Spatial Computing: the TOTA Approach

Marco Mamei, Franco Zambonelli

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

Abstract. *Spatial abstractions promise to be basic necessary ingredients for a novel "spatial computing" approach to distributed systems development and management, suitable to tackle the complexity of modern distributed computing scenarios and promoting self-organization and self-adaptation. In this paper, we analyze the key concepts underlying spatial computing and show how they can be organized around a sort of "spatial computing stack", in which a variety of apparently very diverse mechanisms and approaches can be properly framed. Following, we present our current research work on the TOTA middleware as a representative example of a general-purpose approach to spatial computing. In particular, we discuss how TOTA can be exploited to support the development and execution of self-organizing and self-adaptive spatial computing applications.*

1. Introduction

During the nineties, most researches in distributed computing have focused on the "network of workstations" scenario [CouDK94]. However, in the past few years, a number of novel scenarios have emerged including: *(i)* micro-networks, 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)* ubiquitous networks, 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 networks, 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 recognize some key common characteristics distinguishing the above scenarios from more traditional ones:

- **Large Scale**: the number of nodes 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 to enforce a strict control over their configuration (consider e.g., the nodes of a P2P network) or to directly control them during execution (consider e.g., the nodes of a sensor network distributed in a landscape).

M. Baldoni, F. De Paoli, A. Martelli, and A. Omicini (Eds.): *WOA 2004 – Dagli Oggetti agli Agenti, Sistemi Complessi e Agenti Razionali*, Torino, Italy, November 29th – December 1st 2004. Pitagora Editrice Bologna. ISBN 88-371-1533-4. http://woa04.unito.it

- **Network dynamism**: the activities of components will take place in network whose structure derives from an almost random deployment process, likely to change over time with unpredictable dynamics. This may be due to factors such as environmental contingencies, failures (very likely e.g., in sensor networks and pervasive computing systems), and mobility of nodes (as e.g. in robot teams and in networks of smart appliances). In addition, at the application level, software components can be of an ephemeral or temporary nature (as 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 environment. On the one hand, situatedness can be at the very core of the application goal (as e.g. in sensor networks and pervasive computing systems devoted to improve our interaction with the physical world). On the other hand, situatedness can relate to the fact that components can take advantage of the presence of a structured virtual environment to organize the access to distributed resources (as e.g., in P2P data sharing networks).

The first two characteristics compulsory require systems to exhibit – both at the network and at the application level – properties of self-organization and self-adaptation (or generally, "self-*" properties). 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, in an autonomic way. The last characteristic calls for an approach that elects the environment, its spatial distribution, and its dynamics, to primary design dimensions. In any case, the three aspects are strictly inter-related, in that the enforcement of self-* properties cannot abstract from the capability of the system to become "context-aware", i.e., to have components perceive the local properties of the environment in which they are situated and adapt their behavior accordingly.

In the past few years, a variety of solutions exploiting specific self-* properties to solve specific application problems for large-scale systems in dynamic networks are being proposed [Dim04]. The question of whether it is possible to devise a single unifying conceptual approach, applicable with little or no adaptations to a variety of application problems and to scenarios as diverse as P2P networks and local networks of embedded sensors, is still open.

In this paper, we identify the important role that will likely be played in that process by spatial abstractions, and by their adoption as building blocks for a novel general-purpose "spatial computing" approach for distributed system development and management. A spatial computing approach – by abstracting the network as a continuum space and by having application level activities expressed in terms of sensing the properties of space and navigating in it – can effectively deal with network dynamics in large scale systems, can facilitate the integration of variety of self-* properties in distributed systems, and also suit systems whose activities are situated in an environment.

The remainder of this paper elaborates on spatial computing and is organized as follows. Section 2 introduces the basic concepts underlying spatial computing and discusses their relations with self-* properties. Section 3 proposes a framework around which to organize the basic abstractions and mechanisms involved in spatial computing. Section 4 presents our current research work on the TOTA middleware, as a representative example of a general-purpose approach to spatial computing. Section 5 concludes by sketching a rough research agenda in the area.

2. Spatial Computing

The key principles underlying spatial computing are that:

(i) the central role of the network – a discrete system of variously interconnected nodes – evolves into a concept of space – i.e., an abstraction of a metric continuum built over the network;

(ii) all application-level activities are abstracted as taking place in such space, and rely on the capability of application components of locally perceiving (and possibly influencing) the local properties of space;

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 become "space-aware". On the one hand, they 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 thus by "position", i.e., any entity is characterized by being situated in a specific position in the physical space.

The above characteristics notably distinguish spatial computing from traditional distributed computing models. In *transparent* distributed computing models [CouDK94, ChiC91], components are identified by logical names, applications abstract from the presence of a distributed environment, and only a priori known interaction patterns can be effectively supported. This makes them unable to deal with large-scale systems and with network dynamics. In network-aware models [Wal97], components are typically aware of executing in a network and are identified by their location in it (e.g.., the IP). This enables dealing also with applications executing in large-scale networks, but still call for an explicit and complex handling of dynamic changes in the network or in the position of components. Neither of the two promotes suitable abstractions of environment.

Spatial computing overcomes the above limitations in a very effective way:

- **Large scale**: 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**: the presence of a dynamic network is not directly perceived by components, being 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 whose activities are strictly intertwined with a physical or computational environment.

In addition, as discussed in the following sub-section, spatial computing promotes and support self-* computing.

3.1 Self-* Properties in Spatial Computing

Self-* properties, 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 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 self-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.

1.1 Examples of Spatial Computing Approaches

The shift towards spatial computing is an emerging trend 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 self-constructing a virtual representation of the physical space [NagSB03]. Moreover, they can take advantage of "geographical" communication and routing protocols: messages and events flow towards specific position of the physical/virtual space rather than towards specific nodes, thus surviving in an self-adaptive way the possible dismissing of some nodes [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] 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 self-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 [MamVZ04] explicitly exploit spatial abstractions (and, therefore, a sort of spatial computing model) to organize distributed activities.

3. Framing Spatial Computing

Let us now have a more systematic look at the basic mechanisms that have been exploited so far in distributed computing to promote self-* properties in distributed systems. 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.

3.1. A Spatial Computing Stack

In this section, we introduce the "space-oriented" stack of levels (see Figure 1) as a framework for spatial computing mechanisms. In each level of the stack, by introducing a new paradigm rooted on spatial concepts, it is possible to interpret a lot of proposed self-*

approaches, in different scenarios, in terms of mechanisms to manage and exploit the space (see Table 1). On this basis, it is likely that a simply unifying model for self-* distributed computing – leading to a single programming model and methodology and – can be actually identified.

The "*physical level*" deals on how components start interacting – in a dynamic and spontaneous way – with other components in the systems. This is a very basic expression of self-organizing 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 and to start interacting with each other.

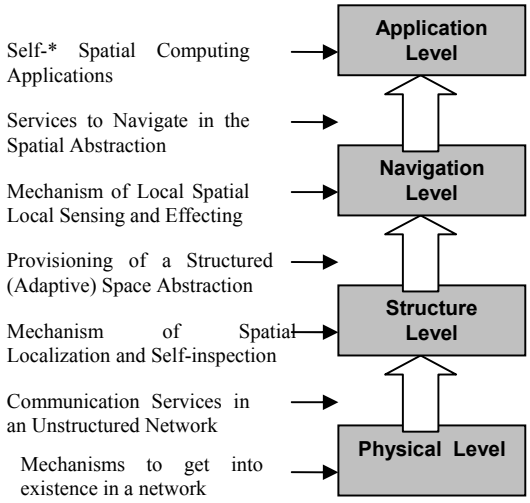

Figure 2. A Spatial Computing Stack.

The "*structure level*" is the level at which a spatial structure is built and maintained by components existing in the physical network. 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 self-organizing and self-adapting 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 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.

The "*navigation level*" regards to the basic mechanisms that components exploit to orient their activities in the spatial structure and to sense and affect the local properties of space. 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 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 [BosM01].

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;
- 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.

	MICRO NETWORKS **Nano Networks, Sensor Networks, Smart Dust, Self-Assembly, Modular Robots**	UBIQUITOUS NETWORKS **Home Networks, MANETs, Pervasive Environments, Mobile Robotics**	GLOBAL NETWORKS **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 to *interact*)	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

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).

4. TOTA: a Middleware Approach to Spatial Computing

The ambitious 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. Earlier in this paper we have strongly advocated the generality, flexibility, and modularity of a spatial computing approach. Although we have do not have the ultimate proof that spatial computing can be effectively put to practice and fulfill all its promises, our experience in spatial computing with the TOTA [MamZ04] middleware can support in part our claims.

The TOTA middleware (short for "Tuples On The Air"), gathers concepts from both tuple space approaches [Cab03, MamZL04] and event-based ones [Car01, Jini] and extends them to provide applications with simple and flexible mechanisms to create, self-maintain, and exploit at the application level a variety of spatial structures, implemented by means of distributed tuples. Unlike traditional shared data space models, tuples are not associated to a specific node (or to a specific data space) of the network. Instead, tuples are injected in the network and can autonomously propagate and diffuse in the network

accordingly to a specified pattern.

To support this idea, the typical scenario of a TOTA application is that of a peer-to-peer network of possibly mobile nodes, each running a local version of the TOTA middleware. Each TOTA node holds references to a limited set of neighboring nodes and can communicate directly only with them.

Upon the distributed space identified by the dynamic network of TOTA nodes, each component is capable of locally storing tuples and letting them diffuse through the network. Tuples are injected in the system from a particular node, and spread hop-by-hop accordingly to their propagation rule. In fact, a TOTA tuple is defined in terms of a "content", and a "propagation rule". **T=(C,P)**. The content **C** is an ordered set of typed fields representing the information carried on by the tuple. The propagation rule **P** determines how the tuple should be distributed and propagated across the network. This includes determining the "scope" of the tuple (i.e. the distance at which such tuple should be propagated and possibly the spatial direction of propagation) and how such propagation can be affected by the presence or the absence of other tuples in the system. In addition, the propagation rules can determine how the content of a tuple should change while it is propagated. Tuples are not necessarily distributed replicas: by assuming different values in different nodes, tuples can be effectively used to build a distributed data structure expressing contextual and spatial information. So, unlike traditional event based models, propagation of tuples is not driven by a publish-subscribe schema, but it is encoded in tuples' propagation rule and, unlike an event, can change its content during propagation (see figure 3).

Distributed tuples must be maintained coherent despite network dynamism. To this end, the TOTA middleware supports tuples propagation actively and adaptively: by constantly monitoring the network local topology and the income of new tuples, the middleware automatically re-propagates tuples as soon as appropriate conditions occur. For instance, when new nodes get in touch with a network, TOTA automatically checks the propagation rules of the already stored tuples and eventually propagates the tuples to the new nodes. Similarly, when the topology changes due to nodes' movements, the distributed tuple structure automatically changes to reflect the new topology.

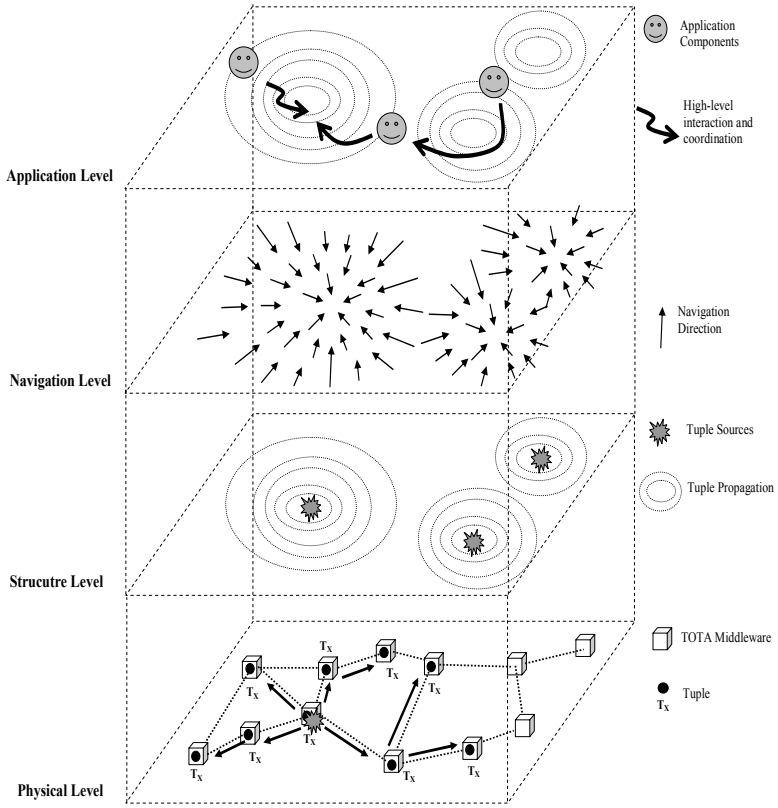

Figure 3: The General Scenario of TOTA in the spatial computing stack: at the physical level there is the network, communication is broadcast of messages encoding TOTA tuples. At the structure level, the space is represented by means of the TOTA distributed tuples. At the navigation level spatial structures can provide basic navigation directions. At the Application level complex coordination tasks can be achieved.

The TOTA middleware supports the spatial computing stack introduced in section 4. In fact, from the application components' point of view, executing and interacting basically reduces to create distributed spatial structures in the network (inject tuples), navigate such spatial structures (sense tuples in a neighborhood), and act accordingly to some application-specific policy.

To clarify and ground the discussion we introduce the following exemplary pervasive computing case study application: tourists with wireless PDAs visit a museum provided with an embedded computer network. We suppose that the PDAs and the embedded devices run the TOTA middleware and that they connect with each other forming a multi-hop mobile wireless network. In the following subsections, working on this case study application, we will detail how TOTA deals with all the levels in the spatial computing stack.

4.1. Physical Level

The physical level deals with how components find and start communicating with each other. At this level, the specific nature of the network scenario has an important role. Since our primary focus is pervasive computing, we mainly consider a wireless network scenario without long-range routing protocols available (like in a “bare” mobile ad-hoc network). In such scenario, it is easy to identify the node's neighborhood with the network local topology (e.g. all the nodes within 10m, for a Bluetooth network). In this case, a TOTA node detects in-range nodes via one-hop message broadcast.

Turning the attention to the case study, each PDA detects neighbor devices, by broadcasting and receiving “here I am” messages. Such discovery operations is executed periodically to take into account the possible movements of users. Upon injecting a tuple, the TOTA middleware broadcasts the tuple to its current neighbors. There, the tuple will be recursively broadcasted hop-by-hop to travel across the network, accordingly to its propagation rule.

To support our experiments, we developed a first prototype of TOTA running on HP IPAQs 36xx equipped with 802.11b wireless card, Familiar LINUX and J2ME-CDC (Personal Profile). IPAQs connect locally in the MANET mode (i.e. without requiring access points) creating the skeleton of the TOTA network. Tuples are being propagated through multicast sockets to all the nodes in the one-hop neighborhood. The use of multicast sockets has been chosen to improve the communication speed by avoiding 802.11b unicast handshake. By considering the way in which tuples are propagated, TOTA is very well suited for this kind of broadcast communication. We think that this is a very important feature, because it will allow in the future implementing TOTA also on really simple devices (e.g. micro mote sensors [Pis00]) that cannot be provided with sophisticate communication mechanisms.

It is important to remark that, despite our focus to wireless networks and pervasive computing, the TOTA mechanisms are general and independent from the underlying physical network. For example, in an Internet scenario (where a long-range routing protocol is available), TOTA identifies the neighborhood of a node with the nodes whose IP address is known (a node can communicate directly with another, only if it knows the other node's address). To realize neighbors discovery, TOTA can either download from a well-known server the list addresses representing its neighbors or it can start an expanding-ring search to detect close nodes [RipIF02]). Given that, the multi-hop propagation of a tuple proceeds as previously described.

4.2. Structure Level

TOTA tuples create a “structure of space” in the network. At the basic level, once a tuple is injected from a node and propagates across the network, it creates a source-centered spatial structure identifying some spatial features relative to the source.

For example, a tuple incrementing one of its fields as it gets propagated identifies a spatial structure defining the network distances from the source. This kind of structure of space provides spatial awareness to application agents. In fact, an agent is both able to infer its approximate distance from the source (in terms of hops – i.e. network link range), and the direction of the source by looking at where the gradient of the tuple descends.

Moreover, TOTA allows to combine different tuples to create more complex spatial representations. A particularly significant example of these mechanisms is the creation of

shared coordinate systems in the network on the basis of mere connectivity. Localization, in general, can rely on the (geometrically intuitive) fact that the position of a point on a surface can be uniquely determined by measuring its distance from at least three non-aligned reference points ("beacons"), via a process of "triangulation" [NagSB03]. Implementing such localization mechanism in TOTA is rather easy. *(i)* A leader election algorithm can elect three beacons nodes. *(ii)* Each beacon "arbitrarily" locates at specific coordinates (without external location information the coordinate system can only be internally coherent [NagSB03]). *(iii)* Each beacon injects a TOTA tuple, increasing its content hop-by-hop and marked with the beacon coordinates. As previously pointed out, this tuple allows other nodes to estimate their distance from the beacon. *(iv)* After at least three beacons had propagated their ranging tuples, nodes can apply a triangulation algorithm to infer their coordinates. Moreover, since TOTA tuples self-maintain, the coordinate system remains up to date and coherent despite network dynamism. If upon a node movement the topology of the network changes, the tuples maintenance triggers an update in the coordinate system, making the latter robust.

A shared coordinate system provides a powerful spatial structure in a network and allows to realize complex navigation and coordination tasks (see later).

In addition, although at the primitive level the space is the network space and distances are measured in terms of hops between nodes, TOTA allows to exploit a much more physically-grounded concept of space.

This may be required by several pervasive computing scenarios in which application agents need to interact with and acquire awareness of the physical space. For instance, one can bound the propagation of a tuple to a portion of physical space by having the propagation procedure - as the tuple propagates from node to node - to check the local spatial coordinates, so as to decide whether to further propagate the tuple or not. In order to bound agents' and tuples' behavior to the physical space, nodes must be provided with some kind of localization mechanism [HigB01]. From our perspective, such mechanisms can be roughly divided into two categories:

- A GPS-like localization mechanism provides absolute spatial information (e.g. it provides latitude and longitude of a node in the network). An actual GPS (Global Positioning System) getting spatial coordinates from satellites naturally belongs to this category. Beacon-based signal triangulation (coupled with beacons actual physical location) is anther example of this category (nodes get their coordinates in an absolute coordinate-frame defined by the beacons [NagSB03])
- A RADAR-like localization mechanism provides local information (e.g. relative distances and orientations between nodes). An actual radar or sonar device belongs to this category (radio and sound waves reflected by neighbor devices enable to infer their distance and orientation). A videocamera installed on a node can serve the same purpose (processing the image coming from the camera, a node can infer where other nodes are). Also network roundtrip-time and signal-strength attenuation may serve this purpose.

The kind of localization mechanism being available strongly influences how nodes can express and use spatial information. GPS-like mechanism are more suitable at defining "absolute" regions. For example, they allow to easily create tuples that propagate across a region defined by means of the coordinates of its corners (e.g. propagate in the square area defined by (0,0) and (100,100)). RADAR-like mechanism are more suitable at

defining “relative” regions, where for example tuples are constrained to travel north form the source or within a specified distance.

It is fair to report that a similar idea has been developed and exploited in the context of a recently proposed language to program a vast number of devices dispersed in an environment [Bor04]. The idea of this programming language is to identify a number of spatial regions relevant for a given application and to access the devices through the mediation of these regions (e.g. for all the devices on the “hill” do that). In [Bor04], the definition of the regions is performed adopting GPS devices and distributed data structures similar to TOTA tuples.

Other than the network and the physical space, one could think at mapping the peers of a TOTA network in any sort of virtual space. This space must be supported by an appropriate routing mechanism allowing distant peers to be neighbors in the virtual space. Such virtual spaces are particularly useful and enable the definition of advanced application such as content-based routing, as in CAN [Rat01]. TOTA concretely supports the definition of these kinds of applications. Also in this case it is fair to report that similar principles have been used in the Multilayered Multi Agent Situated System (MMASS) model [BanMV04]. In MMASS agents' actions take place in a multilayered environment. Each layer provides agents with some contextual information supporting agents' activities. The MMASS environment is thus a hierarchy of virtual spaces built upon one another, where lower layers provide the routing infrastructure for upper ones.

4.3. Navigation Level

TOTA defines a set of API to allow application components to sense TOTA tuples in their one-hop neighborhood and to locally perceive the space defined by them. Navigation in the space consists in having agents act on the basis of the local shape of specific tuples.

As a first simple example we can consider physical navigation. Turning the attention to our case study, it is clear that a PDA injecting a hop-increasing tuple in the network, becomes immediately reachable by other users. Users, in fact, can move following the gradient of the tuple downhill, to reach the tuple source. Moreover, since the tuple shape is maintained despite network dynamism, users can reach the source of a tuple even if it moves.

Navigation is not related to physical movement only. TOTA allows to relate the propagation of a tuple to other tuples already propagated (e.g. a tuple can propagate following another tuple). This can be at the basis of the routing algorithm detailed in the following [Poo00]. In very general terms, when a node “A” wants to send a message to a node “B”, it actually injects the network with a TOTA tuple, that holds: the source identifier i.e. “A”, the message, and the number of hops from the source of the message to the current node. Such structure not only trivially hand-off the message to “B”, but creates a path leading to “A” that can be exploited for further uses. If node “B” wants to reply, it can just send a message that follows the “A”-field downhill towards node “A”. In this case no flooding is involved. The field-like distributed data structures created in this process, can be used further also by other peers to communicate.

Complex spaces enable advanced navigation strategies. A shared coordinate system, like the one described in the previous section, allows, for example, to set-up geographic routing algorithm [BosM01]. A geographic routing algorithm is a mechanism that takes

advantage of the established coordinate frame to send messages to the node closer to a specific location. Such algorithm is suitable in a lot of application scenarios because it inherently supports communication decoupling in that senders and receivers are decoupled by the coordinate frame. For example, a sender can send a message to an unknown receiver located at a specific location and the message will be received by whoever is closer to that location.

4.4. Application Level

The spatial abstractions and tools promoted by TOTA enable to easily realize complex coordination tasks in a robust and flexible way.

Our research, up to now, has mainly focused on the problem of enabling a group of agents to coordinate their respective movements (i.e. distributed motion coordination). Specifically, considering our case study, we focus on how tourists can be supported in planning their movements across a possibly large and unfamiliar museum and in coordinating such movements with other, possible unknown, tourists. Such coordination activities may include scheduling attendance at specific exhibitions occurring at specific times, having a group of students split in the museum according to teacher-specific laws, helping a tourist to avoid crowd or queues, letting a group of tourist to meet together at a suitable location, and even helping to escape accordingly to specific evacuation plans.

An intriguing possibility to realize motion coordination is to take inspiration from the physical world, and in particular from the way masses in our universe move accordingly to the gravitational field. By interpreting (rather roughly) the General Relativity Theory, we can say that the gravitational field actually changes the structure of the space letting particles to globally self-organize their movements. Under this interpretation, particles achieve their "tasks" by simply following the structure of the space.

Realizing this kind of idea with the spatial abstraction promoted by TOTA is rather easy. Under the assumption that users spread hop-counting tuples in the network, it is possible to realize several coordination tasks. A group of tourist following downhill each other tuples will collapse in a single location allowing the tourists to meet somewhere in the building. Analogously, museum's guides could decide to sense each other's tuples (i.e. spaces) so as to maintain a certain distance from each other to improve their reachability by tourists. If a guide has to go away, the same tuples would allows the others to automatically and adaptively re-shape their formation.

Following this approach, agents achieve their goals not because of their capabilities as single individuals, but because they are part of an auto-organized system that leads them to the goal achievement. Such characteristics also imply that the agents' activities are automatically adapted to the environmental dynamism, which is reflected in a changing spatial representation, without forcing agents to re-adapt themselves.

Motion coordination with spatial abstractions is by no means limited to the presented case study. It can be applied to a wide range of scenarios ranging from urban traffic management, mobile software agents on Internet and even self-assembly in modular robots (detailed in the following). A modular or self-reconfigurable robot is a collection of simple autonomous mobile robots with few degrees of freedom. A distributed control algorithm is executed by all the robots that coordinate their respective positions to let the robot assume a global coherent shape or a global coherent motion pattern (i.e. gait).

From a methodological viewpoint, robots can exploit spatial abstraction and TOTA

tuples to self-organize their respective positions in space. In particular, starting from any spatial configuration of robots: *(i)* robots start diffusing specific types TOTA tuples; *(ii)* robots react to locally perceived tuples by trying to follow them downhill/uphill, or by changing their activity state possibly depending on the perceived values of the tuples (i.e. depending on their position in some abstract space); *(iii)* changes in the activity state of robots can lead to inhibiting the propagation of some tuples and/or to the diffusion of new types of tuples in the system, leading back to point *(i)*. One can then apply this process several times, with new types of tuples being propagated in different phases, so as to incrementally have robots self-organize into the required shape [MamVZ04].

In all these application scenario, we verified that the spatial abstractions promoted by TOTA effectively support robust and flexible self-organizing behaviors.

5. Conclusions

By abstracting the execution of distributed applications around spatial concepts, spatial computing promises to be an effective approach towards the identification of general and widely applicable self-* approaches to distributed systems development and management. Our experiences with the TOTA middleware confirm the effectiveness of the approach.

However, besides the claims of this paper and our personal experience, much 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 approach to self*- computing. In particular:

- Is the spatial computing stack depicted in Table 1 meaningful and useful, or a better and more practical framing can be proposed?
- If and when such a unifying model will be found, will it be possible to translate it into a limited set of programming abstractions and lead to the identification of a practical methodology for developing self-organizing distributed computing systems?
- Is a middleware-centered approach like that of TOTA the best direction to follow?
- Several self-organization phenomena disregarded by this paper, deals with concepts that can be hardly intuitively mapped into spatial concepts. Would exploring some sorts of spatial mapping be still useful and practical? Would it carry advantages?
- Possibly most important of all questions: is the search for a unifying model fueled by enough applications? Or it is rather the search for specific solutions to specific problems the best direction to follow?

In our hope, further researches and a larger variety of studies about self-* properties in distributed systems will soon provide the correct answers to the above questions.

References

[BanMV04] S. Bandini, S. Manzoni, G. Vizzari, “Towards a Specification and Execution Environment for Simulations based on MMASS: Managing at-a-distance Interaction”, *Fourth International Symposium From Agent Theory to Agent Implementation (AT2AI'04)*, Vienna, Austria, 2004.

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

[BosM01] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, “Routing with Guaranteed Delivery in Ad Hoc Wireless Networks”, Wirleless Networks 7:609-616, Kluwer Academic Publisher, 2001.

[Cab03] G. Cabri, L. Leonardi, M. Mamei, F. Zambonelli, Location-dependent Services for Mobile Users, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems And Humans*, Vol. 33, No. 6, pp. 667-681, November 2003

[Car01] A. Carzaniga, D. Rosenblum, A. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service", ACM Transaction on Computer System, 19(3):332-383.

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

[CouDK94] G.Coulouris, J. Dollimore, T. Kindberg, *Distributed Systems. Concepts and Design* Addison-Wesley, second edition, 1994.

[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.

[HigB01] Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, vol. 34, no. 8, Aug. 2001, pp. 57–66.

[Jini] JINI, http://www.jini.org

[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.

[MamVZ04] M. Mamei, M. Vasirani, F. Zambonelli, "Experiments of Morphogenesis in Swarm os Simple Mobile Robots*", Journal of Applied Artificial Intelligence* (to appear) 2004.

[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.

[MamZL04] Mamei, M., and F. Zambonelli. 2004b. Co-Fields: a Physically Inspired Approach to Distributed Motion Coordination. *IEEE Pervasive Computing*, 3(2):52-60.

[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.

[Poo00] R. Poor, Embedded Networks: Pervasive, Low-Power, Wireless Connectivity, *PhD Thesis, Massachusstes Institute of Technology*, 2001.

[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.

[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.