# Spray Computers: Frontiers of Self-Organization for Pervasive Computing


Franco Zambonelli[1], Marie-Pierre Gleizes[2], Marco Mamei[1], Robert Tolksdorf[3]

*1) DISMI - Università di Modena e Reggio Emilia – Reggio Emilia – ITALY*

*2) IRIT – Universitè Paul Sebatier – Toulouse – France*

*3) Dept. Computer Science - Freie Universitat of Berlin – Berlin - Germany*



## Abstract

*We envision a future in which clouds of microcomputers can be sprayed in an environment to provide, by spontaneously networking with each other, an endlessly range of futuristic applications. However, beside the vision, spraying may also act as a powerful metaphor for a range of other scenarios that are already under formation, from ad-hoc networks of embedded and mobile devices to worldwide distributed computing. After having detailed the different spray computers scenarios and their applications, this paper discusses the issues related to the design and development of spray computer applications, issues which call for novel approaches exploiting self-organization and emergent behaviors as first-class tools. Finally, this paper presents the key research efforts being taken in the area and attempt at defining a rough research agenda.*



***Keywords****: Spray Computers, Bottom-up Software Engineering, Self-organization, Emergent Behaviors.*


## 1. Introduction

With the MEMS revolution in full swing, micro-sensors are now following manufacturing curves that are at least related to Moore's Law [Pis00]. This trend, when combined with both the push for low power communication and computation devices and for the ubiquitous provisioning of data and services, pave the way for the *spray computers* revolution.

It is not hard to envision a future in which network of micro computers will be *literally* sold as spray cans, to be sprayed in an environment or on specific artifacts to enrich them with functionalities that, as of today, may appear futuristic and visionary [Nag03, ZamM02, Pis03]. The number of potential applications of the scenario is endless, ranging from smart and invisible clothes, intelligent interactive environments, self-assembly materials and self-repairing artifacts.

However, the vision of spray computers may also act as a powerful metaphor for a range of other scenarios that are already under formation. These include distributed applications in embedded, possibly mobile, ad hoc networks, as well as distributed service- and data-oriented activities on the Internet [RowD01, Rat01]. In fact, besides the different physical scale of the components involved and of their interactions (from micro-computers interacting within networks extending across a few meters, to Internet hosts interacting at a world-wide scale), all of these types of spray computer networks raise the same challenges as far as development and deployment of applications is involved, calling for radically novel approaches to distributed systems development and management.

On the one hand, to avoid the unaffordable efforts related to the placement, configuration, and maintenance of such systems, there is the need of approaches enabling of deploying components without any a priori layout effort, and letting components to self-organize their application activities and self-retune their overall behavior depending on specific contingencies (e.g., localized faults and environmental changes) [KepC03]. On the other hand, the autonomous and decentralized nature of the activities in such scenarios, together with the possibly unpredictable dynamics of the operating environments, is likely to make those systems exhibit unexpected, "emergent" behaviors - as recent observations in several types of decentralized networks (i.e., the Internet, the Web, as well as Gnutella) suggest. Therefore, there is also need for methodologies to predict and control the emergence of such behaviors and, when possible, offensively exploit them for the achievement of otherwise impossibly complex distributed tasks [BerGPP02, ParBS02, ZamMR03].

This paper aims at exploring the above issues and will be organized as follow: Section 2 details our vision about spray computers, starting from the micro-scale (i.e., literally sprayable computers), to the medium scale (smart artifacts and MANETs), up to the macro-scale (wide-area networks). Section 3 will present and discuss the major issues arising in the exploiting self-organization for the design and development of applications for spray computers. Section 4 will briefly present a few representative research efforts being taken in this area, and their limitations. Section 5 concludes the paper by attempting to define a roadmap of activities in the area of spray computers.

## 2. Spray Computers and Applications

The concept of spray computers will soon pervade the ICT scenarios at every scale and at every level. In the following we will briefly survey our idea of future computer-based systems from the micro-scale (literally spray computers), to the medium-scale (handheld and wearable computers) to the global scale (Internet and Web computing).

### 2.1 The Micro Scale

As proved in the context of the Smart Dust project at Berkeley [BerG97, Pis00], it is already possible to produce fully-fledged computer-based systems of a few $mm^3$, and even much smaller ones will be produced in the next few years. Such computers, which can be enriched with communication capabilities (radio or optical), local sensing (e.g., optical, thermal, or inertial) and local effecting (e.g., optical and mechanical) capabilities, are the basic ingredients of our spray computers vision.

Spray computers, as we imagine them, are clouds of sub-millimeter-scale microcomputers, to be deployed in an environment or onto specific artifacts via a spraying or a painting process. Once deployed, such components will spontaneously network with each other and will coordinate their actions (i.e., local sensing and effecting) to provide specific "smart" functionalities. We imagine it will be possible, say in 2020, to go to the local store and there buy, for a few dollars, a "pipe repairing" spray, made up of a cloud of MEMS microcomputers capable of navigating in a pipeline, recognizing the presence of holes, and self-assembling with each other so as perfectly repair the pipe. Similarly, we could imagine a spray to transform our everyday desk into an active one, capable of recognizing the positions and characteristics of objects placed on it and letting them meaningfully interact.

Another peculiar application we envision is the "spray of invisibility" (described in [ZamM02]): a spray of micro devices capable of receiving and re-transmitting light emissions in a directional way, and capable of interacting with each other via short-range wireless communications. When an object is covered by a layer of such spray, the emissions of the devices make external observers perceive exactly the same light configurations that they would have perceived if there were nothing in between. In fact sensors on the rear side of the object can receive such configurations and, via distributed coordination, can communicate them to emitters on the observer's side to be retransmitted. Other types of application one could envision include any type of self-assembly artifact [Nag03], there included thing like Terminator T-1000, the nano-swarms of Michael Crichton's novel "Prey" [Cri02], and MEMS-based artificial immune systems and drugs [Pis03].

Whatever the applications one envision, the key characteristics that will distinguish spray computers applications from traditional distributed computing systems are not – as one could at first think – the scale at which processes take place and the fact that processes are likely to be situated in a physical environment and have to strongly interact with the physical world. After all: *(i)* the fact that a process executed on a micro device rather than on a high-end computer does not change it basic nature; *(ii)* distributed computing systems traditionally have to carry on their activities while being situated in a computational environment and have to interact with it. Instead, what we think strongly distinguish spray computers are the facts that:

- their computational activities take place in a network whose structure derives from an almost random deployment process (as a spraying process is), and that is likely to change over time with unpredictable dynamics (due to environmental contingencies, failure of components, or simply mobility of components);
- the number of (hardware and, consequently, software) components involved in a distributed application is dramatically high and hardly controllable. It is neither possible to enforce a strict configuration of software components nor to control their behavior during execution at a fine-grained level.

Both the above characteristics compulsory call for execution models in which applications are made capable of self-configuring and self-tuning their activities in a spontaneous and unsupervised way, adapting to whatever network structure and surviving network dynamics.

### 2.2 The Medium Scale

Besides micro devices to be literally sprayed, spray computers can also act as a power metaphor for describing the key characteristics of the emerging scenarios of ubiquitous and pervasive computing, as enabled by handheld, wearable, and embedded, networked computing systems.

We already typically carry on two or three computers (i.e., a cell phone, a laptop, and possibly a PDA).

Also, our houses our already populated by a variety of microprocessor based furniture (e.g. TVs, phones, etc.). However, at the moment, the networking capabilities of these computer-based systems are under-exploited. Very soon, our world will be densely populated by personal-area networks (e.g., the ensemble of Bluetooth enabled interacting computer-based components we could carry on or we could find in our cars), local ad-hoc networks of handheld computers (e.g., networks of interacting PDAs carried by a team that have to directly interact and coordinate with each other in an open space), and furniture networks (e.g., Web-enabled fridges and ovens able to interact with each other and effectively support our cooking activities in a coordinated way).

What we want to emphasize here is that the above types of networks, although being formed by different types of computer-based devices (let's say, medium-end computers) and at different physical scales than literally spray computers, shares with them the same issues as far as the development and management of distributed applications is concerned. In fact:

- most of these networks will be wireless, with structures dynamically varying depending on the relatives positions of devices, all of which intrinsically mobile (the persons in an ad-hoc network can move around in an environment and the position of home furniture can changed on needs) and characterized by the dynamic arrival dismissing of nodes (a PDA running out of power or a new home furniture being bought).
- even if technically possible, it is simply not commercially and economically viable to consider deploying applications that would require explicit configuration and explicit tuning to meet the amorphous and dynamic nature of the networks in which applications will be expected to operate.

Also in these cases, new approaches are needed to develop applications as if they were to execute on a network of spray computers.

### 2.3 The Global Scale

Also in the case of macro-scale networks made up of high-end computer systems, i.e., the Internet and the Web, the dramatic growth of these networks and of the information and traffic to be managed, together with the increasing request for ubiquitous connectivity and the peculiar structures exhibited by such networks [AlbJB00, RipIF02], have recently raised researchers' attention to the need of novel approaches to distributed systems management.

Traditional approaches to management, requiring human configuration efforts and supervision, fall shorts when the number of nodes in the network (or the number of interrelated services and links in the Web) grows in a fully decentralized way, and when the presence of the nodes in a network is of an intrinsically ephemeral nature, as it is the case of laptops and, with regard to the Web, of several non-commercial data and services. In particular, the need to access data and services according to a variety of patterns and independently of the availability/location of specific servers calls for P2P approaches to distributed application development centered on the idea of overlay networks. The idea (promoted by first generation P2P systems such as Gnutella [RipIF02], and later improved by second-generation P2P systems such as Pastry [RowD01] and CAN [Rat01]) is to have data and services organized in sorts of spontaneously organized virtual networks of acquaintances. The key assumption is that the allocation of software components in need to interact with each other (think, e.g., at file-sharing applications) can be intrinsically amorphous and dynamic, i.e., composed by an unpredictable number of possibly unknown peers placed almost anywhere in the physical network, as if it were a network of spray components. Thus, instead of promoting strict control over the execution of single software components and of their interactions, the idea of overlay network is to promote and support adaptive organization and maintenance of a structured network of logical relationships among components, to abstract from the physical "sprayed" nature of the actual network and survive events such the arrival of new nodes or the dismissing of some nodes.

Overlay networks are currently the most widely investigated approach to promote unsupervised and adaptive approach to distributed application management, and are leveraging a variety of useful applications facilitating access to (and coordination over) a variety of world-wide distributed data and services, they may not be necessarily the only and best approach. In any case, the great deal of attention towards self-organizing overlay network is the body of evidence of the need of novel approaches to distributed application development.

As a final note, we emphasize that, although the micro, medium, and global scale currently represent almost separated worlds, this will not be the case in the near future. All the above systems will probably be in the near future part of a mega decentralized network, including traditional Internet nodes, smart computer-enriched objects and furniture, networks of embedded and dispersed micro-sensors. For instance, the IPv6 addressing scheme will make it possibly to assign an Internet address to every cubic millimeter in the earth surface [ImiG00], thus opening the possibility for each and every computer-based component to become part of a single worldwide network.

## 3 Programming Spray Computers

Programming a spray computer means to engineer a pre-specified, coherent and useful behavior from the cooperation of an immense number of unreliable parts interconnected in unknown, irregular, and time-varying ways. This translates in devising algorithms and control methodologies to let the sprayed computing devices *self-organize* their interaction patterns and their activities: devices have to start working together without the presence of any a-priori global supervisor or centralized facility.

The basic low-level mechanism upon which to rely to enable self-organization appears quite well-understood and are basically the same whatever the scale, whether that of sensor networks or that of wide-area P2P computing. Among the others: dynamic discovery of potential communication partners and of available services via broadcasting; localization and navigation in some sorts of spatial environment, whether physical (as in sensor networks) or computational (as in P2P systems).

What is still missing is an assessed understanding of how to design, develop, and manage, self-organizing applications manage these kinds of systems, leading to some general purpose methodologies and programming environments. The main conceptual difficulty being that, while spray computers enable a direct-engineered control only on components' local activities, a variety of diverse application goals have to be achieved at a global scale.

Identifying some general and abstract solution to enable the design and development – via a proper programming of self-organizing activities – of specific global application goals, would have a dramatic impact in all sketched scenarios (micro, medium and global scale). In this paper, without having solutions at hand, we can anyhow try to identify some key directions to investigate.

### 3.1 Direct Engineering of Self Organization

Direct engineering approaches to self-organization basically aims at defining distributed algorithms that, starting from a few basic mechanisms (e.g., broadcast and localization), and exploiting local interactions and local computations, can provably lead a system (or parts of it) to a final coherent global state. Unlike traditional distributed algorithms, self-organizing algorithms disregard micro-level issues such as ordering of events, process synchronization, and structure of the underlying networks (issues for which no possibility of control is assumed). Rather, they focus on the fact that the algorithm will eventually converge despite micro-level contingencies and that it will keep the system in the stable state despite perturbations (e.g., changes in the network structure).

A typical example of a direct engineering approach to self-organization is distributed self-localization. There, a number of randomly distributed particles can determine their geographical position starting from a few "beacon" particles (possibly self-determined via leader election and acting as reference frame) and recursively applying a local triangulation mechanism to determine their position w.r.t, to close particles, until a global coherent positioning of all particles in the reference frame is reaches [Nag03].

Another example relates to the formation of regular spatial patterns in mobile particles [VasMZ03], which has several applications in computational self-assembly and in landscape exploration. Given a number of particles distributed in an environment, it is possible to devise distributed algorithms that, by locally driving the movements of the particles, eventually lead the system to self-organize in globally regular shapes. For instance (see Figure 1): a simple leader election algorithm can determine the center of gravity of the particles; then, the center of gravity can propagate in the network with a sort of "gravitational field" attracting all other particles toward the center, and until a specific distance from the center is reached. The result is in a circular organization of particles.

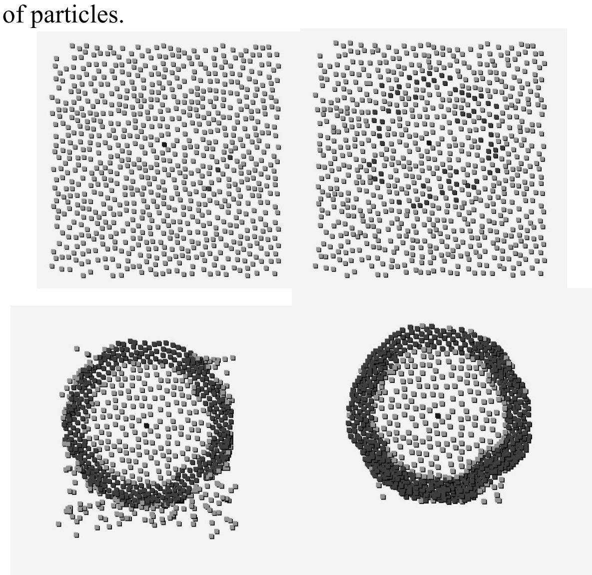

**Figure 1: Sequences of Direct Self-Organization of Cooperative Mobile Robots into a Circle**

Direct engineering approaches to self-organization have the great advantage of enabling engineers to achieve "by design" a specific robust self-organized behavior. Unfortunately, such approaches are feasible only for a limited number of application needs. In fact, with direct approaches, only very simple global states of equilibrium (or, which is the same, only very regular patterns of activity) can be enforced, i.e., all those that can be modeled in simple linear terms. Engineering very complex behavior involving non-linear phenomena such as differentiation in activities, navigation and localization in complex manifolds, enforcement of complex coordination patterns, requires facing very higher complexities. Most importantly, it requires a priori assumptions on the configuration of the system that would limit its self-organizing nature and, consequently, its degree of adaptiveness and its robustness. For instance, making the particles in Figure 1 self-organize into non-symmetrical patterns by direct self-organization requires either a very complex code to be executed by particles or some particles to have some a priori information on where to go, that is, some a priori configuration efforts. For all these cases, a reverse engineering approach may be required.

### 3.2 Reverse Engineering of Emergent Behaviors

Reverse engineering approaches to self-organization aims at achieving complex coordinated behaviors in spray computers by recreating in spray computers (and by adapting to specific application needs) the conditions to make some complex coordinated behaviors observed in other systems and in nature emerge in the computational spray computer system. In these cases, due to the complexity (and non-linearity) of the phenomena involved, engineers have no direct control on the evolution of the system, nor they can somehow prove that the system will behave as needed. Simply, they can be reasonably (i.e., probabilistically) confident that the global evolution of the system will eventually lead to the desired globally coordinated behavior.

Clearly, simulations will be the workhorse of reverse engineering approaches. Simulations of spray computer systems will not only provide a framework on which to test the functionalities of a developed systems, but they will be an integral part of the design and development process. Since the behavior of the components and of their interactions that can lead to the desired global behavior can hardly be modeled and predicted "on paper", simulations appear to be the only tool with which to have feedback on how a system will actually work. In other words, in reverse engineering approaches, the modeling phase consists in verifying via simulations the correctness of an idealized model suitable, but not necessarily close, to the target scenario (this model can be for example a biological or social model), then to refine the model and the simulations (that also realize a prototype implementation of the model) to rend both enough similar to the actual scenario to be taken in consideration as a candidate solution.

In the past few years, several approaches to self-organization relying on the reverse engineering of diverse natural phenomena have been proposed in different areas and have shown their effectiveness in achieving difficult global coordination tasks. For instance, the phenomena of ant foraging [BabMM02, MenT03] and gossiping [BraE02] turn out to be useful to discover path to information and diffuse information, respectively, in dynamic networks of spray computers. Coming back to the example of pattern formation in mobile cooperative robots, it is possible to enforce the emergence of non-symmetrical patterns by getting inspiration from the biological formation of morphogenesis. There, it can be observed that differentiation from regular symmetrical patterns occur due to the contrasting forces induced by cells increasing in number and still have to adhere to each other in a limited physical space. Similarly, in mobile computational particles, one can impose a constraint on the average density of cell, making cells repel each other if a too high density is reached. This constraint, contrasting with the gravitational force that would attract them toward the center of gravity, force them organizing in non-symmetrical patterns (see Figure 2).

Reverse engineering approaches to self-organization have several advantages. First, it is possible to rely on results from other disciplines to explore a variety of complex coordination phenomena to be exploited in spray computers systems. Second, once the basic mechanisms underlying an emergent behavior are understood and properly reproduced via simulation, programming an actual system to exhibit such behavior is dramatically simple, and it reduces to programming typically simple local rules and local interactions. In addition, the resulting system is intrinsically robust and adaptive, the result is typically robust and adaptive.

Unfortunately, approaches relying on complex emergent behavior also incur in several potential drawbacks. Generally speaking, the non-linearity involved in the evolution of the system may cause several potential final states to be reached by a system, each of which potentially stable, without the possibility of predicting which ones will be actually reached after the self-organization process. In some cases, all of this states may be equivalent from the application viewpoint (e.g., in ant foraging, what matters is that a reasonably short path to food/information is reached, no matter what the path actually is). Also, in these cases, the presence of multi stable states may be also advantageous, because this ensure that the system, even if strongly perturbed (e.g., due to network or environmental dynamics), will be able to soon re-organize its activity into another stable state. However, in several other cases, the designer may wish that its system self-organizes to a specific global states, not to any one. For instance, in the case of cooperative mobile robots, specific application needs of self-assembly or of landscape, may require robot to assume a specific non-symmetric form, not any of the ones that could emerge from the self-organization process (see Figure 2). And the same problem applies in the case of systems perturbation: once an already self-organized system gets perturbed, the designer may not wish it to re-stabilize to a different state.

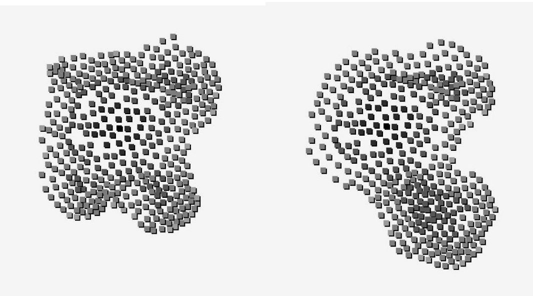

**Figure 2. Emergence of non-symmetrical patterns in mobile cooperative robots**

### 3.3 Control of Emergent Behaviors

When the evolution of a system can lead to several final global states, and only a limited set of these are useful to the specific application purposes, the problem arise on how to control/direct the evolution of the system so as to ensure that it will self-organize as desired. The problem is not easy and introduce the key issue of somewhat mixing together forms of reverse and of direct engineering of self-organization.

On the one hand, introducing some sorts of direct engineered control should be done without undermining the basic advantages of the reverse engineering approach, i.e., its capability to promote the spontaneous formation of complex and robust patterns of activity with little design and coding efforts. On the other hand, for such a mixed approach to be possible, it is necessary that both direct self-organizing algorithms and emergent behaviors are modeled and coded with the same set of basic abstractions.

In the example of cooperative mobile robots, we have successfully enforced some sort of distributed control of emergent behavior, by properly mixing the phenomena making non-symmetrical patterns emerge (described in Subsection 3.2) and sort of leader election algorithm. In particular, by using specific computational fields we have been able to let the system self-select a specific number of particles – equidistant from each other and from the center. These particles can then start to act as if they had a higher gravitational mass (i.e., diffusing a gravitational field of higher value), which makes it possible to control the emergence of polygon-like structures (Figure 3) with any required number of vertices. Overall, the solution preserves enough simplicity and robustness.

Besides this example, the general problem of controlling emergent behaviors in a complex self-organizing system – which, to the authors' opinions, will represent one of the key challenges for the whole research area of autonomic and self-organizing computing – is still open and widely uninvestigated. The urge for appropriate control models and for a uniform approach to direct and reverse engineering of self-organization appears even more compulsory when considering another factor: in several cases, even simple systems engineered with a direct approach may – due to simplifications in their modeling – exhibit unexpected emergent behaviors. Although sometimes such unexpected behaviors may be irrelevant or even useful for them to be offensively exploited (consider, e.g., the emergence of scaling in complex networks, and the advantages it carries to the robustness of the network [AlbJB00]), sometimes they may be damaging and introduce the need to defend from them by proper forms of control.

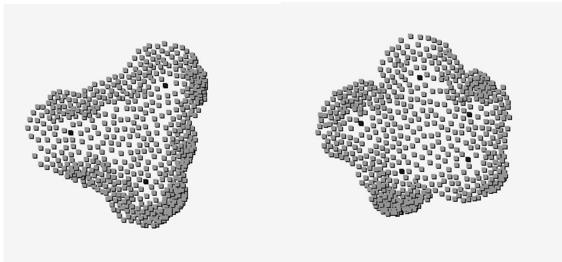

**Figure 3. Controlling the emergence of specific non-symmetrical patterns in mobile cooperative robots.**

## 4 Relevant Research Projects

Several projects around the world are starting to recognize the above needs and are facing issues related, to different extent, to the engineering and programming of spray computers systems and applications. Without the ambition to be exhaustive, we present here a few relevant threads of activities and discuss their shortcomings.

With regard to the micro-scale, the Amorphous Computing project at MIT focuses on the problem of identifying suitable models for programming applications over amorphous networks of computational particles [Nag03]. The particles constituting an amorphous computer have the capabilities of locally propagating sorts of computational fields in the network, and to locally sense and react to such fields. By having particles sense and re-propagate this fields, coordinated patterns of activities emerge in the system independently of the structure of the network. So far, the Amorphous computing project has defined a simple yet effective language for programming particles on the basis of computational fields. On this base, it has been shown how it is possible to exploit such a language to let the particles (directly) self-organize a coordinate systems and self-determine their position in it, and to have a variety of global patterns getting (directly) organized in a system from local interactions. What the project has still not addressed are the problems related to mobile and ephemeral particles: the network is considered static, and the relative position of particles is considered fixed. Our work on pattern formation for cooperative mobile robots [VasMZ03] complement this by exploiting computational fields to drive the movements of mobile particles and goes further, by exploiting computational fields towards reverse engineering approaches to self-organization. Still, both works so far has focused on very simple computational particles – not much different from cellular automata cells – and their effectiveness in coordinating systems of more complex particles is to be verified.

Besides amorphous computers, most of the researches in the area of micro-scale spray computers (i.e., literally spray computers) are performed in the context of the "sensor networks" research community [Est02]. There, the key issues being investigated relate to the identification of effective algorithms and tools to perform distributed monitoring activities by a cloud of distributed sensors in a physical environment (tracing the position and movement of an object, determining the occurrence of specific conditions, reporting sensed data back in an efficient way). These researches are indeed providing good insights on the theme of self-organization and are leading to some very interesting results. Techniques for self-localization, self-synchronization of activities, adaptive data distribution, all of which of primary importance for any type of spray computers, have been widely investigated. Still, we feel these researches are somewhat limited by two main factors. First, the accent is mostly on direct self-organization, and few works explored the possibility of exploiting emergent phenomena in sensor networks (namely, the only examples in these direction relate to routing algorithms based on ant-foraging or gossiping). Second, the accent on "sensing" tends to disregard the "actuating" factor – potential source of a wide range of interesting applications – and the algorithms and tools that could be of use to perform specific actuation works. In addition, most research work is being devoted to the definition of "power-effective" algorithms, aimed at minimizing resource consumption. This is motivated by the current impossibility of providing such small computer systems with enough battery power to last for a long time. However, we think that short-life batteries and the consequent need of power-aware computing models are a current contingent problem, rather than a basic research issues likely to have long-term impact. Scavenging power from sunlight, vibration, thermal gradients, and background RF, next generation of microcomputers will be fully autonomous in terms of power supply, and will be capable of long-lasting, if not ever-lasting, activity. Quoting Kris Pister [Pis03]: "computer-based sensors and actuators, being entirely solid state and with no natural decay processes, may well survive the human race".

Coming the medium scale, as far as we can see most of the researches are focusing either on routing algorithms for mobile ad-hoc networks of handheld computers [Bro98] or on the definition of effective user-level ubiquitous environments [Rom02]. Researches on routing algorithms for mobile networks share several common issues with researches on algorithms for data distribution on sensor networks. In our opinion, these works are, again, too often focused on power and resources limitation problems and mostly disregard higher-level self-organization issues such as coordination of distributed behaviors. Researches on ubiquitous computing environments mostly focus on achieving dynamic interoperability of existing application-level components and of smart-artifact and pervasive computing devices. For instance, the Gaia system developed at PARC [Rom02], defines an architecture based on "active" interaction spaces, as a reification of a specific real-world environment (e.g., a meeting room), where pre-existing (and pre-programmed) devices and user-level software components can dynamically enter, leave and interoperate dynamically with each other according to specific patterns specified as part of the active environment. Although such an approach is very important to organize user-level activities and their interactions with a smart environment, neither Gaia nor most of the other proposals in this direction has something to say on the issue of designing, developing, and controlling self-organizing coordinated distributed applications.

As far as the global scale is involved, most research on adaptive and unsupervised computing focus, as we have already stated, on the key idea of self-organizing overlay networks for P2P computing, which can be considered as a typical example of a direct engineering approach to self-organization. However, we do not think this is the best approach. In fact, building and maintaining globally coherent overlay networks at a worldwide scale may be very costly. Thus, despite the simulation on small-scale systems show the feasibility of the approach, it is not very clear how this could scale to millions of nodes. In addition, although most of the proposals for overlay networks prove their effectiveness in re-organizing a coherent structure upon dynamic changes in the structure, such studies are typically performed by testing the sequential arrival/dismissing of single nodes, and it is not clear if higher degree of networks dynamics (with concurrent arrivals/dismissing of nodes) would be sustained equally well. In our opinions, P2P systems based on overlay networks are being of great help to understand the basic property of dynamic networks and the basic requirements for adaptive applications, although next generation P2P should better rely on more flexible and light-weight approaches, possibly exploited reverse engineering approaches. For instance, approaches based on artificial ants [BabMM02, BonDT99, BraE02, MenT03] and virtual computational fields [MamZ04] appear very promising. As an example of an approach based on artificial ants, Anthill [BabMM02] support the design and development of adaptive peer-to-peer applications by relying on distributed mobile components ("ants") that can travel and can indirectly interact and cooperate with each other by leaving and retrieving bunches of information (to act as synthetic pheromones) in the visited hosts. The key objective of anthill is to build robust and adaptive networks of peer-to-peer services by exploiting the capabilities of ants to re-organize their activity patterns accordingly to the changes in the network structure. As an example of an approach based on computational fields, TOTA (Tuples On The Air) [MamZ04] relies on spatially distributed tuples for both supporting adaptive and uncoupled interactions between agents, and context-awareness. Agents can inject these tuples in the network, to make available some kind of contextual information and to interact with other agents. Tuples are propagated by the middleware, on the basis of application specific patterns, defining sorts of "computational fields", and their intended shape is maintained despite network dynamics, such as topological reconfigurations. Agents can locally "sense" these fields and can rely on them for both acquiring contextual information and carrying on distributed self-organizing coordination activities. However, the generality of this approach in supporting the design and development of a variety of applications and their power in supporting very large-scale applications for highly dynamic networks is still to be proved.

Whether the micro, medium, or global scale is involved, most of the researches so far focus on direct engineering approaches to self-organization or on the exploitation a limited number of biologically and socially inspired reverse engineering approaches. Emergent behaviors in complex distributed systems have been mostly studied in terms of structural properties (e.g., the scale-free structures of networks such as Gnutella [RipIF02] and the Web [AlbBJ00]). However, as far as spray computers and likes are involved, little is known so far about their dynamic behavior, possible emergence of complex patterns of activity, and methodologies to control the emergence of these patterns.

# 5 Research Agenda

To conclude, we sketch a rough research agenda for what we believe are the key challenges to be faced in the area of self-organization for the design, development, and control, of spray computer applications.

First, we think that researches should rely on a deeper understanding of the global behavior of spatially distributed systems of autonomous and interacting components, in any area. This could be used to exploit self-organization principles both offensively (i.e., to use them so as to achieve in a simple way globally coordinated behaviors) and defensively (i.e., to prevent the potential emergence of possibly dangerous self-organizing behaviors). Both cases may requires the study of mechanisms and tools to somehow direct and engineering such systems in a decentralized way, so as to enforce some sorts of control over these systems despite the impossibility of controlling them in their full. As previously anticipated, some recent approaches already take inspiration from phenomena of self-organization in real-world systems to defines adaptive and reliable solutions to specific contingent problems (e.g., ant-inspired algorithms and coordination based on computational fields). Currently underestimated phenomena occurring in other types of spatially distributed systems of autonomous components (e.g., macro-ecology patterns of population distribution and biodiversity, physics of granular media, emergence of synchronization, morphogenesis) are worth to be explored too. Also, more simulation work to possibly predict what types of behaviors the emergent scenarios of spray computers will exhibit will be compulsory.

Once the above understanding will be quite assessed, we think there will be need to define a general purpose programming model for designing and deploying applications in such dynamic networks of spray computers, together with the development of associated middleware infrastructure and tools. One very ambitious objective could be for such a model to enable people to program, deploy, and control self-organizing and adaptive distributed applications (exploiting both direct and reverse engineering approaches) with a minimal background knowledge – the same as a undergraduate students can currently develop excellent distributed Web-based Java applications – and independently of the specific application scenario, sensor networks rather than wide-area distributed applications – the same as an undergraduate student can easily and with minimal efforts adapt its applications for execution on both a Linux workstation and a Cellular phone. The definition of such a model will clearly require the identification of a minimal set of abstractions enabling the modeling of salient characteristics of spray computers and their operational environments. In our opinion, approaches based on computational fields [VasMZ03, MamZ04] are very promising to this purpose, by enabling to uniformly model a wide variety of distributed self-organizing behaviors (both with direct and reverse engineering) and to effectively model also ant-inspired approaches [MenT03]. However, this opinion is still to be verified.

Eventually, all the above researches will definitely increase our understanding on the potentials of spray computers at any scale, and will likely cause a range of new application areas to come to the fore. For instance, systems such as worldwide file sharing and artifacts like the cloak of invisibility could have simply never been conceived a few years ago. The new software and hardware technology will call also for visionary application-oriented thinkers, to unfold in full the newly achieved application potentials.

As a final note, we also want to emphasize that the widespread and pervasive diffusion of self-organizing distributed computing systems to which we will assist in the next few years will not come without dangers. Even without referring to the (scientifically improbable) catastrophic scenarios depicted by Michael Chricton, more pragmatic problems will have to be faced such as pollution due to (literally) spray computers being dispersed in the environment and garbage collection of obsolete spray computer software.

## References


[AlbJB00] R. Albert, H. Jeong, A. Barabasi, “Error and Attack Tolerance of Complex Networks”, Nature, 406:378-382, 27 July 2000.

[BabMM02] O. Babaoglu, H. Meling, A. Montresor, “Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems”, 22nd International Conference on Distributed Computing Systems, IEEE CS Press, Vienna (A), June 2002.

[BerG97] A. A. Berlin, K. J. Gabriel, “Distributed MEMS: New Challenges for Computation”, IEEE Computing in Science and Engineering, Vol. 4, No. 1, Jan.-March. 1997, pp. 12-16.

[BerGPP02] C. Bernon, M.P. Gleizes, S. Peyruqueou, G. Picard, “ADELFE: a Methodology for Adaptive Multi-Agent Systems Engineering”, 3rd International Workshop Engineering Societies in the Agents World, LNAI No. 2577, Sept. 2002.

[BonDT99] E. Bonabeau, M. Dorigo, G. Theraulaz, “Swarm Intelligence”, Oxford University Press, 1999.

[BraE02] D. Braginsky, D. Estrin, “Rumor Routing Algorithm For Sensor Networks”, 1ST Workshop on Sensor Networks and Applications (WSNA), Sept. 2002.

[Bro98] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, “A Perfomance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols”, ACM/IEEE Conference on Mobile Computing and Networking, ACM Press, Dallas (TX), Oct. 1998.

[Cri02] M. Crichton, “Prey: a Novel”, HarperCollins, 2002.

[Est02] D. Estrin, D. Culler, K. Pister, G. Sukjatme, “Connecting the Physical World with Pervasive Networks”, *IEEE Pervasive Computing*, 1(1):59-69, Jan. 2002.

[ImiG00] T. Imielinski, S. Goel, “Dataspace - querying and monitoring deeply networked collections in physical space”, IEEE Personal Communications Magazine, October 2000, pp. 4-9.

[KepC03] J. Kephart, D. M. Chess, "The Vision of Autonomic Computing", IEEE Computer, 36(1):41-50, Jan. 2003.

[MamZ04] M. Mamei, F. Zambonelli, “Distributed Motion Coordination with Co-Fields: A Case Study in Distributed Traffic Management”, IEEE Pervasive Computing, 2004, to appear.

[MenT03] R. Menezes, R. Tolksdorf, “SwarmLinda: a New Approach to Scalable Linda Systems based on Swarms“, 16th ACM Symposium on Applied Computing, Melbourne (FL), March 2003.

[Nag03] R. Nagpal, A. Kondacs, C. Chang, “Programming Methodology for Biologically-Inspired Self-Assembling Systems”, AAAI Spring Symposium on Computational Synthesis, March 2003.

[ParBS02] V. Parunak, S. Bruekner, J. Sauter, "ERIM’s Approach to Fine-Grained Agents", NASA/JPL Workshop on Radical Agent Concepts, Greenbelt (MD), Jan. 2002.

[Pis00] K. Pister, “On the Limits and Applicability of MEMS Technology”, Defense Science Study Group Report, Institute for Defense Analysis, Alexandria (VA), 2000.

[Pis03] K. Pister, Invited Plenary Talk, The 23rd International Conference on Distributed Computing Systems, Providence (RI), May 2003.

[Rat01] S. Ratsanamy,, P. Francis, M. Handley, R. Karp, "A Scalable Content-Addressable Network", ACM SIGCOMM Conference 2001, Aug. 2001.

[RipIF02] M. Ripeani, A. Iamnitchi, I. Foster, “Mapping the Gnutella Network”, IEEE Internet Computing, 6(1):50-57, Jan.-Feb. 2002.

[Rom02] M. Roman et al., “ Gaia : A Middleware Infrastructure for Active Spaces”, IEEE Pervasive Computing, 1(4):74-83, Oct.-Dec. 2002.

[RowD01] A. Rowstron, P. Druschel, “Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems”, 18th ACM Conference on Middleware, Heidelberg (D), Nov. 2001.

[Ten00]D.Tennenhouse, "Proactive Computing", Communications of the ACM, 43(5):43-50, May 2000.

[VasMZ03] M. Vasirani, M. Mamei, F. Zambonelli, “Morphogenesis of Cooperative Mobile Robots with Minimal Capabilities”, presented at the 1st European Workshop on Multiagent Systems, Oxford (UK), Dec. 2003.

[ZamM02] F. Zambonelli, M. Mamei, “The Cloak of Invisibility: Challenges and Applications”, IEEE Pervasive Computing, 1(4):62-70, Oct.-Dec. 2002.

[ZamMR03] F. Zambonelli, M. Mamei, A. Roli, “What Can Cellular Automata Tell Us About the Behaviour of Large Multi-Agent Systems?”, in Software Engineering for Large Scale Agent Systems, LNCS No. 2603, April 2003.