Multi-Agent Systems on the Internet: Extending the Scope of Coordination towards Security and Topology

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

[1] LIA - DEIS - Università di Bologna
Viale Risorgimento, 2 40136 – Bologna, Italy
Ph.: +39 051 6443087- Fax: +39 051 6443073
{mcremonini, aomicini}@deis.unibo.it
[2] DSI - Università di Modena
Via Campi 213b 41100 – Modena, Italy
Ph.: +39 059 376735- Fax: +39 059 376799
franco.zambonelli@unimo.it

Abstract. The Internet is rapidly becoming the privileged environment for today's Multi-Agent Systems. This introduces new issues in MAS' design and development, from both a conceptual and a technological viewpoint. In particular, the dichotomy between the openness of the execution environment and the need for secure execution models makes governing agents' interaction a really complex matter, especially when *mobile agents* are involved. If *coordination* is managing the interaction, then the issue of agent coordination is strictly related with the issues of *topology* (how the space where agents live and possibly move is modelled and represented), *authentication* (how agents are identified), and *authorisation* (what agents are allowed to do). To this end, we first discuss the TuCSoN model for the coordination of Internet agents, then show how it can be extended to model the space where agents live and interact as a hierarchical collection of locality domains, where programmable coordination media are exploited to rule agent interaction and to support intelligent agent exploration. This makes TuCSoN result in a single coherent framework for the design and development of Internet-based MAS, which takes coordination as the basis for dealing with network topology, authentication and authorisation in a uniform way.

Keywords: Coordination, Multi-Agent Systems, Internet Agents, Agent Mobility, Security

1 Introduction

Multi-Agent Systems (MAS henceforth) are spreading all over the Internet: more and more, multi-agent metaphors and technologies are exploited to build complex Internet applications. In this context, we see the Internet as a multiplicity of agent execution environments, heterogeneous and physically distributed. Since

these environments are typically under decentralised control and often highly dynamic, the Internet is usually an unpredictable space, unlikely to be completely known *a-priori* by agents.

As an agent space, the Internet is then the environment where agents live and possibly move, and where they interact with other agents and with resources, too. *Coordination* generally deals with managing the interaction among components [18]: so, in the context of MAS, it addresses the issue of how agents interact. Like agent architectures and languages support in designing, engineering and experimenting agents [23], *coordination languages* and *models* [11] are meant to provide for the abstractions and tools required to get agents together and build a multi-agent system [3].

However, coordination models alone cannot face the complexity intrinsically related with Internet-based MAS. First, the dichotomy between the need for open application environments and the ever-growing requirement of safe and reliable execution models makes coordination and security two dual and strictly connected topics. Then, agents exploring the heterogeneous and unpredictable space of the Internet put network topology and coordination in relation. More precisely, when defining an effective framework for the design and development of Internet-based MAS, the issue of coordination turns out to be in strict relation with the issues of *(i)* how agents are identified (*authentication*), *(ii)* what they are allowed to do (*authorisation*), and *(iii)* how the space where agents live and possibly move is modelled (*topology)*.

From this perspective, this paper discusses and extends the TuCSoN model for the coordination of Internet agents [20]. Here, coordination is exploited as the conceptual basis where all the abstractions and mechanisms for modelling the Internet and managing security are rooted, providing for a uniform approach for the authentication, authorisation and topology issues.

Interaction within Internet nodes is ruled by the TuCSoN *tuple centres* [7] working as the core of the whole framework. From a coordination viewpoint, tuple centres are *programmable coordination media* [6] whose behaviour can be defined in order to rule component interaction. From a security viewpoint, tuple centres mediate between agents and resources, so that they can be exploited to restrict interaction and implement access control policies. From a topology viewpoint, tuple centres are exploited as knowledge repositories for agents, containing information about network topology, which is modelled by means of the *place*, *domain*, and *gateway* abstractions.

2 Coordination in Context

Coordination is concerned with managing the interaction among software components [11]. Coordination models and languages [3] aim at providing the abstractions and mechanisms which are most suitable for the effective design and development of multi-component software systems, like MAS, where active components (e.g., agents) communicate, synchronise, cooperate and compete within an execution framework.

Choosing the Internet as the environment for MAS extends the scope of agent coordination beyond the conceptual boundaries set up by the current research on coordination [5, 21]. Since agents live and possibly move through a collection of heterogeneous and physically distributed execution environments, and there interact with both protected resources and other agents, both *topology* and *security* strictly relate to coordination, as discussed in the following subsections.

2.1 Coordination and Topology

Two main problems have to be addressed in the coordination of network-aware agents: *(i)* how the space where agents live is modelled (*network modelling*), and *(ii)* how the knowledge about the structure of that space is made available to the agents (*network knowledge*).

The problem of network modelling is particularly evident when dealing with intrinsically structured domains, as Internet environments frequently are. In fact, Internet nodes are often grouped in clusters, subject to highly coordinated management policies and possibly protected by a firewall. Moreover, 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, where a single large cluster enclose all the academic nodes and defines basic management policies. Different enclosed clusters, such as the one of the single research laboratories, provide protected domains with their own policies, typically under the administration of a single system manager.

As far as network knowledge is concerned, agents may either have an *a-priori* knowledge of the space they live in, or acquire it dynamically by need. The former choice would not affect the agent interaction protocol, which could simply exploit the topological information that agents are statically provided with. However, this approach is unsatisfactory, since it makes difficult to deal with the typical dynamicity and unpredictability of Internet-based domains. With the latter approach, instead, an agent first acquires information about the structure and properties of the agent space, then interacts and possibly moves through that space according to the knowledge gained. This actually affects the coordination protocol, since part of the agent interaction concerns the acquisition of information about topology, and makes network knowledge a coordination-related issue.

2.2 Coordination and Security

While the Internet intrinsically promotes the openness of the application environments, its exploitation in domains where safe and reliable execution models are required is growing quite fast. Roughly speaking, while coordination technology for the Internet is typically concerned with enabling interaction and making it fruitful, security technology is typically meant to bound interaction so as to make it harmless. This makes coordination and security two strictly related topics, even though somehow dual. More precisely, while coordination deals with

how agents interact, security (according to the interpretation of the term adopted in this paper, which includes access control models) deals with *(i)* how agent are identified (*authentication*), and *(ii)* what they are allowed to do (*authorisation*).

The absence of a centralised control for both the MAS and the Internet nodes raises the issue of authentication: before accepting an incoming agent, or a query from a remote agent, a node should verify its identity. So, the first interaction between an agent and an execution environment always concerns agent authentication. A coordination framework should then endorse a global agent naming scheme and support an identification protocol, univocally mapping agents onto names. Coherently, a coordination model should embody a suitable notion of agent identity, by allowing any communication operation to be associated to the proper agent identifier. Even more, a broader notion of identity should be supported, which enables a MAS to be denoted as a whole, and single agents to be characterised by both their MAS' identity and their *role* in the MAS.

Authorisation is what more directly concerns interaction, and may somehow be seen as the conceptual security counterpart of coordination. While a coordination language may allow an agent to perform a communication primitive, an authorisation policy may instead limit the agent interaction protocol, by forbidding, for instance, its access to some resources, or by preventing some communication operations to be performed. According to that, a coordination framework should enable authorisation policies to be defined over the coordination language, so as to make it possible to achieve the best compromise between expressive power and safety in the interaction.

Finally, in a distributed environment like the Internet, security policies may be either completely decentralised (every node defines and implements its own authentication and authorisation patterns), or partially/totally structured according to the network model, by exploiting some form of (either implicit or explicit) delegation. The latter choice, by relating the network structure to both authentication and authorisation, relates security to topology, too.

3 Coordination of Internet Agents in TuCSoN

3.1 Communication in TuCSoN

TuCSoN is a model for the coordination of Internet agents [20], where agents interact through a multiplicity of independent coordination media, called *tuple centres*, spread over Internet nodes. Agents exchange tuples via tuple centres by means of a small set of communication primitives (`out`, `in`, `rd`, `inp`, `rdp`) having basically the same semantics of Linda ones. In short, `out` puts a tuple in the tuple centre, while the *query primitives* (`in`, `rd`, `inp`, `rdp`) send a tuple template (in their *pre phase*) and expect a matching tuple back from the tuple centre (in their *post phase*). More in detail, `in` and `inp` delete the matching tuple from the tuple centre, while `rd` and `rdp` leave it there; `in` and `rd` wait until a suited tuple becomes available, while `inp` and `rdp` fail if no such a tuple is found.

Each tuple centre is associated to a node and has a unique name: in particular, a tuple centre can be denoted either by its full Internet (*absolute*) name or by its

local (*relative*) name. More precisely, tuple centre *tc* provided by the Internet node *node* can be referred to by means of its absolute name *tc@node* from everywhere in the Internet, and by means of its relative name *tc* in the context of node *node*. Correspondingly, the TuCSoN interaction space can be seen either as a *global interaction space*, featuring a collection of uniquely denoted tuple centres (when considering absolute names), or as a collection of *local interaction spaces*, replicating the name space on each node (when considering relative names).

The general form for any admissible TuCSoN communication operation performed by an agent is *tc?op(tuple)* asking tuple centre *tc* to perform operation *op* using *tuple*. Since *tc* can be either an absolute or a relative tuple centre's name, agents can adopt two different forms of primitive invocation, according to their contingent needs: the *network* and the *local* one, respectively. The network communication form *tc@node?op(tuple)* is used by agents when behaving as network-aware entities, by denoting tuple centres by their absolute names in the global TuCSoN interaction space. In its turn, the local communication form *tc?op(tuple)*, which refers by definition to the local tuple centre's implementation of the current execution node of an agent, is used by agents when behaving as local components of their current hosting environment. As shown in [20], the availability of both forms particularly suits mobile agents.

3.2 TuCSoN Tuple Centres

TuCSoN tuple centres are tuple spaces enhanced with the notion of *behaviour specification*: the behaviour in response to communication events of every tuple centre can be defined according to the system requirements. The interaction space provided by a TuCSoN node relies on a multiplicity of tuple centres: so, TuCSoN shares the advantages of models based on multiple tuple spaces and goes beyond, since different coordination media can encapsulate different coordination laws, providing system designers with a finer granularity for the implementation of global coordination policies.

The behaviour of a stateful coordination media like a tuple centre is naturally defined as the observable state transition following a communication event. Correspondingly, defining a new behaviour for a tuple centre basically amounts to specifying a new state transition in response to a standard communication event. This is achieved by allowing any admissible TuCSoN communication event to be associated to specific computations, called *reactions*. In particular, a *specification tuple* `reaction(Op,R)` associates the event generated by an incoming communication operation *Op* to the reaction *R*.

A reaction is defined as a sequence of *reaction goals*, which may access the properties of the communication event triggering the reaction, perform simple term operations, and manipulate tuples in the tuple centre. In particular, operations on the tuple space (`out_r`, `in_r`, `rd_r`, `no_r`) work similarly to communication operations, and can trigger further reactions in a chain. Reaction goals are executed sequentially, each with a success/failure semantics, and a reaction as a whole is either a *successful* or a *failed* one depending on whether *all* its reaction goals succeed or not. Each reaction is executed with a transactional semantics:

a successful reaction can atomically modify the tuple centre state, a failed reaction yields no result at all. Moreover, all the reactions triggered by a given communication event are executed before taking into account serving any other communication event, so as to ensure that agents perceive only the final result of the execution of the communication event *and* the set of all the triggered reactions. (More details on the reaction specification language and execution model can be found in [7].)

As a consequence, the result of the invocation of a communication primitive from the agent's viewpoint is the sum of the effects of the primitive itself and of all the reactions it triggered, perceived altogether as a single-step transition of the tuple centre state. This makes it possible to uncouple the agent's view of the tuple centre (which is perceived as a standard tuple space) from the tuple centre actual state, and to connect them so as to embed the laws of coordination. Since the reaction specification language is Turing-equivalent [7], any computable coordination law can be encapsulated into the coordination medium. In particular, as shown in [20], this enable TuCSoN to address some of the typical issues in the design and development of Internet-based MAS, such as the heterogeneity of the agent execution environments, the support of incremental system development, and the definition and enforcement of security policies.

Since each TuCSoN tuple centre stores both its data (ordinary tuples) and its behaviour specifications (specification tuples) locally, coherently with the distributed nature of component-based systems, coordination intelligence can be spread where needed all over the agent space. Furthermore, thanks to their uniform representation, agents can manipulate data tuples and behaviour specification tuples adopting the same conceptual protocol. So, in principle, intelligent agents may modify/integrate the behaviour of a TuCSoN MAS in the same straightforward way as they communicate with other agents, that is, by adding, removing, and reading (specification) tuples.

3.3 Security in TuCSoN

The TuCSoN security model defines neither a naming scheme nor an authentication mechanism, instead it simply assumes that it is somehow possible to authenticate an agent and to denote it with a suitable identifier. Given that, TuCSoN provides a simple yet expressive authorisation scheme, based on the definition of an access control model over tuples. Each tuple centre may be made either visible or invisible, and its tuples (both ordinary and specification ones) readable/writable/removable, to any (class of) agent(s). Both tuples and specification tuples (that is, those defining reactions) are marked with the owner identity, that is, the identifier of the agent who put the tuple in the tuple centre. Then, *(i)* any communication operation on a tuple centre performed by an agent may either succeed or fail also according to the rights granted to the agent itself, and *(ii)* any reaction associated to the operation may either succeed or fail also according to the rights granted to the agent owning the corresponding specification tuple.

Thus, the effect of a communication primitive is defined by the joint effects of both the coordination and authorisation policies: for instance, the effect of an `out` operation performed by agent `a` on tuple centre `tc` depends on *(i)* the default operation's semantics, *(ii)* the reactions possibly associated to it by the `tc`'s behaviour specification, and *(iii)* the access rights of `a` with respect to `tc`.

4 Extending TuCSoN

4.1 Concepts and Terminology

In order to enable it to effectively deal with the security and topology issues, we extend TuCSoN with new *locality abstractions* which enable it to model the Internet as a hierarchical collection of *places*, *domains*, and *gateways*, as well as with a notion of *agent identity* rooted in standard Internet security mechanisms.

The *place* provides the abstraction for the agent execution environment. The *domain* concept is used to group places sharing common policies and privileges. Each domain is associated to a *gateway*, which is the place in charge of agent authentication on behalf of domain's places, as well as of inter-domain routing for both incoming and outgoing agents. Domains can be nested, so that a place of a domain can be the gateway for a sub-domain. As a result, the collection of the Internet nodes hosting a TuCSoN MAS can be represented with a hierarchical structure: a TuCSoN *tree*. Places are the leaves of a TuCSoN tree, and gateways are the nodes whose children are the associated domain's places, so each sub-domain is represented by a sub-tree. The tree root is the most external gateway, which works as a bridge with the Internet and possibly connects different TuCSoN domains.

All parties associated with a TuCSoN MAS are known via *principals* [15], that is, identities (unique names) that can be authenticated. Thus, agents and places are identified via the *Developer*, *User*, and *Node* principals, exploiting X.509 certificates issued by a certified authority of a PKIX infrastructure: the *Developer* represents the responsible for the agent class, the *User* is the responsible of the agent instantiation, and the *Node* is the responsible for the agent execution environments. Attribute certificates [9] enable both the definition of a global notion of MAS' identity, and the association of a *role* to each agent, representing its specific task within the MAS. So, authentication in TuCSoN associates to each agent both the global identity of the agent's MAS and the specific agent's role within the MAS. This notion of agent identity can then be exploited for agent authorisation, that is, to provide the agent with the *permissions* needed to access and modify tuples in the tuple centre.

4.2 Understanding TuCSoN

In order to help understanding the extended TuCSoN model and its features, we presents a simple example of a TuCSoN MAS – the `bookreader` MAS – where some of the agents are mobile and roam the Internet nodes of a University

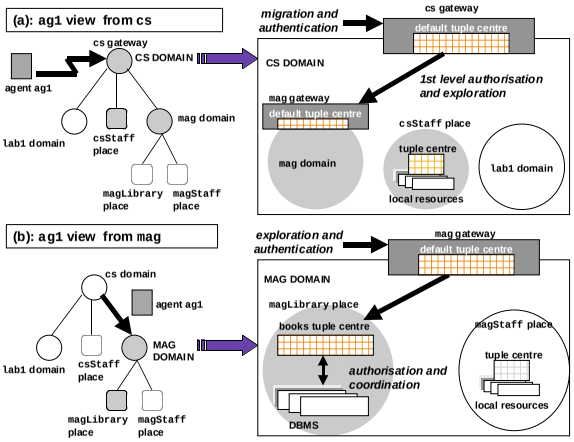

Fig. 1. An example of a TuCSoN framework.

looking for book references. The University has one central library and several departmental libraries, along with a bunch of book collections owned by single research groups. Actually, this represents a typical application context suitable to be addressed by multi-agent technology, as well as an ideal scenario for our framework, due to *(i)* the hierarchical organisation of the structure, *(ii)* the decentralisation of control, *(iii)* the need for global policies, *(iv)* the cooperation required to each domain, and *(v)* the intrinsic heterogeneity of resources.

The topology aspect is obviously addressed by exploiting the locality abstractions introduced above so as to model the Internet agent space. As an example, Fig. 1 depicts a University department network modelled as a TuCSoN tree: the root is the Computer Science department gateway (`cs`), associated to the `cs` domain, which contains as its children nodes places like `csStaff` and gateways for both laboratories (such as `lab1`) and research groups, such as the Multi-Agent Group one (`mag`). In its turn, the `mag` sub-domain contains the `magLibrary` and `magStaff` places. In particular, as a TuCSoN node, `magLibrary` provides for the `books` tuple centre, representing the catalogued books of the Multi-Agent Group. Mobile agents of the application `bookreader`, in charge of finding book references, will then explore all the `books` tuple centres available in the `cs` domain and in its sub-domains, including the one provided by the `magLibrary` place.

In TuCSoN, the role of gateways is fundamental to support and control the execution of the agents moving along the network or remotely accessing places. Each gateway can authenticate agents on behalf of its associated domain, by verifying agents' identity and by propagating it by default to all domain's places and sub-gateways. More precisely, agent authentication must be performed by each gateway receiving an agent request from an external source, as usual in Internet applications with security requirements. Since the information about the agent identity is propagated to the sub-tree, 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. Even though any gateway is left free to fully authenticate an agent autonomously, wherever the agent comes from, authentication delegation between gateways permits to gain in performance and to reduce the complexity of security requirements for mobile agent applications. For instance, in Fig. 1, once authenticated by the `cs` gateway, the `bookreader` agent `ag1` recognises the presence of an inner gateway, the `mag` one, which it may be interested to explore. Then, as `ag1` comes to `mag`, it needs not to be re-authenticated, but is instead recognised as a `bookreader` agent coming from `cs`, since `mag` trusts on `cs` authentication.

Besides acting as centralised points for authentication, TuCSoN gateways work as *knowledge repositories*, providing agents with a multi-layered description of the agent space. For this purpose, the default tuple centre of any gateway (the one accessed when no tuple centre identifier is specified in the invocation of a communication primitive) contains knowledge about its associated domain, and is programmed to make that information available based on agent's identity and role. Thus, an agent can dynamically retrieve from a gateway the set of the accessible places and sub-domains in the domain, as well as the set of the tuple centres made visible to it by each place of the domain. In this context, TuCSoN tuple centres work not only as a mere coordination media, but are exploited also as authorisation engines, given that authorisation policies can be charged upon both the behaviour specification and the access control model of a tuple centre. Moreover, this supports intelligent agent exploration, since agents are not supposed to have a complete, *a-priori* knowledge of the agent space, but can instead gain information about the agent space incrementally, by need.

For example, according to Fig. 1(a), agent `ag1` accesses the default tuple centre of the `cs` gateway, and discovers what places are in the domain and which tuple centres are accessible to it. By providing the corresponding knowledge in form of tuples (see also Fig. 2), `cs` authorises `ag1` to access the `mag` gateway and the `csStaff` place (grey in figure) but forbids it to access to the `lab1` domain (white in figure), whose existence is hidden to `ag1`. When `ag1` moves to the `mag` gateway, as shown in Fig. 1(b), it can access the `magLibrary` place only, because the existence of `magStaff` is kept hidden to it by the behaviour specification of the default tuple centre of `mag`. As a result, both authorisation and dynamic acquisition of knowledge about the agent space are achieved by gateways in a uniform way by exploiting the coordination media, *i.e.*, the tuple centres.

4.3 Impact on Agent Design

As shown in [8], the adoption of tuple centres as the coordination media has a positive impact on the way in which agents of a MAS are designed. Since the behaviour of a tuple centre can be programmed so as to embody any computable coordination law, agents can be freed of the load of coordination, and designed around very straightforward interaction protocols, where single agent computation is kept cleanly separate from multi-agent coordination.

– agent exploration –	
`<goto d>`	migration to gateway `d`
`<identify>`	gateway `d` authenticates the agent on behalf of all the places of its associated domain
`?rd(subdomlist)`	access the default tuple centre of gateway `d` to obtain information about domain structure, in terms of accessible subdomains (`subdomlist`), places (`placelist`), and tuple centres (`commspace`), filtered according to agent's identity and role
`?rd(placelist)`	
`?rd(commspace)`	
`<for pl in placelist do>`	exploration of the accessible places of the domain `d`
`<goto pl>`	migration to place `pl`
– agent interaction –	
`<for tc in commspace do>`	for any visible tuple centre `tc` of place `pl`
`tc?op(tuple)`	ask tuple centre `tc` of place `pl` to execute `op(tuple)`
– agent exploration (sub-domains) –	
`<for sd in subdomlist do>`	exploration of the accessible subdomains
`<goto sd>`	migration to gateway `sd`
`<...>`	keep on exploration and access, in a recursive fashion

Fig. 2. A possible interaction protocol for a mobile agent in TuCSoN.

In addition, extending the scope of the TuCSoN coordination model towards topology and security has further advantages, as pointed out by Fig. 2, which shows the scheme of a possible exploration protocol of an agent roaming an environment modelled as a TuCSoN tree. On the one hand, the example emphasises the clear distinction between the exploration of the agent space (*agent exploration* in Fig. 2) and the management of the agent interaction within the execution environment (*agent interaction* in Fig. 2). On the other hand, it shows how the same interaction protocol is adopted for the definition of both authorisation and coordination policies, which are mediated by the programmable coordination media: this makes it possible to find the right balance between the dual issues of security and coordination.

Finally, agents are allowed to know in advance which operations they are allowed to perform within TuCSoN nodes. This makes it possible to design intelligent agents which are able to *(i)* plan their course of actions so as to minimise the cost of exploration and access to resources, and *(ii)* reduce as much as possible the load of handling exceptions and access denials.

5 Related Works

As MAS become more and more a profitable solution in the Internet arena, new issues and challenges are raised, from both a conceptual and a technological viewpoint. In this paper, we presented an extension to the TuCSoN model for the coordination of Internet agents, which takes coordination as a basis for addressing both the topology and security issues in a uniform framework.

Despite the great deal of activity in both the coordination and security areas, their integration is still a neglected issue, as well as the definition of a comprehensive model meant to support system design. In particular, the mobile agent research field still lacks in-deep proposals addressing the issues of agent motion in a structured environment and agent interaction with resources. Most proposals in the area of intelligent agents address the issue of inter-agent interactions

for knowledge exchange, without paying attention to architecture and security issues [10, 12].

The notion of programmable coordination medium [6], exploited in TuCSoN to deal with security and topology in a uniform way, has been already applied in the $\mathcal{ACLT}$ system [8] in the field of heterogeneous multi-agent systems, by MARS [1] in the context of mobile agents, and by Law-Governed Linda [19] to address security and efficiency issues in distributed environments. PageSpace [4] defines a general architecture for the coordination of Internet agents, where special-purpose agents are provided to influence the coordination activity of the applications agents. The ActorSpace model [13] is a comprehensive framework for building agent ensembles which addresses the coordination issue explicitly, but does not support agent exploration. In T Spaces [17], agents can add any new primitive to the tuple space, in order to implement any needed transaction on the stored data. However, none of the above coordination systems addresses in an integrated way the issues of authentication and authorisation in the access to the coordination media, which is instead one of the TuCSoN's key-features.

In the security area, well-known systems as Aglets and D'Agents [14] focused on the immediate usability of their tools, possibly supporting traditional cryptographic mechanisms and basic forms of access control lists (ACLs). In addition, a number of proposals are emerging, which address agent security with new interesting mechanisms, such as mobile cryptography [22]. However, the above systems lack a likewise attention to the definition of both appropriate models for the network topology and suitable coordination models.

To the best of our knowledge, the only proposals which address the issue of the hierarchical structure of many Internet application domains and explicitly model the migration of active entities across protected domains are Ambit [2] and Discovery [16]. However, both the above models fall short in supporting agent exploration and in providing for suitable coordination abstractions.

Acknowledgements

This work has been carried out under the financial support of the MURST (the Italian "Ministero dell'Università e della Ricerca Scientifica e Tecnologica") in the framework of the Project MOSAICO "Design Methodologies and Tools of High Performance Systems for Distributed Applications".

References

[1] G. Cabri, L. Leonardi, and F. Zambonelli. Reactive tuple spaces for mobile agent coordination. In *Proceedings of the 2nd Workshop on Mobile Agents*, volume 1477 of *LNCS*. Springer-Verlag, September 1998.

[2] L. Cardelli and A. D. Gordon. Mobile ambient, 1997. `http://www.research.digital.com/SRC/personal/Luca_Cardelli/Ambit`.

[3] P. Ciancarini. Coordination models and languages as software integrators. *ACM Computing Surveys*, 28(2), June 1996.

[4] P. Ciancarini, R. Tolksdorf, F. Vitali, D.Rossi, and A. Knoche. Coordinating multiagent applications on the WWW: A reference architecture. *IEEE Transactions on Software Engineering*, 24(5):362–375, 1998.

[5] *Coordination Languages and Models*, volume 1594 of *LNCS*. Springer-Verlag, 1999. 3rd Intl. Conf., COORDINATION'99, Amsterdam, The Nederlands, April 1999.

[6] E. Denti, A. Natali, and A. Omicini. Programmable coordination media. In *Coordination Languages and Models*, volume 1282 of *LNCS*, pages 274–288. Springer-Verlag, 1997.

[7] E. Denti, A. Natali, and A. Omicini. On the expressive power of a language for programming coordination media. In *Proceedings of SAC'98*, Atlanta, USA, 1998.

[8] E. Denti and A. Omicini. Designing multi-agent systems around an extensible communication abstraction. In *Proceedings of the 4th ModelAge Workshop on Formal Models of Agents*, LNAI. Springer-Verlag, 1999.

[9] S. Farrell. An Internet Attribute Certificate Profile for authorisation, August 1998. Internet Draft.

[10] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In *Proc. of the Third International Conference on Information and Knowledge Management*, Gaithersburg, Maryland, November 1994.

[11] D. Gelernter and N. Carriero. Coordination languages and their significance. *Communications of the ACM*, 35(2):97–107, February 1992.

[12] M.R. Genesereth and R.E. Filkes. Knowledge Interchange Format: Version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[13] N. Jamali, P. Thati, and G.A. Agha. An actor-based architecture for customising and controlling agent ensembles. *IEEE Intelligent Systems – Special Issue on Intelligent Agents*, 1999. To appear.

[14] N.M. Karnik and A.R. Tripathi. Design issues in mobile-agent programming systems. *IEEE Concurrency*, 6(3):52–61, 1998.

[15] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

[16] S. Lazar, I. Weerakoon, and D. Sidhu. A scalable location tracking and message delivery scheme for mobile agents. In *Proc. of the IEEE WETICE'98*, June 1998.

[17] T.J. Lehman, S. McLaughry, and P. Wyckoff. T Spaces: The next wave. `http://www.almaden.ibm.com/TSpaces/`.

[18] T. Malone and K. Crowstone. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[19] N. Minsky and J. Leichter. Law-governed Linda as a coordination model. In *Object-Based Models and Languages*, volume 924 of *LNCS*, pages 125–145. Springer-Verlag, 1994.

[20] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 1999. Special Issue on Coordination Mechanisms and Patterns for Web Agents.

[21] *Proceedings of the 1999 ACM Symposium on Applied Computing (SAC '99)*, The Menger, San Antonio, Texas, February 28 - March 2 1999. ACM. Track on Coordination Models, Languages and Applications.

[22] T. Sander and C.F. Tschudin. Towards mobile cryptography. In *IEEE Symposium on Security and Privacy*, May 1998.

[23] M. Woolridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.