Using a properties based naming system in mobile agents environments for pervasive computing

F. Tarantino, M. Zambrini, A. Ravani,

Abstract— **Mobile Agents model is widely accepted as a good solution to simplify the coordination in a complex and distributed environment. Pervasive computing, presents many issues regarding resources discovery and heterogeneous routing on different network topologies.**

This paper is concerned about the architecture of a dynamic name properties based system able to manage Agents and places without a network dependent architecture. This approach allows managing places and user profiling, directly at naming level. Different migration patterns can be performed by the agents just working on attributes matching. The mobile agents framework will solve the dynamic binding between properties and resources.

Index Terms—**Mobile Agents, Property based naming system, pervasive applications.**

I. Introduction

A Pervasive Computing environment is "saturated with computing and communication capability, yet so gracefully integrated with users that it becomes a Technology that disappears" [5]. The challenging scenario of pervasive computing means in fact a deep analysis of the system requirements and a careful design of different modules of the "Pervasive Computing Environment". Most of the mechanisms and of the abstractions needed in pervasive computing are currently available in different applications. The main focus is now on system integration, resource discovery and management.

Code mobility allows an easier approach to satisfy some industrial strength requirements, like fault tolerance and scalability. Mobile Agents model also matches most requirements to build effective pervasive applications. In this paper we focused on naming system in order to satisfy some requirements related to Resource Discovery [8].

Naming is an issue that is easily overlooked but is nonetheless fundamental in distributed systems design. Names are needed to refer entities, they are beyond the scope of any single service. A Naming system must grant strong requirements [6]:

- **Unification:** it is often convenient for resources managed by different services to appear together under the same naming scheme; in a pervasive environment locality it becomes a key property to complete the name binding with the resource
- **A Long Lifetime:** many changes will occur in the name space and in the name binding.
- **High availability:** most other systems depend upon the name service; they can't work when it is broken.
- **High expressiveness:** name binding in a heterogeneous environment has to describe different services; expressiveness is provided often through specific context definition.

To achieve these requirements a name system should be designed at low level in framework architecture: low level design often means lack of expressivity.

In a Mobile Agents environment, naming issues are normally related with Agent naming [7]; in pervasive computing, Resource Discovery becomes actually a more critical issue [2], [8]. There are different standard, from Jini to UPnP, but all standard approaches lack of dynamicity and flexibility to manage resources' locality and mobility in a pervasive environment.

If we define a Place as a container of resources locally available for a traveling agent in the net [1], we see that Place Discovery represents the abstractions of the problem in a mobile agents environment.

An interesting approach is property-based naming [6]: any resource is linked through different attributes to express different features. Property based naming is suitable to build extended directory services (Yellow Pages) and X.500 is a related standard. In pervasive computing this model allows to characterize resources in an extended manner, making them more reachable.

We focus on Java to build a complete framework, a porting on

F. Tarantino is R&D Manager in Jakala spa, he is leading a project related the development of multi-channel applications in pervasive environment (WAMP project). Email: ftarantino@jakala.com

M. Zambrini is Project Engineer in Jakala spa, he is working on mobile agent's framework of WAMP project. Email: mzambrini@jakala.com

A. Ravani in Project Engineer in Jakala spa, he is working on multi-channel web dispatching in WAMP project. Email: aravani@jakala.com

Microsoft .Net will also be possible [3]. The actual diffusion of J2Me Virtual Machine on all the wireless devices of 2.5G and 3G [9] defines Java as the standard de facto in building such type of applications.

In the following sections we describe some features of an extended Mobile Agents and Web Framework called WAMP (Wireless Application Multi-channel Platform) developed in Java and focused on building business pervasive applications.

II. Application Scenario – The Ultimate Traveller's Guide (UTG)

John is a business traveller, he needs to go around Italy but he doesn't speak Italian very good. He uses *the ultimate traveller's guide* on his last generation connected handheld device because he needs help to reach his new destination. The only information that John is able to provide to the application is the destination address and the preferred arrival time. UTG application starts a path finder algorithm matching the available transport systems capable to reach the destination. UTG knows many information related to John' travels and it has a complete profile of John (he doesn't like flying, he has a specific budget for all his journey, etc.). UTG is also able to collect different information, like traffic situation, arrival time constraint or unpredictable event (like strikes).
John wants to go from Milan, Montenapoleone Street to Rome, Fori Imperiali Street. The system can find different solutions and different ways to reach the final destination. Providing some solutions: by plane, by train, or by bus. Matching all the information, UTG chooses the train, selecting it, booking a good seat and buying the corresponding ticket. Now UTG must give assistance to John to reach "Milano Centrale" Station: UTG can now choose between Subway, bus or taxi. As the time constraint is critical, John will take the taxi: UTG calls the taxi service and tells John to wait 5 minute for the car. When taxi arrives the taxi driver already knows the destination to reach, the John's device and taxi software make an automatic transaction to pay the trip. At this moment UTG tells John the platform number and the booking numbers. When John arrives in Rome the UTG tells him how to take the Subway and how to reach Fori Imperiali Street.
In this scenario there are two main issues: UTG pro-activity and Resource Discovery. In the following section we propose a Property based Naming System (PbNS) suitable to manage a so complex environment. UTG application is a Mobile Agent Application suitable for a mobile environment; the following sections take into account this hypothesis.

III. A Property-based Naming System (PbNS)

All the resources are collected in Places [4], an Agent has to find a Place in order to access to a specific resource. The pattern of an Agent seeking a resource is strictly related to the structure of the naming system.
We are going to use a PbNS in this scenario:
A Place can register itself on a PbNS with a unique key (Place ID) and with a dynamic list of properties. The properties identify the local resources available and the place configurations.

The semantic of the different properties depends from context. A specific application has its own context, for this reason PbNS's basic architecture is *Application centric*. An Application is defined as stand alone software providing a specific set of solutions running in the Mobile Agent Framework. This means that every different application, working in the Mobile Agent Framework, has a different scope for the naming system.

The properties are described by a simple string, without any further semantic codification hypothesis. A specific semantic for the property is implicit in the use made by the application; it is a very simple approach that allows the integration of the PbNS directly in the main routing service of the Mobile Agent Framework. This solution provides a more powerful abstraction for Agent's mobility: *Property-based mobility*.

For example:
An Agent needs a *printer service*; the application has defined one place enabled with such service. During the start-up phase, this place adds the attribute *"printer"* in his properties list. An Agent normally uses a specific method to migrate to a new place: ***this.go(newDestination);*** using PbNS is also able to use a new set of methods, strictly related to the place's properties matching; for example ***this.go("printer")***
There is no need to search the correct place: routing and naming are collapsed in one service and name solving becomes implicit.
If the Agent needs to match different properties to find a specific resource, it is possible to use Boolean expressions to create queries on PbNS: ***this.go("(printer AND laser) OR plotter")***

This feature allows creating different solutions spaces of active places matching the requirements. There are few possible policies regarding the selection of a specific place in a group; the policy described here, moves directly the Agent to the first place available, always granting a deterministic behaviour.

This type of services needs different patterns and rules to correctly manage the navigation between the places.
First of all there are two different types of properties:

- **Exclusive properties:** only one place at time can set this type of property. The match is always made by string comparison. Setting a specific property as *exclusive* means that the attribute must remain unique and no other place can have the same attribute.
- **Non exclusive properties (or multiple properties):** there are no specific restrictions with the registration

of the property. Many places can add the same attribute in their property list

A. *Exclusive Properties*

An exclusive property is a unique key (but not necessarily the Place Id); it is normally related on an identifier of a specific service.
If for example we are talking about a collaborative working application, the exclusive property should be the user's username:

Place A Fabio
Place B Alessio
Place C Mario

In order to reach a specific user, an Agent can move easily to the local place. PbNS completely solves the search algorithm:

- The Agent obtains the user's username to interact
- The Agent **goes** close to the user (***this.go("Fabio")***)

The PbNS must be robust and fault tolerant; it must grant naming tables consistency, even in high dynamic situations. A place should complete a un-register action each time it goes down. Possible faults of the network, or simply a hard shutdown by the user (he may simply switch off the device) must be supported by the PbNS without a complex consistency recovery.
This session consistency problem can be avoided using exclusive properties; when a place try to register an exclusive property that is already linked to another place, the system discards the previous registration and completes the new one (*win the last registration)*. This policy could seem unsafe, but actually the application context grants the consistency demanding at application level. All security issues are solved by the allowance of a place to enter in the group of places used by the application.

B. *Places Pool*

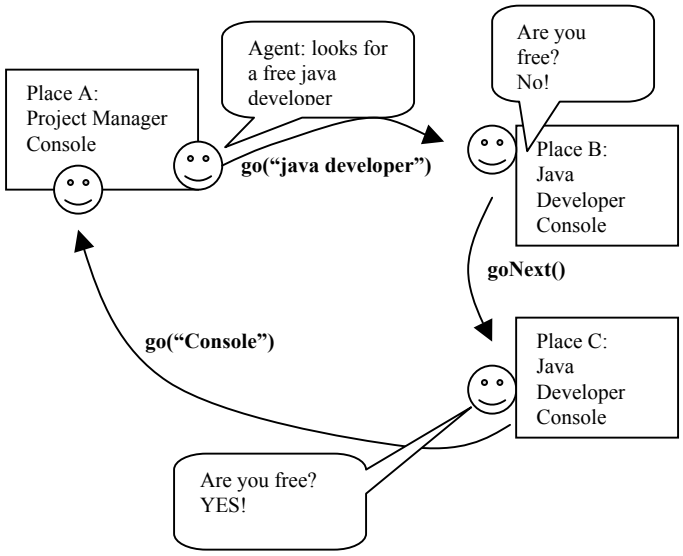
Using non exclusive properties the application can describe all the features of a place. The semantic of such properties is obviously application dependent; a property should be a service name provided either by the place or by a features or a role. It is possible to create many different groups of places through the definition of specific properties.
If we think about a mobile agent application that needs to manage many different data sources, we can produce the following properties configuration:

Place A Console mainGui
Place B ServerB logfile datasource1 datasource2
Place C ServerC datasource2
Place D ServerD datasource2
Place E ServerE datasource2

There are different places and different resources, for example, only *Place B* is able to access to *datasource1*. If an agent needs to access to *datasource2* a pool of places is available: *B, C, D and E.* A pool can be used into two different ways:

- **Warm Resources Replication:** the different places offer the same services, for scalability and fault tolerance reasons the application chooses to have a pool of places instead of one.

Application dependent role management: every place in the pool is identified by an attribute that defines a specific role in the execution (for example a role property should be *"user java developer"* vs. *"user project manager"*). This property becomes the focus of a navigation pattern for the application's Agents.



The control of the navigation sequence is obtained by a new specific migration method: ***this.goNext()***; There is a new status in the session of the agent, the cursors in the list of available places. For sake of simplicity all the session must remain isolated in the agent status; this mean that the position in the list must be stored by the agent.
The complete *goNext()* declaration becomes as follow: ***this.goNext(query, currentPlace)***

The place list can't be stored by the agent, because an high dynamic system doesn't allow to manage potentially obsolete data. Every time an agent wants to make a *goNext()* action, the system have to recalculate the result place list.
This policy creates a possible mismatch between the cursor and the place list, for this reason the system provides a registration policy time dependent: each time a new place is registered, it is appended in the bottom of the list, in order to grant cursor consistency.

IV. Applying PbNS to UTG application

In the UTG scenario the available resources are different type of transport systems. Every transport system has different properties that define its capability to match user needs.
The most important properties are:

- **The service location**: where the service is located. This property identifies not only the physical position of the service itself, but it defines if the service is available in the time constraints of the travel
- **The reachable destinations**: understanding a destination could be very complex; we assume this problem is solved by a specific service in to a set of information with different granularities (Country, City, street, room etc.)
- **The service peculiarities:** it is not possible an *a priori* definition of the service peculiarities, the agent can *react* to specific properties or looking for specific requests provided by the user.

These properties should have different granularity depending of the range of action, for example the train system is useful to reach cities, and subway system allows reaching an address inside a city. A property based approach avoids a complex data model design that could not be enough flexible to manage all possible configurations.
If we think about the specific example in section 2, Italian Railway transport system is represented in Milan by "Milano Centrale Station" and train resource can be identified by these attributes: "*Location-City : Milan*", "*Location-Address : Duca D'Aosta*","*Cost : low*". Inside Railway Station's Place a service is available to verify train's timetables and to book tickets.
If an UTG's Agent needs some information related to train's routes, it can reach Railway Station's Place by the query *(Location-City:Milan AND railway)*. UTG can also add some condition related to the user's profile.
Using a defined interface, the agent can collect travel's information in all places. With PbNS model the attributes can change dynamically to variable conditions and it is possible to access easily on different location with different information's granularity, for example Milan subway system can be defined with many Places with the following attributes: "*Location-Address:Loreto*", "*Subway-Number:2*" or "*Location-Address:Turro*", "*Subway_Number:2*".

V. Results

The use of a PbNS in a Mobile Agents Framework allows new navigation patterns and simplifies the Agent's development. There are many applications in pervasive computing regarding this solution.
We are developing different agent based application using PbNS and we also think that pervasive computing applications will take a benefit from it.
If we think for example about instant messaging systems (a message exchange system through different wired and wireless devices), through PbNS, it is possible to manage profiling of the users directly to the routing system. If a user wants to send a message to all the people that *"like Pink Floyd"* the system solves the query directly at routing time.

The most important benefit of PbNS is related to geographical localization of devices. Localization could become a property for the place; different applications can manage different localization granularities: an application can use attributes like *Milan* or *Madrid*, an other application can use directly longitude and latitude for a finest detail. The flexibility of a PbNS is very high and it is possible to manage optimally many different application's structures.

VI. Related and future work

A complete PbNS, compliant with the policies described here, was implemented in a extended Mobile Agents and Web Framework called WAMP (wireless application multi-channel platform). WAMP is developed by Jakala spa in collaboration with DEIS department, Faculty of Engineering, Bologna University, Italy.
Future work is focused on direct applications of WAMP in collaborative computing and 3G value added services providing. Pervasive computing applications for Collaborative and Work Force Automation is the long term goal of the project.

References

[1] A. Corradi, M. Cremonini, C. Stefanelli*: Locality Abstractions and Security in a Mobile Agent Environment,* **WET ICE '98**, Stanford University, California, USA, June 1998, Collaboration in Presence of Mobility, IEEE Computer Society PressN.M. Karnik and A. Tripathi. "Design Issues in Mobile-Agent Programming Systems". *IEEE Concurrency*, July-Sep\tember 1998.

[2] A. Di Stefano, C. Santoro: *Locating Mobile Agents in a Wide Distributed Environment***, IEEE Transaction on Parallel and Distributed Systems** August 2002 (Vol. 13, No. 8) pp. 844-864.

[3] M. Delamaro, G. Picco: *Mobile Code in .NET: A Porting Experience,* In Proceedings of the 6th International Conference on Mobile Agents (**MA 2002**), Barcelona (Spain), N. Suri ed., Springer Lecture Notes on Computer Science vol. 2355, pp. 16-31, October 2002.

[4] P. Bellavista, A. Corradi, C. Stefanelli: *A Mobile Agent Infrastructure for Terminal, User and Resource Mobility*, Proceedings of the IEEE/IFIP Network Operations and Management Symposium (**NOMS 2000**), Honolulu, Hawaii, April 10-14, 2000, pp. 877-890, IEEE Press

[5] Satyanarayanan, M. *Pervasive computing: vision and challenges*, Personal Communications, IEEE p. 10-17, Volume: 8, Issue: 4, Aug 2001

[6] G. Coulouris, J. Dollimore, K. Kinderberg, *Distributed Systems: Concepts and Design*, Addison-Wesley, 2000.

[7] A. Di Stefano, L. Lo Bello, C. Santoro: *Naming and locating mobile agents in an Internet environment*, Proceedings. Third International Enterprise Distributed Object Computing Conference, 1999. **EDOC '99**. p 153-161

[8] S. Helal: *Standards for service discovery and delivery,* **Pervasive Computing, IEEE** p.: 95- 100, Volume: 1, Issue: 3, 2002

[9] G. Hattori, S. Nishiyama, C. Ono, H. Horiuchi: *Making java-enabled mobile phone as ubiquitous terminal by lightweight FIPA compliant agent platform,* Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(**PerCom 2003**), 23-26 March 2003, p.: 553 -556