Spray Computers: Frontiers of Self-Organization for Pervasive Computing

Marco Mamei, Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia – Via Allegri 13 – Reggio Emilia – ITALY
{ mamei.marco, franco.zambonelli}@unimo.it

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 [Abe00, 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 [ZamP03], as well as distributed service- and data-oriented activities on the Internet [CabLZ02, 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 the need of appropriate 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 [ParBS02, ZamP03, ZamMR03].

This paper aims at exploring the above issues with a strong application-orientation and with a touch of inter-disciplinarity (by analyzing potentially-related research findings in different areas). To this end, the final paper

will be organized as follow: Section 2 will detail 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. 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).

1.1 The Micro Scale

As proved in the context of the Smart Dust project at Berkeley [BerG97, KahKP00], it is already possible to produce fully-fledged computer-based systems smaller than few cm^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-2, the nano-swarms of Michael Chricton's novel "Prey" [Chr02], 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.

2.1. 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.

1.2 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 Chord [Sto01], 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 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 prespecified, 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 control methodologies and algorithms to let the sprayed computing devices to *auto*-organize their interaction patterns: computing devices have to start working together without the presence of any a-priori global supervisor or centralized facility.

This problem is further exacerbated by the fact that both technological reasons and scalability issues call for a strictly local perception of the environment and for only strictly local interactions between components. Moreover, mobility and environment dynamism, on the one hand, and network and components' failures, on the other hand, call for even more robust and fault-tolerant approaches.

Unfortunately, we still do not know how to program and manage these kinds of systems. The main conceptual difficulty is that we have direct-engineered control only on component local activities, while the application task is often expressed at the global scale (e.g. in sensor network or modular robot) [SheS02, BonDT99]. Bridging the gap between local and global activities is not nearly easy, but it is although possible: distributed algorithms for modular robots have been proposed and successfully verified, routing protocols is MANET (in which devices coordinate to let packets flow from sources to destinations) have been already widely used.

The problem is still that the above successful approaches are tailored ad-hoc to a specific application domain and it is very difficult to generalize them to other scenarios.

There is a great need for general, widely applicable, techniques, engineering methodologies, middleware and APIs for this kind of systems [Abe00, ZamP03]. General and abstract solutions would have an impact in all the above scenarios (micro, medium and global scale), because the interaction patterns and control flows of those systems are similar and the underlying principles the same.

To face this situation, several researchers turned their attention to engineering approaches that take inspiration from natural systems already providing robustness and performance despite uncertainty and environment dynamism e.g. the embryo's development, the immune system and the collective behavior of social animals, like ants. Robustness in these systems arises from the fact that they have the ability to achieve its goal even when some individuals die or fail to perform their task; flexibility arises from the fact that the patterns of interactions between the individuals are not fixed by contract and can be instead dynamically re-shaped to self-adapt to changing environments. Following other authors [BonDT99], we refer to those kind of systems as swarm intelligent systems, to stress the fact that their features and capabilities are not embedded in the single components of the system, but emerge by the coordinated activities of a swarm of individuals.

In the following of this section, we introduce two approaches, weaved in the swarm intelligence philosophy, that in our opinion are the embryos of a radically new engineering methodology to deal with spray computers. The first – direct engineering – aims at devising control algorithms from scratch: adopting standard engineering principles (modularity, composition, etc.) together with low-level, swarm-inspired interaction mechanisms (e.g. ants' pheromone communication). The second – reverse engineering – aims at devising control algorithms by trying to recreate in the spray computer those behaviors arising in systems with similar features (e.g. cellular automata) that appear to be useful in the targeted application scenario.

3.1. Direct Engineering via Self Organization

The key idea of this approach is to combine standard engineering principles together with low-level, swarm-inspired interaction mechanisms.

In this approach, the high-level cooperation mechanism are formalized by means of programming languages or middleware-level APIs, with explicit primitives, means of combination, and means of abstraction, thus providing a framework for the design and analysis of systems. These general high-level operations are then automatically compiled into a series of low-level swarm-inspired interaction patterns that enable the sprayed computers to perform the cooperation tasks in a robust and effective way.

The spray computer is thus programmed with abstractions that are suitable to consider it *as a whole* and the single low-level code executed by the particles is invisible form the programmer point of view.

The power of this approach is that it allows us to take advantage of traditional computer science techniques for managing complexity, while eventually relying on biological models, like gradient diffusion or stigmergy, for achieving robustness at the local level.

This has many advantages: *(1)* It is possible to rely on results from other disciplines to determine the classes of coordination patterns that can be formed *(2)* The primitives themselves can be made robust by relying on

mechanisms inspired by biological systems *(3)* The analysis of a complex system becomes tractable because it is built in understood ways from smaller parts *(4)* The compiler makes it possible to easily specify complex behavior.

Let us clarify the above procedure with an example; in [Nag03] a system providing a language for specifying shape formation on an intelligent reconfigurable sheet composed of thousands of identically-programmed but locally-interacting flexible agents is presented. The system uses a novel approach: the desired global shape is specified at an "abstract" level as a folding construction on a continuous sheet of paper, which is then automatically compiled to produce the program run by the identically-programmed agents (see figure 1). All agent behavior is constructed from a set of five general-purpose robust primitives, inspired by epithelial cell morphogenesis and cell differentiation in multicellular organisms such as the Drosophilae. The global-to-local compilation is achieved by composing these primitive building blocks in a principled way, using a set of geometry axioms taken from paper-folding mathematics. The resulting process is versatile and reliable in the face of random agent distributions, varying numbers of agents and random agent death, without relying on global coordinates, a global clock, or centralized control.

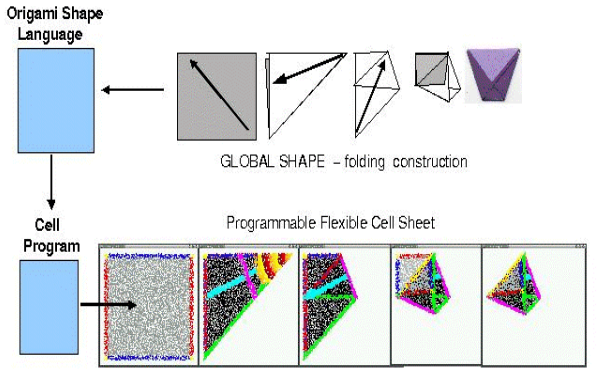

Fig 1. **Spray computer direct engineering: a global shape, described in terms of a folding construction, is compiled into a set of low-level, swarm inspired interaction patterns executed by the sprayed particles.**

3.2. Reverse Engineering of Emergent Behaviors

This approach aims at recreating in the spray computer those (emergent) behaviors arising in systems with similar features that appear to be useful in the targeted application scenario. Simulations will be the workhorse of this approach. Simulations of spray computer systems will not only provide a framework on which to test the functionalities of a developed systems, but they could be fully integrated in the development process. Engineering spray computer applications is difficult because the collective behaviors of the particles can hardly be foreseen, but can be typically analyzed only *a-posteriori* in a simulated environment. In this way, simulations provide a strong feedback and actually become the principal tool of the modeling phase.

Following this approach, the modeling phase consists in verifying via simulations the correctness of an idealized model suitable, but not necessarily close, to the target IT 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 IT scenario to be taken in consideration as a candidate solution.

Let us clarify the above procedure with an example: asynchronous cellular automata can be regarded as minimalist spray computer systems and for this reason the patterns emerging in cellular automata are expected to arise also in deployed spray computer systems. Such emerging patterns can be of use in different application scenarios: for example, to differentiate the state of the spray computer cells, or to let them assume a homogeneous coordinated state.

Starting form these considerations, a control algorithm for a spray computer system could be based on applying to the sprayed particles the interaction rules of a suitable cellular automaton, eventually trying to control the formation of proper patterns. A similar control procedure has been exploited in cellular automata in which the emergence of specific spatial patterns has been induced by external stimuli capable of perturbing the cellular automata state [ZamMR03].

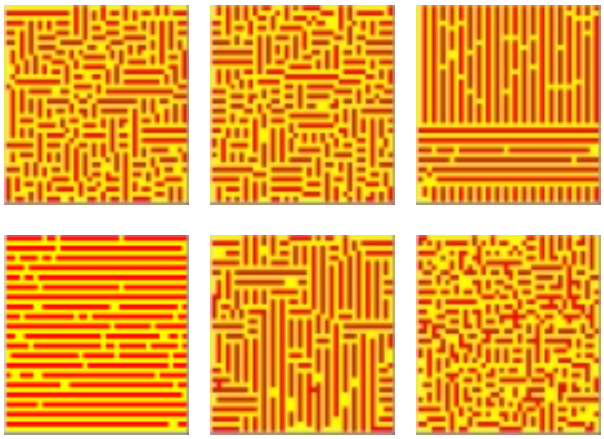

Fig 2. **External stimuli controlling the emergence of regular patterns on an asynchronous cellular automaton.**

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 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 "particles", for the sake of enabling the emergence of coherent patterns of activity in the network [But02, Nag03]. The particles constituting an amorphous computer have the basic capabilities of locally propagating sorts of computational fields in the network, and to locally sense and react to such fields. In particular, particles can transfer an activity state towards directions described by fields' gradients, so as to make coordinated patterns of activities emerge in the system independently of the specific 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 self-organize a coordinate systems and self-determine their position in it, and how it is possible to let a variety of global patterns getting-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. Also, the project so far has focused on very simple particles as finite-state machine with a limited number of states – not much different from cellular automata cells. The effectiveness of their model and programming language in programming systems of complex particles is still 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, whose most representative research group is headed at UCLA [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 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. Second, most research work is being devoted to the definition of "power-aware" and "power-effective" algorithms for distributed sensing (where distributed sensors tends to self-organize their activities so as to minimize 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, it is the opinion of the authors 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. As also Kris Pister (the inventor of the Smart Dust technology) envision [Pis00, 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, HesC03]. 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 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. 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 network 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. For instance, approaches based on artificial ants [BabMM02, BonDT99, BraE02, MenT03] and virtual computational fields [MamZL03, BanMS02] 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) [MamZL03] 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. With the notable exception of ant-based system like Anthill, a coherent global behavior is always achieved by direct design of algorithms that can provably lead to the desired global behavior. Little or none is said about the possibility of having a global systems behavior arise from "emergence", that as a complex results of simple rules attracting a systems (i.e., its patterns of activities) towards a specific configuration. 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, Bar02]). However, as far as spray computers are involved, their dynamic behavior (i.e., the evolution of their patterns of activities) is of possibly higher interest. Our research group has performed an interesting set of experiments with a new class of cellular automata [ZamMR03] for which the external environment can somehow perturb the natural evolution of the cells. We have observed that stable macro-level global structures emerge from local interactions among cells. Since perturbed cellular automata express characteristics strongly resembling those of spray computer systems, we expect that similar sorts of macro-level behaviors are likely to emerge and need to be studied, controlled, and possibly fruitfully exploited (e.g., the emergence of global structures from local interactions can be effectively exploited to achieve globally coordinated patterns of activity at low cost). A preliminary set of experiments reporting two ways of indirectly controlling the behavior of dissipative cellular automata are reported and discussed w.r.t. the possibility of applying similar sort of indirect control on large spray computer systems.

5. Research Agenda

In this section we will try to sketch out 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 of all, we think that researches in this area rely on a deep 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.

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

[Abe00] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss, “Amorphous Computing”, Communications of the ACM, 43(5), May 2000.

[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.

[BanMS02] S. Bandini, S. Manzoni, C. Simone, “Space Abstractions for Situated Multiagent Systems”, 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna (I), ACM Press, pp. 1183-1190, July 2002.

[Bar02] L. Barabasi, “Linked”, Perseus Press, 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.

[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.

[But02] W. Butera, “Programming a Paintable Computer”, PhD Thesis, MIT Media Lab, Feb. 2002.

[CabLZ02] G. Cabri, L. Leonardi, F. Zambonelli, “Engineering Mobile Agent Applications via Context-Dependent Coordination”, IEEE Transactions on Software Engineering, 28(11):1040:1058, Nov. 2002.

[Chr02] M. Chricton, “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.

[HesC03] C. K. Hess, R. H. Campbell, “A Context-

Aware Data Management Systems for Ubiquitous Computing Applications", 23rd International Conference on Distributed Computing Systems, IEEE CS Press, Providence (RI), June 2003.

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

[KahKP00] J. M Kahn, R. H. Katz, K. S. J. Pister, "Emerging Challenges: Mobile Networking for Smart Dust", Journal of Communications and Networks, 2(3):188-196, Sept. 2000.

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

[MamZL03] M. Mamei, F. Zambonelli, L. Leonardi, "Distributed Motion Coordination with Co-Fields: A Case Study in Distributed Traffic Management", 6th IEEE Symposium on Autonomous Decentralized systems, IEEE CS Press, Pisa (I), April 2003.

[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: From Basic Building Blocks to High Level Functionality, 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 IFIP/ACM Conference on Distributed Systems Platforms, Heidelberg (D), Nov. 2001.

[SheS02] W. Shen, B. Salemi, P. Will, "Hormone-Inspired Adaptive Communication and Distributed Control for Self-Reconfigurable Robots", IEEE Trans. on Robotics and Automation 18(5):1-12, 2002.

[Sto01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", ACM SIGCOMM 2001 Conference, ACM Press, San Diego (CA), Aug. 2001.

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

[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.

[ZamP03] F. Zambonelli, V. Parunak, "Signs of a Revolution in Computer Science and Software Engineering", 3rd International Workshop on Engineering Societies in the Agents' World, LNCS No. 2577, April 2003.