7
1.4 Scientific objectives . . . 9

2 Scientific and technical programme, project organisation **10**
2.1 Task description . . . 10
2.1.1 Task 0: Project Management . . . 10
2.1.2 Task 1: Cartography of Distributed Computing Models . . . 11
2.1.3 Task 2: Pertinence of Existing Hierarchies . . . 12
2.1.4 Task 3: Holistic Approaches . . . 14
2.1.5 Task 4: Layered Distributed Computing . . . 15
2.2 Schedule . . . 17
2.3 Consortium description . . . 17
2.3.1 Partners description and relevance, complementarity . . . 17
2.3.2 Qualification of the coordinators . . . 19
2.4 Scientific and technical justification of the requested ressources . . . 23
2.4.1 Equipment . . . 23
2.4.2 Staff . . . 23
2.4.3 Missions . . . 24
2.4.4 Other expenses . . . 26
2.4.5 Total . . . 26

3 Exploitation of results, global impact of the proposal **26**
3.1 Design techniques . . . 26
3.2 Dissemination of knowledge . . . 27
3.3 Bottleneck identification . . . 27
3.4 Diffusion . . . 27

4 References **27**

Table of participants

Laboratory	Last name	First name	Position	MM	Role
Partner 1:	Bordeaux (Bordeaux University) – Coordinator				
LaBRI	GAVOILLE	Cyril	PR	29	Global coordinator
LaBRI	ILCINKAS	David	CR/CNRS	29	Deputy coordinator
LaBRI	JOHNEN	Colette	PR	19	
LaBRI	MÉTIVIER	Yves	PR	19	
LaBRI	MILANI	Alessia	MCF	19	
LaBRI	TRAVERS	Corentin	MCF	19	Task 1 coordinator
Partner 2:	Paris (Paris Diderot University)				
IRIF	DELPORTE	Carole	PR	29	Task 2 coordinator
IRIF	FAUCONNIER	Hugues	PR	29	Local coordinator
IRIF	FRAIGNIAUD	Pierre	DR/CNRS	29	Task 4 coordinator
IRIF	KOSOWSKI	Adrian	CR/INRIA	20	
IRIF	VIENNOT	Laurent	DR/INRIA	10	
Partner 3:	Rennes (Rennes 1 University)				
IRISA	RAYNAL	Michel	PR	34	Local coordinator
IRISA	TAÏANI	François	PR	12	
Vérimag (Grenoble)	DEVISMES	Stéphane	MCF	19	
LIF (Marseille)	GODARD	Emmanuel	PR	14	
LINA (Nantes)	MOSTEFAOUI	Achour	PR	24	Task 3 coordinator

1 Context, position and objectives of the proposal

1.1 Our general ambition

The OSI model characterizes and standardizes the internal functions of a communication system by partitioning it into abstraction layers. Although existing communication systems (e.g. the Internet) are deliberately not as rigidly designed as the OSI model, it remains a fact that the OSI standard provides an essential conceptual model for building communication systems. This is because it allows designers to break up complex problems into smaller manageable pieces in a way that is practical, generic, and protocol-independent.

Turning our attention to the focus of this proposal, that is, to *distributed computing systems*, one cannot fail to notice that practitioners and engineers have proposed a number of reusable frameworks and services to implement specific distributed services (from Remote Procedure Calls with Java RMI or SOAP-RPC, to JGroups for group communication, and Apache Zookeeper for primary backup replication). In spite of the high conceptual and practical interest of such frameworks, many of these efforts lack a sound grounding in distributed computation theory (with the notable exceptions of JGroups and Zookeeper), and only provide punctual and partial solutions for a narrow range of services. We argue that this is because we still lack a generic framework that unifies the large body of fundamental knowledge on distributed computation that has been acquired over the last 40 years.

> *This proposal aims at bridging this gap, by developing a systematic model of distributed computation that organizes the functionalities of a distributed computing system into reusable modular constructs.*

These constructs should be assembled via well-defined mechanisms that maintain sound theoretical guarantees on the resulting system. This vision is in a way analogous to what has been done for communication systems thanks to the OSI model. We advocate that this modular unification must occur now for two crucial reasons:

- *First*, the lack of any unifying framework, relying on formal foundations, makes it very difficult for practitioners to design, develop and maintain efficient and robust distributed systems. When they create a distributed system, these practitioners must face many challenges (adversaries) simultaneously (process asynchrony, unknown network structure and dynamics, component failures, variety of communication mediums, etc.), with no a priori insights about the order in which these challenges should be addressed, or about the interplay between them.
- *Second*, distributed computing systems are now ubiquitous, and can be observed at all levels of modern computing infrastructures, ranging from multi-core processors to the cloud. As a compounding factor, our inherently "interconnected world" is likely to generate even more of these systems, which share a limited amount of technical characteristics, yet have a lot in common in terms of functionalities and conceptual aspects.

This situation creates a tension between applications and social needs for distributed computing systems, and the lack of a fundamentally sound and systematic way to realize them. This project arises from the strong belief that the distributed computing community has now acquired enough fundamental knowledge on the inherent nature and limitations of distributed computation to resolve this tension, and that this tension has to be resolved quickly as new distributed computing systems continue to flood our everyday life while lacking solid theoretical foundations.

In terms of relevance and impact, the DESCARTES project will contribute to the challenge of the ANR entitled *Société de l'information et de la communication*, and more precisely to its axis *Fondement du numérique*. Indeed, DESCARTES aims at developing fundamental research in distributed computing susceptible to have a strong impact on the design, analysis, and maintenance of robust and efficient distributed computing systems. We foresee in particular that the outcome of the project will

not only be conceptual, but will also eventually produce breakthroughs in several practical aspects of the design of distributed computing systems, in a way similar to OSI-like models, which are not only conceptual but also at the core of the practical design of actual communication systems.

The DESCARTES project will also contribute significantly to the dissemination of knowledge [1], by making it easier to teach distributed computing systems to students at universities and engineering schools. Indeed, the modular partitioning we envision will provide a natural teaching framework based on a progressive and principled exposition of the field. Such a partitioning will allow students to approach the aforementioned difficulties separately, and to understand how the solutions developed to address them can be composed in a smooth and systematic manner.

Last but not least, by offering a modular decomposition with a strong theoretical grounding, the DESCARTES project should greatly help identify performance bottlenecks in distributed computing systems. The modular strategy we propose will help pinpoint the components and/or the interfaces to be enhanced in order to improve the global behavior of the whole system, by highlighting which aspects might require deeper scientific investigations among the jungle of concepts, models, problems, and constraints that are characteristic of the field.

1.2 Difficulties and obstacles

Communication systems and parallel systems. To illustrate the difficulties and obstacles to be faced when organizing the functionalities of a distributed computing system into reusable modular constructs, let us first confront the nature of distributed systems to two types of apparently closely related environments, namely *communication systems*, and *parallel computing systems*. The former have been essentially layered according to the aforementioned OSI model: starting from basic hardware transmission technologies, at the physical layer, the model is built up, until applications can be executed at the top layer. More concretely, the capability to exchange electric signals makes it possible to transmit frames, which in turn makes it possible to transmit packets, and so forth and so on, until a set of abstract APIs (Application Programming Interfaces) is obtained that enables applications to exchange all kinds of information. Overall, one can therefore see communication systems as having been organized *bottom-up*.

By contrast, parallel computing systems, which are essentially interested in computing performances, can be conceptually viewed as being organized *top-down*. From the early age of parallel computing, algorithm designers have felt the need to manipulate data at a sufficiently abstract level, be it in the form of high level computing libraries (e.g., ScaLAPACK), or high level programming models (e.g., PVM or MPI for message passing, OpenMP for shared memory multiprocessing, or, more recently, MapReduce, Storm, S4 or Spark for processing and generating large amounts of data on a cluster). The implementation of these libraries and programming models then depends on the kind of parallel platform upon which they are executed, potentially requiring sophisticated data formatting techniques, as well as efficient mechanisms for barrier synchronization, scheduling, and load balancing. At a lower level, if one assumes that the programmer should be responsible for exposing the parallelism, then programming languages like Cilk and its derivatives are designed for the efficient manipulation of threads in multi-threaded parallel computing environments. At the bottom level, specific architectures (e.g., multi-core, switches, etc.), coupled with appropriate compilers have been designed to extract as much parallelism as possible from (sequential or parallel) programs.

Having in mind the above very rough description of communication systems and parallel computing systems, one can argue that, conceptually, communication systems have to *cope with interoperability*, and parallel computing systems have to *cope with efficiency*. Both types of systems have benefited from the enormous effort of the networking and parallel computing communities, respectively, over the last

1. For the first scientific task of the project will be to *cartography* the world of distributed computing systems.

decades, resulting in tremendous progresses in the way networks and all kinds of parallel computers can be used.

Distributed computing. Differently from communication systems and parallel systems, distributed computing has to *cope with uncertainty*. This uncertainty arises from asynchrony, failures, remoteness, unstable behaviors, non-monotonicity, system dynamism, mobility, low computing capability, scalability requirements, etc. Mastering one form of uncertainty or another is pervasive in all distributed computing problems [Ray14]. The major obstacle to mastering uncertainty is that it is *ubiquitous*. Hence, bottom-up approaches (i.e., that progressively build layers upon lower level layers), or top-down approaches (i.e., that progressively refine the basic mechanisms required to perform a task efficiently) applied to distributed computing have only succeeded in analyzing small fragments of the field (cf. Section 1.3).

One further reason that explains why uncertainty is so difficult to control is the fact that even the "elementary particles" of distributed computing (i.e., the basic tasks at the core of the field) are directly impacted by the "uncertainty principle" inherent to the discipline. For instance, *symmetry breaking* cannot be achieved locally, and requires communications at a distance growing with the size of the system [2]. Even more dramatic is the fact that a system composed of a collection of computing entities (e.g., asynchronous crash-prone Turing Machines) can be *computably weaker* than a system composed of a single element [3]. In other words, it appears that there are no extremities from which distributed computing can be solidly grabbed, as each extremity is at best extremely difficult to solve, and, at worst, just impossible to solve. As a result, the field developed by growing different branches (e.g., the Java Concurrency package, the synchronizers, etc.), which hardly interoperate.

The DESCARTES project. The DESCARTES project has been set up based on the strong belief that the distributed computing discipline is now mature enough to overcome the "uncertainty obstacle", and is ready to tackle the design of a modular approach that is able to articulate and interface the numerous components of distributed computing, while remaining grounded on rigorous theoretical principles. This belief results from the recent effort of the French distributed computing community to adopt the same terminology and the same language for tackling the so many different issues to be faced by the discipline (which, again, spans scenarios at all scales, from multicore to Internet, from the access to critical sections to the maintenance of consistent replicas in a large distributed database). This includes notions of *tasks*, *reduction*, *adversaries*, etc., as well as all concepts at the core of computability and complexity theory, adapted to the distributed computing setting. Hence, thanks to several years of intense collaborations, while the distributed computing field remains scattered and split in several subfields, the French community, and particularly the members of DESCARTES, are *no longer* scattered. As a consequence, we strongly believe that not only gaps between sub-fields of distributed computing can be bridged, but, that more ambitiously, the functionalities of a distributed computing system can be organized into a modular structure according to a rigorous theoretical understanding of the field.

Apart from the usual technical difficulties to be faced during the execution of a scientific project, DESCARTES is facing an enormous obstacle: parts of distributed computing are currently just a huge jungle of models and hypotheses, and it may seem hopeless to encompass all of them in a single theory. Our answer to this legitimate criticism is that DESCARTES is not aiming at designing a "theory of

2. Cf. the lower bound [Lin92] for MIS for which N. Linial received the Dijkstra Prize in Distributed Computing 2013 (the highest recognition in the distributed computing community). See also the work of P. Fraigniaud on the role of identities in local distributed computing, for which he received the SIROCCO "Innovation in distributed computing" Award in 2014.

3. Cf. the impossibility result [FLP85] for *consensus*, which deserved the Dijkstra Prize in Distributed Computing 2001 to their authors. See also the work of M. Raynal on distributed agreement for which he received the SIROCCO "Innovation in distributed computing" Award in 2015.

everything", but instead at designing a modular model. Modularity is central to the project. This concept offers two main advantages. First, modularity ensures that the project will be able to make progress step-by-step, potentially leaving some part of the jungle unexplored, but at least providing a coherent structure to large parts of it. Second, modularity ensures that DESCARTES will be able to elaborate a *proof of concept* to demonstrate the relevance of our approach. A 4-year project may be too short for elaborating a fully consistent modular framework encompassing the whole setting of distributed computing systems. Nevertheless, we have no doubts that this proposal can lead to a perhaps partial but quite convincing modular model, opening the way to the near-future design of a practically relevant, and theoretically sound, modular model for distributed computing systems.

1.3 State of the art

In the following, we list a few examples of some of the aforementioned branches of the distributed computing field in which embryos of modular approaches of the discipline can be identified, at least implicitly. We are by no means aiming at being exhaustive. Instead, the goal of this section is merely to illustrate frameworks that appear promising as starting points for our project. Each of the frameworks discussed below exhibits articulations and/or connections between different aspects of distributed computing. We start our quick survey by notions mostly present in the context of distributed networked systems, and then move on to notions mostly present in tightly coupled distributed systems (i.e., shared-memory).

Synchronizers. A *synchronizer* is an algorithm that can be used to run a synchronous algorithm on top of an asynchronous network, thus enabling the asynchronous system to run as a synchronous network. In other words, a synchronizer produces *logical* synchrony (but not physical synchrony). Synchronizers therefore form the primary example of an interface between two different aspects of distributed computing: synchronous computing and asynchronous computing. The concept of synchronizer was originally proposed by B. Awerbuch [Awe85] along with three synchronizer algorithms, named alpha, beta, and gamma, which provided different tradeoffs in terms of time and message complexity. The zeta synchronizer, having the same tradeoff as gamma, was introduced in [AW04]. The gamma synchronizer is essentially optimal if the simulation is round-by-round. Improved synchronizers have been proposed for specific settings, like [AP90, PU89a] (see also [Lyn96, Pel00, Ray13b] for general discussions about synchronizers).

Self-stabilization. *Self-stabilization* is a concept of fault-tolerance in distributed computing, introduced by E. Dijkstra in 1974, and popularized by L. Lamport (Turing Award 2013) a decade later. A distributed system is *self-stabilizing* if it ends up in a correct state after a finite number of steps, no matter which state it starts from. A popular and modular approach in the self-stabilizing design consists in the use of so-called *composition techniques*, e.g., fair, collateral, hierarchical, or conditional compositions, to only quote a few of them. For example, in the *collateral composition* [Tel01] of two algorithms A and B, both algorithms run concurrently, yet A uses as input the output of B. *Silent* self-stabilization adds the requirement that, once the system has stabilized in a correct state, this state remains unchanged. This latter notion is closely related to *non-deterministic local computing* in network, thanks to the concept of *proof-labeling scheme* [KKP10], a mechanism for certifying the correctness of a state. Tradeoffs between certificate size and efficiency of self-stabilizing algorithms were recently established in [BFPS14] (see also [BF15]). Other mechanisms baring similarities with proof-labeling schemes have been described in [AD02, BDDT07].

Locality. *Locality* is a guiding principle in many different frameworks of distributed computing. In the framework of network computing, it captures the ability to solve tasks (e.g., coloring, MIS, etc.),

by having each node inspecting the inputs of nodes in its close neighborhood. Solving tasks locally turns out to be challenging, and often impossible [Lin92]. Surprisingly, as established in [NS95], there are strong connections between construction tasks and decision tasks in local network computing, with significant impact on the power of randomized algorithms (see also [FF05]). Computability classes related to local computing have been extensively studied in [FKP13]. Recently, it was proved [EPSW14] that every task that can be solved (and verified) by a randomized anonymous algorithm can also be solved by a deterministic anonymous algorithm provided that the latter is equipped with a 2-hop coloring of the input graph. Hence 2-hop coloring is somewhat universal in the context of anonymous networks.

Spanner. A *spanner* is a subgraph spanning all the nodes of a network, but including only a small part of the original network links. A spanner can thus be seen as a skeleton of a original network. It provides a natural graph theoretical tool for expressing trade-offs between the edges density in the spanner, and the distance stretch w.r.t. the distance in the original graph. Formally introduced by Peleg and Schäffer in [PS89], graph spanners implicitly appears in [Awe85, PU89a] as a way of implementing synchronizers. Each of the alpha, beta, and gamma synchronizers is acutally based on a specific spanner construction. Nowaday, the basic concept of spanner arises not only in distributed computing, and in communication networks, but also in many fields of computer science, including computational biology, computational geometry, and robotics.

As early observed in [PU89b], algorithms for constructing spanners can be used to provide efficient routing service for general networks. In particular, for every connected weighted graph, a spanner provides routing tables using a sublinear number of entries per node, while guaranting routes with constant stretch factor w.r.t. the distance in the original graph. The relationship between compact routing schemes and graph spanners is at the heart of an active field of research over the last decades, with prominent results such as: [TZ01, AGM+08, RT15]. All these works tend to demonstrate that the trade-offs between edge-density and distance-stretch for spanners, and between routing table size and route length stretch for networks are extremely closely related [GS11].

Failure detectors. *Failure detectors* [CT96] are distributed oracles that encapsulate information about crash failures. According to the type and the "quality of service" of this information, different classes of failure detectors can be defined. Not only failure detectors may be compared by reduction, but it is also possible to associate with each problem P, its weakest failure detector [JT08]. Interestingly, the reduction order on failure detectors induces an order on problems: a problem P is harder (as far as the information on failures to solve it ids required) than a problem P' if the weakest failure detector needed for solving P' is weaker than the weakest failure detector needed for solving P. There are strong relationships between asynchrony properties and failure detectors [ADFT08] and furthermore partially synchronous models may be characterized by the failure detector they enable to implement.

Indulgent algorithms. The notion of indulgent was introduced in [Gue00]. Its underlying theory is developed in [GL08], and fault-tolerant distributed algorithms are presented [GR07, Ray10].

An indulgent algorithm is a distributed algorithm based on a failure detector that has the following properties. If the underlying failure detector satisfies its specification, the algorithm terminates and its results are correct. If the the underlying failure detector behaves arbitrarily (i.e., it never satisfies its specification), the algorithm can never terminate, but if it terminates its results are correct. Hence, an indulgent algorithm never returns incorrect results, and its termination is strongly related to the correct behavior of its underlying failure detector.

Indulgent algorithms are very attractive from a practical point of view. Intuitively, the (long enough) periods during which the underlying failure detector satisfies is specification is what is called

"stable periods", while the other periods are "unstable periods".

Borowsky-Gafni (BG) simulation. The BG simulation is a powerful simulation technique initially proposed for colorless tasks [BG93, BGLR01]. Those are the tasks for which each process may adopt the input or the output value of any other process. The BG simulation proves that, from a colorless task computability point of view, a system of $(f+1)$ processes in which up to f processes may crash is equivalent to a system of n processes in which up to f processes may crash. "Equivalent" means here that any algorithm in a system may be simulated in the other. Roughly speaking, the BG simulation means that, f-resiliency may be reduced to wait-freedom. Interestingly, the BG simulation has been extended to colored tasks in [Gaf09, IR09].

Consensus universality. The notion of *consensus universality* result [Her91] applies to state machine replication. Any sequential service modeled by a state machine can be replicated over any number of processes, when using consensus objects to ensure the liveness of the service in presence of crashes. This result was extended to k-set agreement [GG11, RST14]. More precisely, it is shown in [GG11] that, when replacing consensus by k-set agreement, any number of processes may emulate k state machines of which at least one makes an infinite number of step. It is shown in [RST14] that, if consensus is replaced by to (k,ℓ)-set agreement objects, at least $\ell \in [1..k]$ state machines progress forever.

Consensus hierarchy (Herlihy's hierarchy). This hierarchy introduced by M. Herlihy in [Her91] is on the wait-free implementations of atomic objects. This hierarchy is based on the notion of a *consensus number*. Namely, the consensus number of an object X is the maximum number of processes for which consensus can be solved with the help of atomic registers and objects X. Herlihy's hierarchy allowed many results on the possibility or impossibility to wait-free implement concurrent objects (defined by a sequential specification on total operations) to be proved.

This infinite hierarchy is strict in the sense that some atomic objects (e.g., test&set) are strictly weaker in the hierarchy than objects (e.g., compare&swap): no object with consensus number x can wait-free implement an object whose consensus number is $y > x$. Such a hierarchy appears now in textbooks on "modern" synchronization (e.g., [Ray13a, Tau06]).

1.4 Scientific objectives

The main objective of DESCARTES is to provide a layered or modular model for distributed computing systems to be viewed as a set of *services* linked by appropriated *interface* mechanisms enabling to compose these services, while remaining grounded on rigorous theoretical principles. It will not necessarily be the case that these services pile up as the layers of networking models. Instead, it is expected that they will be used as bricks, to be glued so as to form a structure that may not be 1-dimensional. By specifying each brick as well as the interplay between these bricks appropriately, the project aims at providing a systematic way to approach the design of distributed computing systems, despite the fact that these systems may be of very different natures: from multi-cores to the cloud, from ad hoc wireless networks to swarms of robots, from static networks to rapidly evolving dynamic systems, etc.

As toy examples illustrating the objectives of the DESCARTES project, consider a *synchronizer service* that enables developers to design protocols by assuming a synchronous system, while the underlying system may actually be fully asynchronous. Similarly, a *fault-tolerance service* will enable to design protocols as in fault-free systems, while the underlying system may actually be crash prone, and a *routing service* will enable to design protocols as in a completely connected network, while the underlying network may actually be poorly connected. The challenges addressed by DESCARTES are

of course far more complex than these toy examples as the space of scenarios to be encompassed when designing a distributed computing system is enormous and multi-dimensional. One dimension is the degree of asynchrony; another is the kind of communication medium; a third dimension is the type of potential failures; a fourth one is the type of consistency criteria; et cetera, not to mention the kinds of access to shared resources, the degree of privacy, and so on, and so forth. Each of these dimensions has been thoroughly investigated by the distributed computing community, and hence we believe that the expertise acquired by this scientific community, and in particular the members of DESCARTES, has now reached a level of maturity which is sufficient to solve the whole puzzle, that is, to cease studying each piece separately, but to address the issue of composing and assembling them appropriately.

Slightly more specifically, there are roughly two types of *simulations* that are well understood and widely used in distributed computing. One is *model-simulation*, that is, for two distributed computing models M and M', the definition of a mechanism enabling any algorithm A designed for model M to be rewritten as an algorithm A' for model M', with the same input-output specification as A. The core example of such simulation is BG-simulation, which allows a set of $f+1$ processes, any f of which may exhibit stopping failures, to "simulate" a larger number n of processes, also with at most f failures. The other type of simulation is *task-simulation*, that is, for a fixed distributed computing model M, the proof that some task T can be solved whenever some other task T' can be solved. The proof might be either existential, or constructive, by providing an algorithm A for T using a black box procedure solving task T'. The core example of such a simulation is the universality of *consensus* in crash-prone asynchronous shared-memory systems. That is, in such systems, a consensus black box enables to solve *all* tasks. Similarly, as said before, *distance-2 coloring* is universal in anonymous networks. The central object of study in the DESCARTES project is a third type of simulation, called *service-simulation*.

DESCARTES will heavily base its investigations on the notion of *service*, defined by its *interface* and its *specification*[4]. Roughly, Service S is implementable from Service S' under "hypothesis" H (specifying some context of execution) if the service S can be provided under H by using the service S' as a sub-routine. Service-simulation is eventually aiming at encompassing model-simulation and task-simulation into the same framework. Our ultimate objective (see Task 4 for more details) is to organize the functionalities of a distributed computing system as a graph $\mathcal{G}$ whose vertex set consists in services, and whose edge set consists of implementation relations (where each edge is labeled by a context). Note that such a graph may have double-edges and self-loops. Using an appropriate algebra on the hypotheses regarding the various context of executions, including operators such as conjunction or disjunction, we will be able to characterize the weakest hypotheses required to implement a service S from a service S', by analyzing the various paths leading from S' to S in $\mathcal{G}$. It is expected that distributed system developers and students will be able to use this modular organization of distributed computing the same way network developers uses the layered OSI model. Moreover, it is also expected that specialists in distributed algorithms with eventually participate in enhancing and enlarging our modular description, by adding new services, and new implementation relations between services.

2 Scientific and technical programme, project organisation

2.1 Task description

2.1.1 Task 0: Project Management

Global coordinators: Cyril Gavoille and David Ilcinkas (Bordeaux)
Local coordinators: Hugues Fauconnier (Paris), Michel Raynal (Rennes)

4. It may turn out that such a service is nothing else than the standard notion of *object* [Her91]. Nevertheless, we prefer to stick to the terminology "service" for it is better understood by the developers unaware of abstract notions of distributed computing.

An intensive collaboration between the three partners is crucial to the success of the project, involving frequent meetings and inter-partner visits. DESCARTES is therefore planning a total of seven 2-day meetings during which all partners will present their most recent results related to the project (note that we do not plan to limit the attendance to the members of the project only). As detailed in Section 3, we also plan to organize a workshop associated to the PODC conference and a final 1-week school.

In addition, our budget requests a grant for a 1-week visit per year per permanent member (that can be used by the member himself or by his students). The coordinators of the project will pay a specific attention that frequent exchanges between the partners are done. They will also encourage international exchanges in the form of short visits.

The fundamental nature of the project requires that the results will be made available to the community as quickly as possible. For this purpose, we will organize a web site for the project, enabling rapid diffusion of our preliminary results inside the project (via a private web site), and a rapid advertising of the finalized results outside the project (via a public web site). This web site will also include slides of all contributed talks given during plenary meetings.

2.1.2 Task 1: Cartography of Distributed Computing Models

Coordinator: Corentin Travers (Bordeaux)

A plethora of distributed computing models can be found in the literature. This reflects the various difficulties (asynchronous communication, changing and/or unknown network structure, process failures, variety of communication mediums, etc.) that have to be overcome when dealing with distributed systems.

Many fundamental services, including, e.g., consensus, routing, mutual exclusion have been studied extensively in many models. For example, a large, almost exhaustive body of knowledge has been collected on consensus. Consensus can be implemented in synchronous systems, tolerating various types of processes malfunction ranging from crash to byzantine failures. For each of these failure models, often tight, lower and upper bounds have been established on the complexity of consensus implementations. In asynchronous systems, consensus cannot be solved when at least one process can crash but can be implemented under some additional hypothesis such as using randomization, failures detector or assuming partial synchrony. In each of these refined models, again, lower and upper bounds for consensus implementation have been exposed. Most often, these bounds have been established in an ad-hoc manner, specific to the model under study. Nevertheless, a close examination of the literature reveals the existence of common, model-independent, basic building blocks in consensus implementations. For instance a simple service (called an adopt-commit object in shared memory, a ratifier in randomized protocols or introduced as the alpha abstraction in message passing), with minor variations in its specification, is at the heart of many consensus protocols.

Simulations have been developed to show the connection between different models. For example, the celebrated ABD [5] simulation relates the shared memory and message passing asynchronous model. It shows that as long as a majority of the processes are non-faulty, shared memory can be emulated in asynchronous, crash-prone message passing systems. The ABD simulation is universal, in the sense that it allows to translate any shared memory protocol into a message passing one. Besides the communication medium, other dimensions have been related by simulations including timing assumptions (how to simulate asynchronous models in a synchronous model), fault tolerance (the BG-simulation in asynchronous shared memory, how to give the illusion of crash failures in stronger failure models in synchronous message passing systems) or the strength of adversarial schedulers in self-stabilization.

5. The Dijkstra Prize 2011 was awarded to Hagit Attiya, Amotz Bar-Noy and Danny Dolev for the paper Sharing Memory Robustly in Message-Passing Systems.

Differently from the ABD simulation, some simulations are not universal as they apply only to some restricted classes of protocols. For instance, the BG-simulation was initially proposed only for colorless tasks [6].

When a model A can simulate a model B, whether or not a protocol exists for some task in model B reduces to a related question about the simulating model A. Besides distributed computability questions, whenever some service can be implemented in model B, the simulation may be used to automatically implement the same service in model A. The complexity of the resulting implementation is the complexity of the original implementation combined with the cost of the simulation. In most cases, the complexity of the simulation is not known. Exceptions include the synchronous message passing systems in which essentially optimal (in terms of communication rounds) simulations of crash failures on top of more severe message omission or byzantine failures have been designed. Even worst, between some pairs of models, the existence of simulations have been established but no effective protocol is known.

The objective of Task 1 is to build a map of distributed computing models and the relations linking them. The map will serve as a ground for the investigation conducted in the other tasks. We expect a Phd Student to strongly participate in the construction of this map, which involves the following sub-tasks:

- Identifying regions. A first step is to identify the main properties to define a model in the large body of distributed computing literature. The base map will consist in models: each region on the map will represent a model.
- Connecting regions. There is an arrow from a model A to a model B if it is possible to simulate B starting from A provided that some conditions are verified. We can imagine that different kinds of arrows may be used according to the status of the simulation (e.g., whether it is universal or restricted to some class of protocols, whether its complexity is optimal or not, etc.). While doing so, we will strive to identify common constructs among simulations techniques. Another possible outcome is the design of new simulations and/or when simulations already exists, improvements of their complexity.
- Exploring regions. Finally, for any fixed model, we would like to represent the problems that have been solved in this model and reductions among them, aiming again at identifying simple model-independant modular constructs.

2.1.3 Task 2: Pertinence of Existing Hierarchies

Coordinator: Carole Delporte (Paris)

Task 2 concerns the fundamentals tools for the cartography of Distributed Models. Among the success stories in the context of distributed computing, the identification of *hierarchies* (and *simulations*) is one of the prominent successes.

The main objective of Task 2 is to identify which hierarchies and simulations are relevant and can be viewed as an abstraction of a hierarchy of services and which one can be discarded- as far as setting up a modular model for distributed computing system is concerned.

A priori several interesting hierarchies may be identified.

- The most famous is probably the Herlihy's hierarchy of wait-free objects based on the *consensus number* $h(o)$ of an object o. This hierarchy enables to separate distributed objects into a strict (infinite) hierarchy. For example, from the Herlihy's hierarchy we deduce that a shared memory system with queue or stack is strictly weaker than a shared memory system with compare&swap. This hierarchy would play an important role in the cartography fo Distributed Computing models and we can sketch what could be our approach in DESCARTES. Task 2 has to determine in which

6. Roughly speaking, a colorless specification does not involve processes ids.

way the notion of service in the cartography may correspond to the notion of shared object in the sense of the Herlihy's hierarchy, and we guess it could be possible to define the consensus number of a service. Then the universality of consensus among m processes induces that any shared memory system with such a consensus can implement any service of consensus number m. In this way, such a hierarchy enables to separate and regroup (by simulation and/or implementation) models in the cartography.

- Another classical hierarchy is the hierarchy of failure detectors. Recall that failure detectors may be compared by reduction and that for each problem there is a weakest failure detector enabling to solve the problem, then we get hierarchies of the hardness of problems based on their weakest failure detector. Moreover, failure detectors correspond to properties of partially synchronous systems and from a hierarchy of failure detectors we get a hierarchy of partially synchronous models that may be defined in term of liveness properties and a priori partially synchronous models will be an important part of the cartography.
- Concerning f-resiliency many tools may concern the cartography of DESCARTES.
  First, with classical replication techniques, as soon as consensus is possible, any task may be realized whatever the number of faults is. This result of universality may be extended considering k-set agreement (in the k-set agreement up to k values may be decided) with a weaker liveness condition. Adapting these kinds of simulation in the DESCARTES framework is one of the goal of Task 2.
  The Borowsky-Gafni simulation allows a set of $f+1$ processes with at most f failures to simulate a larger number n of processes with at most f failures and this simulation should enable to get the same kind of results in the cartography of Distributed Model.
  There are strong relationship between f-resiliency, f-set agreement, f-concurrency that could have an important impact on the cartography. For example, the hierarchy of f-set agreement is equivalent to the hierarchy of f-concurrency (assuming f-set agreement f-concurrency may be implemented and reciprocally). Concerning process failures, with the notion of adversary characterized by the sets of processes that may fail the disagreement number provides another hierarchy that corresponds in some way to the ability of solving k-set agreement. Hence we have a set of potential tools that, helping to reduce process failure to wait-freeness and k-set agreement, would simplify the cartography.
- Several dimensions of hierarchies can also be identified in the framework of synchronous fault-free distributed network computing, that is, under the LOCAL and CONGEST models. One is related to the presence of identities, from fully anonymous networks (where nodes have no IDs or are not willing to reveal their IDs), to networks with nodes provided with pairwise distinct identities, including the scenario in which IDs are taken from a bounded set of values (like in IPv6), and the scenario in which they are arbitrarily large, as well as the presence or absence of a local orientation (i.e., some symmetry-breaking mechanism). Another dimension in the a priori knowledge given to the nodes about their environment, including their number, or an upper bound on this number, and graph parameters such as maximum degree, maximum edge weights, etc. A third dimension is related to the presence of distributed data structures such as distance oracles or any kinds of informative labeling scheme, as well as multiplicative or additive spanners, distance-2 coloring, etc. Of course, the access to private or shared sources of randomness can also be viewed as yet another type of hierarchy.

In Task 2, DESCARTES will complete a full survey of those hierarchies, and will organize them in a comprehensive and systematic way, as a preliminary step for their modular organisation, to be completed in Task 4. The PhD student will be strongly involved in this survey and the organisation of those hierarchies will be his first work. The risk of failure exists but is quite limited for Task 2. Indeed, DESCARTES includes experts in all the fields of distributed computing mentioned in this task. It may however happen that the presence of too many different hierarchies yields exponentially many

different scenarios, leading us to loose the big picture. DESCARTES will overcome this danger by preventing overspecialization, which will be achieved by making sure that every identified hierarchies is accessible to, and understandable by most members of the project, and not only by a small group of hyper-specialized experts.

2.1.4 Task 3: Holistic Approaches

Coordinator: Achour Mostefaoui (Rennes)

For different reasons such as dealing with complex systems/problems, favoring reusability of code, principles and programs, allowing several groups of programmers to work on different parts of the same global system or allowing to change part of a whole without having to rebuild from scratch, several concepts have been introduced. Among these concepts, we can cite layered architectures (e.g. OSI network system), modularity (e.g. modules, functions, and packages in programming languages), and abstractions (e.g. data structures like arrays and sets, control structures like the loop and the conditional). This allows to reach machine/architecture independence. Each level, abstraction or module is specified independently of the outside environment; only the interface has to be fixed.

With the arrival of parallelism, several new abstractions have been introduced to deal with concurrency such as semaphores, locks, transactions (for databases) that allow to master the complexity induced by the multiplicity of activities (processes, servers, clients). Indeed, while a sequential program generates a unique sequence of events a parallel program generates a partial order of events that can lead to billions of possible interleavings of events that may themselves lead to conflicting accesses to resources (e.g. data, peripherals) [Lam78]. The problem became more complex with distributed computing, indeed it introduces two more uncertainty sources that need to be mastered: asynchronism and failures.

When one has only to deal with distribution and asynchronism, RPC (Remote Procedure Call) is a good abstraction as it allows to hide distribution by extending to notion of a function call to the outside of a machine starting the era of client-server systems [BN84]. The state-machine abstraction is also a good abstraction that allows to emulate a unique server over a replicated architecture [Sch90]. The combined effect of asynchronism and failures lead to the impossibility to solve many problems such as consensus, atomic broadcast, election, etc. [FLP85]. However this combined effect has another effect related to the abstraction. It makes the abstraction and modularity non trivial when not impossible. For example the concept of failure detectors is a good abstraction/module for abstracting timing hypotheses when designing distributed applications. It has been proved that, when some processes may exhibit a Byzantine behavior, the stacking of an application over a failure detector is not possible [DGGS99]; each process has to be aware of the exact behavior of the other processes. Years of research in the distributed computing field has not yet produced the ultimate abstractions as even for the computing model there is no unique vision. Whereas in communication networks the multiplicity of communication media (radio, optical fiber, Ethernet, etc.) does not prevent machines from having an end-to-end uniform communication without being aware of the exact routing and the exact nature of the network or the sub-networks that are interconnected.

As explained above, more than the architecture, timing assumptions and failures are the two key parameters that have to be considered when tackling distributed computing. The hardness of this task leads to the design of solutions that are specially tailored for a given computing model that makes the approach holistic. The solution and the context (environment and underlying system) are seen as a whole. After decades of such fragmented contributions, it is perhaps time to try to extract abstractions and design patterns that may allow to design solutions that work on a system that can be synchronous or asynchronous and prone to crash failures or omission failures with a different quality of service.

First steps have already been taken if we consider a family of algorithms called indulgent algo-

rithms [GR07]. As an example, consensus algorithms designed for asynchronous systems prone to process crashes and based on eventual failure detectors are inherently indulgent in the sense that even if the failure detector black box considered does not behave correctly the consensus built atop will never behave wrongly. Either it delivers a correct result or it stalls. In such a situation, it is impossible to bound the time complexity of the algorithm and master its efficiency. On the other side, in real asynchronous distributed systems prone to process failures, there are periods of time where communication delays are chaotic and there are long periods of time where these systems are nearly synchronous. Moreover, even though failures may occur this seldom happens or fewer failures occur simultaneously. The question is then is it possible to design a protocol that implements a solution to a problem (e.g. consensus) such that if the system is quasi-synchronous it benefits from it and when the system is really asynchronous and/or failures do occur, then the protocol benefits from the indulgent property and always delivers a correct output, possibly with longer delays [Gue00]. When the cost induced by this versatility is reduced we say that the solution gracefully degrades [DG02]. Such solutions have been proposed but they are problem and system oriented.

The questions that arise are why such approach cannot be generalized to other problems not starting over from scratch, but through concepts/abstractions, not necessarily universal but differents approachs for different classes of problems/systems. Three possible directions are proposed below.

- Distributed systems being heterogeneous from communication and connectivity point of view, it is interesting to design replicated data structures with different consistency criteria. The mutual consistency between the operations executed by different nodes are not viewed in the same way, according to the fact that the nodes are close to each other or distant (deal with system heterogeneity).
- Try to define classes of problems in such a way that any problem P of a given class satisfies the following property. Let us first consider that the problem P is specified by a series of properties (termination, obligation, agreement, validity, etc.). Given two systems $S1$ and $S2$ any solution that implements problem P in system $S1$ can be used in system $S2$ with the guaranty that some properties are satisfied and some others are only satisfied eventually or with high probability (deal with system changes).
- Establish limits to modularity and abstraction. This means defining bounds such as: there is no solution to any problem of a given class that can work both on two kinds of systems. In such a situation a holistic approach becomes necessary in some way or another.

2.1.5 Task 4: Layered Distributed Computing

Coordinator: Pierre Fraigniaud (Paris)

Task 4 is aiming at producing the modules of a Layered Distributed Computing (LDC) model. This model is expected to consist in a collection of services and interfaces enabling to connect these services. DESCARTES is conjecturing that there might exist a one-to-one correspondence between, on the one hand, the services, and, on the other hand, the abstract mechanisms connecting models, and that this correspondence makes it possible to relate and compare different distributed computing models. Similarly, it is conjectured that there might exist a one-to-one correspondence between the interfaces and the distributed computing models. To illustrate such correspondences, consider an asynchronous fault-free distributed computing model, its synchronous variant, and a synchronization mechanism that makes it possible to execute synchronous algorithms within the asynchronous computing model. This synchronization mechanism provides a service S_i, with two interfaces, one towards asynchronous systems (H_j), and one towards synchronous systems (H_k) — see Figure 1. It is worth mentioning that the above are just conjectures, and it is precisely the role of Task 4 to investigate the structure of the LDC model, with no a priori of any kind. The main object in consideration in Task 4 is the aforementioned notion of *service-simulation*. Recall that a *service* is defined by its *interface* and its

specification, and that Service S is implementable from Service S' under "hypothesis" H, i.e., under some context of execution, if the service S can be provided under H by using the service S' as a sub-routine. This simulation is denoted by

$$S' \xrightarrow{H} S.$$

Figure 1 is depicting a collection of service-simulations as a graphs in which services are connected by simulations, labeled by their contexts of execution. This kind of modular decomposition might be the way the main outcome of the project will look like.

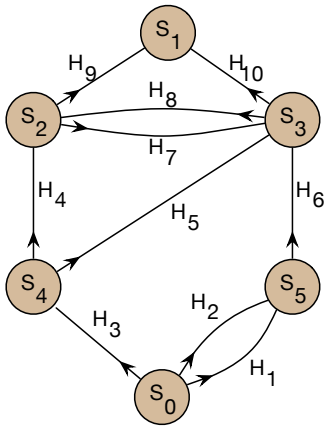

Figure 1: *Layered Distributed Computing (LDC) model.*

A LDC model such as depicted on Figure 1 would enable to answer questions such as: given service S_0, it is possible to implement service S_1 under H? An affirmative answer to this question is related to the existence of path P from S_0 to S_1 in the LDC graph with $H \preceq \cap_{i \in P} H_i$, i.e., H is stronger than any of the execution contexts encountered along P. Note that, in the LDC graph, there might be multiple edges — e.g., from S_0 to S_5, indicating that S_5 can be obtained from S_0 under H_1 as well as under H_2, with two incomparable execution contexts H_1 and H_2. The LDC graph might also include symmetric edges — e.g., between S_2 to S_3, indicating that these two services are equivalent under $H_7 \cap H_8$.

The outcome of Task 3 will serve as a benchmark for analyzing the pertinence of the LDC model, in terms of both efficiency and practicability. Indeed, it might well be the case that, in certain cases, applying stricto sensu the LDC model will result in a lost of efficiency, compared to the holistic approach (e.g., compared to indulgent algorithms). This is probably unavoidable, in the same way gaining efficiency in the context of communication systems may require to bypass some layer(s) in the OSI model. Task 4 will aim at evaluating such performance degradation, to be compared with the enormous gain in term of modularity and practicability.

As a starting point, Task 4 will mostly focus on two radically different frameworks which vary according to almost aspects, and whose hierarchies were studied in Task 2, namely: *synchronous fault-free distributed network computing* (i.e., the LOCAL and CONGEST models [Pel00]) and *asynchronous crash-prone shared memory computing* (i.e., the WAIT-FREE and t-RESILIENT models [AW04]). This will be the core of a PhD thesis. We believe that the risk of failure in both frameworks is limited. However, our high-risk/high-gain objective is to succeed to develop *the same* methodology in both frameworks, that is, using the same notion of services, and the same notion of interfaces.

Evaluation of success. The success of Task 4 (and, to some extend, of the whole project) will be measured according to our ability to produce fragments of the Layered Distributed Computing (LDC)

model, for few but highly representative branches of study within distributed computing, such as the aforementioned LOCAL/CONGEST and WAIT-FREE/t-RESILIENT classes of models. The project will already be a success if we are able to decompose each of these frameworks in modular constructs, as on Figure 1. The result of the project will be considered outstanding if we are also able to incorporate the notion of faults and asynchrony in the former classes, and the notion of network and synchrony in the latter classes, so that to essentially connect the two modular decompositions, and by doing so, making a giant step in the theory of distributed computing.

2.2 Schedule

The scientific methodology of the DESCARTES project consists in four scientific tasks (in addition to a fifth task, Task 0, related to the management of the project and the dissemination of its results). The first two tasks will start in parallel right at the beginning of the project, and end at mid-term. The last two tasks will start after one year, and will be then carried on until the end of the project, reaching their stationary regime at mid-term of the project. More specifically, Task 4 will deliver the main outcome of the project, while Task 3 will serve as a benchmark for analysing the pertinence of this outcome, in terms of both efficiency and practicability.

Year 1	Year 2	Year 3	Year 4
1. Cartography of Distributed Computing Models			
2. Pertinence of Existing Hierarchies			
	3. Holistic Approaches		
	4. Layered Distributed Computing		

2.3 Consortium description

2.3.1 Partners description and relevance, complementarity

The consortium gathers members mainly from three French leading labs in Distributed Computing: LaBRI, IRIF, IRISA[7]. Each of these labs has a worldwide reputation of excellence in the design and analysis of distributed algorithms. DESCARTES has paid attention to include experts of most sub-topics in distributed computing, including network computing, self-stabilization, fault-tolerant computing, failure detector theory, etc.

In order to illustrate the high level of expertise of the DESCARTES partners, let us just mention that the consortium includes several ACM PODC and DISC Program Committee Chairs (which are the top int'l conferences in the domain), several DISC and SIROCCO Steering Committee Chairs, and regular Program Committee members of most conferences in distributed computing (ACM PODC, DISC, SIROCCO, OPODIS, SSS, IEEE ICDCS, IPDPS, etc.) and Editorial Board members of top int'l journals (e.g., IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Parallel and Distributed Computing). The consortium also includes the last two winners of the Prize for Innovation in Distributed Computing (2014 & 2015).

The quality of the consortium is more than the sum of the quality of its members. Indeed, the main interest of DESCARTES will be its ability to tackle the main research topics and models in distributed computing in a modular way. The consortium includes two well identified sub-groups, with each of the sub-groups scattered over the three collaborative parterns. Roughly, one sub-group includes individuals

7. The Rennes partner, whose members are mainly from IRISA, includes also experts from LINA, LIF and Vérimag.

dealing with problems related to asynchrony and faults, while the other one includes partners dealing with network computing. It should be mentioned that many participants regularly attend to the same conferences gathering both communities.

We do hope, and actually strongly believe, that the richness of the consortium, beyond the individual quality of its members, and now the strong collaboration between these two sub-groups will achieve the main target of the project, that is a layered or modular model for distributed computing.

Bordeaux. The LaBRI (Laboratoire Bordelais de Recherche en Informatique) is a research unit supported by the CNRS (UMR 5800), the University of Bordeaux, and Bordeaux INP. It includes around 300 members (academics, researchers, PhDs, etc.). The members of the laboratory are grouped in six teams, each one combining basic research, applied research and technology transfer. The LaBRI participants to DESCARTES are all members of the Combinatorics and Algorithmics team, which has an internationally high visibility in combinatorics, graph theory, and distributed computing. More specifically, the LaBRI partner includes world-wide experts on various models and topics in the scope of the project, in particular fault tolerance issues, local computations, network and communication algorithms, and distributed data structures.

Paris. The IRIF (Research Institute on Fundamental Informatics) is supported jointly by the French National Center for Scientific Research (CNRS) and by the University Paris Diderot - Paris 7. IRIF is member of the Foundation Sciences Mathématiques de Paris (FSMP) and of the Paris Research Federation in Mathematics (MP). IRIF hosts two INRIA projects-teams, and is affiliated to the doctoral school in Mathematical Sciences of Paris (ED 386). IRIF results from the merging of the two research units LIAFA and PPS on January 1st, 2016. The scientific objectives of IRIF are at the core of Computer Science, focusing on the following aspects: (1) Mathematical Foundations of Computer Science, (2) Computational Models and Proofs, and (Models, Algorithms and System Design. The one hundred members of IRIF (academics, researchers, and PhDs) are divided in six research teams: Algorithms and Complexity, Combinatorics, Distributed Algorithms and Graphs, Automata and applications, Modeling and verification, and Proofs, programs and systems. DESCARTES involves all members of the team Distributed Algorithms and Graphs whose research interests are dealing with distributed computing. All these people are worldwide experts in distributed computing, covering most of the topics in the field, both theoretically-oriented and practically-oriented (they are all members of the INRIA team-projet GANG).

Rennes. The ASAP (As Scalable As Possible) team, lead by Anne-Marie Kermarrec, is part of IRISA/INRIA Rennes a research unit associated mainly with University of Rennes, CNRS and INRIA. The research activities of ASAP range from theoretical foundations to practical protocols and implementations for (mainly large-scale and dynamic) distributed systems in order to cope with the recent and tremendous evolution of distributed systems. Effectively we observed huge evolutions: (i) Scale shift in terms of system size, geographical spread, volume of data, and (ii) Dynamic behaviour due to versatility, mobility, connectivity patterns. The research of the ASAP Project-Team is along two main themes: Distributed computing models and abstractions and Peer-to-peer distributed systems and applications. These research activities encompass both long term fundamental research seeking significant conceptual advances, and more applied research to validate the proposed concepts against real applications. ASAP group involves 12 people, including two professors, one senior researcher, and two researchers. The proposal involves also three other researchers. Achour Mostefaoui who is coordinating Task 3, and two researchers from Vérimag (Grenoble) and LIF (Aix-Marseille). The first one, S. Devismes, is a specialist of self-* systems. The second one, E. Godard, has interests in distributed computability for static and dynamic networks.

2.3.2 Qualification of the coordinators

Cyril GAVOILLE (LaBRI) **[Global coordinator]**

45 *years old*, *PhD in* 1996
e-mail: gavoille@labri.fr
webpage: dept-info.fr/~gavoille
h-index: 35 (Google Scholar)

Junior member of the prestigious "Institut Universitaire de France" (2009-2014), Cyril Gavoille is currently in charge of a research axis of a LabEx at Bordeaux University, axis including 80 faculties from three labs and a budget of 330 K€/year. He was Deputy Director of the LaBRI (300 people including 150 faculties) in charge of the scientific affairs (2009-2010). He was General Chair of the *30th ACM Symp. on Principles of Distributed Computing* (PODC'10) and treasurer in 2009. He was also co-Chair of the *37th Int'l Coll. on Automata, Languages and Programming* (ICALP'10), and chairman/co-chair for several workshops. He participated in more than 20 international program committee conferences, in particular ESA'11, DISC'12 and '15, SODA'12, PODC'13 and '14, ICALP'13 and '16 for the last few years, and is currently PC Chair of DISC'16. He is Chair of the 2016 Dijkstra Prize Award Committee. He has more than 40 publications in refereed international journals, and more than 90 in refereed international conferences with proceedings.

Selected publications:

1. S. Alstrup, C. Gavoille, E. B. Halvorsen, and H. Petersen. *Simpler, faster and shorter labels for distances in graphs.* In 27th Symposium on Discrete Algorithms (SODA), pp. 338-350. ACM-SIAM Press, 2016
2. I. Abraham, C. Gavoille, A. Gupta, O. Neiman, and K. Talwar. *Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs.* In 46th Annual ACM Symposium on Theory of Computing (STOC), pp. 79-88. ACM Press, 2014
3. I. Abraham, S. Chechik, and C. Gavoille. *Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels.* In 44th Annual ACM Symposium on Theory of Computing (STOC), pp. 1199-1217. ACM Press, 2012

David ILCINKAS (LaBRI) **[Deputy coordinator]**

35 *years old*, *PhD in* 2006
e-mail: david.ilcinkas@labri.fr
webpage: www.labri.fr/perso/ilcinkas
h-index: 19 (Google Scholar)

David Ilcinkas received his PhD in computer science in 2006 from the Paris-Sud University. Since 2007, he is "Chargé de Recherche" CNRS at the LaBRI in Bordeaux. Until June 2016, he is local coordinator for Bordeaux of a 4-year ANR project. He is also currently member of the steering committee of the "Pôle ResCom du GdR ASR". He participated in a dozen of program committees of international events, including DISC, SIROCCO, ICDCN, OPODIS, and currently co-chairs AlgoTel 2016. He is the co-author of 20 publications in refereed international journals and 32 publications in refereed international conferences with proceedings (ICALP, STACS, PODC, DISC, etc.), on distributed computing for networks, with a special emphasis on distributed computing by mobile agents.

Selected publications:

1. E. Bampas and D. Ilcinkas. *On Mobile Agent Verifiable Problems.* In 12th Latin American Theoretical Informatics Symposium (LATIN), to appear, 2016.
2. H. Arfaoui, P. Fraigniaud, D. Ilcinkas, and F. Mathieu. *Distributedly Testing Cycle-Freeness.* In 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), LNCS 8747, pages 15–28, 2014

3. C. Gavoille, C. Glacet, N. Hanusse, and D. Ilcinkas. *On the Communication Complexity of Distributed Name-Independent Routing Schemes.* In 27th International Symposium on Distributed Computing (DISC), LNCS 8205, pp. 418–432, 2013

Corentin TRAVERS (LaBRI) **[Task 1 coordinator]**

35 *years old*, *PhD in* 2007
e-mail: travers@labri.fr
webpage: www.labri.fr/perso/travers
h-index: 17 (Google Scholar)

Corentin Travers received his PhD in computer science in 2007 from University of Rennes. He obtained an associate professor position in 2010 and is member of the LaBRI laboratory, Bordeaux. He is the co-author of 50 publications, including 13 journal papers (SICOMP, Distributed Computing, JPDC, etc.), on fault-tolerance, distributed agreement and simulations. He received two best paper awards (DISC'11, ICDCN'11). He was member of 6 program committees, in particular for the conferences DISC'13, ICDCS'13 and ICDCS'16.

Selected publications:

1. F. Ellen, P. Fatourou, E. Kosmas, A. Milani, and C. Travers. *Universal constructions that ensure disjoint-access parallelism and wait-freedom.* Distributed Computing 1-27, 2016
2. P. Fraigniaud, S. Rajsbaum, and C. Travers. *Locality and checkability in wait-free computing.* Distributed Computing 26(4): 223-242, 2013
3. D. Alistarh, S. Gilbert, R. Guerraoui, and C. Travers. *Of choices, failures and asynchrony: the many faces of set agreement.* Algorithmica 62(1-2): 595-629, 2012

Hugues FAUCONNIER (IRIF) **[Local coordinator]**

61 *years old*, *PhD in* 1982
e-mail: hugues.fauconnier@liafa.univ-paris-diderot.fr
webpage: www.irif.univ-paris-diderot.fr/~hf
h-index: 22 (Google Scholar)

Member of Commission Recherche and Conseil Académique of the Université Paris-Diderot. Hugues Fauconnier is co-responsable of the ANR project DISPLEXITY until June 2016. He is Director of the Computing Departement (UFR d'informatique) at University Paris Diderot. He participated in many international program committee conferences, among them top distributed computing conferences like PODC, DISC, OPODIS, SSS, ICDCS, ICDCN. He has 17 publications in refereed international journals (in particular J. ACM, Distributed Computing, Inf. Comp., TCS, JPDC), and more than 50 in refereed international conferences with proceedings.

Selected publications:

1. C. Delporte-Gallet,H. Fauconnier, E. Gafni, and P. Kuznetsov. *Wait-freedom with advices.* Distributed Computing 28(1): 3-19 (2015)
2. M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg *Partial synchrony based on set timeliness.* Distributed Computing 25(3): 249-260 (2012)
3. C. Delporte-Gallet, H. Fauconnier, and R. Guerraoui. *Tight failure detection bounds on atomic object implementations.* Journal of the ACM 57(4) (2010)

Carole DELPORTE (IRIF) **[Task 2 coordinator]**

56 *years old*, *PhD in* 1983
e-mail: carole.delporte@liafa.univ-paris-diderot.fr

webpage: www.irif.univ-paris-diderot.fr/~cd
h-index: 19 (Google Scholar)

The main research interest of Carole Delporte is distributed computing, and specifically fault tolerance. She participated in many international program committee conferences (PODC, DISC, ICDCS, OPODIS, ICDCN, SSS ...). During the last 10 years she has 17 publications in refereed international journal (J. ACM, Distributed Computing, TCS, ...), and 54 in referred international conferences with proceeding (PODC, DISC, ICDCS, OPODIS, ...). She is co-Chair of Netys'15 Conference this year. She is in charge of the Master 2 of computer science at the university of Paris Diderot.

Selected publications:

1. C. Delporte-Gallet, H. Fauconnier, P. Kuznetsov, and E. Ruppert. *On the space complexity of set-agreement.* In 34th ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, pp. 271-280, 2015.
2. C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, A.-M. Kermarrec, E. Ruppert, and H. Tran-The. *Byzantine agreement with homonyms.* Distributed Computing 26(5-6): 321-340 (2013)
3. C. Delporte-Gallet, H. Fauconnier, and S. Toueg. *The minimum information about failures for solving non-local tasks in message-passing systems.* Distributed Computing 24(5): 255-269 (2011)

Pierre FRAIGNIAUD (IRIF) **[Task 4 coordinator]**

53 *years old*, *PhD in* 1990
e-mail: pierre.fraigniaud@liafa.univ-paris-diderot.fr
webpage: www.irif.jussieu.fr/~pierref
h-index: 43 (Google Scholar)

Pierre Fraigniaud is "Directeur de Recherche" CNRS. Since 2010, he is director of the "Research Institute on Fundamental Informatics" (IRIF) at University Paris Diderot. His main research interest is parallel and distributed computing, and specifically the design and analysis of distributed algorithms and data structures for networks. He is member of the Editorial Boards of the journals *Distributed Computing* (DC), and *Theory of Computing Systems* (TOCS). He was Program Committee Chair for the *41st Int'l Coll. on Automata, Languages and Programming* (ICALP'14, Track C), the *30th ACM Symp. on Principles of Distributed Computing* (PODC'11), the *19th Int'l Symp. on Distributed Computing* (DISC'05) and the *13th ACM Symp. on Parallel Algorithms and Architectures* (SPAA'01). He was Management Committee Chair of the European COST Action 295 "DYNAMO" on Algorithmic Aspects of Dynamic Networks. In 2012, he received the Silver Medal from CNRS, and, in 2014, the Prize for Innovation in Distributed Computing.

Selected publications:

1. P. Fraigniaud, G. Giakkoupis. *Greedy routing in small-world networks with power-law degrees.* Distributed Computing 27(4): 231-253 (2014)
2. P. Fraigniaud, A. Korman, and D. Peleg. *Towards a complexity theory for local distributed computing.* Journal of the ACM 60(5): 35 (2013)
3. P. Fraigniaud, S. Rajsbaum, and C. Travers. *Locality and checkability in wait-free computing.* Distributed Computing 26(4): 223-242 (2013)

Michel RAYNAL (IRISA) **[Local coordinator]**

66 *years old*, *PhD in* 1975
e-mail: michel.raynal@irisa.fr
webpage: www.irisa.fr/prive/raynal
h-index: 52 (Google Scholar)

Michel Raynal is a professor of computer science at university of Rennes. His main research interests are (a) distributed algorithms, distributed computability, dependability, and (b) the fundamental principles that underlie the design and the construction of distributed computing systems. He belongs to the editorial board of five international journals, including IEEE Transactions and the *Journal of Parallel and Distributed Computing*. During his career he was a member of more than ten international conferences steering committees (including ACM PODC, DISC, IEEE ICDCS), chaired more than 20 international conferences, and was a PC member of more than 200 international conferences. Since 2010, he has written four books (two published by Morgan & Claypool, anf two by Springer), and received four *Best Paper Awards* (PODC'15, DISC'10, SSS'11, and Europar'10). In 2013 he was appointed as an *adjunct* professor at the Hong Kong Polytechnic University, and received the 2015 Prize for Innovation in Distributed Computing. Michel Raynal is a senior member of the prestigious "Institut Universitaire de France" and a member of Academia Europaea.

Selected publications:

1. M. Raynal, *Distributed algorithms for message-passing systems.* Springer, 510 pages, 2013 (ISBN: 978-3-642-38122-5).
2. M. Herlihy, S. Rajsbaum, M. Raynal, *Power and limits of distributed computing shared memory models.* Theoretical Computer Science, 509:3-24, 2013.
3. A. Mostéfaoui, H. Moumen, M. Raynal, *Signature-free asynchronous binary Byzantine consensus with $t < n/3$, $O(n^2)$ messages, and $O(1)$ expected time.* Journal of ACM, 62(4), Article 31, 21 pages, 2015.

Achour MOSTEFAOUI (LINA) **[Task 3 coordinator]**

47 *years old*, *PhD in* 1994
e-mail: achour.mostefaoui@univ-nantes.fr
webpage: www.univ-nantes.fr/mostefaoui-a
h-index: 31 (Google Scholar)

Achour Mostefaoui is Professor at the University of Nantes. He is co-head of the GDD research team within the LINA Lab. He is also in charge of the Bachelor program of computer science at the university of Nantes. His reserach interests include calculability, synchronization and fault-tolerance issues in asynchronous fault-prone distributed systems. He published more than 30 papers in the most prestigious journals (e.g. JACM, SICOMP, Distributed Computing, IEEE TPDS, IEEE TC, JPDC) and more than 80 conferences including (STOC, PODC, DISC, DSN, ICDCS, IPDPS, SPAA). He participated in more than 30 international program committee conferences (e.g. DISC, ICDCS, EUROPAR, EDCC, SIROCCO, NCA).

Selected publications:

1. D. R. Kowalski and A. Mostéfaoui. *Synchronous byzantine agreement with nearly a cubic number of communication bits: synchronous byzantine agreement with nearly a cubic number of communication bits.* In 32th ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, pp. 84-91, 2013
2. A. Mostéfaoui, M. Raynal, and C. Travers. *Narrowing power vs efficiency in synchronous set agreement: Relationship, algorithms and lower bound.* Theoretical Computer Science 411(1): 58-69 (2010)
3. A. Mostéfaoui, S. Rajsbaum, M. Raynal, and C. Travers. *The Combined Power of Conditions and Information on Failures to Solve Asynchronous Set Agreement.* SIAM Journal on Computing 38(4): 1574-1601 (2008)

2.4 Scientific and technical justification of the requested ressources

Preliminary notes. To simplify the different cost estimations, we will assume, on one hand, that the participation of all members will be perfectly regular and equivalent (for example, no member will ever miss a project meeting) and, on the other hand, that only the permanent members and the staff paid by the ANR will participate in the project (for example, the costs induced by future PhD students who are not paid by the ANR but who participates in the project is not considered).

In the following, we will use the average participation rate in the project, which we define as the average, taken over all the permanent members of the project, of their participation rate to the project. Bordeaux, Paris, and Rennes invest respectively 134, 110, and 103 man-months, while the total of 16 permanent members could have provided 16×48 man-months. The average participation rate is thus 348/768, which is roughly 45.3%. All calculations will be rounded to the closest multiple of 100 €.

Since most of the requested resources are estimated in the same manner for each of the three partners, we present the following justification of the requested resources according to the type of resources, and for each of these, partner by partner, rather than in the opposite way.

2.4.1 Equipment

We will provide a working station (a laptop, a screen, and a docking station) to the person (PhD student or postdoctoral fellow) funded by the ANR. Its cost is estimated to 2 500 €. Considering the duration of the project, the permanent members will also need the renewal of their working station. However, the average participation rate must be taken into account in this case. This leads to the following estimations:

For Bordeaux: $(1 + 6 \times 0.453) \times 2\,500$ € $= 9\,300$ €.
For Paris: $(1 + 5 \times 0.453) \times 2\,500$ € $= 8\,200$ €.
For Rennes: $(1 + 5 \times 0.453) \times 2\,500$ € $= 8\,200$ €.

2.4.2 Staff

PhD student and Master internships for Bordeaux.

PhD student. We request the funding of a PhD position, which will be directly related to Task 1 and Task 4. The PhD student will start by working on Task 1 and will progressively shift towards Task 4, with a continuous focus on the interfaces. More precisely, the student's first work will consist of the study of the various simulations connecting different models of distributed computing, with an emphasis on their complexity. His/her focus will then move to Task 4, in which the student will be mostly involved in evaluating the performances degradation incurred by the Layered Distributed Computing (LDC) model. The student will start the PhD at the end of Year 1, in order to take benefit of the initial progress made in the project.
The cost incurred by this PhD position is estimated to 95 000 €.

Master internships. They will be proposed in relation with the different aspects of the project. We expect to supervise two master students in total, for a four months period each. Considering an expected cost of 560 €/month, we obtain a total of $2 \times 4 \times 560$ € $= 4\,500$ €.

PhD student and Master internships for Paris.

PhD student. We request the funding of a PhD position, which will be directly related to Task 2 and Task 4. To benefit of the first outputs of Task 2, the thesis will begin six months after the beginning of the project and will be devoted to the formalization of services of Task 4.

More precisely, as a preliminary work the PhD will be involved in the complete survey of the distributed computing hierarchies and their modular organisation (Task 2). Then he will focus more specifically on two different frameworks namely: *synchronous fault-free distributed network computing* (i.e., the LOCAL and CONGEST models [Pel00]) and *asynchronous crash-prone shared memory computing* (i.e., the WAIT-FREE and t-RESILIENT models [AW04]) and will try to produce fragments of the Layered Distributed Computing (LDC) model (Task 4) for these frameworks. Finally he will then have enough of experience to be able to produce some other modules of the LDC as on Figure 1 (Task 4).
The cost incurred by this PhD position is estimated to 95 000 €.

Master internships. They will be proposed in relation with the different aspects of the project. We expect to supervise two master students in total, for a four months period each. Considering an expected cost of 560 €/month, we obtain a total of $2 \times 4 \times 560$ € = 4 500 €.

Postdoc and Master internships for Rennes.

Postdoc. We request a 12-month funding for a post-doc position, which will be related mainly to tasks 1, 3 and 4. The cost of this post-doc is estimated to 55 000 euros.
The post-doc is planned to start after 12 months (or 18, but no more) to benefit as much as possible from the early results of the project. The aim will be to investigate, from a distributed software point of view, the impact due to failures and asynchrony on the layered architecture of distributed computing systems. A first step in this direction will address the notions of *message adversaries* (introduced a long time ago in [SW89]) and *process adversaries* (investigated in several papers, e.g. [DFGT11, IR11, JM13, Kuz12, RS13]). The aim of these notions is to consider failures, not as "bad events", but as part of the normal behavior of a system. As an example, when considering round-based algorithms, a message adversary is a daemon which, at every round, is allowed to suppress some messages. The aim is then, given a problem P, to find the strongest adversary under which P can be solved ("strongest" means here that giving more power to the adversary makes the problem impossible to solve). As we can see the work of the post-doc will be to enlarge both the proposed *layered* approach for distributed computing (task 4), and the *map* of distributed computing models (task 1). Notice early efforts in this direction appeared in [AG13, RS13].

Master internships. They will be proposed in relation with the different aspects of the project. We expect to supervise two master students in total, for a four months period each. Considering an expected cost of 560 €/month, we obtain a total of $2 \times 4 \times 560$ € = 4 500 €.

This gives the following total for the "Staff" budget:

	Total	PhD & Postdoc	Masters internships
Bordeaux	99 500 €	95 000 €	4 500 €
Paris	99 500 €	95 000 €	4 500 €
Rennes	63 300 €	58 800 €	4 500 €

2.4.3 Missions

To achieve a better estimation, we divide the missions into four categories.

Project plenary meetings. As explained in Task 0, we will organize seven 2-day project plenary meetings, a workshop collocated with a PODC conference, and a final 1-week school. Due to the uncertainty about the location of the future PODC conferences, and the exact form of the season

school, these two special events are treated as the ordinary plenary meetings as far as budget is concerned.

Based on an average cost of 300 € per meeting per member (weighted by the average participation rate), and remembering that the persons (the two PhD students and the postdoctoral fellow) funded by the ANR will not participate the four years, we obtain for this category:

For Bordeaux: $(6 \times 0.453 \times 9 + 1 \times 6) \times 300$ € = 18 000 €.
For Paris: $(5 \times 0.453 \times 9 + 1 \times 6) \times 300$ € = 15 300 €.
For Rennes: $(5 \times 0.453 \times 9 + 1 \times 2) \times 300$ € = 14 100 €.

Inter-partner meetings. We estimate that each member will perform, in average, two 1-week visit to an other partner of the project. Based on an average cost of 600 € for such a mission, and again taking into account that the staff funded by the ANR will not participate the four years, we obtain for this category:

For Bordeaux: $(6 \times 2 + 1 \times 2) \times 600$ € = 8 400 €.
For Paris: $(5 \times 2 + 1 \times 2) \times 600$ € = 7 200 €.
For Rennes: $(5 \times 2 + 1 \times 1) \times 600$ € = 6 600 €.

International exchanges. In the framework of this project, we plan to collaborate with international experts in various aspects of distributed computing and distributed systems. For each of the three partners, we request the budget for one short (typically two weeks long) international visits (from or to abroad) per year. Considering a cost of 1 500 € per such visit, we obtain for this category:

For Bordeaux: $4 \times 1\,500$ € = 6 000 €.
For Paris: $4 \times 1\,500$ € = 6 000 €.
For Rennes: $4 \times 1\,500$ € = 6 000 €.

Dissemination. It is expected that most of the results will be communicated to top-level international conferences in distributed computing, but also to workshops, GdR meetings, etc. This category being difficult to estimate, we propose the following reasoning. Over the last four years, the number of publications in international conferences by all the permanent members of the project is around 200. We consider an average cost of 1 500 € for each of these missions to attend an international conference. Due to co-authoring effects, we consider that each member attends a conference in half of the cases. We multiply now by the average participation rate and we obtain a total budget, for the three partners, of 68 100 € for this category. This total amount is then distributed among the three partners in proportion of the number of man-months they invest:

For Bordeaux: $(134/348) \times 68\,100$ € = 26 300 €.
For Paris: $(110/348) \times 68\,100$ € = 21 600 €.
For Rennes: $(103/348) \times 68\,100$ € = 20 200 €.

To summarize, the "Missions" budget is:

	Total missions	Project plenary meetings	Inter-partner meetings	International exchanges	Dissemination
Bordeaux	58 700 €	18 000 €	8 400 €	6 000 €	26 300 €
Paris	50 100 €	15 300 €	7 200 €	6 000 €	21 600 €
Rennes	46 900 €	14 100 €	6 600 €	6 000 €	20 200 €

2.4.4 Other expenses

We estimate that 1 500 € per partner should be sufficient for all small expenses of the project such as books, backup hard drives, spare laptop batteries, etc.

2.4.5 Total

The total amount requested to the ANR, considering the extra 8% of managing and "structural" fees is:

	Total+8%	Equipment	Staff	Missions	Other expenses
Bordeaux	182 520 €	9 300 €	99 500 €	58 700 €	1 500 €
Paris	172 044 €	8 200 €	99 500 €	50 100 €	1 500 €
Rennes	129 492 €	8 200 €	63 300 €	46 900 €	1 500 €
	484 056 €				

3 Exploitation of results, global impact of the proposal

DESCARTES is aiming at developing fundamental research in distributed computing susceptible to have a strong impact on the *conception*, *analysis*, and *maintenance* of robust and efficient distributed computing systems. As we said before, it is foreseen that the outcome of the project will not only be *conceptual*, but will also eventually produce breakthroughs in several *practical aspects* of the conception of distributed computing systems, in a way similar to OSI-like models, which are not only conceptual but also at the core of the practical design of actual communication systems. More specifically, the exploitation of results and the global impact of the proposal can be organized along three complementary axes: conception techniques, dissemination of knowledge, and bottleneck identification.

3.1 Design techniques

The Layered Distributed Computing (LDC) model as detailed in Task 4 is expected to be accessible to non experts in distributed computing, with the help of a complete specification of the services and interfaces, together with the graph articulating them. Analogically to the OSI model in which the designer of an application does not need to know the way data frames are treated by the data link layer, it is expected that, using the LDC model, a designer of an application will only need to be aware of the specifications of the services she is aiming at using, and of the context of execution in which those services will be invoked.

As we pointed out in the description of Task 4, we are essentially aiming at providing a *proof of concepts*, by decomposing classical distributed computing frameworks in modular constructs, and, expectedly, using the LDC model for connecting frameworks that are typically treated completely separately by the distributed computing experts. Nevertheless, beyond the proof of concepts, we are ultimately aiming at providing *implementations* of some of our services, e.g., under the form of Java class libraries, inspired by the work in Java Concurrency, and in particular the efforts by the community for formalizing concurrent data structures in the Java framework (cf., Herlihy and Shavit's book the art of multiprocessor programming [HS08]). That is, we shall take the package *java.util.concurrent* for shared memory applications as a reference source of inspiration for designing Java packages capturing the essence of our services. We expect that such packages will be easily handled by programmers with only basic knowledge in distributed computing.

3.2 Dissemination of knowledge

DESCARTES will also aim at significantly contributing to the dissemination of knowledge, including the way distributed computing systems are taught to students at universities and engineer schools. The modular partitioning of distributed computing systems that is expected to be eventually delivered by the project will indeed provide an ideal teaching framework enabling distributed computing systems to be approached progressively, by considering the difficulties of distributed computing separately but in a consistent and well articulated manner. Importantly, the modular decomposition of the distributed computing landscape that should come out of the project will enable teaching only parts of the topic, without discarding crucial topics, but by discarding topics which have a lower significance with respect to the teaching program. For instance, in teaching programs dedicated to the design of distributed applications (say, for the cloud), only services that are related to this "level" will be taught. Instead, in teaching programs dedicated to the design of system protocols addressing lower levels of the system architecture, a different set of services might be taught. More precisely, it may happen that the same services will be taught in both teaching programs, but with different interfaces, dedicated to each specific environment of executions in connection with the specificities of the course.

In connection to the above, DESCARTES is aiming at organizing a *1-week season school* on Year 4, dedicated to Master and PhD students, during which the outcome of the project will be taught.

3.3 Bottleneck identification

By modularizing the conception and analysis of distributed computing systems, DESCARTES will also enable to identify the *bottlenecks* for improving the performances and usages of distributed computing systems. The modular approach should indeed facilitate the identification of the services and/or interfaces to be enhanced in order to improve the global behavior of the system. Interestingly, the modular approach should help significantly in identifying the exact targets requiring deeper scientific investigations in the very precise and well specified terminology of services and interfaces. This should in turn enable the distributed computing community to address well defined issues instead of wondering around in the jungle of concepts, models, problems, and constraints typical of distributed computing.

3.4 Diffusion

The main expected results of this project are scientific advances in the investigated research fields. Research results are expected to be presented at the leading international conferences and in the leading scientific journals. In particular, the annual flagship conference in distributed computing, namely the *ACM Symposium on the Principles of Distributed Computing* (PODC) is inviting proposals for workshops that are closely related to the scope of the conferences, i.e., on topics related to the theory, design, specification or implementation of distributed systems (these workshops are in conjunction with the conference on the day immediately preceding or following the conference program). We plan to organize such a workshop, on DESCARTES-related topics, which will be an opportunity to promote our scientific results and to draw the attention of the international community. This will be done on Year 3 of the project.

Also, DESCARTES will maintain the project web site that will make available the project results. In particular, it will be a useful portal open to all researchers for accessing papers and other materials.

4 References

[AD02] Yehuda Afek and Shlomi Dolev. Local stabilizer. *Journal of Parallel and Distributed Computing*, 62(5):745–765, May 2002.

[ADFT08] Marcos Kawazoe Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. On implementing omega in systems with weak reliability and synchrony assumptions. *Distributed Computing*, 21(4):285–314, 2008.

[AG13] Yehuda. Afek and Eli Gafni. Asynchrony from synchrony. In *Proc. Int'l Conference on Distributed Computing and Networking (ICDCN'13)*, number 7730 in LNCS, pages 225–239. Springer, 2013.

[AGM+08] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Transactions on Algorithms*, 3(4):Article No. 37, June 2008.

[AP90] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 514–522. IEEE Computer Society Press, October 1990.

[AW04] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics (2nd ed.)*. Wiley-Interscience Publication, 2004.

[Awe85] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[BDDT07] Joffroy Beauquier, Sylvie Delaët, Shlomi Dolev, and Sébastien Tixeuil. Transient fault detectors. *Distributed Computing*, 20(1):39–51, 2007.

[BF15] Lélia Blin and Pierre Fraigniaud. Space-optimal time-efficient silent self-stabilizing constructions of constrained spanning trees. In *35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015.

[BFPS14] Lélia Blin, Pierre Fraigniaud, and Boaz Patt-Shamir. On proof-labeling schemes versus silent self-stabilizing algorithms. In *16th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 18–32, 2014.

[BG93] Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 91–100, May 1993.

[BGLR01] Elizabeth Borowsky, Eli Gafni, Nancy A. Lynch, and Sergio Rajsbaum. The BG distributed simulation algorithm. *Distributed Computing*, 14(3):127–146, 2001.

[BN84] Andrew Birrell and Bruce Jay Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, 1984.

[CT96] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[DFGT11] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Andreas Telmann. The disagreement power of an adversary. *Distributed Computing*, 24(3–4):137–147, 2011.

[DG02] Partha Dutta and Rachid Guerraoui. Fast indulgent consensus with zero degradation. In *4th European Dependable Conference Computing (EDCC)*, pages 191–208, October 2002.

[DGGS99] Assia Doudou, Benoît Garbinato, Rachid Guerraoui, and André Schiper. Muteness failure detectors: Specification and implementation. In *3rd European Dependable Conference Computing (EDCC)*, pages 71–87, September 1999.

[EPSW14] Yuval Emek, Christoph Pfister, Jochen Seidel, and Roger Wattenhofer. Anonymous networks: randomization = 2-hop coloring. In *33rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 96–105. ACM Press, July 2014.

[FF05] Laurent Feuilloley and Pierre Fraigniaud. Randomized local network computing. In *27th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM Press, 2005. To appear.

[FKP13] Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *Journal of the ACM*, 60(5):Article No. 35, October 2013.

[FLP85] Michael J. Fisher, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[Gaf09] Eli Gafni. The extended BG-simulation and the characterization of t-resiliency. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 85–92. ACM Press, 2009.

[GG11] Eli Gafni and Rachid Guerraoui. Generalized universality. In *22nd International Conference on Concurrency Theory (CONCUR)*, pages 17–27, September 2011.

[GL08] Rachid Guerraoui and Nancy A. Lynch. A general characterization of indulgence. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4), 2008.

[GR07] Rachid Guerraoui and Michel Raynal. The alpha of indulgent consensus. *Computer Journal*, 50(1):53–67, 2007.

[GS11] Cyril Gavoille and Christian Sommer. Sparse spanners vs. compact routing. In 23^{rd} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 225–234. ACM Press, June 2011.

[Gue00] Rachid Guerraoui. Indulgent algorithms (preliminary version). In *19th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 289–297, July 2000.

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124–149, 1991.

[HS08] Maurice Herlihy and Nir Shavit. *The art of multiprocessor programming.* Morgan Kaufmann, 2008.

[IR09] Damien Imbs and Michel Raynal. Visiting Gafni's reduction land: from the BG simulation to the extended BG simulation. In *11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 5873 of Lecture Notes in Computer Science, pages 369–383. Springer-Verlag, 2009.

[IR11] Damien Imbs and Michel Raynal. A liveness condition for concurrent objects: x-wait-freedom. *Concurrency and Computation: Practice and Experience*, pages 2154–2166, 2011.

[JM13] Flavio Junquera and Keith Marzullo. A framework for the design of dependent-failure algorithms. *Concurrency and Computation: Practice and Experience*, 19(17):2255–2269, 2013.

[JT08] Prasad Jayanti and Sam Toueg. Every problem has a weakest failure detector. In *27th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 75–84. ACM Press, 2008.

[KKP10] Amos Korman, Shay Kutten, and David Peleg. Proof labeling system. *Distributed Computing*, 22(4):215–233, May 2010.

[Kuz12] Petr Kuznetsov. Understanding non-uniform failure models. *Bulletin of EATCS*, 106:53–77, 2012.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[Lin92] Nathan Linial. Locality in distributed graphs algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[Lyn96] Nancy A. Lynch. *Distributed algorithms.* Morgan Kaufmann, 1996.

[NS95] Moni Naor and Larry Stockmeyer. What can be computed locally. *SIAM Journal on Computing*, 24(6):1259–1277, December 1995.

[Pel00] David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM Monographs on Discrete Mathematics and Applications, 2000.

[PS89] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[PU89a] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18(4):740–747, 1989.

[PU89b] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[Ray10] Michel Raynal. *Communication and Agreement Abstractions for Fault-Tolerant Asynchronous Distributed Systems.* Morgan & Claypool Publishers, 2010.

[Ray13a] Michel Raynal. *Concurrent programming: algorithms, principles, and foundations.* Springer, 2013.

[Ray13b] Michel Raynal. *Distributed algorithms for message-passing systems.* Springer, 2013.

[Ray14] Michel Raynal. What can be computed in a distributed system? In *Workshop "From Programs to Systems: The Systems Perspective in Computing" in honor of Professor Joseph Sifakis*, volume 8415 of Lecture Notes in Computer Science, pages 209–224. Springer-Verlag, 2014.

[RS13] Michel Raynal and Julien Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proc. 32nd ACM Symposium on Principles of Distributed Computing (PODC '13)*, pages 53–77. ACM Press, 2013.

[RST14] Michel Raynal, Julien Stainer, and Gadi Taubenfeld. Distributed universality. In *18th International Conference on Principles of Distributed Systems (OPODIS)*, volume 8878 of Lecture Notes in Computer Science, pages 469–484. Springer, 2014.

[RT15] Liam Roditty and Roei Tov. New routing techniques and their applications. In 34^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, July 2015. To appear.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

[SW89] Nicolas Santoro and Peter Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS'89)*, number 349 in LNCS, pages 304–316. Springer, 1989.

[Tau06] Gadi Taubenfeld. *Synchronization algorithms and concurrent programming*. Pearson Education/Prentice Hall, 2006.

[Tel01] Gerard Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2nd edition, 2001.

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.