Ruling Agent Motion in Structured Environments

Marco Cremonini[1], Andrea Omicini[1], Franco Zambonelli[2]

[1] LIA - DEIS – Università di Bologna
Viale Risorgimento 2 – 40126 Bologna – ITALY
{mcremonini, aomicini}@deis.unibo.it
[2] DSI -Università di Modena e Reggio Emilia
Via Campi 213/b – 41100 Modena – ITALY
franco.zambonelli@unimo.it

Abstract. The design and development of cooperative Internet applications based on mobile agents require appropriate modelling of both the physical space where agents roam and the conceptual space of mobile agent interaction. The paper discusses how an open, Internet-based, organisation network can be modelled as a hierarchical collection of locality domains, where agents can dynamically acquire information about resource location and availability according to their permissions. It also analyses the issue of how agent motion can be ruled and constrained within a structured environment by means of an appropriate coordination infrastructure.

1 Introduction

Mobile agents are a promising technology for the design and development of cooperative applications on the Internet [3, 5, 12, 13]. Due to their capability of autonomously roaming the Internet, mobile agents can move locally to the resources they need – let them be users, data, or services – and there interact with them. This can provide for saving bandwidth and, by embedding some sort of intelligence into the agents, for enabling complex and dynamic interaction protocols to be defined, as needed in an open and unpredictable environment as the Internet.

Nowadays, researches in the area of mobile agents have greatly emphasised the dichotomy between mobile, possibly intelligent agents, and static ones. On the one hand, the community of distributed artificial intelligence has mostly focussed the aspects of agent intelligence and of autonomous planning capability [11]. On the other hand, the community of distributed systems has widely discussed the effectiveness of mobility with respect to traditional client/server solutions [8] and developed systems to make it practical. This way, the problems mostly tackled by designers of mobile agent applications have been how mobility and intelligence could be exploited to achieve complex cooperative tasks (for example in the e-commerce, or in the information retrieval area) and which technological solutions are better suited to face issues that mobility makes more critical, such as reliable message delivery and security.

However, in order to fully exploit the potentiality of mobile agents, we argue that also structural models for the environment where agents move and interact are needed and have to be supported by appropriate infrastructures.

In this paper, we suggest to take more deeply into account the need of a supporting infrastructure, and we focus on two specific issues of agent mobility:

- Most of the resources available in the Internet may be *a-priori* unknown to agent systems, and they are likely to be subject to different access control policies. In this context, we argue that an infrastructure should be in charge of supporting agents in the process of acquiring knowledge about the environment and the resources they need to handle.
- The Internet is not merely a flat collection of nodes, but mostly a dynamically structured network of domains, belonging to different organisations and ruled in different ways. In this context, we argue that the agent motion should be governed by an infrastructure that models such a scenario.

With regard to the former issue, we believe that an infrastructure should help improving agent efficiency both by driving them along specific itineraries, and by providing them with the ability of acquiring knowledge about the structure of the network in a dynamic way.

In our opinion, the two issues are strictly related. In this paper, we argue that the same abstractions and mechanisms exploited by an infrastructure to make agents aware of the network topology and of the organisation structure can be used to control and govern their motion. Agents should have the possibility to move and interact only with those part of the infrastructure and with those resources strictly needed to complete their tasks. Access control techniques applied to agent interaction and mobility should be adopted for this purpose. This topic is discussed in Section 2.

Then, in Section 3, we describe an infrastructure for mobile agent applications in structured environments, which is taken as a case study. The proposed infrastructure, by exploiting the TuCSoN coordination model based on programmable tuple space [13], provides mobile agent applications with:

- *dynamic knowledge acquisition*, which let agents free from the charge of an a-priori knowledge of the environment;
- *dynamic exploration constraining*, which forces agents to follow only specifically authorised paths within an organisation's network;
- *uniform information-based interaction pattern*, for both agent-to-agent communication and resource access.

Section 4 concludes the paper and briefly discusses some related works.

2 Ruling Agent Motion

In this section, we assume virtual enterprises as a case study to discuss some problems related to agent mobility in the Internet. Furthermore, we show how an appropriate infrastructure should support agents in the dynamic acquisition of knowledge about the network structure and in their motion.

2.1 The Case of Virtual Enterprises

More and more, business activities cross the boundaries of single companies and encompass federations – possibly of a world-wide size – of independent partners, working together on common projects. These federations, of either short or long duration, require support for the electronic management of inter-organisational business processes. This is likely to be a complex task, due to the need of adapting the processes of single companies to the emerging ones of the federation, as well as to the inescapable heterogeneity of data and services that the overall electronic infrastructure of the virtual enterprise is likely to exhibit.

For these reasons, and due to the possibly world-wide area of the virtual enterprise, mobile agents are likely to be fruitfully exploited in this context. In fact, agents can move across the nodes of the virtual enterprise and effectively handle problems such as the dynamic adaptation of processes or the heterogeneity of data and services, by exploiting their embedded intelligence and the capability of locally accessing data and resources.

However, in this context, it is unrealistic to assume that agents have a complete knowledge of the whole infrastructure, of the enterprise's network organisation, as well as of its resources. In fact, virtual enterprises are intrinsically dynamic, may be highly distributed, and have no centralised authority, being composed by independent partners. Consequently, it is very complex and often not convenient to provide agents with the detailed knowledge about the structure of all the federation's nodes and resources each time a new federation is formed. Also, basic security reasons suggest that a company is not likely to disclose all its data and resources to the virtual enterprise applications, but only a sub-set of them (i.e., the ones required for the processes in the virtual enterprise to proceed correctly). Thus, agents must be able to deal with the fact that the access to several of the resources and/or to several of the nodes in the network might be denied.

From the virtual enterprise example, it results that a suitable infrastructure for mobile agents in the Internet must provide two facilities:

- *dynamic knowledge acquisition* – during the agent motion both the resource availability and the organisation topology should be considered, in general, *a-priori unknown* or only *partially known*. Therefore, agents should be supported by the organisation infrastructure in the acquisition of knowledge about resource availability and location in order to plan their actions and achieve their goals.
- *dynamic exploration constraining* – both the desired destinations of a mobile agent and the ones it is effectively permitted to reach within an organisation network should be considered *a-priori unpredictable*. Therefore, agents should not be allowed to freely roam an organisation sub-network. On the contrary, their motion and interaction should be ruled and constrained by the infrastructure according with access control policies.

We call *exploration* the interaction of an agent with an infrastructure that provides for the above two facilities, and *explorable topology* the resulting interaction space, whose knowledge an agent acquires dynamically, incrementally, and accordingly to access control policies, by querying and moving around the infrastructure.

2.2 Dynamic Knowledge Acquisition

In our everyday experience, if we want to search and retrieve some information from the Web, we rarely get it directly from the first page of a site. Since we might not know the exact location of the page we need, we usually exploit search engines and then navigate through a set of links until some interesting information is found.

When mobile agents are involved, the scenario is likely to be the same. For example, consider a mobile agent application in charge of retrieving information from Internet sites by using agents that roam between Internet sites and query information sources. Agents, in general, cannot retrieve the required information from an a-priori known Internet host, but they are forced to explore complex structures to reach the location where information is stored. For instance, when information about the product of a specific company is desired, the agent could start the exploration by migrating to the company's main location (correspondingly to the home page on the Web). There the support for agent applications should allow it to produce its query and reply with information useful to refine the exploration. For example, in the case of a virtual enterprise composed by a main company and several subsidiaries, the main company's site, when required to provide information about a product, could reply with the location of the subsidiary company responsible for the product itself. Then, the agent could migrate to the node of the subsidiary company and repeat its query. There, the subsidiary's node could provide the agent with the commercial office location, thus incrementally augmenting the agent knowledge about the organisation' topology with this additional permitted destination. This way, the agent is allowed to migrates within an infrastructure and the exploration goes on.

The point here is that information is seldom centralised and often its management is delegated to local authorities (e.g. each of the federated subsidiary). This is true for the Web as well as for virtual enterprises. Thus, mobile agent applications have necessarily to deal with these situations where information are incrementally acquired and organised within a complex network infrastructure.

Supporting agents in dynamically acquiring knowledge is also likely to reduce the number of exceptions that agents must handle. The exploration plan could be refined or re-built as needed at each step by exploiting the information gathered, so as to prevent agents to move to nodes that will deny them the access to their resources, thus forcing them to handle a number of exceptions.

In order to support the dynamic exploration of agents, the infrastructure has to model the structure of the network, to keep into account the locality relationship (either logical or physical) between the nodes of the network. This requires different roles to be assigned to the nodes of the network. The nodes where agents execute to access specific data and resources play the role of *places*. The nodes exploited by agents to acquire information about the network structure and resources play the role of *gateways*. Since a gateway provides information for a limited set of places (a single centralised repository is unfeasible in complex and large environments), we call *domain* the set of nodes composed by the gateway and the places for which it provides information.

Moreover, a hierarchical topology of gateways seems the most manageable model for organisational structures (see Figure 1). A hierarchical structure naturally maps into the typical physical structure of the Internet, where the nodes of an administrative domain are often protected by a firewall and possibly clustered in higher level structures.

It is worth noting that the concepts of gateway and place do not automatically imply the definition of a unique hierarchical structure: a place can be part of different domains (there can be more than one gateway to provide information about that node), and a gateway can be a place in its turn (it can have local resources, as well). This can be very useful to model complex and dynamic network topologies, as those deriving from virtual organisations. As a result, a virtual enterprise, formed by several federated organisations, could be modelled as a composition of tree-like infrastructures of gateways and domains.

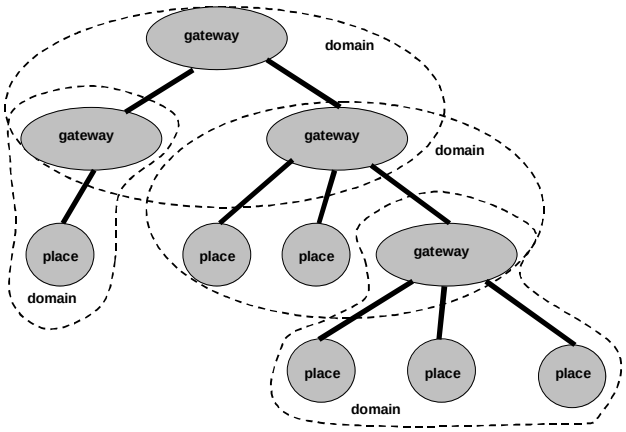

Figure 1. The supporting infrastructure

The abstractions introduced above are used both to model the physical topology of a company and to rule the motion of mobile agents [6]. Gateways are the key elements for the support of the agent dynamic acquisition of knowledge. They are 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). By allowing information about the network topology and the organisation to be acquired incrementally by need, whenever crossing gateways and entering new domains, the task of mobile agents result simplified, because the complexity of knowledge management is shifted from the agents to the infrastructure.

2.3 Dynamic Exploration Constraining

In the above subsection, we have described how an infrastructure can model the physical and logical topology of a network and help agents in the exploration phase.

From the system viewpoint, further considerations have to be done: an agent should not be allowed to move to those nodes which do not hold accessible resources. On the one hand, computational resources would be wasted if agents were free to reach nodes where no resources can be accessed. In this case, access control should be enforced by limiting the number of unauthorised attempts of accessing local resources. On the other hand, agents may be required to dynamically handle exceptions when they are denied access to a resource. Since exception handling negatively impact on

application efficiency, also from the agent viewpoint unauthorised accesses to resources should prevented.

For the aforementioned reasons, when agents have to interact with the gateway infrastructure to acquire knowledge, gateways could also be exploited to guide and constrain agent movements and their access to resources. Information provided to agents by gateways should depend from agents' identities and from privileges that an agent is enabled to acquire. In this acceptation, we elect knowledge about the network structure as a first-class information resource, to be managed accordingly to specific access control policies, as any other network resource. Only agents which are authorised to access to specific resources on a node will be given information about that node and about the resources allocated there.

However, the pre-condition for applying an access control technique is that entities whose access might be authorised (mobile agents, in our case) must be authenticated. As a result, gateways could be also exploited for the authentication of incoming agents on the behalf of all domains' places and sub-gateways. Places and sub-gateways of a given domain should reject each agent that was not previously authenticated by a gateway. By enforcing this policy, the agent motion can be controlled and limited within a hierarchical infrastructure. From the implementation viewpoint, this can be efficiently built upon current security frameworks and cryptographic tools [2].

In summary, the approach we adopted is to state that the agent motion must be governed following the basic principle of the *least privilege*, as usually done for the resource access control. This means that, if a place holds resources that an agent is not allowed to interact with, then also the migration of the agent to that place should be not authorised. The same consideration could be extended to a sub-tree of the whole structure: each sub-tree of the infrastructure could be explored by an agent only if it logically contains some resources needed by the agent to complete its task.

3 The TuCSoN Infrastructure

In this section we describe the infrastructure that we have defined to rule agent motion in a structured environment. The infrastructure is based on the TuCSoN coordination model, which enables the interaction between agents and resources to be uniformly managed in an information-oriented fashion [13].

3.1 The TuCSoN Coordination Model

The TuCSoN coordination model 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* [13]. 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 local 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 via either its full Internet (absolute) name or its local (relative) name. This supports the twofold role of agents as network-aware entities explicitly accessing to a remote tuple centre, 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 original Linda model [9], 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 performed by a given agent can be associated to specific computational activities, called *reactions* [7]. 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 has triggered. Thus, a new observable behaviour can be defined for a tuple centre, where global coordination laws can be embedded.

Due to their programmability, different tuple centres can exhibit different behaviours in response to the same access event. In addition, since an access event is also characterised by the identity of the agent that performs it, the same tuple centre can behave with different behaviours in response to the same access event performed by two different agents. This can be fruitfully exploited to make tuple centres act both as the media used by agent to communicate with each other (by integrating in the form of behaviour specification any needed communication protocol) and as the media used for accessing any resource provided by a node (by representing data in the form of tuples and services in the form of behaviour specification).

3.2 TuCSoN as the Engine for Agent Motion and Access Control

TuCSoN tuple centres can be exploited for ruling agent motion in a structured environment and to handle the access control policies. To support the agent motion, and by considering the agent/gateway interaction, we have mediated both knowledge acquisition and constraints on agents' movements via tuple centres. Differently from nodes holding resources, for which more application-oriented tuple centres could be defined, each gateway is supposed to implement a *default tuple centre*, which represents the standard communication media that all the agents know and are able to interact with (see Figure 2). Hence, when an agent arrives on a gateway, it can query the default tuple centre to get the structure of the domain, in order to be enabled to proceed in the exploration. The default tuple centre provides the agent with:

- information about those nodes that authorise the execution of the agent;
- information about the available resources located on the nodes, i.e. those tuple centres that the agent can access;
- information about the privileges granted by tuple centres to the agent.

This way, the agent is dynamically informed about the structure of the network, and bounded in the motion towards the resources it can actually access. Avoiding useless migrations improves both agents and supporting infrastructure efficiency. Even more, since a tuple centre can be programmed to behave differently to accesses performed by different agents, it is possible to provide different agents with different information about the structure of the domain, depending on the permissions these agents have.

Similarly, when an agent arrives on a place, it can access the resources via local tuple centres. Again, a suitable programming of the tuple centres permits to develop the required access control policies to protect local resources. This leads to a very uniform model, where inter-agent communication, agent's access to the resources on a node, and agent's access to the information about the network structure, all exploit the same tuple-based interaction model.

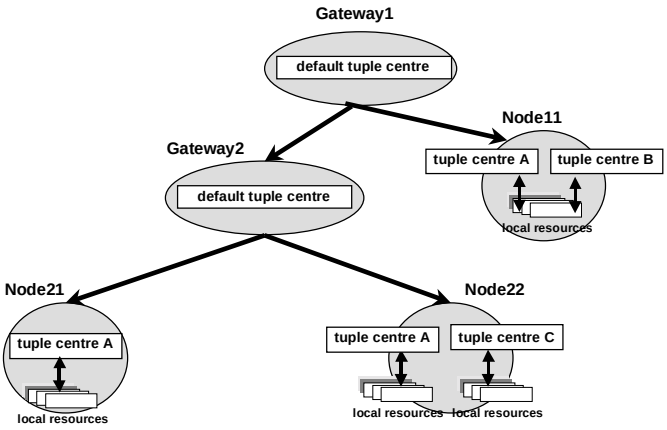

Figure 2. Tuple centres and hierarchical topology

Table 1 reports the typical pseudo-code that an agent executes while moving in a structured network ruled by the TuCSoN infrastructure.

Table 1. Pseudo-code for the agent exploration protocol in TuCSoN

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

`?read(subdomlist)`
`?read(placelist)`
`?read(commspace)`
	access the default tuple centre of the gateway and obtain information about accessible sub-domains (`subdomlist`*), places (*`placelist`*), and tuple centres (*`commspace`*).*

`<for pl in placelist do>` `<goto pl>`	*exploration of the accessible places of the domain* *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` *on* `Tuple`*, if authorised by* `pl`

4 Conclusions and Related Work

The novelty of this paper is to discuss how a homogeneous, well-balanced infrastructure for agent systems could be arranged to deal with the topological

structure of a dynamic and unpredictable environment, the coordination of interactions among autonomous, mobile agents, and the access control applied to the exploration of an infrastructure. These issues have been often recognised as being relevant in agent-based systems, but not yet fully addressed in their complexity and in their reciprocal dependencies by the agent researches. Hence, it has been described how the TuCSoN framework allows to support that infrastructure in a coherent and sound fashion.

Considering systems related with TuCSoN, *ActorSpace* [10] is one of the most relevant proposals in the area of agent-oriented infrastructures and provides an underlying platform for agent systems that enables to control the access to, and the management of resources. ActorSpace explicitly models the location of agents on particular hosts and the amount of computational resources that an agent is allowed to consume. Resource interaction are mediated by means of proxies, which are components of the agent's behaviour specifically customised to interact with a certain type of resource. Differently from ActorSpace, in TuCSoN agents interact in a standard way with resources and have no need to embed components (i.e. proxies) specifically tailored for the different resource types. This better management of the heterogeneity of the interaction space is one of the main benefits derived from the adoption of a tuple-based coordination model. Moreover, issues derived from the structure of the environment are not explicitly addressed in ActorSpace.

LIME [14] proposes an interesting way for coordinating mobile agents and hosts (possibly mobile, too). In LIME, each mobile agent carries a tuple space. Then, when two or more agents are co-located in a host, their tuple spaces are dynamically recomputed in such a way that, for each mobile agent, the content of a tuple space is logically merged with those of the other agents and the one of the host. The resulting global tuple space is shared among all the agents and the host. Differently from TuCSoN, LIME principally investigates problems related to inter-agent communication, while aspects concerning the need of an infrastructure seems quite ignored. Consequently, also problems concerning access control and topology are not considered.

Secure Spaces [15] is a relevant proposal in the area of coordination models. It modifies Linda in order to be able to offer security guarantees to mutually untrusting programs. Some locking mechanisms to protect data stored in the tuple set have been developed by means of cryptographic techniques. If compared with TuCSoN's goal of controlling the agent interaction with an organisational infrastructure, it addresses the issue of controlling the agent access to available tuples in a shared blackboard.

Ambit [4] is a formal model that recognises the hierarchical structure of the Internet and explicitly models the migration of active entities across protected domains. It is one of the few works that, like TuCSoN, considers together the structure of a network and security problems in an agent-based scenario. Differently from TuCSoN, it is focused on the formal representation of the model, while a real infrastructure defined in terms of effective technological solutions is not provided.

Further work will be devoted to the identification of suitable high-level design patterns for mobile agent [1] and to the integration with the Jini technology [16], whose JavaSpaces coordination model is suitable to be enhanced according to the TuCSoN's model.

References

1. Aridor, Y., Lange, D.B., Agent Design Patterns: Elements of Agent Application Design, in Proceedings of Autonomous Agents '98, ACM Press, 1998.
2. Arsenault, A. and Turner, S., Internet X.509 Public Key Infrastructure PKIX Roadmap. IETF Internet Draft, PKIX Working Group, March 1999.
3. Cabri, G., Leonardi, L., Zambonelli, F., Coordination Models for Internet Applications based on Mobile Agents, IEEE Computer, Feb. 2000.
4. Cardelli, L., Gordon, A., Mobile Ambients, Foundations of Software Science and Computational Structures, LNCS no. 1478, Springer-Verlag, 1998.
5. Chess, D., Harrison, C., Kershenbaum, A., Mobile Agents: Are They a Good Idea?, RC 19887, IBM Research Division, 1994.
6. Cremonini, M., Omicini, A., Zambonelli, F., Multi-agent systems on the Internet: Extending the scope of coordination towards security and topology. In F.J. Garijo and M.Boman, (ed.), Multi-Agent Systems Engineering - Proc. of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAMAAW'99), LNAI no. 1647, pages 77-88, Springer-Verlag, 1999.
7. Denti, E., Natali, A., Omicini, A., On the expressive power of a language for programming coordination media. In Proceedings of the 1998 ACM Symposium on Applied Computing (SAC'98), pages 169-177, Atlanta (GA), 1998.
8. Fuggetta, G., Picco, G.P., Vigna, G., Understanding Code Mobility, IEEE Transactions on Software Engineering, May 1998.
9. Gelernter, D., Carriero, N., Coordination languages and their significance, Communications of the ACM, 35(2) 97-107, Feb. 1992.
10. 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, 38-44, March-April 1999.
11. Jennings, N. R., Sycara, K., and Wooldridge, M., A Roadmap of Agent Research and Development International Journal of Autonomous Agents and Multi-Agent Systems 1(1) 7-38, 1998.
12. Karnik, N. M., Tripathi, A. R., Design Issues in Mobile-Agent Programming Systems, IEEE Concurrency, 6(3), July-Sept. 1998, pp. 52-61.
13. Omicini, A., Zambonelli, F., Coordination for Internet Application Development, Journal of Autonomous Agents and Multi-Agent Systems, 2(3), Sept. 1999.
14. Picco, G.P., Murphy, A., Roman, G.C., LIME: Linda Meets Mobility. In Proc. of the 21st International Conference on Software Engineering (ICSE'99), Los Angeles, California, May 1999.
15. Vitek, J., Bryce, C., Oriol, M., Coordinating Agents with Secure Spaces, In Proceedings of Coordination '99, Amsterdam, The Nederlands, May 1999.
16. Waldo, J., The Jini Architecture for Network-centric Computing. Communications of the ACM, 42(7), July 1999.