Spatial Computing: an Emerging Paradigm for Autonomic Computing and Communication

Franco Zambonelli, Marco Mamei

DISMI - Università di Modena e Reggio Emilia
Via Allegri 13, 42100 Reggio Emilia – ITALY
franco.zambonelli@unimore.it, mamei.marco@unimore.it

Abstract. *Emerging distributed computing scenarios call for novel "autonomic" approaches to distributed systems development and management. In this position paper we analyze the distinguishing characteristics of those scenarios, discuss the inadequacy of traditional paradigms, and elaborate on primary role of "space" in modern distributed computing. In particular, we show that spatial abstractions promise to be basic necessary ingredients for a novel "spatial computing" paradigm, acting as a unifying framework for autonomic computing and communication. On this base, we propose a preliminary "spatial computing stack" to frame the key concepts and mechanisms of spatial computing. Eventually, we try to sketch a research agenda in the area.*

1. Introduction

In the past few years, a variety of novel distributed computing scenarios have emerged that, although apparently very different from each other, share some key characteristics. By considering scenarios as diverse as P2P networks, multi-agent ecologies, pervasive computing systems, sensor networks, and robot swarms, one can easily recognize that [ZamP04]: *(i)* they all involve distributed computational and communication activities taking place in decentralized networks with a very large number of components and *(ii)* with a highly dynamic structure; *(iii)* components are embedded in some external dynamic environment, whether physical or computational, and their activities are influenced by their position in that environment.

The large size and the dynamics of the network, as well as the unpredictability induced by environmental dynamics, make traditional approaches to distributed systems management – involving humans-in-the-loop and typically assuming the capability of centralized control – fall short. Novel approaches supporting autonomous self-configuration and self-adaptation of activities in response to network and environmental dynamics are required.

A variety of solutions exploiting specific forms of self-organization and self-adaptation to solve specific application problems are being proposed (see [Dim04] for a comprehensive overview). The question of whether it is possible to devise a single unifying conceptual framework, applicable with little or no adaptations to scenarios as diverse as P2P networks and local networks of embedded sensors, is still open.

In this position paper, without having the ambition of providing a definitive answer to the above question, we will try to identify the important role that will likely be played in that process by spatial abstractions. In addition to the fact that spatial abstractions naturally suit systems whose activity are situated in some environment, the key point is that a spatial computing model can facilitate the integration of autonomic feature in distributed systems. In particular: *(i)* the central role of the network is substituted by an abstraction of space, built over the network in an autonomous and adaptive way; *(ii)* all application-level activities are abstracted as taking place in such space; *(iii)* autonomic behavior emerge form both the capability of the system of dynamically adapting the structure of the space as well as from the capability of application-level components of sensing, acting in, and navigating that space.

This paper is organized as follow. Section 2 outlines the key characteristics of modern scenarios and discusses the inadequacy of traditional approaches. Section 3 introduces "Spatial Computing" and discusses its basic concepts and advantages, and its relations with autonomic computing and communications. Section 4 proposes a "Spatial Computing Stack", as a framework to organize and understand the basic abstractions and mechanisms involved in spatial computing. Section 5 sketches a rough research agenda in the area and concludes.

2. Modern Distributed Systems Scenarios and the Need for Novel Approaches

A variety of modern distributed computing scenarios exhibit characteristics challenging traditional approaches to network and distributed systems management.

2.1 Key Characteristics

Such scenarios include *(i)* micro-scale ones, i.e., networks of low-end computing devices typically distributed over a geographically small area (e.g., sensor networks [Est02], smart dusts [Pis00] and spray computers [Zam04]); *(ii)* medium-scale scenarios, i.e., networks of medium-end devices, distributed over a geographically bounded area, and typically interacting with each other via short/medium range wireless connections (pervasive computing systems and smart environments [GelSB02] and cooperative robot teams); *(iii)* global-scale scenarios, characterized by high-end computing systems interacting at a world-wide scale (the physical Internet, the Web, P2P networks [RipIF02] and multiagent systems ecologies [Kep02].

Despite clear dissimilarities in structure and goals, one can also easily recognize some key common characteristics:

- **Large Scale**: the number of nodes in all the above types of networks and consequently the number of components involved in a distributed application is typically very high and, due to decentralization, hardly controllable. It is not possible neither to enforce a strict control over the configuration of components (consider e.g., the nodes of a P2P network) nor to directly control each of them during execution (consider e.g., the nodes of a sensor network distributed in a landscape).
- **Network dynamism**: the activities of components will take place in network whose structure derives from an almost random deployment process, and that is likely to change over time with unpredictable dynamics. Factors that may contribute to dynamically change the network topology include environmental contingencies or

failure of components (very likely e.g., in sensor networks and pervasive computing systems) and mobility of nodes (as e.g., in robot teams and in network of smart appliances). In addition, at the application level, software components can be of an ephemeral or temporary nature (consider e.g. the peers of a P2P network).

- **Situatedness:** The activities of components will be strongly related to their location in either a physical or a virtual computational environment. On the one hand, situatedness can be at the very core of the application (e.g. in sensor networks and in pervasive computing systems the very goal is to exploit the physical location of nodes and their capabilities to collect environmental data and to improve our interaction with the physical world). On the other hand, situatedness can relate to the fact that components can take advantage of their environment to organize the access to distributed resources (as e.g., in P2P data sharing networks).

The first two characteristics (large size and network dynamism) compulsory call for self-organizing and self-adapting approaches, enabling those systems to exhibit – both at the network and at the application level – autonomic behavior. In fact, if the dynamics of the network and of the environment compulsory require dynamic adaptation, the impossibility of enforcing a direct control over each component of the system implies that such adaptation must occur without any human intervention. The last characteristic, situatedness, calls for an approach that elects the environment, its spatial distribution, and its dynamics, to primary design dimensions, aspects which have been mostly disregarded by traditional approaches. In any case, the capability of self-organization and self-adaptation cannot abstract from the capability of the system of becoming "context-aware", i.e., of letting components to perceive the local properties of the space in which they are situated, and to act and adapt their behavior on this basis.

Summarizing: all presented scenarios of distributed computing share very similar characteristics and all require novel approaches promoting both autonomic behavior and an explicit modeling of situatedness. On this base, one could imagine that a single general-purpose distributed computing and communication paradigm, suitable for a variety of scenarios and enabling to face a variety of problems in a uniform way can be conceived. Unfortunately, traditional distributed computing paradigm appears not suitable to this purpose.

2.2 Inadequacy of Traditional Approaches

Early researches in parallel and distributed computing promoted a *transparent distributed computing* paradigm, in which the presence of an underlying network was totally hidden from application components [ChiC91]. The key motivation was that, to avoid the complexities inherent in having to deal with a distributed environment, it was necessary to hide distribution and enable components to execute and interact with each other as if they were all executing on a single, centralized, node. Figure 1a summarizes this by outlining that a component can interact with another one by simply "naming" it and disregarding its actual position.

Unfortunately, a transparent approach to distributed computing is totally unsuitable for modern scenarios. First, promoting transparency is very costly and can hardly scale to large-scale systems, in that it requires the presence of complex global naming services. In addition, transparent – i.e., "by name" – interactions can be effectively supported only in the presence of static interaction patterns and closed systems, definitely not in the presence of network dynamics. Finally, a transparent distributed computing paradigm does not provide an appropriate abstraction to deal with the situatedness of components in a distributed environment: under this paradigm, the distributed environment does not exist.

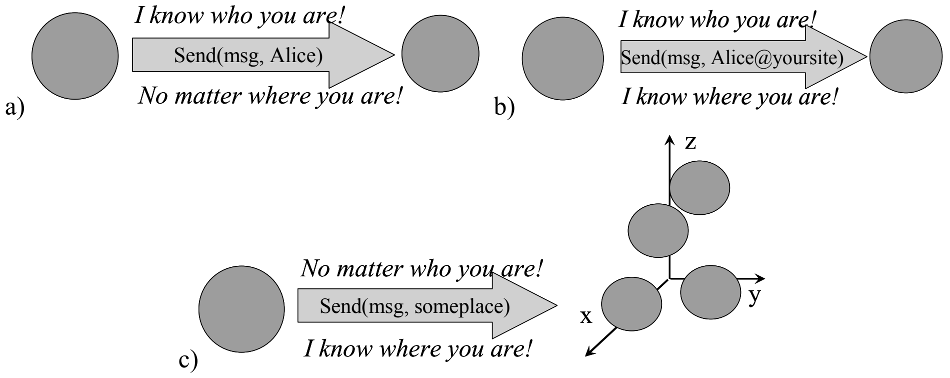

Figure 1. Interactions in: (a) a transparent distributed computing model; (b) a network-aware distributed computing model; (c) a spatial computing model.

The limitations of transparent distributed computing became evident in the mid 90's, with the advent of the Internet and of world-wide distributed computing. Such global scenarios outlined the need for *network-aware computing* models, in which application components were made aware of the distributed and decentralized nature of their operational environment [Wal97]. The assumption in network-aware computing is to make all interactions rely on the explicit knowledge of the network allocation – i.e., the IP – of components and resources. Figure 1b summarizes this with regard to the interaction between two distributed components: an interaction relies on the knowledge on the local name of a component on a specific network node.

Network-aware computing is suitable for large-scale network systems. First, it enables to explicitly take into account the costs involved in distributed interactions. Second, it involves local naming services which can scales well. However, network-aware computing does not provide at hand solutions to deal with network dynamics. If the structure of the network can dynamically change, and if nodes (and software components over them) can be of an ephemeral nature (i.e., intermittently available) a network-aware computing model requires to handle explicitly the exceptions caused by nodes unavailability. This naturally complicates the system's execution and management and also increases the costs and complexity of discovery services. Also, as far as situatedness is concerned, the only actual environmental abstraction reduces to be the network itself, an abstraction which is quite low level and does not easily enable modeling the logical relations between the network components.

3. Spatial Computing

To overcome the limitations of network-aware computing without losing its advantages it is necessary to identify a general-purpose model that:

- Hides the complexities intrinsic in dealing with a dynamic network.

- Provides suitable and conceptually simple environmental abstractions.
- Preserves an explicit awareness of the distribution of the scenarios.

3.1 Key Concepts in Spatial Computing

To this end, the key idea underlying spatial computing is to:

- Make the concept of "network" – a discrete system of variously interconnected nodes – evolve into a concept of "space" – i.e., a metric continuum.
- Let distributed components be aware of their surrounding space and to let them perceive (and possibly influence) the local properties of space.
- Rely on such spatial perception for all management-level and application-level activities.

In particular, in spatial computing, any type of networked environment is hidden below some of virtual metric n-dimensional space, mapped as an overlay over the physical network. The nodes of the network are assigned a specific area of the virtual space, and are logically connected to each other accordingly to the spatial neighborhood relations. Accordingly, each and every entity in the network, being allocated in some nodes of the network, is also automatically situated in a specific position in space. In this way, components in the network are no longer "network-aware" but rather "space-aware". On the one hand, components perceive their local position in space as well as the local properties of space (e.g., the locally available data and services) and possibly change them. On the other hand, the activities of components in that space are related to some sort of "navigation" in that space, which may include moving themselves to a specific different position of space or moving data and events in space according to "geographical" routing algorithms. The primary way to refer to entities in the network is by "position", i.e., any entity is characterized by being situated in a specific position in the physical space. In other words, the concept of "names" loses its primary role. This is summarized in Figure 1c: an entity interacts to another entity by sending data to a position in space.

Spatial computing models appear very suitable for the identified key characteristics of modern distributed computing scenarios:

- **Large size**: the size of a network does not influence the models or the mechanisms, which are the same for a small network and for a dramatically large one.
- **Network dynamics**: since the presence of the network is not directly perceived by components, the fact that it can be of a highly dynamic nature is irrelevant. The network is hidden behind a stable structure of space that is maintained despite network dynamism.
- **Situatedness**: the abstraction of space is a conceptually simple abstraction of environment, which also perfectly matches the needs of those systems, such as pervasive computing systems and sensor networks, whose activities are strictly intertwined with the physical space.

3.2 Examples of Spatial Computing Approaches

We do not claim to have invented the spatial computing paradigm from scratch. Rather, we consider spatial computing as an emerging trend that is, more or less explicitly, making its appearance in diverse scenarios.

As an example, consider a sensor network scenario with a multitude of wireless sensors randomly deployed in a landscape to perform some monitoring of environmental conditions [Est02]. There, all activities of sensors are intrinsically of a spatial nature. First, each sensor is devoted to local monitoring a specific portion of the physical space (that it can reach with its sensing capabilities). Second, components must coordinate with each other based on their local positions, rather than on their IDs, to perform activities such as detecting the presence and the size of pollution clouds, and the speed of their spreading in the landscape. All of this implies that components must be made aware of their relative positions in the spatial environment by re-constructing a virtual representation of the physical space [NagSB03]. Moreover, they can take advantage of "geographical" communication and routing protocols in which messages, data, and events, flow towards specific position of the physical/virtual space [RaoP03].

Another example in which spatial concepts appear in a less trivial way is world-wide P2P computing. In P2P computing, an overlay network of peers is built over the physical network and, in that networks, peers act cooperatively to search specific data and services. In first generation P2P systems (e.g., Gnutella [RipIF02]), the overlay network is totally unstructured, being built by having peers randomly connect to a limited number of other peers. Therefore, in these networks, the only effective way to search for information is message flooding. More recent proposals [Rat01, RowD01] suggest structuring the network of acquaintances into specific regular "spatial shapes", e.g., a ring or an N-dimensional torus. When a peer connects to the networks, it occupies a portion of that spatial space, and networks with those other peers that are neighbors accordingly to the occupied position of space. Then, data and services are allocated in specific positions in the network (i.e., by those peers occupying that position) depending on their content/description (as can be provided by a function hashing the content into specific coordinates). In this way, by knowing the shape of the network and the content/description of what data/services one is looking for, it is possible to effectively navigate in the network to reach the required data/services. That is, P2P networks define a spatial computing scenario in which all activities of application components are strongly related to positioning themselves and navigating in an abstract metric space. It is also worth outlining that recent researches promote mapping such spatial abstractions over the physical Internet network so as to reflect the geographical distribution of Internet nodes (i.e., by mapping IP addressed into geographical physical coordinates [Row04]) and, therefore improve efficiency.

In addition to the above examples, other proposals in areas such as pervasive computing [Bor04] and self-assembly [MamVS04] explicitly exploit spatial abstractions (and, therefore, a sort of spatial computing model) to organize distributed activities.

3.3 Autonomic Features in Spatial Computing

Autonomic features, including the capability of a distributed system of self-configuring its activity, self-inspecting and self-tuning its behavior in response to changed conditions, or self-healing it in the presence of faults, are necessary for enabling spatial computing and, at the same time, are also intrinsically promoted by the adoption of a spatial computing model.

On the one hand, to enable a spatial computing model, it is necessary to envision mechanisms to build the appropriate overlay spatial abstraction and to have such spatial abstraction be coherently preserved despite network dynamics. In other words, this requires the nodes of a network to be able to autonomously connect with each other, set up some sort of common coordinate systems, and self-position themselves in such space. In addition, this requires the nodes of the network to be able to self-reorganize their distribution in the virtual space so as to *(i)* make room for new nodes joining the network (i.e., allocate a portion of the virtual space to these nodes); *(ii)* fill the space left by nodes that for any reason leave the network; *(iii)* re-allocate the spatial distribution of nodes to react to node mobility. It is also worth outlining that, since the defined spatial structure completely shields the application from the network, it is also possible for a system to dynamically tune the structure of the space so as enforce some sorts of autonomic management of the network, transparently to the higher application levels. As an example, load unbalances in the network can be dynamically dealt, transparently from the application level, by simply re-organizing the spatial structure so as to have overloaded nodes occupy a more limited portion of the space.

On the other hand, the so defined spatial structure can be exploited by application level components to organize their activities in space in an autonomous and adaptive way. First of all, it is a rather assessed fact that "context-awareness" and "contextual activity", i.e., the capabilities of a component to perceive the properties of the operational environment and of influencing them, respectively, are basic ingredients to enable any form of adaptive self-organization and to establish the necessary feedback promoting self-adaptation. In spatial computing, this simply translates in the capability of perceiving the local properties of space, which in the end reflect some specific characteristics of either the network or of some application-level characteristics and of changing them. Second, one should also recognize that the vast majority of known phenomena of self-organization and self-adaptation in nature (from ant-foraging to reaction-diffusion systems, just to mention two examples in biology and physics) are actually phenomena of self-organization in space, emerging from the related effect of some "component" reacting to some property of space and, by this reaction, influencing at its turn the properties of space. Clearly, a spatial computing model makes it rather trivial to reproduce in computational terms such types of self-organization phenomena, whenever they may be of some use in a distributed system.

4. Framing Spatial Computing

Let us now have a detailed look at the basic mechanisms that have been exploited so far in distributed computing to promote self-organization and autonomic behavior. We will show that most of these mechanisms can be easily interpreted and mapped into very similar spatial concepts, and that they can be framed in a unifying flexible framework.

4.1. A Spatial Computing Stack

As shown in Figure 2, we frame these mechanisms according to a "space-oriented" stack of levels. For each level, we can recognize that different mechanisms, exploited in different scenarios, can serve similar purposes and aim at providing similar "spatial services" (see Table 1). In other words, by introducing a new paradigm rooted on spatial concepts, it is possible to interpret a lot of proposed self-organizing and autonomic approaches in terms of mechanisms to manage and exploit the space. On this basis, it is likely that a simply unifying model for autonomic computing and communication – leading to a single programming model and methodology and – can be actually identified.

The lowest "*physical level*" is about how components start interacting – in a dynamic and spontaneous way – with other components in the systems. This is a very basic expression of autonomic behavior which is a pre-requisite to support more complex forms of autonomy and of self-organization at higher levels. To this end, the basic mechanism exploited is broadcast (i.e. communicate with whoever is available). Radio broadcast is used in sensor networks and in pervasive computing systems, and different forms of TCP/IP broadcast (or of dynamic lookup) are used as a basis for the establishment of overlay networks in wide area P2P computing. Whatever the case, this physical level can be considered as in charge of enabling a component of a dynamic network application to get into existence in it and to start interacting with the other components.

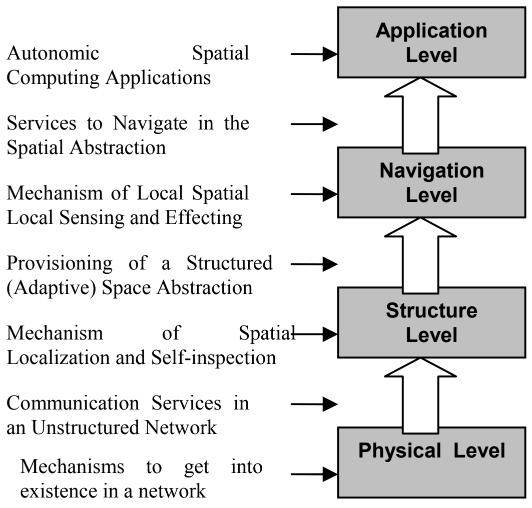
Autonomic Spatial Computing Applications
Application Level
Services to Navigate in the Spatial Abstraction
Mechanism of Local Spatial Local Sensing and Effecting
Navigation Level
Provisioning of a Structured (Adaptive) Space Abstraction
Mechanism of Spatial Localization and Self-inspection
Structure Level
Communication Services in an Unstructured Network
Mechanisms to get into existence in a network
Physical Level

Figure 2. A Spatial Computing Stack.

The "*structure level*" is the level at which some sort of spatial structure is built and maintained by components existing in the physical network. As already outlined, the fact that a system is able to create a stable spatial structure capable of surviving network dynamics and adapting the working conditions of the network is an important expression of autonomic behavior *per se*. However, such spatial structure is not a goal for the application, and it is instead used as the basic spatial arena to support higher levels of organization of activities.

The various mechanisms that are used at the structure level in different scenarios are – again – very similar to each other. Sensor networks as well as self-assembly systems typically structure the space accordingly to their positions in the physical space, by exploiting mechanisms of geographical self-localization. Pervasive computing systems, in addition to mechanisms of geographical localization, often exploit logical spatial structures reflecting some sorts of abstract spatial relationships of the physical world (e.g., rooms in a building) [Bor04]. Global scale systems, as already anticipated, exploits overlay networks built over a physical communication network. Although early approaches (e.g., Gnutella) give no metric structure to such overlay space, more recent approaches (as already anticipated) typically exploit metric overlay spaces and aims at making the spatial overlay match the spatial distribution of Internet nodes. A possibility which is currently under-investigated relates to the possibility – at the structure level – of dynamically adapting the structure of the space (and not simply of preserving a stable structure) to reflect and adapt to changing working conditions in the network.

The "*navigation level*" concerns the basic mechanisms that components exploit to orient their activities in the spatial structure and to sense and affect the local properties of space (i.e. mechanism to actually "use" the available spatial structure). If the spatial structure has not any well-defined metric, the only navigation approaches are flooding and gossiping. However, if some sort of metric structure is defined at the structure level (as, e.g., in the geographical spatial structures of sensor networks or in metric overlay networks) navigation approaches typically relate in following the metrics defined at the structure level. For instance, navigation can imply the capability of components to reach specific points (or of directing messages and data) in the space based on simple geometric considerations as in, e.g., geographical routing.

Starting from the basic navigation capability, is also possible to enrich the structure of the space by propagating additional information to describe "something" which is happening in that space, and to differentiate the properties of the space in different areas. One can say that the structure of space may be characterized by additional types of spatial structures propagating in it, and that components may direct their activities based on navigating these additional structures. In other words, the basic navigation capabilities can be used to build additional spatial structures with different navigation mechanisms. Typical mechanisms exploited at these additional levels are computational fields and pheromones. Despite the different inspiration of the two approaches (physical versus biological), we emphasize that they can be modeled in a uniform way, e.g., in terms of time-varying properties defined over a space [MamZ03]. The basic expression of self-organization that arises here derives from the fact that the structures propagated in the space – and thus the navigation activity of application components – are updated and maintained to continuously reflect the actual structure and situation of the space.

At the "*application level*", navigation mechanisms are exploited by application components to interact and organize their activities. Applications can be conveniently built on the following self-organizing feedback loop: *(i)* having components navigate in the space (i.e., discriminating their activities depending on the locally perceived structure and properties of the space) and *(ii)* having components, at the same time, modifying existing structure due to the evolution of their activities.

Depending on the types of structures propagated in the space, and on the way components react to them, different phenomena of self-organization can be achieved and modeled. For example, processes of morphogenesis (as needed in self-assembly, modular robots and mobile robotics), phenomena mimicking the behavior of ant-colonies and of flocks, phenomena mimicking the behavior of granular media and of weakly correlated particles, as well as a variety of social phenomena, can all be modeled in terms of:

- entities getting to existence in a space;
- having a position in a structured space and possibly influencing its structure;
- capable of perceiving properties spread in that space;

	MICRO SCALE **Nano Networks, Sensor Networks, Smart Dust, Self-Assembly, Modular Robots**	MEDIUM SCALE **Home Networks, MANETs, Pervasive Environments, Mobile Robotics**	GLOBAL SCALE **Internet, Web, P2P networks, multiagent systems**
"Application" Level (exploiting the spatial organization to achieve in a *self-organizing and adaptive way* specific app. goals)	Spatial Queries Spatial Self-Organization and Differentiation of Activities Spatial Displacement Motion Coordination & pattern formation DATA: environmental data	Discovery of Services Spatial Displacement Coordination and Distribution of Task and Activities Motion coordination & pattern formation DATA: local resources and environmental data	P2P Queries as Spatial Queries in the Overlay Motion Coordination on the Overlay Pattern formation (e.g., for network monitoring) DATA: files, services, knowledge
"Navigation" Level (dealing with the mechanism exploited by the entities living in the space to *direct activities and movements in that space*)	Flooding Gossiping (random navigation) Geographical Routing (selecting and reaching specific physical coordinates) Directed Diffusion (navigation following sorts of computational fields) Stigmergy (navigation following pheromone gradients)	Computational fields Multi-hop routing based on Spanning Trees Pattern-matching and Localized Tuple-based systems	Flooding Gossiping (random navigation) Metric-based (moving towards specific coordinates in the abstract space) Gossiping (random navigation) Stigmergy (navigation following pheromone gradients distributed in the overlay network)
"Structure" Level (dealing with mechanisms and policies to adaptively *shape a metric space* and let components find their position in that space)	Self-localization (beacon-based triangulation)	Self-localization (Wi-Fi or RFID triangulation) Definition and Maintenance of a Spanning Tree (as a sort of navigable overlay)	Establishment and Maintenance of an Overlay Network (for P2P systems) Referral Networks and e-Institutions (for multiagent systems)
"Physical" Level (dealing with the mechanism necessary to *get into existence* in a network)	Radio Broadcast Radar-like localization	Radio Broadcast RF-ID identification	TCP broadcast – IP identification Directed TCP/UDP messages Location-dependent Directory services

Table 1. Spatial Mechanisms in Modern Distributed Computing Scenarios

- capable of directing their actions based on perceived properties of such space and capable of acting in that space by influencing its properties at their turn.

Still, the ultimate goal of a uniform modeling approach capable of effectively capturing the basic properties of self-organizing computing, and possibly leading to practical and useful general-purpose modeling and programming tools, is far from close.

4.2 Multiple Spaces and Nested Spaces

In general, different scenarios and different application problems may require different perceptions of space and different spatial structures. For instance, a world-wide resource-sharing P2P network over the Internet may require – for efficiency reason – a 2-D spatial abstraction capable of reflecting the geographical distribution of Internet nodes over the earth surface. On the other hand, a P2P network for social interactions may require a spatial abstraction capable of aggregating in close regions of the virtual space users with similar interests. Also, one must consider that in the near future, the different network scenarios we have identified will be possibly part of a unique huge network (consider that IPv6 addressing will make it possible to assign an IP address to each and every square millimeter on the earth surface). Therefore, it is hard to imagine that a unique flat spatial abstraction can be effectively built over such a network and satisfy all possible management and application needs.

With this regard, the adoption of the spatial computing paradigm does not prescribe at all to adopt the same set of mechanisms and the same type of spatial structure for all networks and for applications. Instead, being the spatial structure a virtual one, it is possible to conceive both *(i)* the existence, over the same physical network, of multiple complimentary spatial abstraction independently used by different types of applications; and *(ii)* the existence of multiple layers of spatial abstractions, built one over the other in a multi-layered system.

With regard to the former point, in addition to the example of the different types of P2P networks calling for different types of spatial abstractions, one could also think at how different problems such as Internet routing, Web caching, virtual meeting points, introduce very different problems and may require the exploitation of very different spatial concepts.

With regard to the latter point, one can consider two different possibilities. Firstly, one can think at exploiting a first-level spatial abstractions (and the services it provides) to offer a second-level spatial abstraction enriching it with additional specific characteristics. For examples, one can consider that a spatial abstraction capable of mapping the nodes of the Internet into geographical coordinates can be exploited, within a campus, to build an additional overlay spatial abstraction mapping such coordinates into logical location (e.g., the library, the canteen, the Computer Science department and, within it, the office of Prof. Zambonelli). Such additional spatial abstraction could then be used to build semantically-enriched location dependent services. Secondly, one could think at conceiving a hierarchy of spatial abstractions that provides different levels of information about the space depending on the level at which they are observed, the same as the information we get on a geographical region are very different depending on the scaling of the map on which we study it. As an example, we can consider that the spatial abstraction of a wide-area network can map a sensor network – connected to the large network via a gateway – as a "point" in that space, and that the distributed nature of the sensor networks (with nodes having in turn a specific physical location in space) becomes apparent only when some activity takes place in that point of space (or very close to it).

In any case, although we have strongly advocated the flexibility and modularity of a spatial computing paradigm, whether and how it could be put to practice and could fulfill its promises is an open research issue.

5. Research Agenda

Spatial computing, by abstracting the execution of distributed applications around spatial concepts (localization and navigation in some sorts of metric space), and by exploiting the same set of spatial abstractions to perform adaptive network management activities, promises to be an effective paradigm for modern distributed computing scenarios.

In any case, besides the considerations made in this paper, much formal and practical work is needed to asses the potentials of spatial abstractions in distributed computing, and to verify whether they can actually pave the way to a sound and general-purpose engineered approach to autonomic computing and communication. In particular:

- Is the research for a unifying model fueled by enough application problems? In other words, is there a compulsory need for a unifying approach to promote a uniform autonomic treatment of a variety of problems in different scenarios? Or is instead the current way of doing (i.e., researching specific special-purpose autonomic solutions to specific problems in specific scenarios) the most economic and effective one?
- If the research of a unifying modes is worth (as we believe), is the proposed spatial computing stack meaningful and useful? Here, we have proposed it as a preliminary attempt to frame some basic concepts, and we are well aware that not everything fits perfectly in it. Nevertheless, our opinion is that, once all the layers will be properly defined, they will support a better engineering of such systems, promoting separation of concerns and clearly identifying the duties of the different levels.
- Beside the fact that spatial computing seems promising, can really all (or at least a large portion of) conceivable autonomic features – from the network management level up to the application level – be effectively expressed and implemented in such terms? With regard to network management, earlier in this paper we have sketched a rough idea on how spatial computing could be of some use to adaptively promote load balancing in a network. However, could issues such as QoS management, service personalization, emergence of pathological congestion patterns, take any advantage of a spatial computing approach? With regard to the application level, a variety of application problems requiring self-organizing and adaptive behavior deals with concepts that can be hardly intuitively mapped into spatial concepts. Consider, for instance, phenomena of adaptive division of labor or phenomena or adaptive evolution. For these problems, would exploring some sorts of spatial mapping to face them still be useful and practical? Would it carry advantages?
- If a unifying model spatial computing model can be found, can it be translated into a limited set of general-purpose and manageable protocols, tools, and programming abstractions? Also, can it lead to the identification of a sound methodology for developing and deploying autonomic features in modern distributed systems?

All the above issues need to be investigated, complemented by researches aimed at better understanding and framing the behavior of self-organizing and self-adaptive systems.

To conclude, we simply point out that our research group is currently involved in the development of a middleware called TOTA [MamZ04]. TOTA, suitable for both large-scale wide-area networks and local ad-hoc networks of small computing devices, defines *(i)* a set of network-level services to build and self-maintain spatial overlay structures over dynamic networks; *(ii)* an API to be exploited by application agents to sense and effect the local properties of space and to navigate in the spatial structure. In particular, TOTA exploits sorts of potential fields to be propagated in space to implement in a simple and modular way a variety of overlay spatial abstractions, and it has been successfully experienced to achieve self-organization and self-adaptation in a variety of distributed applications. This by no means imply that we have already found an appropriate solution to all the above mentioned research issues.

References

[Bor04] C. Borcea, “Spatial Programming Using Smart Messages: Design and Implementation”, *24th Int.l Conference on Distributed Computing Systems*, Tokio (J), May 2004.

[ChiC91] R. S. Chin, S. T. Chanson, “Distributed Object-Based Programming Systems”, *ACM Computing Surveys*, 23(1), March 1991.

[Dim04] G. Di Marzo, A. Karageorgos, O. Rana, F. Zambonelli (Eds.), *Engineering Self-organizing Systems: Nature Inspired Approaches to Software Engineering*, LNCS No. 2977, Springer Verlag, May 2004.

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

[GelSB02] H.W. Gellersen, A. Schmidt, M. Beigl, “Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts”, *Mobile Networks and Applications*, 7(5): 341-351, Oct. 2002.

[Kep02] J. Kephart, “Software Agents and the Route to the Information Economy”, *Proceedings of the National Academy of Science*, 99(3):7207-7213, May 2002.

[MamZ03] M. Mamei, F. Zambonelli, “Co-Fields: a Unifying Approach to Swarm Intelligence”, *3rd Workshop on Engineering Societies in the Agents’ Word*, LNCS No. 2677, April 2003.

[MamZ04] M. Mamei, F. Zambonelli, “Programming Pervasive and Mobile Computing Applications with the TOTA Middleware”, *2nd IEEE Conference on Pervasive Computing and Communications,* Orlando (FL), IEEE CS Press, March 2004.

[NagSB03] R. Nagpal, H. Shrobe, J. Bachrach, “Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network”, *2nd International Workshop on Information Processing in Sensor Networks*, Palo Alto (CA), April, 2003.

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

[RaoP03] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, I. Stoica. “Geographic Routing Without Location Information”. *ACM Mobicom Conference*. San Diego (CA), USA, 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.

[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.

[Row04] A. Rowstron et al., “PIC: Practical Internet Coordinates”, *24th International Conference on Distributed Computing Systems*, IEEE CS Press, Tokyo (J), May 2004.

[Wal97] J. Waldo et al., “A Note on Distributed Computing”, *Mobile Object Systems*, LNCS No. 1222, Feb. 1997.

[Zam04] F. Zambonelli, M.P. Gleizes, R. Tolksdorf, M. Mamei, Spray Computers: Frontiers of Self-organization“, *1st International Conference on Autonomic Computing*, IEEE CS Press, New York (I), May 2004.

[ZamJW03] F. Zambonelli, N. Jennings, M. Wooldridge, “Developing Multiagent Systems: the Gaia Methodology”, *ACM Transactions on Software Engineering and Methodology*, 12(3):410-470, July 2003.

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

[ZamP04] F. Zambonelli, V. Parunak, “Towards a Paradigm Change in Computer Science and Software Engineering”, *The Knowledge Engineering Review*, Vol. 18, 2004.