Programming Ubiquitous and Mobile Computing Applications with TOTA Middleware

Marco Mamei, Franco Zambonelli, and Letizia Leonardi • Universita di Modena e Reggio Emilia

Tuples on the Air (TOTA) facilitates the programming of distributed pervasive applications by relying on distributed data structures spread over a network resembling an electromagnetic field. Application agents use these structures to extract contextual information and to coordinate with each other effectively.

Computer-based systems will be increasingly embedded in many of our everyday objects and in much of our environment. Typical systems will be communication enabled and able to interact with each other in the context of complex distributed applications.

Such a scenario introduces peculiar and challenging requirements in the development of distributed software systems. Because new agents (we use this term generically to indicate a distributed application's active components) can depart out of and arrive into the network at any time and roam across different environments, applications must be able to deal with such activity adaptively and independently. In addition, the activities of software systems are often contextual—that is, closely connected to the setting (for example, location, time, or environment) in which the systems execute. Setting/contextual characteristics are typically a priori unknown and thus require dynamic enforcement of context awareness.

Adherence to these requirements must not conflict with the need to promote a simple programming model requiring light supporting infrastructures, possibly suited for resource-constrained and power-limited devices. Unfortunately, current practice in distributed software
development, as supported by available middleware infrastructures, isn't likely to address the above requirements effectively. The first obstacle is that because application agents are typically strictly coupled in their interactions (as in message-passing models and middleware), promoting and supporting spontaneous interoperations is difficult. Second, agents are provided with either no contextual information at all or only low-level information (raw local data or simple events) that can't be easily exploited for complex coordination activities. Therefore, both the application and the supporting environment are more complex.

The approach we propose, Tuples on the Air (TOTA), builds on the lessons of uncoupled coordination models such as event-based and tuple-space programming. We aim to provide agents with effective contextual information, still keeping the supporting environment to a minimum and promoting programming simplicity while facilitating the contextual activities of application agents and the definition of complex distributed coordination patterns.

**TOTA**

Tuples on the Air is middleware in which all interactions between application agents occur in a fully uncoupled way via tuple exchanges. However, TOTA has no notion of a centralized, shared tuple space. Rather, tuples can be injected into the network from any node and can propagate and diffuse according to tuple-specific propagation patterns. Agents can exploit a simple application programming interface to define and inject new tuples into the network and to locally sense both tuples and events associated with changes in the tuples' distributed structures (for example, the arrival and departure of tuples). So, agents can easily acquire contextual information to serve as a basis for performing complex distributed activities in a fully distributed and adaptive way.

It helps to think of TOTA tuples as electromagnetic fields: instead of interacting directly with each other, particles locally sense the physical fields' current values. These fields' distributed shape, continuously updated to reflect particle movements, drives local motion activities and leads to distributed and adaptive complex motion patterns.
Our main objective with TOTA is to exploit a unified, flexible mechanism to deal with both context representation and component interaction, leading to simpler, more easily supported applications. We propose relying on distributed tuples to represent contextual information and enable uncoupled interaction among distributed application components. In fact, TOTA tuples form a type of spatially distributed data structure that can express not only messages to be transmitted between application components but also, more generally, contextual information about the distributed environment.

To support this idea, we designed TOTA as a peer-to-peer network of potentially mobile nodes, each running a local version of TOTA. Each node holds references to a limited set of neighboring nodes. The nodes automatically maintain and update the network's structure (determined by the neighborhood relations) to support dynamic changes resulting from either node mobility or node failure. The network scenario's nature determines how each node finds its neighbors. For example, in a MANET (mobile ad hoc network) scenario, TOTA nodes occur within the range of their wireless connection. On the Internet, an expanding-ring search (the same search used in most Internet peer-to-peer systems) finds them.

Within the distributed space identified by the dynamic network of TOTA nodes, each component can store tuples locally and let them spread throughout the network. Tuples injected into the system from a particular node spread hop-by-hop according to their propagation rule. In fact, we define a TOTA tuple in terms of content and a propagation rule, $T = (C, P)$.

Content is an ordered set of typed fields representing the information the tuple carries. The propagation rule determines how the tuple should be distributed and propagated throughout the network. This includes determining the tuple's scope (the distance at which it should be propagated and possibly the propagation's spatial direction) and how the presence or absence of other tuples in the system can affect such propagation. In addition, the propagation rule can determine how a tuple's content should change while it is being propagated. Tuples are not necessarily distributed replicas. By assuming different values
in different nodes, tuples can effectively serve to build a distributed overlay data structure expressing some kind of contextual and spatial information. So, unlike traditional event-based models, tuple propagation isn't driven by a publish-subscribe schema but is directly encoded in a tuple's propagation rule. Moreover, in contrast with an event, a tuple can change its content during propagation.

The spatial structures resulting from tuple propagation must be kept coherent despite network dynamism (see Figure 1). To this end, TOTA middleware supports tuple propagation actively and adaptively. By constantly monitoring the network's local topology and the arrival of new tuples, the middleware automatically repropagates tuples as soon as conditions are appropriate. For instance, when new nodes first contact a network, TOTA automatically checks the propagation rules of the tuples already stored and eventually propagates the tuples to the new nodes. Similarly, when node movements change the topology, the distributed tuple structure automatically changes to reflect the new topology. Figure 1 shows how TOTA can keep a distributed tuple's structure coherent in a MANET scenario despite dynamic network reconfigurations.

Figure 1. TOTA keeps a distributed tuple's structure coherent in a MANET scenario despite dynamic network reconfigurations: (a) A peer Px propagates a tuple that increases its value by one at every hop. (b) When the tuple source Px moves, all tuples are updated to take into account the new topology.
From the application components' viewpoint, executing and interacting are basically reduced to injecting tuples, perceiving local tuples and events, and acting according to some application-specific policy.

Software components on a TOTA node can inject new tuples into the network, defining their content and their propagation rule. The software components can fully access the middleware's local content (that is, the local tuple-space content) and can query the local tuple space via a pattern-matching mechanism to check for the local presence of specific tuples. In addition, the TOTA middleware can notify components of locally occurring events, such as changes in tuple-space content and in the network neighborhood's structure.

Again using the analogy of electromagnetic fields, we find that the overall result is applications whose components can

- Influence the TOTA space by propagating application-specific tuples
- Execute according to the influence of the locally sensed tuples on their internal and coordination activities
- Implicitly tune their activities to reflect network dynamics, as enabled by the automatic reshaping of TOTA middleware tuple distributions.

**Implementation**

Our first TOTA prototype runs on laptops and Compaq IPAQs equipped with IEEE Std. 802.11b and Personal Java. IPAQs connect locally in the MANET mode (no access points required), creating the TOTA network skeleton. Tuples propagate through multicast sockets to all one-hop neighbor nodes. We chose multicast sockets to improve communication speed by avoiding the 802.11b unicast handshake. Because TOTA tuples are propagated in a breadth-first, epidemic manner, this kind of broadcast communication is appropriate. We
think this broadcast capability is important because it will enable TOTA implementation on very simple devices (for example, microsensors) that cannot be provided with sophisticated communication mechanisms. In addition to this communication mechanism, TOTA middleware has at its core a simple event-based engine. This component can collect subscriptions of interesting events and invoke reactions on the subscribed agents in a call-back fashion.

We have only a dozen IPAQs and laptops on which to run the system. Because effective TOTA testing would require more devices, we've implemented an emulator to analyze TOTA behavior in the presence of hundreds of nodes. The emulator, developed in Java, lets us examine TOTA behavior in a MANET scenario, in which node topology can be rearranged dynamically either by a drag-and-drop user interface or by the movement of autonomous nodes. Our emulator's strength is that the adoption of well-defined interfaces between it and the application layers permits the same code installed on the emulated devices to be installed on real Personal Java devices (such as IPAQs) enabled with wireless connectivity. This lets us test applications in the emulator first and then transfer them directly to a network of real devices. To match the emulated and actual scenarios as closely as possible, we emulated devices' battery consumption and wireless network glitches as well.

**Routing on mobile ad hoc networks**

A major challenge in wireless ad hoc networks is the design of routing mechanisms. Over the past decade, several ad hoc network routing protocols have been proposed. Most build a sort of routing overlay structure dynamically by flooding the network with routing packets; then they exploit this overlaid structure for a much finer routing. Rather than proposing a new routing protocol, we'll show how the TOTA model effectively allows creation of a routing overlay structure and the associated routing mechanism (similar to those already proposed).

Here's the basic routing algorithm: When node X wants to send a message to node Y, it injects into the network one tuple representing the message to be sent and another tuple for creating an overlay
routing structure. We describe the latter tuple as

\[ C = ("structure", \text{nodeName}, \text{hopCount}) \]
\[ P = \text{(propagate to all the nodes, increasing hopCount by one at every hop)} \]

The tuple conveying the message will be

\[ C = ("message", \text{sender}, \text{receiver}, \text{message}) \]
\[ P = \text{(if a structure tuple having my same receiver can be found, follow downhill its hopCount; otherwise, propagate to all the nodes)} \]

This routing algorithm is very simple: Structure tuples create an overlay structure so that a message tuple following a structure tuple's hopCount downhill can reach the node that created that particular structure. Wherever such information is absent, the routing simply floods the network with the message tuples. Despite this model's simplicity, it captures the basic underlying principle of several different MANET routing protocols. The basic mechanism-tuples defining a structure to be exploited by other tuples' propagation-is fundamental to the TOTA approach and provides great flexibility. For example, it permits implementation with TOTA content-based routing algorithms, analogous to the algorithm adopted in the Internet peer-to-peer scenario.

In content-based algorithms/models, peers forming an unstructured and dynamic community exchange data and messages not on the basis of the IP addressing scheme but on the basis of message content (for example, "I need the MP3 of 'Hey Jude,' no matter who can provide it to me"). To this end, content-based algorithms/models propose a communication mechanism based on a publish-subscribe model and rely on a properly built overlay space. A peer publishes information by sending it to a particular point in the overlaid space, while another peer reads this information by finding it in the same point of space. (Typically, the process involves a hash function shared by all peers that maps keywords associated with the information content to points in space.) TOTA can realize such systems by using a first layer of tuples to define the overlay space. It then uses other tuples whose propagation rules let them propagate efficiently in the
Although we have been able to realize many examples with TOTA, we still do not have a general engineering methodology or a primitive tuple type on which to build and generalize other kinds of applications. However, this limitation is general and not specific to our work. A general methodology for dealing with bottom-up approaches, like the one TOTA promotes, is still unknown. Such a methodology might yet emerge as research proceeds. Our final goal would be to develop a complete engineering procedure for this kind of model. Meanwhile, deploying applications will definitely help to identify current shortcomings and suggest possible improvements as researchers pursue this goal. We will base much of our future work on using the TOTA model to develop new applications on amorphous sensor networks.

Further information about this project is available at http://polaris.ing.unimo.it.

Marco Mamei is a PhD student in computer science at the University of Modena and Reggio Emilia. He obtained the Laurea degree in computer science in 2001 from the University of Modena and Reggio Emilia. His research interests include distributed and pervasive computing, complex and adaptive systems, and multiagent systems. He is a member of AIIA and TABOO. Contact him at mamei.marco@unimo.it.

Franco Zambonelli is a professor of computer science at the University of Modena and Reggio Emilia. He has a PhD in computer science from the University of Bologna. His current research interests include distributed and pervasive computing, multiagent systems, and agent-oriented software engineering. He is a member of IEEE, ACM, AIIA, and TABOO. Contact him at franco.zambonelli@unimo.it.

Letizia Leonardi is a professor in the department of information engineering at the University of Modena and Reggio Emilia. She has a PhD in computer science from the University of Bologna. Her research interests include design/implementation of mobile-agent systems. She is a member of TABOO and AICA. Contact her at letizia.leonardi@unimo.it.