Building Mobile Agent Applications in HiMAT

Marco Cremonini, Andrea Omicini
LIA - Dipartimento di Elettronica Informatica e Sistemistica – Università di Bologna - ITALY
E-mail: {mcremonini, aomicini}@deis.unibo.it

Franco Zambonelli
Dipartimento di Scienze dell'Ingegneria
Università di Modena e Reggio Emilia - ITALY
E-mail: franco.zambonelli@unimo.it

Abstract

An engineered approach to the design of mobile agent applications requires appropriate abstractions for both the space where agents move and the conceptual space of mobile agent interaction. The paper introduces the HiMAT *framework for mobile agent applications, which models the Internet as a collection of hierarchical domains, where programmable coordination media rule mobile agent's interaction within each domain and locally to each node. This provides the designers of mobile agent systems with a single, coherent framework enabling them to effectively deal with network topology, authentication, authorisation and coordination in a uniform way.*

1 Introduction

The mobile agent technology promotes a new paradigm for distributed computing which is particularly suited to the design and implementation of Internet applications [12, 17]. Distributed applications should no longer be based only on a set of components assigned to given nodes and remotely interacting through the network. Instead, they can integrate active computational entities (*mobile agents*) capable of moving to different nodes and locally accessing the resources they need. This permits to gain in efficiency, save bandwidth, and make application less sensitive to failures in the communication network.

Nowadays, researches in the mobile agent area have been mainly focused on the basic technological issues to enable Internet mobility. This includes mobile code systems [12], supports for secure execution [8, 16] and for interaction between mobile entities [4, 23]. However, we feel that the novelty and peculiarity of the mobile agent's paradigm requires not only technological support but also structural models to precisely define the environment where agents move and interact.

The general role of mobile agents in Internet applications is to *(i)* proactively move through a multiplicity of different execution frameworks, and *(ii)* there interact with local resources and other mobile agents. Then, an effective model for the engineering of Internet applications based on mobile agents should define and structure both *(i)* the space where agents move and *(ii)* the conceptual space of mobile agent interaction.

From the mobile agent design viewpoint, issue *(i)* is essentially a *topology* one [9]: how agents view the space they roam, whether their representation is partial or complete, and whether it is statically given or dynamically acquired. From the hosting node's viewpoint, the same issue is related to the notion of agent *identity*: the space representation may introduce relationships between nodes and, then, may influence the agent authentication process [18].

Issue *(ii)* concerns the *coordination* of mobile agents and local resources [3, 13]: how their interaction can be constrained and driven so as to result in a system behaviour accomplishing the global system's requirements, how a mobile agent can deal with the heterogeneity of the hosting nodes, and with the unpredictability of other agents behaviour. All the above issues are also strictly related to the *authorisation* problem [21]: agent mobility and interaction have to be coherently ruled according to both global policies, shared by a collection of nodes, and local policies, specialising the global one according to peculiar local needs.

The aim of this paper is to introduce the HiMAT framework for the design and development of mobile agent applications, aimed at addressing all the issues stressed above. HiMAT starts from the TuCSoN coordination model for Internet agents [23], and extends it by allowing the Internet to be modelled as a collection of hierarchical locality domains, where agents can exploit programmable tuple spaces both to coordinate their interactions and to dynamically acquire knowledge about the space they roam. On the one hand, locality domains reflect the hierarchical topology of Internet administrative domains and makes agent motion and authentication easier to be managed. On the other hand, TuCSoN programmable tuple spaces (called *tuple centres*) uniformly act as both the authorisation engines providing agents with the knowledge about the domain resources they can access, and the media for actually interacting with those resources.

The above features make HiMAT a coherent framework for mobile agent applications, which enables system designers to effectively deal with network topology, authentication, authorisation and coordination, in a uniform way. This eases application design and leads to the development of more structured applications, where typical Internet issues such as heterogeneity and unpredictability can be addressed, as we show by means of an application example in the area of distributed information retrieval.

The paper is organised as follows. Section 2 introduces some preliminary concepts: it briefly describes the coordination model adopted (TuCSoN) and discusses its limitations in terms of topological abstractions. Section 3 presents the HiMAT framework. Section 4 shows a simple application example for HiMAT, while Section 5 discusses related works.

2 Preliminaries

HIMAT extends the TuCSoN coordination model [23] with the aim of fully exploiting its power (mostly relying on its programmable communication abstractions called *tuple centres*), and addressing its limitations in modelling the space for agent's motion.

2.1 TuCSoN

As shown in [4], the design of Internet applications based on mobile agents may be strongly influenced by the coordination model adopted. In particular, coordination models exploiting tuple spaces *à la* Linda [5] provide for many features which are essential in design and development of Internet applications based on mobile agents. Among the others, while associative access to information makes it possible to easily deal with partial knowledge and dynamicity of the resources, temporal and spatial uncoupling intrinsically suits autonomous and dynamic entities like mobile agents.

Starting from the above considerations, TuCSoN defines an interaction space spread over a collection of Internet nodes and built upon a multiplicity of independent tuple-based communication abstractions called *tuple centres* [10, 11]. Each tuple centre is associated to a node and is denoted by a locally unique identifier. Each node possibly hosts a multiplicity of tuple centres, providing its own version of the same TuCSoN name space (the set of the tuple centre identifiers), and virtually implements each tuple centre as an Internet service. Any tuple centre can then be identified either via its full Internet (absolute) name or via its local (relative) name. This supports the double role of mobile agents as network-aware entities explicitly accessing to a remote tuple space, and as local entities of their current execution node. An operation on a remote tuple centre must be invoked specifying its full Internet name, as in `tcID@some.node?op(tuple)`, while a local tuple centre can be accessed by specifying its local identifier only, as in `tcID?op(tuple)`.

To overcome the limits of the Linda model, TuCSoN tuple centres enhance tuple spaces with the notion of *behaviour specification*: each tuple centre can be programmed so as to implement its own observable behaviour in response to communication events. Instead of simply triggering the basic pattern–matching mechanism of the Linda model, the invocation of any of the TuCSoN basic communication primitives can be associated to specific computational activities, called *reactions*, having a success/failure transactional semantics. The result of the invocation of a communication primitive is perceived by agents as a single-step transition of the tuple centre state, which combines altogether the effects of the primitive itself and of all the reactions it triggered. Thus, a new observable behaviour can be defined for a tuple centre, where global coordination laws can be embedded.

As shown in [23], coordinating mobile agent's interaction with TuCSoN leads to several interesting features:

- interactions related to different application contexts can be encapsulated and modularised, by exploiting the multiplicity of the tuple centres, each one separately and independently programmable;
- agent's interaction protocols can be made independent of the particular execution framework architecture. This makes it easy to deal with typical heterogeneity of an Internet-based application domain, which is a mandatory issue for the definition of a mobile agent framework;
- coordination policies can be charged upon programmable coordination media, thus freeing agents from the burden of coordination–awareness;
- agent's accesses to resources can be easily controlled, since they are always mediated by a tuple centre. In addition, the possibility of programming specific behaviours enables the definition or enhancement of any required access control policy.

Obviously, tuple centre coordination does not prevent in principle direct agent-to-agent communication, which could be useful in case of data-intensive communication. Instead, tuple centres simply work as the interaction kernel, and could be exploited for instance when negotiating the protocol for establishing a direct communication channel connecting two agents.

2.2 Modelling the Network

As a pure coordination model, TuCSoN falls short in modelling complex and secure application environments, as actually needed to develop real Internet-based application: the TuCSoN interaction space is flat, and the TuCSoN model currently neglects some security issues, by assuming that agent authentication has to be solved by the engine for mobile agent execution.

Instead, Internet-based applications typically deal with intrinsically structured domains: *(i)* Internet nodes are often grouped in clusters, subject to highly coordinated management policies and possibly protected by a firewall; *(ii)* large clusters can be further characterised by the presence of enclosed sub-clusters, in a hierarchical structure of protected administrative domains. A typical example can be found in most academic environments: a single large cluster encloses all the academic nodes and defines basic management policies; different enclosed clusters, such as the ones of single research laboratories, provide protected domains with their own policies, typically under the administration of a single system manager.

The first observation *(i)* has led to several proposals that model the Internet as a collection of locality abstractions, both in the area of mobile agent systems, such as MA [8] and Telescript [26], and in the area of distributed open programming environments, such as ActorSpace [15]. In this context, we propose the *place*, *domain*, and *gateway* locality abstractions. The *place* provides the abstraction for the mobile agent's execution environment, i.e., the Internet node, where agents execute their code and interact with local resources. The *domain* concept is used to model subnetworks by grouping a set of places sharing common policies and privileges. For instance, one domain could represent a LAN subnetwork of a department and provide for a security policy common to all the department places. Specialised security policies may then be defined locally to each place, in order to provide a fine–grained control of the interaction with local resources. Finally, mobile agents moving from one domain to another one rely on a specific abstraction, the *gateway*, in charge of inter-domain routing for both incoming and outgoing agents.

The second consideration *(ii)* suggested assuming a *hierarchical* structure, where domains can contain both places and other (sub-)domains. This provides for higher flexibility in modelling complex network structures, which are often intrinsically hierarchical, and for a better support for decentralised management, and also enables the definition of global policies at different organisation levels, given the clear separation of all the administrative domains of a network.

As shown in Figure 1, the distribution of domains on the network can be represented with a tree structure where each node is a domain (containing places) and each arc is a sub-domain gateway. Each hierarchical domain has a most external gateway (gateway 1 in Figure 1) bridging it with Internet and enabling communication between different domains. Here Internet acts as an unstructured communication network between domains. However, note that also the internal structure of a domain is Internet-based: in this context, the distinction is between the portion of the net modelled hierarchically according to the locality abstractions defined above, and the rest of the Internet.

The hierarchical definition adds a new meaning to the role of gateways with respect to a non-hierarchical system. Besides acting as centralised points for domain access control, along with authentication of mobile agents, gateways can be naturally exploited to provide mobile agents with a multi–layered description of the network topology, where each gateway only describes a single level (the structure of its associated domain). This enables a better management of the system knowledge, by delegating to each domain (through its gateway) the representation and management of what is related with its inner structure only, security issues included.

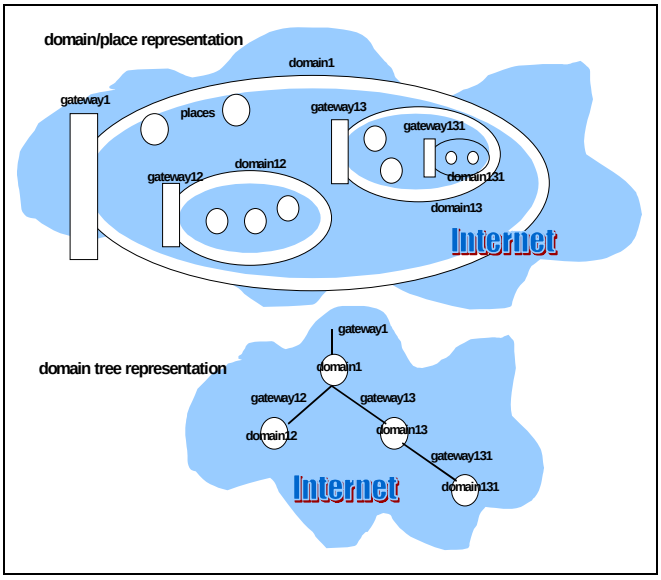

Figure 1: Domain/place and domain tree representations

Moreover, this simplifies the task of mobile agents when dealing with network topology, since information about domains can be acquired incrementally on need, whenever crossing gateways and entering new domains. As a result, mobile agents, during their migration towards local resources, could see and gain knowledge only of the sub-network strictly related with their task and application-dependent role. By shifting the complexity of knowledge management from agents to domains and gateways, this approach avoids the agents need for an *a priori*, complete knowledge of the system topology.

In principle, this approach can be used to model both physical and logical network structures. By grouping the nodes of a domain according to application/agent identity, we may easily think of modelling the network according to the specific needs of every mobile agent application. For instance, all the nodes delegating gateway *G* for the authentication of the agents of the application *A* belong to the same logical domain with respect to identity *A*. Therefore, several logical, application-dependent, possibly overlapping trees can then be mapped onto the same physical network.

3 The HiMAT framework

HiMAT extends TuCSoN and defines a coherent framework suitable to supporting all the phases of the life of a mobile agent. These include its movement over the network (*topology*), the verification of its identity (*authentication*), the acquisition of the necessary knowledge and permissions to access resources (*authorisation*), and the management of its interactions with other agents and with local resources (*coordination*).

3.1 The HiMAT architecture

The above four aspects are modelled by the overall HiMAT infrastructure. The first (*topology*) and the last (*coordination*) are achieved by exploiting the features of TuCSoN extended with a hierarchical framework: the hierarchical organisation of domains, and the distribution of programmable tuple centres coordinating interaction, respectively. In addition, TuCSoN is exploited as the authorisation engine with respect of resource access, and gateways are used as filters controlling agent's movement and domain representation in an application-oriented environment.

More in detail, the *topology aspect* of HiMAT is achieved by exploiting the locality abstractions defined in Subsection 2.2 to model a network context. Each domain groups a set of places along with other (sub)-domains, implicitly defining a tree structure (HiMAT *tree*). The hierarchical structure of a HiMAT tree can be explored traversing each sub-domain *gateway*. For example, with respect to the schemata of Figure 2a, **`mag`** gateway is a place of **`cs`** domain (where **`mag`** stands for Mobile Agent Group and **`cs`** for Computer Science Department).

Given this structure, an agent moving along a HiMAT topology and accessing a new domain through its gateway can explore the (ordinary) places and gateways belonging to that domain, as the agent **`ag1`** in Figure 2.

The role of gateways is fundamental to support and control the execution of the agents roaming along the network (*authentication* and *authorisation* issues). Gateways are the natural places where to perform the *authentication* of incoming agents for its associated domain, by verifying agents' identity and by propagating it by default to all domain's places and sub-gateways. More in detail, an agent coming from an external Internet source must be authenticated by a gateway as in common Internet-based applications with security requirements (i.e. by relying on public key infrastructures managed by trusted third-parties). Once that the agent begins the exploration of the inner topology, deeper gateways could decide to trust the authentication already performed by the higher gateway and perform only a weaker form of authentication: for example, by simply verifying that the agent comes from the higher gateway (i.e. by relying on an internal public key infrastructure), as shown in Figure 2b. This permits to gain in performances and to reduce the complexity of security requirements in mobile agent applications. Of course, any gateway is left free to autonomously fully authenticate an agent, wherever the agent comes from.

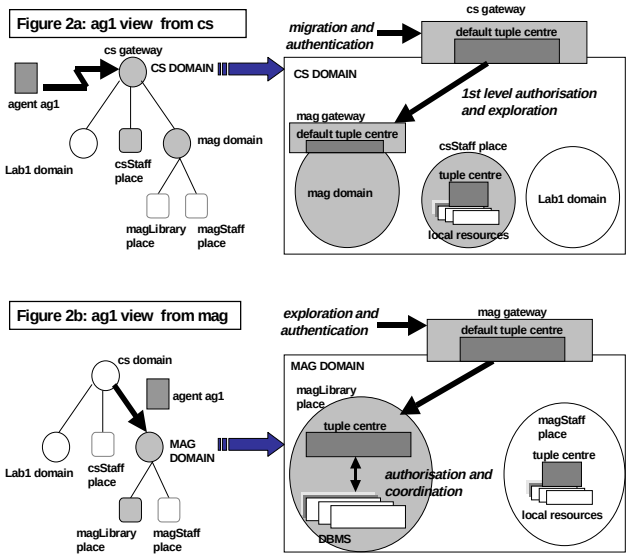

Figure 2 : The HiMAT architecture (*white boxes are place/gateway hidden to the agent; grey boxes are places/gateways accessible by the agent*)

In addition to its authentication role, a gateway works also as a knowledge repository, providing agents with information about the structure of its associated domain, and can filter such information according to agents' identity. This makes gateways implicitly work as the *first authorisation level* of HiMAT. A mobile agent can dynamically retrieve from a gateway the set of the accessible places and sub-gateways in the domain, as well as the set of visible tuple centres locally provided by each place of the domain. For example, in Figure 2a, the **`cs`** gateway authorises agent **`ag1`** to access the **`mag`** sub-gateway and the **`csStaff`** place but prevents it to access the **`lab1`** domain, by hiding **`lab1`** existence to **`ag1`**. When **`ag1`** moves to the **`mag`** gateway, as in Figure 2b, it can access the **`maglibrary`** place only, because the existence **`magStaff`** is kept hidden to it. In HiMAT gateways, both the function of dynamic knowledge acquisition and authorisation are achieved via a single tuple centre, programmed so as to make agent exploration easier and to manage the interaction with local resources. This, however, do not actually force agents to physically move to all the gateways of a tree to get the required resource information. According to the TuCSoN model, in fact, agents can interact with gateways also remotely, possibly performing a sort of *virtual exploration* before actually migrating to some places.

In this paper, we don't address the issue of specific services definition (i.e. directory, accounting, billing, auditing services, administration tools). However, we argue that any required service can be defined and implemented by exploiting the programmability of tuple centres (for example, by masking services behind tuples) and their distribution on the gateways.

3.2 The HiMAT exploration protocol

To better understand HiMAT, Table 1 reports the scheme of a possible exploration protocol of a mobile agent moving through an environment modelled as a HiMAT tree. We label with *network topology* the operations executed by the mobile agent to roam the tree and to deal with HiMAT locality abstractions, and with *local interaction* the ones executed by the mobile agent to interact with local resources and other mobile agents through the tuple centres provided by each HiMAT place.

Network topology	
`<goto d>`	*migration to gateway* `d`
`<identify>`	*gateway* `d` *authenticates the agent on behalf of all the places of its associated domain*

`?read(subdomlist)`
`?read(placelist)`
`?read(commspace)`
	access to the default tuple centre of the gateway to obtain information about domain structure, in terms of accessible sub-domains (`subdomlist`*), places (*`placelist`*), and tuple centres (*`commspace`*), filtered according to agent's identity and credentials, provides for the first authorisation level*

`<for pl in placelist do>`	*exploration of the accessible places of the domain*
`< goto pl>`	*migration to place* `pl`
Local interaction	
`<for tc in commspace do>`	*for all the visible tuple centres of place* `pl`
`tc?op(Tuple)`	*ask tuple centre* `tc` *of place* `p` *to execute* `op` *over* `Tuple`*, if authorised by* `pl`
Network topology (sub-trees)	
`<for sd in subdomlist do>`	*exploration of the accessible sub-domains*
`<goto sd>`	*migration to gateway* `sd`
`<...>`	*keep on exploration and access, in a recursive fashion*

Table 1. The HiMAT exploration protocol

The example points out the positive impact of HiMAT on agent design. First, there is a clear distinction between the modelling of the space through which mobile agents roam (*network topology*) and the management of the interaction among mobile agents and resources (*local interaction*). In addition, HiMAT enables *(i)* security policies and access control to be developed in a structured environment where agents migrate and *(ii)* the exploitation of the same interaction protocol, mediated by the TuCSoN programmable communication abstractions, to define both authorisation and coordination policies.

As a result, HiMAT grants the proper level of autonomy and isolation of each domain (for administrative, security, management reasons), by making it possible to find a good balance between the dual issues of security and usability, enabling at the same time a more structured agent design.

4 Exploiting HiMAT

Let us consider an application in which mobile agents look for book references through the Internet nodes of University libraries. In this scenario, there is generally one central library and several departmental libraries, along with a bunch of book collections owned by each research group. This represents a typical application context suitable to be addressed by mobile agent technology, as well as an ideal scenario for our HiMAT model, due to:

- the hierarchical organisation of the context;
- the decentralisation of management control and decision as well as the presence, potentially at each level of the hierarchy, of global policies;
- the cooperation required to each domain involved in the application;
- the intrinsic heterogeneity of local resources.

Let us consider again the Computer Science department, where the Mobile Agents research Group operates with its own research library. In this case, the physical organisation of the network directly maps into a HiMAT tree with the CS domain (**`cs`**) as the root and all the research group domains as children, such as the MAG one (**`mag`**). Then, both **`cs`** and **`mag`** have a gateway with a default tuple centre where their internal structure is recorded.

The **`mag`** domain defines some places representing different agent execution environments (typically Internet nodes) bound with different local resources. Each place has tuple centres as communication abstractions handling all the interactions between agents and resources. In particular, the two places, **`maglibrary`** and **`magvarious`**, of the **`mag`** domain both provide for the **`books`** tuple centre. However, the **`books`** tuple centre of **`maglibrary`** keeps track of already catalogued book only, while the **`books`** tuple centre of **`magvarious`** keeps also track of not–yet catalogued books.

The application **`bookreader`** is in charge of exploring the **`books`** tuple centres for finding book references, by exploiting two classes of agents: the first one (role **`ordinary`**) has to take into account catalogued

books only; the second one (role **advanced**) has to considers not–yet–catalogued books, too. According to that, the **cs** gateway must authenticate agents' identity, i.e., application identity (**bookreader**) and role identity (either **ordinary** or **advanced**). The **mag** gateway, by its side, can decide to trust **cs** and delegate it the authentication of **bookreader** agents.

Once authenticated by the **cs** gateway, agents access its default tuple centre to discover what places are in the domain and which tuple centres are available. In addition, agents recognise the presence of an inner gateway, the **mag** one, which they may be interested to explore. As **bookreader** agents arrive to the **mag** gateway, they are not re-authenticated, since **mag** delegates this task to **cs**. However, based on their role, **mag** can decide to assign them a particular view of its internal domain structure. In particular, in the example:

- A **bookreader** agent with **ordinary** role (**bookreader:ordinary**), as it comes to the **mag** gateway and access its default tuple centre, is made aware of the **maglibrary** place only and can access its **books** tuple centre. However, it has no way of accessing the **magvarious** place (Figure 3a).
- A **bookreader** agent with **advanced** role (**bookreader:advanced**) is made aware of both the **maglibrary** and **magvarious** places, and can access both associated **books** tuple centres (Figure 3b).

The capability of providing different tuple centre views to different agents – achieved by appropriate behaviour specifications of the default tuple centre – is likely to notably simplify both agent design and management. In fact, agents are not in charge of authorisation issues in tuple centres access, because the gateway simply denies to agents the possibility of acquiring the knowledge about the existence of tuple centres they are not authorised to access to.

A further simplification of agent design comes from providing to agents with a uniform view of all the resources of a domain. Let us suppose that the two **books** tuple centres of **maglibrary** and **magvarious** mediate agent's access to different information structures (Figure 3). Catalogued books are recorded in the **maglibrary** place exploiting a DBMS. The DBMS interfaces with the local tuple centre **books** through a wrapper, which translates tuples into queries and back answers into tuples. Not–yet–catalogued books, since they are typically a small number, are instead directly recorded as tuples of the form **book(*Author*, *Title*)** in the **books** tuple centre provided by the **magvarious** place. The general problem, then, is that semantically homogeneous information (concerning books in our example) are often represented in an heterogeneous way by different information sources.

HiMAT addresses the above problem by considering that agents dynamically acquire knowledge about the domain. Moreover, the tuple centres of a domain can be programmed to dynamically provide agents with a uniform view of the resources they access. This allows agents to adopt the same, straightforward interaction protocol, independently of the (architecture of) their current hosting node, and makes it possible to deal with heterogeneity. Let us suppose agents ask both **books** tuple centres for tuples **bookTitle(*Title*)**. On the one hand, the **maglibrary** tuple centre can be programmed to dynamically produce a tuple representing a DBMS query, put it into the tuple centre, get the consequent answer tuple, and translating it into tuples of the form **bookTitle(*Title*)**. On the other hand, the **magvarious** tuple centre can be programmed to dynamically transform the tuples in the form **book(*Author*, *Title*)** already stored into tuples of the form **bookTitle(*Title*)**.

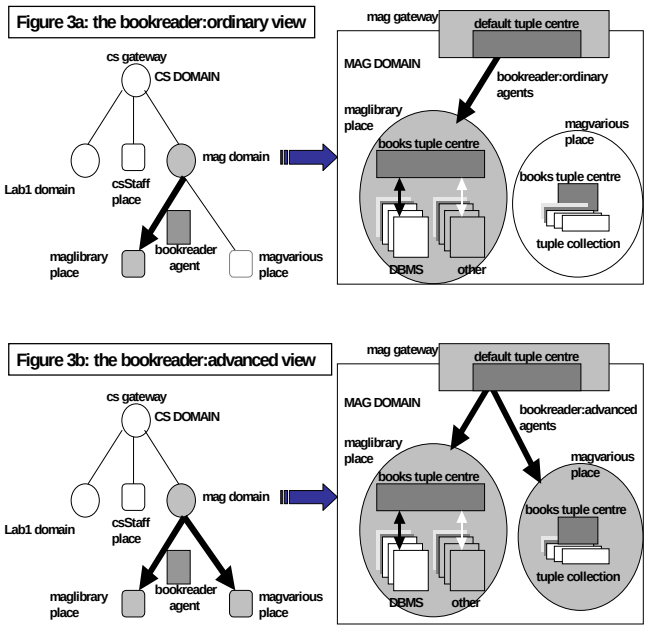

Figure 3: An example of HiMAT application space

5 Related works

Despite the great deal of activity in the mobile agent area [17], only a few proposals focus on the definition of a comprehensive model extensively supporting the migration of agents in a structured environment and the coordination of their interaction with information resources.

Most of the systems focus on how to support mobile agent applications to allow agent motion in the unstructured Internet network, thus generally assuming flat (single-level) architectures (Aglets [19], Ara [24], D'Agents [14], Jumping Beans [1]). Telescript [26] proposes a limited form of hierarchy (i.e. Telescript's nested places), whose goal is simply to provide a mechanism to manage policies with raising degrees of security on the same interpreter, called engine. Differently from HiMAT, Telescript does not

exploit its nested places to develop a hierarchical topology mapping both the organisation structure and the application spaces.

As far as security is concerned, all the systems cited above support traditional cryptographic mechanisms and exploits general-porpoise protocols [25, 27]. They also define basic solutions to deal with authorisation by means of resource access control lists [1] and credentials (called allowances in Ara). Basic security mechanisms could be sufficient to develop some applications in unstructured environments. However, when considering more complex scenarios, we argue that security should be strictly integrated with the organisation topology and exploited to define protocols to rule the motion of agents within the organisation network. In this paper we only sketched the HiMAT security model, which exploits all the mechanisms usually employed for Internet applications and integrates them with tuple centres and the gateway hierarchy. The actual definition and formalisation of HiMAT security support is in progress.

To the best of our knowledge, the only proposals which addresses the issue of the hierarchical structure of many Internet application domains and explicitly models the migration of active entities across protected domains are Ambit [6] and Discovery [20]. However, both the above models fall short in supporting agent exploration and in providing for suitable coordination abstractions.

More in general, the lack of appropriate coordination models is a common weakness of several today's mobile agent systems. Most of them rely on message passing for inter-agent communication and on the client-server model for access to the local information system [17]. As already discussed in this paper, these approaches often do not suit Internet applications based on mobile agents. The notion of programmable coordination medium [10], exploited by TuCSoN in HiMAT, has been already applied in the *ACLT* system, which exploits programmable tuple spaces for the coordination of distributed applications based on intelligent heterogeneous agents [11]. In the context of mobile agents, MARS [3] adopts programmable tuple spaces for mobile agent coordination. Developed in the context of an affiliated project, MARS is mostly oriented to network management duties, rather than to handle accesses to highly dynamic and heterogeneous information sources. The PageSpace project [7] aims to define a general architecture for the coordination of Internet agents: interactions occur via Linda tuple spaces that are not programmable in itself. Instead, special-purpose agents are provided, which are capable of accessing the space permits and changing its content to influence the coordination activity of the applications agents. The ActorSpace model [15] provides a comprehensive framework for building agent ensembles which addresses the coordination issue explicitly, but provides no support for agent exploration. However, none of the above coordination systems addresses in an integrated way the issues of authentication and authorisation in the accesses to the coordination media, which is instead one of the key–features of HiMAT.

6 Conclusions

HiMAT provides a comprehensive and uniform solution to several critical problems raising when designing and developing mobile agent applications, such as topology, authentication, authorisation and coordination. This is achieved by exploiting both locality abstractions for modelling agent's motion through application domains, and programmable communication abstractions for managing agent's interaction.

At the time of writing, we are completing the implementation of a mobile agent system fully supporting the HiMAT model. This system is being implemented by integrating the Java implementations of the MA mobile agent system (which also provides for MASIF compliance [22]) and TuCSoN. Further work will be devoted to a deeper investigation of the issues of the HiMAT security and access control models, as well as on the identification of suitable high-level design patterns for mobile agent applications to be used for HiMAT application development [2].

Acknowledgements: This work has been supported by the Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project "MOSAICO: Design Methodologies and Tools of High Performance Systems for Distributed Applications".

References

1. Ad Astra Engineering, Inc., Jumping Beans White Paper, December 1998. http://www.AdAstraEng.com/
2. Aridor, Y., Lange, D.B., Agent Design Patterns: Elements of Agent Application Design, Proceedings of Autonomous Agents '98, ACM Press, 1998.
3. Cabri, G., Leonardi, L., Zambonelli, F., Reactive Tuple Spaces for Mobile Agent Coordination, 2nd International Workshop on Mobile Agents, Lecture Notes in Computer Science, vol. 1477, 237-248, Springer-Verlag, Sept. 1998.
4. Cabri, G., Leonardi, L., Zambonelli, Coordination Models for Internet Applications based on Mobile Agents, IEEE Computer, 1999, to appear.
5. Carriero, N., Gelernter, D., Linda in Context, Communications of the ACM, 32(4):444-458, 1989.
6. Cardelli, L., Gordon, A., Mobile Ambients, Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, vol. 1378, 140-155, Springer-Verlag, 1998.
7. Ciancarini, P., et al., Coordinating Multiagent Applications on the WWW: A Reference Architecture, IEEE Transactions on Software Engineering, 24(5):362-375, May 1998.
8. Corradi, A., Cremonini M., Stefanelli, C., Locality Abstractions and Security Models in a Mobile Agent Environment, 7th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Applications, Stanford (CA), June 1998.

9. Cremonini, M., Omicini, A., Zambonelli, F., Modelling Network Topology and Mobile Agent Interaction: an Integrated Framework, Proceedings of the 1999 ACM Symposium on Applied Computing, 410-412, San Antonio (TX), Feb. 1999.
10. Denti, E., Natali, A., Omicini, A., Programmable Coordination Media, Proceedings of the 1997 Conference on Coordination Languages and Models, 274-288, Berlin (D), 1997.
11. Denti, E., Natali, A., Omicini, A., On the Expressive Power of a Language for Programming Coordination Media, Proceedings of the 1998 ACM Symposium on Applied Computing, 169-177, Atlanta (G), 1998.
12. Fuggetta, A., Picco, G., Vigna, G., Understanding Code Mobility, IEEE Transactions on Software Engineering, 24(5):342-361, May 1998.
13. Gelernter, D., Carriero, N., Coordination Languages and their Significance, Communications of the ACM, 35(2):97-107, Feb. 1992.
14. Gray, R., Kotz, D., Cybenko, G., Rus, D., D'Agents: Security in a Multiple-Language, Mobile-Agent System. 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, Brussels (B), July 1998
15. Jamali, N., Thati, P., Agha, G. A., An Actor-based Architecture for Customising and Controlling Agent Ensembles, IEEE Intelligent Systems, Special Issue on Intelligent Agents, 1999, to appear.
16. Karjoth, G., Lange, D., Oshima, M., A Security Model for Aglets, IEEE Internet Computing, 1(4), July/August 1997.
17. Karnik, N. M., Tripathi, A. R., Design Issues in Mobile-Agent Programming Systems, IEEE Concurrency, 6(3):52-61, July-Sept. 1998.
18. Lampson, B., Abadi, M., Burrows M., Wobber, E. P., Authentication in Distributed Systems: Theory and Practice, ACM Transactions on Computer Systems, 10(4):265-310, November 1992.
19. Lange, D., Oshima, M., Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley Longman, Inc., 1998.
20. Lazar, S., Weerakoon, I., Sidhu, D., A Scalable Location Tracking and Message Delivery Scheme for Mobile Agents, 7th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Applications, Stanford (CA), June 1998.
21. Lea, D., Nagaratnam, N., Role-based Protection and Delegation for Mobile Object Environments, 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, Brussels (B), July 1998.
22. Milojicic, D., et al., MASIF The OMG Mobile Agent System Interoperability Facility, 2nd International Workshop on Mobile Agents, Lecture Notes in Computer Science, vol. 1477, 50-67, Springer-Verlag, Sept. 1998.
23. Omicini, A., Zambonelli, F., Coordination of Mobile Information Agents in TuCSoN, Journal of Internet Research, 8(5):400-413, 1998.
24. Peine, H., Stolpmann, T., The Architecture of the Ara Platform for Mobile Agents, Lecture Notes in Computer Science, vol. 1219, 50-61, Springer-Verlag, 1997.
25. Sander, T., Tschudin, C.F., Towards Mobile Cryptography, IEEE Symposium on Security and Privacy, May 1998.
26. White, J. E., Telescript Technology Mobile Agents, General Magic White Paper, 1996.
27. Wilhelm, U., Staamann, S., Buttyan, L., Protecting the Itinerary of Mobile Agents, 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, Brussels (B), July 1998.