Autonomous Agents and Multi-Agent Systems, 7, 29–48, 2003

The RETSINA MAS Infrastructure

KATIA SYCARA katia@cs.cmu.edu
Carnegie Mellon University, Robotics Institute, 5000 Forbes Ave, Pittsburgh, PA 15232, USA

MASSIMO PAOLUCCI paolucci@cs.cmu.edu
Carnegie Mellon University, Robotics Institute, 5000 Forbes Ave, Pittsburgh, PA 15232, USA

MARTIN VAN VELSEN vvelsen+@cs.cmu.edu
Carnegie Mellon University, Robotics Institute, 5000 Forbes Ave, Pittsburgh, PA 15232, USA

JOSEPH GIAMPAPA garof@cs.cmu.edu
Carnegie Mellon University, Robotics Institute, 5000 Forbes Ave, Pittsburgh, PA 15232, USA

Abstract. RETSINA is an implemented Multi-Agent System infrastructure that has been developed for several years and applied in many domains ranging from financial portfolio management to logistic planning. In this paper, we distill from our experience in developing MASs to clearly define a generic MAS infrastructure as the domain independent and reusable substratum that supports the agents' social interactions. In addition, we show that the MAS infrastructure imposes requirements on an individual agent if the agent is to be a member of a MAS and take advantage of various components of the MAS infrastructure. Although agents are expected to enter a MAS and seamlessly and effortlessly interact with the agents in the MAS infrastructure, the current state of the art demands agents to be programmed with the knowledge of what infrastructure they will utilize, and what are various fall-back and recovery mechanisms that the infrastructure provides. By providing an abstract MAS infrastructure model and a concrete implemented instance of the model, RETSINA, we contribute towards the development of principles and practice to make the MAS infrastructure "invisible" and ubiquitous to the interacting agents.

Keywords: MAS, Multi-Agent System, infrastructure, agent, architecture.

1. Introduction

Multi Agent Systems are becoming increasingly important: as a scientific discipline, as a software engineering paradigm, and as a commercially viable and innovative technology. Despite the considerable research that has gone into the formation of theories, scientific principles and guidelines for MAS, there is relatively little experience with the building, fielding and routine use of MASs. It is admittedly the case that the development of a MAS is extremely challenging, both in the laboratory but especially in the real world. However, MAS research will not fulfill its potential until we have a critical mass of fielded systems, components, and services. To achieve this goal, a stable, widely used, widely accessible and extensible MAS infrastructure is crucial. Various standards bodies (e.g. FIPA) are attempting to define standards for various aspects of MAS infrastructure, such as Agent Communications Languages. In addition, industrial organizations (e.g. SUN) are developing and making accessible software that could constitute a part of a MAS infrastructure, such as JINI for service discovery. Various labs and companies are developing agent toolkits that could be reused for building agents and multiagent systems (see section 5).

However, there is no coherent account of what constitutes a MAS infrastructure, what functionality it supports, what characteristics it should have to enable various value-added abilities, and what should be its possible relation with and requirements it may impose on the design and structure of single agents. This is what this paper is all about.

The Intelligent Agents Group at Carnegie Mellon University[1] has had a long history in researching various issues in MAS, such as MAS stability [42], MAS learning [1], MAS coordination [8, 43]. In addition, we have been building and experimenting with MAS [7, 40].

In this paper, we will distill our experience of recent years into an account of what constitutes MAS infrastructure, and specifically, what characteristics and abilities different parameters within the infrastructure afford. Our definition and treatment of MAS infrastructure will not be as encompassing as the one proposed in [15]. It will be concerned mainly with technology development, applications and use, rather than involving scientific and educational MAS activities.[2] This account of the MAS infrastructure has resulted from our vision that the computational world will soon be populated with multiagent societies that are heterogeneous in agent structure, multiagent organization and functionality. Our thinking on MAS infrastructure was guided by the desire to enable the flexible design, building and operation of such societies. One important element that our account articulates is the relation between infrastructure for a *single agent* and the infrastructure for the MAS in which the agent participates. We consider MAS infrastructure to be the domain independent and reusable substratum on which MAS systems, services, components, live, communicate, interact and interoperate, while the single agent infrastructure is the generic parts of an agent that enable it to be part of a multiagent society, i.e to be socially aware.

In developing our own multiagent infrastructure, RETSINA, we made various design decisions that were motivated by our assumptions of what is the best added value that future MAS could provide. In this paper, we will describe the RETSINA infrastructure as an implemented instantiation of the proposed abstract infrastructure model and point out the particular design decisions and characteristics it embodies. The RETSINA infrastructure has evolved over the years. We have used it to implement a variety of applications in order to test the generic features of the infrastructure to make sure of its generality. Each subsequent application guided the refinement of the infrastructure towards increased generality and flexibility.

Since there is no standard MAS infrastructure in existence, we will use characteristics derived from our abstract model of infrastructure as dimensions along which to compare various MAS systems reported in the literature (see section 5.)

The rest of the paper is organized as follows: in section 2 we clearly define what we mean by MAS infrastructure; in section 3 we discuss the implementation of the RETSINA infrastructure; in section 4 we briefly present some applications that have been developed using the RETSINA infrastructure; in section 5 we present related work and finally we conclude in section 6.

2. MAS infrastructure

Agents in a MAS are expected to coordinate by exchanging services and information, to be able to follow complex negotiation protocols, to agree on commitments and to perform other socially complex operations. We define the infrastructure of a MAS as the set of services, conventions, and knowledge that facilitate such complex social interactions. Agents need services to enable them to find each other in open environments, to communicate, to warrant that the proper security constraints are satisfied. Conventions, such as Agent Communication Languages (ACLs), and conversational policies are the basis for achieving interoperability and agreement on what the agents are doing and what they are achieving; knowledge of how to use the infrastructure, ACL and protocols as well as a common ontology are needed by the agents so that they can be effective participants in the community.

Crucially, the above definition does not mention what the infrastructure should know about the internals of the agents in the system. We claim that from the point of view of the MAS infrastructure, agents are "socially aware" programs[3] that communicate, interact among themselves and with the infrastructure components, and whose behavior conforms to the rules of the MAS. An agent's problem solving capabilities, however, are a black box to the infrastructure.

Figure 1 shows how the different services provided by a MAS infrastructure are organized in an abstraction hierarchy, in which the higher levels rely on the functionalities implemented by the lower levels.[4] The infrastructure diagram has two parts: the MAS infrastructure, and the single agent infrastructure that allows an agent to be part of a MAS. The diagram also shows how the components of the infrastructure are reflected in the internal structure of an agent.[5] In the diagram, the Problem Solving layer of an agent is absent precisely because the infrastructure does not make any assumptions about it.

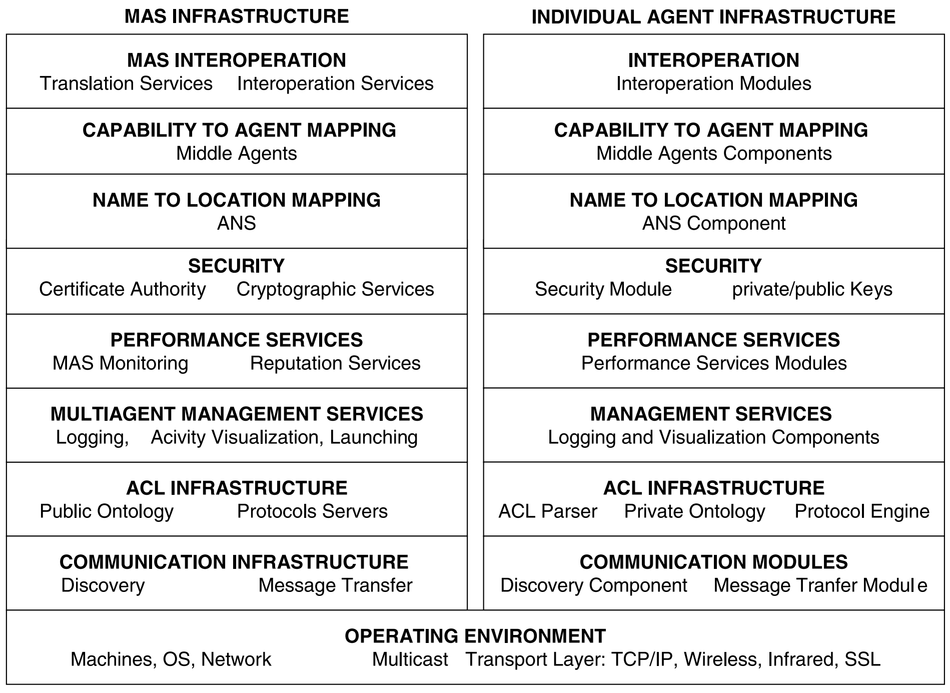

Figure 1. MAS infrastructure and individual agent infrastructure that allows an agent to be part of a MAS.

Our claim has profound consequences: first it defines MASs as inherently heterogeneous, in the sense that any agent can enter the system and interact with the other agents independently of its internal architecture and model of the world. Similarly, the MAS infrastructure is agnostic on the points of particular coordination regimes. We claim that the MAS infrastructure should be general enough to facilitate any coordination scheme such as team behavior [18, 30], negotiation [24], Contract Nets [8, 38] etc. This is why there is no coordination layer in the figure. In addition, we feel that social norms [3] are not part of the infrastructure but are particular to the design of a given MAS society.

In the following, we provide a description of the infrastructure layers.

2.1. *Operating environment*

At the bottom of the conceptual layering of the infrastructure, a MAS relies on an *Operating Environment*, i.e: on physical computers, on their operating system, on different types and topologies of the networks that connect different agents and different means of information transport. Single agents also use this infrastructure without any additional components or awareness. This is why the "operating environment" layer runs across both the MAS infrastructure and the single agent infrastructure portion of the figure. This level of abstraction should be totally transparent to the agents and the MAS, which should work across different platforms and networks.

2.2. *Communication infrastructure*

A MAS is implemented on top of a *Communication Infrastructure* that transfers messages between the agents as well as between the agents and the MAS infrastructure. Current communication channels have various modalities, such as wired, wireless, infrared etc. To ensure maximum flexibility in MAS communications, the communication channel should support different modalities of communication between agents, such as synchronous or asynchronous communications, as well as be abstracted from the actual transport layer and the ACL used [34]. ACL independence does not mean that the ACL can be underspecified within a specific MAS, rather it means that the same communication infrastructure can be reused by different MAS that use different ACLs.

Independence from the transport layer guarantees that agents can communicate whenever there is an open connection between them, independent from the way in which this connection is implemented and from contingency situations that are not under the control of the agents. For example an agent should be able to be connected to other agents via a socket connection, or via infrared or with some sort of wireless radio connection. No matter what media is used, if there is an open connection between the agents, they should succeed in communicating.

Within an individual agent, communication infrastructure is needed, i.e: an ACL-independent communication module that formulates an agent's messages, taking into consideration particular communication channel characteristics (e.g. wired, wireless).

Another important infrastructure service at this layer is the *discovery of infrastructure components*. For example, when an agent first comes up in an open environment, it may want to register itself with agent name services (see the discussion on ANS in subsection 2.7). Instead of having hardwired IP addresses for such services, the MAS infrastructure and the corresponding single agent infrastructure can facilitate the discovery of existing ANSs. UPnP and JINI are examples of such discovery protocols. (See also section 3 for description of such infrastructure discovery protocols implemented in the RETSINA infrastructure).

2.3. ACL infrastructure

An essential part of creating a community of agents is the specification of a language that is understood by all the agents in that community. For this reason, the specification of an ACL, protocols and conversational policies used by the agents is an essential part of the specification of the MAS and it constitutes a part of the MAS infrastructure.

An ACL should specify the syntactic form of the messages exchanged. In addition, it should specify the semantic interpretation of the messages, so that an agent understands what the messages that it receives are all about. The interpretation of the messages relies on the specification of a shared ontology in which the terms used are defined. In turn, the ontology can be used to extract the meaning of the messages themselves. Conversational policies [20] and protocols embody the roles and social context [36] of agent communication. The social context constitutes the pragmatics against which agent communications are interpreted and used.

Correspondingly, an individual agent's infrastructure should support interpretation of a message by an agent, and facilities for allowing an agent to send messages. In addition, the agent should know what to do with the message it receives, i.e: how to parse the message, and how to interpret it in the context of an on-going conversation. Therefore, along with the ACL there should be a definition of a set of protocols [37] and conversational policies [21] that specify what an agent's role is and how a message fits in the general scheme of the messages exchanged by the agents. For instance, a request for information should be followed by an answer or by a "sorry message": an acknowledgment that the agent cannot provide an answer. An agent's language infrastructure should support the understanding of some public ontology that expresses the conversational content.

2.4. Multiagent management services

MAS infrastructures should also provide additional system operation services which we labeled *Multiagent Management Services* in Figure 1. Such services provide facilities that support the work of a MAS over time: they include *Logging* facilities that record the messaging activity of agents in the MAS; *Management Tools* that monitor and visualize the activity of the MAS; and *Installation Services* and *Launching Services* that ease the burden of starting and configuring the many agents that comprise a MAS.

2.5. Performance measurement

Because MASs are in general heterogeneous, the agents differ in their ability, efficiency, reliability etc. The MAS should provide *Performance Measurement* to monitor the performance of the agents. For example Performance Measurement services could be used to optimize the distribution of tasks across agents. Such services could rank S services in terms of performance, so that the more efficient would be more likely to receive requests. Also, the reputation of agents might be monitored [46]. Any agent that provides false or unreliable information would lose credibility within the MAS and it would not be used by any agent that needs its service. In addition, failures could be monitored and the information collected could be used for failure tracking or facilitating failure recovery.

Although the performance measurement services for MAS could operate without the individual agents being aware of them, there could be corresponding services within an individual agent that increase agent effectiveness as a MAS participant. For example, an agent could be *self-aware*, i.e monitor its own performance and try to optimize it. Or, an agent could monitor its own failures and try to recover from them.

2.6. Security

Agents in an Open MAS, where agents can join and leave the society dynamically and where agents have been designed by different development groups, meet as perfect strangers. Each agent knows very little or nothing about the agents with whom it interacts. Therefore, security services are needed to ensure that agents do not misbehave.[6]

The security layer of the MAS infrastructure deals with these problems. It defines a set of trusted services, as for example certificate authorities, that help authenticate the identity of the agents, and a set of protocols that are guaranteed to prevent voluntary and involuntary losses of goods, services, or other values during the interaction.

Individual agent infrastructure should make sure that agents in the system can interact with these Security services. Such an example would be an agent interacting with the Certificate Authority to retrieve be granted certificates necessary to perform its transactions. Furthermore, agents should know and be able to handle encryption and to follow the secure protocols.

2.7. Mapping names to agent locations

A MAS infrastructure includes facilities to find agents by some identifying feature, such as a name. MAS can be divided in two classes: systems that abstract from the physical location of agents and systems that do not. CGI-BIN scripts on the Web are an example of a MAS that employ fixed locations. Each CGI-BIN script is addressed by the name of the web server on which it is running and the exact location within such a server. When the CGI-BIN script is moved to another location, all the references to it should also be updated, but there is no provision in the HTTP protocol or anywhere else that does it automatically. Furthermore, while new CGI-BIN scripts are constantly added and removed from the web, any reference to them is hardwired either in a HTML form or in other CGI-BIN scripts, since there is no mechanism nor provision that allows web pages to reconfigure automatically to make use of new services provided nor to detect when services that they used to access are no longer available.

In the general case, agents can join and leave a MAS dynamically and unpredictably. Agents that are not bound to a particular physical location can appear anywhere on the net and still be part of the community of agents. While this flexibility provides an essential advantage because an agent developer does not need to care where an agent is located, it requires services for mapping the agent name dynamically to the agent location. In addition, such facility provides the basis for agent mobility. No agent that is bound to a precise location can move and still be part of a MAS. To abstract from the physical location of the agent, the MAS infrastructure should maintain a registry to map the name of the agent to a physical location so that it can eventually be reached. Such a registry is represented in the Figure 1 as the ANS: Agent Name Server. An ANS is like a DNS but with increased flexibility for real time updates, discovery services (see Communication Infrastructure Layer in Figure 1), automatically "pushing" agent name registration to other ANSs etc. Systems that are based on the CORBA ORB [5], such as the Sensible Agent Testbed [2], or on JINI [26], such as the Grid [4], or on the RETSINA ANS infrastructure (see section 3.7) use an underlying infrastructure that automatically abstracts from the physical location of the agents.

The *ANS Component* in the figure is the corresponding individual agent infrastructure that registers and unregisters with the ANS and initiates lookup requests for a desired agent.

2.8. Mapping capabilities to agents

A general MAS should support an open rather than closed agent world.[7] Open systems allow agents to enter, and exit, the system dynamically and unpredictably, while closed systems employ a fixed set of agents that are known a priori. Since in an open MAS the set of agents is not known a priori, the infrastructure should provide ways for its agents to locate each other based not only on name but on functionality or capability. Locating agents by capability is solved by employing a set of infrastructure agents called *Middle Agents* [6]. Some examples of middle agents reported in the literature include the OAA Facilitator [29], the RETSINA Matchmaker [41] and the Infosleuth Broker [33]. Middle Agents maintain an up-to-date registry of agents that have made themselves known to the MAS community, along with the services that each agent provides. This information is called the agent's capability *advertisement* and is provided by the agent to a middle agent. When an agent needs another that has some required capability, it sends a middle agent a *request* specifying the desired capability. The middle agent matches requests and advertisements. In general, there could be a variety of middle agents that exhibit different matching behaviors and have different performance characteristics. In prior research, we have identified 28 middle agent types and have experimented with different performance characteristics, such as load balancing, fault tolerance etc. [6, 45].

Whether the system allows Middle Agents or not affects the infrastructural requirements of a single agent. An agent must have the ability to construct advertisements to make itself known to the agent community and also construct requests to take advantage of services provided by other agents. If an agent lacks these abilities, it would be stand alone and isolated from the MAS activities.

2.9. Interoperation

As the number of MAS created by different groups increases, the need for MAS interoperation will increase also. The development of sharable ontologies, conversational policies, ACLs and translation services will go a long way towards allowing individual agents to interoperate. However, additional infrastructure is needed to take care of MAS architectural mismatches, for example between a centrally controlled MAS that uses a

Facilitator as middle agent and a distributedly controlled MAS that uses a Matchmaker as middle agent. Each MAS may have its own architecture-specific features, such as: agent registration, agent capability advertisement, agent communication language, agent dialogue mediation, default agent query preference, and agent content language. Since MAS are in general open, there is the further requirement that interoperation must be done in real-time so as to capture the dynamics of the agent world. If an agent enters one MAS community, agents in the other MAS communities should have ways of finding and transacting with this agent, if it matches a required capability.

Currently, only a couple of research interoperation systems exist (see section 3.9 and 5) between architecturally different open MAS, but we believe this area will receive increased attention, as more MAS get developed and deployed.

2.10. Agent interaction with the MAS infrastructure

The infrastructure provides a set of components that allow agents to interact with each other. Agents in the MAS use these components as tools to discover the agent landscape to find other agents and to interact with them. While heterogeneous MAS infrastructure should not make any assumption on the internal structure of the agents, Figure 1 shows that the agents do make assumptions on the infrastructure, and use these assumptions to be part of the whole MAS. These assumptions consist of the services the MAS infrastructure provides, of the protocols that the different services of the MAS infrastructure require and of the communication channels that the MAS infrastructure implements.

Yet this is not enough for an efficient use of the MAS infrastructure. Agents also need to distinguish between the different services provided by the infrastructure and to know how to use the infrastructure for error recovery. For instance, brokers and matchmakers are both middle agents, but they are used in very different ways. A broker indirectly provides a way to anonymize a requester so that its requests are not disclosed to the providers, also a broker may perform load balancing tasks that a matchmaker can not do and it may take care of managing part of the task for the requester [6, 7]. On the other hand the broker constitutes a single point of failure, and it may easily become a bottleneck because it manages all the interactions between the requesters and the providers. In order to use the MAS infrastructure efficiently, the agents in the MAS need to understand, or have rule for, the tradeoffs that are intrinsic in different modules of the infrastructure [27].

In conclusion, the agent does not only need modules that control the interaction with the infrastructure components, but also it needs rules to decide how to make use of such components and what to do with the information provided by the infrastructure. How this information is used may lead to failure recovery, coordination, team behavior and ultimately to the intelligent and autonomous behavior of the agents.

One of the main topics of research in MAS is coordination and how agents can interact with each other without destroying each other's work. Above, we claimed that the MAS infrastructure should support any coordination regime and therefore be agnostic with respect to the coordination methods. Indeed, coordination behavior results from the MAS infrastructure in two ways: either it is programmed as an additional layer to the infrastructure, as shown below in section 4.3, or it emerges as a "social behavior" from the independent problem solving of the agents, as in the case of team behavior.

Team formation requires that the teammates become team members on the basis of the role they have to play in the team plan and ultimately of their capabilities [18]. This functionality is supported by the middle agents. Crucially, the infrastructure does not dictate a team plan to the agents, or what the agents need to do. The execution of the team plan is supported by the infrastructure secure communication channels; furthermore, through the infrastructure the teammates can monitor whether the team is still intact and what progress the teammates made toward the joint goals. Finally, the infrastructure facilitates failure repair by facilitating the location of agents that may join the team to replace failing teammates or to assist in recovering from failures that emerged during execution.

3. The RETSINA MAS infrastructure

RETSINA is an open MAS infrastructure that supports communities of heterogeneous agents. The RETSINA system has been implemented on the idea that agents in the system should form a community of peers that engage in peer to peer relations. Any coordination structure in the community of agents should emerge from the relation between the agents rather than being imposed by the infrastructure. Following this premise, RETSINA does not employ any centralized control on the MAS, rather it implements distributed infrastructural services that facilitate the relations between the agents instead of managing them.

The organization of the RETSINA MAS infrastructure is displayed in Figure 2. It shows how the various components are organized on the basis of the infrastructure model in Figure 1. In the rest of this section we will describe these components.

3.1. Operating environment

The RETSINA MAS is independent of the platform on which the infrastructure components and the agents run, and it automatically handles different types of transport layers.

Applications of the RETSINA MAS are routinely distributed on different platforms ranging from different versions of Windows, different versions of Linux, and Sun OS. Furthermore, they include agents running on PalmPilots and iPAQ. The agents used have been implemented in different languages, such as Java, C, C++, Python, LISP, and Pearl. Communication transport layers handled by RETSINA include TCP/IP, wireless, SSL, infrared, and serial connection.

3.2. Communication infrastructure

RETSINA is based on two types of communication channels: one provides message transfer for direct peer to peer communication between the agents, the other is based on multicast used for a Discovery process that lets the agents find infrastructure components.

Direct message transfer is supported in an individual agent by the RETSINA Communicator [34] that provides an abstraction over the physical transmission layer abstracting over the type of network used. The Communicator supports synchronous as

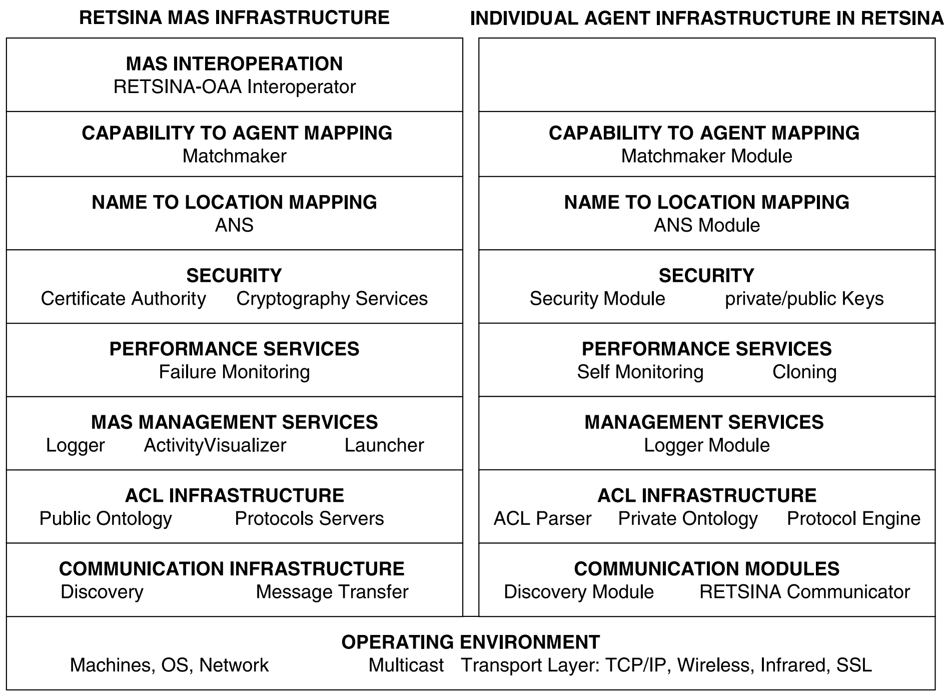

Figure 2. The RETSINA MAS infrastructure and individual agent infrastructure.

well as asynchronous communication, and it manages multithreaded communication that allows the agent to maintain conversations with multiple agents at the same time.

Discovery uses multicast to connect agents to the infrastructure components. For example, an ANS announces its presence by multicasting. An agent can also announce its presence by multicasting a request for an ANS. If the agent finds an ANS, it registers with it, or performs a lookup request. To reduce the load on the multicast channel, no negotiation happens directly on multicast; direct transactions between the agents and the infrastructure are performed on a direct channel [28].

The use of Discovery allows flexible entrance of agents and infrastructure in the RETSINA MAS. An agent can enter the MAS when no infrastructure is yet present, and wait until infrastructure components enter the system. After these components multicast their presence, the agent registers with them and from that moment on it is effectively a reliable resource for the agent community. Discovery is also very useful for agents running on mobile platforms. While these agents might not be mobile themselves [14, 39], they may move around just because the platform they are running on is moved. Using Discovery in the new place, the agent can re-orient itself and find the local components of the infrastructure.

3.3. ACL infrastructure

The ACL used in the RETSINA MAS is KQML [10]. Messages exchanged by the agents have two components: one is the specification of the content of the message, the other is an envelope that specifies information such as sender, receiver, performative, thread of conversation, ontology and language used in the content part. The RETSINA infrastructure dictates the format of the envelope, because it is used to deliver the message, but it does not make any assumption on the content of the message itself. Any content would do as long as the agent that receives the message can understand it.

The specification of the language does not guarantee that the agents understand each other. They also need to have a shared dictionary which specifies the meaning of the words that the agents use. For this reason RETSINA provides an ontology based on domain-specific taxonomies of concepts derived from the Wordnet[8] [9].

The taxonomies are used to measure similarity between terms within messages. For example, the ontology recognizes the similarity between "location" and "city", because the first is a super-concept of the second. The use of taxonomic similarity adds flexibility to the communication process since agents are not forced to use exact terms in their messages.

Finally, the RETSINA MAS provides a protocol engine and a protocol language to specify agents' roles and the messages to be exchanged and expected in the context of a protocol. The protocols employ social semantics [36]. Currently, the implementation of the protocols is in the form of finite I/O automata [8].

3.4. MAS management services

The management of applications of a MAS proves to be a very complex task that is becoming more and more difficult as the application size, the number of agents and machines involved increases. Tools are needed to help monitor the activity of the agents, to debug MAS applications and to launch these applications. The RETSINA MAS includes three management components: the *Logger, ActivityVisualizer* and *Launcher* that form an initial set of tools that help with monitoring, debugging and launching MAS applications.

The *Logger* records the activity of the agents. Specifically, the Logger records agent entry to and exit from the system, and the exchange of messages. In addition, the Logger records states and transitions within the agents, as for instance whether an agent is active or waiting for a query from other agents. Since the Logger cannot spy on the agents, the agents need to implement a *Logger Module* that relays to the Logger information about their state and their communications.

The Logger is connected to the *ActivityVisualizer* that displays the activity within the system. The ActivityVisualizer uses the information provided by the Logger to display in real time which agents are in the system, their state and indicate when they exchange messages. Furthermore, the Logger and the ActivityVisualizer can be used in play-back mode thus permitting the agent programmers to review and analyze activity in the MAS.

An additional component of this MAS infrastructure layer is the *Launcher*, which automatically configures and starts both infrastructure components and agents on different machines, platforms and operating systems from a single point of control, greatly reducing the launching and operational costs of distributed applications. The launcher is of great value especially as different agent versions get developed and as agents may change resource requirements or as they need to be moved and restarted on different machines [16].

3.5. Performance services

RETSINA MAS does not include MAS performance service or reputation service. We have however built performance service monitors in simulation as well as distributed checkpointing and roll-back upon agent failure. We have experimented with different monitoring services in the context of contract net family of protocol [38]. Some agents implement a self monitoring mechanism [35] that predicts when the agent is going to be overwhelmed by the load of tasks it performs and *clones* itself producing a brand new agent with the same functionalities and delivering the same service as the original agent. The set of tasks of the original agent is split and re-distributed between the old agent and the clone, thus allowing for increased system throughput.

3.6. Security

Since RETSINA is an open system, unknown and possibly untrustworthy agents can enter at any time. These agents can damage the system in many ways: they can spy on other agents, steal goods or information, and damage the content of the infrastructure components. For instance, a malicious agent might prevent the MAS from working, by unregistering all the agents from an ANS. The security infrastructure of the RETSINA MAS prevents such problems from happening.

In RETSINA, we guarantee three types of security: agent authentication via a Certificate Authority, communication security, which guarantees that the communication between agents cannot be eavesdropped, and integrity of the components that guarantees that no component can be inappropriately manipulated. Communication security is achieved by giving agents unique IDs, as private keys which are verified using public keys, and by layering SSL underneath the communication interface used by the agents. Integrity of the MAS components, such as the ANS, is also guaranteed by relying on the unique IDs of agents and by adding access control mechanisms.

The security components of the RETSINA infrastructure for the individual agent are the Security Module in the agent and the Certificate Authority in the MAS infrastructure. The Security Module generates the private and public keys of the agent and it requests certification of the public key from the Certification Authority, which binds the requester's ID to its public key [44].

3.7. RETSINA ANS

An ANS provides a means of abstraction from the physical location of agents by mapping an agent ID to its address in the system. The ANS is then queried by agents when they need the address of other agents, for instance when they need to send messages. An ANS does not participate in the transaction between agents, it only provides them with addresses that they can cache, removing the need for unnecessary lookups. In addition an ANS provides robustness of agent communication in the event of an ANS failure, since the agents can continue their transaction even when no ANS is available in the system.

Since an ANS plays a crucial role in the system, it should not become a single point of failure that would prevent the whole MAS from functioning. This is prevented in two ways: first by limiting the role of the ANS in the interaction between agents, and by using a system of *multiple* and redundant ANSs. Multiple ANSs can be present in the system at the same time. ANS servers find each other through Discovery using multicast within a LAN. Since it is not feasible to use multicast outside a LAN, therefore RETSINA uses a discovery mechanism based on P2P protocols such as Gnutella that allows discovery of agents and infrastructure services over WANs [28]. Through reference ANSs the lookup search for an agent can be spread to a much wider network and possibly the whole Internet.

Within an individual agent, the ANS component enables the agent to register and unregister with an ANS and request lookups of desired agents.

3.8. Middle agents

Agents enter a MAS to exchange services with other agents, but since RETSINA is an open system, no agent can be sure of what services are available in the MAS at any given time, and who provides them. It is a task of the infrastructure to provide a registry of services available in the system and to allow agents to search for them in this registry.

RETSINA solves the service location problem by using a set of middle agents called Matchmakers distributed across the MAS [25]. Each Matchmaker records a mapping between agents in the system and the services that they provide. A Matchmaker uses two types of data: the advertisements of the services provided, and the requests from agents that need a service, both of them expressed in the LARKS [41] language. The task of a Matchmaker is to find which advertisements match the requests. To accomplish this task a RETSINA Matchmaker uses the LARKS matching engine that performs both syntactic and semantic analysis of the advertisements and requests to find exact or partial matches.

The RETSINA Matchmakers differ from other Middle Agents such as the OAA Facilitator [29] and Infosleuth's Broker [33] in that they do not stay in the middle of the interaction between the providers and the requesters. A requester agent gets from a Matchmaker the contact information of relevant providers and asks them directly to perform a service. This crucial difference makes the RETSINA Matchmakers less of a single point of failure, since after a requester has been given a list of providers, it can continue its transactions directly even when no Matchmaker is present. In addition, a requester can cache providers' contact information and reuse them without resorting to a Matchmaker every time.

A Matchmaker supports two types of protocols: "single shot" and "monitor". A single shot request to a Matchmaker results in the list of providers whose advertisements match the request. A monitor query instead, results in the list of matching agents, but also in updates whenever one of the providers exits the system, or new relevant providers enter. Agents select the protocol depending on whether they need a snapshot of the agent landscape or they need to be kept up-to-date on the changes in the system.

3.9. RETSINA-OAA InterOperator

Imagine an OAA agent trying to enter the RETSINA MAS. Such an agent would be totally lost and unable to interact with either the agents or with the infrastructure components. It would not be able to communicate with any agent because it would "speak" the Prolog-based OAA ICL, while every agent in the RETSINA system

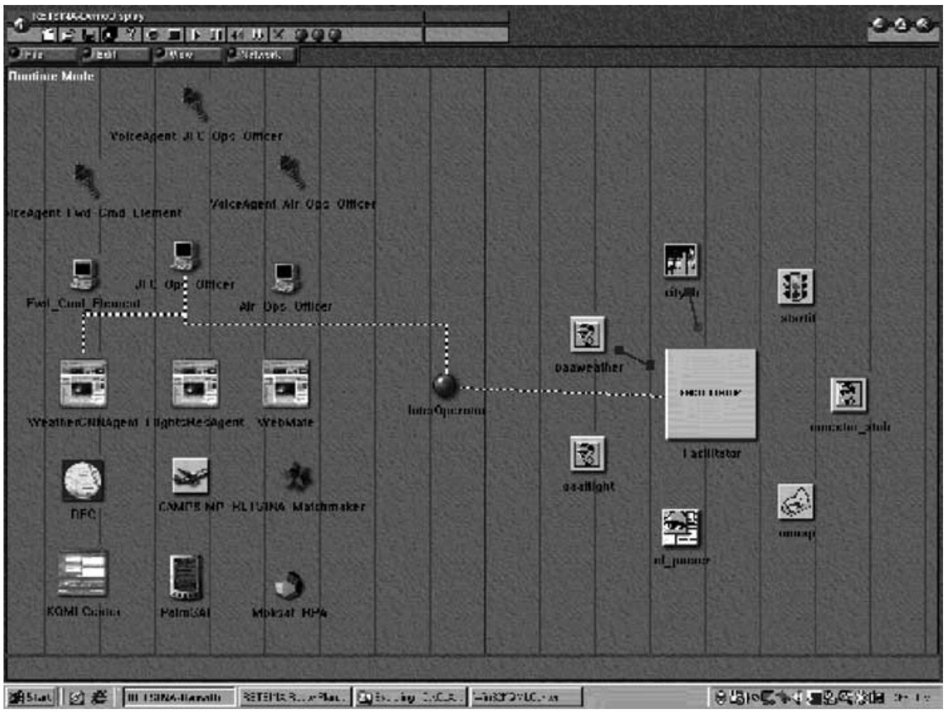

Figure 3. The InterOperator agent mediates between the RETSINA MAS (on the left) and the OAA MAS (on the right).

"speaks" KQML. Furthermore, it would expect to deal with a Facilitator, but may end up dealing with a Matchmaker instead, with the result that it would not be able to ask for services nor to interpret what is returned by the middle agent.

While many claims have been made about openness of MAS, the current practice is that MAS developers make such strong assumptions on the agents they develop that natural interoperation across MAS boundaries is virtually impossible.[9] To interoperate between OAA and RETSINA, we developed the RETSINA-OAA InterOperator [17] Figure 3. The task of the InterOperator is to allow any agent in the RETSINA system to access any service or information provided by OAA agents, and for any agent in the OAA system to access services or information provided by RETSINA agents.

The RETSINA-OAA InterOperator, shown in Figure 3, "bridges" the two worlds of RETSINA and OAA by performing two types of tasks: first it makes the two systems visible across MAS boundaries, second it allows agents to exchange messages across MAS. The first task is accomplished by collecting all the advertisements of RETSINA agents, translating and registering them with the OAA Facilitator. Similarly, the advertisements of OAA agents with the Facilitator are collected and advertised with the RETSINA Matchmaker. Therefore, through the RETSINA-OAA InterOperator, the two systems are able to "see" each other's agents. The second task is accomplished by translating the queries of the agents of one MAS to the agents of the other MAS, and then translating the answers back.

Due to fundamental differences in the architectures and ACLs of the RETSINA and OAA multi-agent system architectures, it is not possible for all forms of agent-to-agent interaction of one MAS architecture to be translated to the other. Nevertheless, the RETSINA-OAA InterOperator does adequately allow for the necessary agent interactions to occur across MAS boundaries.

4. Applications

The RETSINA MAS infrastructure has been used to develop many applications that range from supporting teams of human decision makers in crisis response, to financial portfolio management, and E-commerce. In the following we show how these applications are supported by the infrastructure and we highlight some of the infrastructure features.

4.1. Warren: Financial portfolio management

The WARREN system [7] is an application of RETSINA to financial portfolio management. Warren is composed of three types of agents: interface agents that display the portfolios to the users, task agents that assist the user in the management of her portfolio, and information agents that gather information about stocks in the portfolio (for example, stock prices, news and company financial reports.) Through the interface agent, the investor buys and sells stocks, monitors the value of her own portfolio, and monitors news about the stocks in the portfolio. Agents support the investor decision making by reporting current stock quotes, and by advising on risk associated with changes in the portfolio.

Because prompt and accurate information is essential in this domain, the loss of the information agents may result in actual loss for the investor. The agents in Warren use the infrastructure to discover whether any agent is temporarily unreachable and when this is the case they automatically replace the agent with another with similar capabilities. Warren shows how a MAS uses the infrastructure to dynamically reconfigure itself to prevent failures totally transparently to the user.

4.2. Aiding teams of humans in crisis response

MAS reconfiguration and failure repear are essential in Crisis Response domains. In one application of RETSINA, the agents help human decision makers to plan an hypothetical evacuation of civilians out of Kuwait City. In this scenario, the humans are distributed in space, but they are assisted by an interface agent, called Messenger, to communicate with each other. Messengers eavesdrop the conversation to identify and anticipate information needs that help in the decision process. Also, they use the MAS infrastructure to identify the agents that monitor the information sources of interest to the decision makers and to substitute failing agents.

The RETSINA OAA InterOperator agent (described above in 3.9) is used in this context to lower the boundaries between the two systems:agents on the RETSINA side could query agents on the OAA side; similarly, agents on the OAA side could query agents in the RETSINA side. Furthermore, the InterOperator agent allows substitutability of agents across MAS boundaries, so if one of the agents on the RETSINA side is suddenly no longer available, the MAS would reconfigure to use an equivalent agent on the OAA side.

Crisis response also offers the opportunity to experiment with coordination and team behaviors. Each decision maker was assisted by a "Mission Agent" that negotiated a shared plan freeing the decision makers from the burden of dealing with all details of constructing a common plan of action. The Mission Agents perform information gathering, shared planning and monitoring the plan execution. The Mission Agents automatically construct a shared plan and monitor its execution, negotiating changes when failures occur [18].

4.3. Coala: Buyer coalition and e-commerce auctions

Coala [43] shows how auction coordination is implemented as an additional layer using the RETSINA infrastructure. Buyer agents use the MAS infrastructure to find auctions of items the buyers want to buy. The system supports a variety of auction protocols such as English, Dutch and Spanish Auction, as well as collective purchasing by bundling large groups of buyers into coalitions through a pre-negotiation protocol and a variation of sealed-bid reverse auction that allows suppliers to disclose their discount policies to the buyers.

4.4. MOCHA: Connecting devices and people

MOCHA implements an agent based system designed to assist humans in daily communication and information retrieval tasks. The MOCHA agents take on such tasks as communication planning, relevant information retrieval related to ongoing communications and communication device management. In MOCHA, agents are associated with devices such as printers, faxes and mobile phones, as well as with tasks. Agents are also running on a variety of platforms ranging from desktops, PalmPilots, IPaq and other PDA devices.

MOCHA creates an open dynamic and robust communications infrastructure, based on the RETSINA infrastructure, that supports a network of devices in which not all devices are available at the same time. The monitoring and discovery features of the RETSINA infrastructure allow MOCHA to keep track of which agents are available and of the most efficient way to communicate between people, thus providing ubiquitous connectivity and computing.

5. Related work

Above we gave a functional definition of the infrastructure for MAS as a set of services, conventions and knowledge that support the agents' social interaction. We then described RETSINA as an implemented and fully functional MAS infrastructure. In this section we will discuss how these functionalities are implemented in different MAS. As previously in the paper we refer to the layer architecture presented in Figure 1.

5.1. ACL infrastructure

The definition of a communication language is an essential part of creating a community of agents. Most implemented research MASs, for example, RETSINA, DECAF

[19], Infosleuth [31], Jade [23] among others, use KQML [11] or FIPA ACL [12] to communicate.

OAA agents instead exchange messages in the form of PROLOG predicates. One key difference between the ACL used by OAA and KQML or FIPA is that in the OAA ACL there are only two performatives: “solve” that is used to query other agents, and “solved” that is used to answer the query. But there is no way to express a performative equivalent to assertions like the “tell” in KQML. As a consequence, OAA agents are forced to maintain a precise history of the message exchange and infer from it what kind of message they received and what they should do with that message.

5.2. MAS management services

Launching many agents on multiple platforms at the same time is a very time consuming process. In RETSINA we developed a launching and management system for our agents. RETSINA also provides tools monitoring the activity within the MAS and management facilities.

A similar system is used by ZEUS [32] that implements a visual editing system that allows the programmer to construct the MAS and to specify the interactions between the agents. The editing system can also be used for monitoring and management facilities. OAA implements an application called “startit” that starts, manages and shuts down the system.

5.3. Security

Security is a concern in MAS implementations because, as we discussed above, agents can misbehave by cheating other agents or by affecting the integrity of the system. Yenta [13] as well as RETSINA implements a security system to protect the integrity of its Matchmaker. Security is a major concern in the mobile agents community [22], since agents have access to a remote host and their misbehavior might damage the host as well as the MAS infrastructure the agent belongs to.

5.4. Mapping between agents, capabilities and locations

RETSINA and DECAF implement Matchmakers and ANSs as lookup services: the Matchmaker maps capabilities to agents, the ANS maps agents to locations. OAA and Infosleuth [31] implement brokers that map capabilities to agents and their locations. The first difference between the two approaches is that the Matchmaker does not manage the interaction between the agents, while both the OAA Facilitator and the Infosleuth Broker do. The distribution of services implemented by RETSINA and DECAF increases the reliability of the system. Furthermore, advertisements in RETSINA [41] and DECAF represent the functionalities of an agent by specifying the types of inputs that it requires and the types of outputs it generates. In contrast, the advertisement of an OAA agent is just a predicate representing a sample query; it does not specify what information the agent requires to compute an answer or what information it returns. Finally, the advertisement in Infosleuth is the DB schema of the data provided by the agent, instead of the agent capabilities.

6. Conclusions

MASs are more than just a set of agents gathered in the same system, and more than an extension of single agents in some distributed fashion. To work together, agents need a way to find each other, a common communication language, a shared ontology to understand each other's messages. The role of the MAS infrastructure is to provide location services, ontologies, and language that allow agents to interoperate. The result is that MASs emerge by the aggregation of agents around an infrastructure which is the "glue" that keeps the agents together, rather than being a by product of the collaboration between agents.

The contributions of this paper are two fold. First, we provide a model of what constitutes a MAS infrastructure as a set of services and conventions that allow agents to interoperate. Our proposed model of infrastructure also shows how the MAS infrastructure should be reflected within a single agent so that it can become part of the MAS. Second, we present RETSINA as a fully implemented MAS infrastructure that adheres to the proposed model.

Acknowledgments

The authors would like to acknowledge the contribution of the many past and present members of the Intelligent Agents Research Group at CMU. Without their ideas, the devotion and their enthusiasm this research could not have been possible. We are especially grateful to Hao Chi Wong for her help on the discussion of security issues. This research has been sponsored in part by the Office of Naval Research Grant N-00014-96-16-1-1222 and by DARPA grant F-30602-98-2-0138.

Notes

1. More information on the activity of the Intelligent Agents Group can be found at http://www.cs.cmu.edu/~softagents.
2. Of course having a technological infrastructure does positively impact those two activities also.
3. We are NOT defining agents as socially aware programs, we say that they are such from the point of view of the MAS infrastructure. Each agent taken individually may have an architecture that satisfies different principles (for instance, it could be based on the BDI model), but in order to be part of the MAS, it should (explicitly or implicitly) implement the modules that we describe here.
4. We do not claim that our list of components is complete; rather it emerges from our experience in developing MAS applications.
5. We do not impose any implementation requirements on the modules of the individual agent infrastructure, we only claim that explicitly, or implicitly in its behaviors, the agent need those modules to interact with the MAS infrastructure.
6. Most common security issues include communication security and infrastructure integrity. Communication security guarantees that a message cannot be eavesdropped, authentication, so that the agents cannot spoof each other, and non-repudiation i.e: disallow agents to deny having taken part in a transaction. Infrastructure integrity guarantees that no agent can manipulate the information stored in the infrastructure components such as the ANS and the Matchmaker. In addition, Communication Integrity guarantees that the contents of a message cannot be changed by an unauthorized agent.
7. In closed MAS each agent knows the name, location and capability of the others. Thus agent interactions can

be statically predefined. This makes agent design and construction simple, but makes the MAS brittle and not extensible.

8. The Wordnet taxonomy could not be used as a direct taxonomic structure for a number of reasons: it is too big; it does not allow the user to browse the concepts in the ontology; furthermore, it is differentially sparse, which creates enormous problems for similarity assessment of concepts, so we encoded multiple smaller taxonomies to allow more efficient concept retrieval and a more precise similarity measurement.
9. Attempts at standardizations such as FIPA [12] are likely to reduce the problem, but not solve it. Differences will remain in the Ontologies used, the interaction protocols and the MAS architecture.

References

1. S. Arai, K. Sycara, and T. R. Payne, "Multi-agent reinforcement learning for scheduling multiple-goals," in *ICMAS2000*, 2000.
2. K. S. Barber, D. N. Lam, C. E. Martin, and R. M. McKay, "Sensible agent testbed infrastructure for experimentation," in *Agents 2000: Workshop on Infrastructure for scalable MAS*, Barcelona, Spain, 2000.
3. C. Castelfranchi, "Modelling social action for AI agents," *Applied Artificial Intelligence*, vol. 103, pp. 157–182, 1998.
4. Coabs, "Grid Web Site," http://coabs.globalinfotek.com/, 2000.
5. Corba, "Corba Web Site," http://www.corba.org/, 2000.
6. K. Decker, K. Sycara, and M. Williamson, "Middle-agents for the internet," in *Proceedings of IJCAI97*, 1997.
7. K. Decker, K. Sycara, and D. Zeng, "Designing a multi-agent portfolio management system," in *Proceedings of the AAAI-96 Workshop on Internet-Based Information Systems*, Portland, OR, 1996.
8. G. Economou, M. Paolucci, M. Tsvetovat, and K. Sycara, "Interaction without commitments: An initial approach," in *Agents 2001*, 2001.
9. C. Fellbaum, *WordNet: An Electronic Lexical Database*, MIT Press, 1998.
10. T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language," in J. Bradshaw, (ed.), *Software Agents*, MIT Press, 1995.
11. T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language," in J. Bradshaw, (ed.), *Software Agents*, MIT Press, 1997.
12. FIPA, "Foundation For Physical Agents," http://www.fipa.org/, 2000.
13. L. N. Foner, "A security architecture for multi-agent matchmaking," in *ICMAS-96*, 1996.
14. S. Funfrokcen, "Transparent migration of Java-based mobile agents: Capturing and reestablishing state of Java programs," in *MA98*, Berlin, Germany, 1998.
15. L. Gasser, "MAS infrastructure: Definitions, needs, and prospects," in T. A. Wagner and O. Rana, (eds.), *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, LNCS. Springer-Verlag, 2001.
16. J. A. Giampapa, O. Juarez-Espinosa, and K. Sycara, "Configuration management for multi-agent systems," in *Agents 2001*, 2001.
17. J. A. Giampapa, M. Paolucci, and K. Sycara, "Agent interoperation across multagent system boundaries," in *Agents 2000*, 2000.
18. J. A. Giampapa and K. Sycara, "Conversational case-based planning for agent team coordination," in *ICCBR-2001*, 2001.
19. J. R. Graham and K. S. Decker, "Towards a distributed, environment-centered agent framework," in N. Jennings and Y. Lespérance, (eds.), *Intelligent Agents VI*, Lecture Notes in Artificial Intelligence, Springer-Verlag: Berlin, 2000.
20. M. Greaves, H. Holback, and J. Bradshaw, "What is a conversation policy?," in *Agents 99: Workshop on Specifying and Implementing Conversation Policies*, 1999a.
21. M. Greaves, H. Holmback, and J. M. Bradshaw, "What is a conversation policy?," in *In Agents99 Workshop on Specifying and Implementing Conversation Policies*, 1999b.
22. M. S. Greenberg, J. C. Byington, and D. G. Harper, "Mobile agents and security," *IEEE Communications*, 1998.
23. JADE, "Programmer's Guide," http://sharon.cselt.it/projects/jade/, 2000.

24. N. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 275–306, 1998.
25. S. Jha, P. Chalasani, O. Shehory, and K. Sycara, "A formal treatment of distributed matchmaking," in *Agents 1998*, 1998.
26. S. Jini, "Jini Web Site," http://www.sun.com/jini, 2000.
27. M. Klusch and K. Sycara, "Brokering and matchmaking for coordination of agent societies: A survey," in A. Omicini et al., (eds.), *Coordination of Internet Agents*, Springer, 2001.
28. B. Langley, M. Paolucci, and K. Sycara, "Discovery of infrastructure in multi-agent systems," in *Agents 2001 Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.
29. D. Martin, A. Cheyer, and D. Moran, "The open agent architecture: A framework for building distributed software systems," *Applied Artificial Intelligence*, vol. 13, nos. 1–2, pp. 92–128, 1999.
30. T. Milind, "Towards flexible teamwork," *Journal of Artifical Intelligence Research*, vol. 7, pp. 83–124, 1997.
31. M. Nodine, W. B. Amd, and A. Ngu, "Semantic brokering over dynamic heterogeneous data sources in InfoSleuth(tm)," in *Proceedings of the 15th International Conference on Data Engineering*, 1999.
32. H. Nwana, D. Ndumu, L. Lee, and J. Collis, "ZEUS: A tool-kit for building distributed multi-agent systems," *Applied Artifical Intelligence Journal*, vol. 13, no. 1, pp. 129–186, 1999.
33. B. Perry, M. Taylor, and A. Unruh, "Information aggregation and agent interaction patterns in InfoSleuth," in *cia99*, ACM Press, 1999.
34. O. Shehory and K. Sycara, "The retsina communicator," in *Agents 2000*, 2000.
35. O. Shehory, K. Sycara, P. Chalasani, and S. Jha, "Increasing resource utilization and task performance by agent cloning," in M. S. V. A. Rao and M. Wooldridge, (eds.), *In Lecture Notes in AI: Intelligent Agents*, Springer Verlag, 1998.
36. M. P. Singh, "Agent communication languages: Rethinking the principles," *IEEE-Computer*, vol. 11, 1998.
37. I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback, "Designing conversation policies using joint intention theory," in *ICMAS98*, IEEE Press, 1998.
38. R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.
39. N. Suri, J. M. Bradshaw, P. T. G. Maggie R. Breedy, G. A. Hill, T. S. M. Renia Jeffers, B. R. Pouliot, and D. S. Smith, "NOMADS: Toward a strong and safe mobile agent system," in *Agents 2000*, ACM Press, 2000.
40. K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng, "Distributed intelligent agents," *IEEE-Expert, Intelligent Systems and their Applications*, vol. 11, no. 6, pp. 36–45, 1996.
41. K. Sycara, M. Klusch, S. Widoff, and J. Lu, "Dynamic service matchmaking among agents in open information environments," *Journal ACM SIGMOD Record*, vol. 28, no. 1, pp. 47–53, 1999.
42. J. D. Thomas, K. Sycara, and T. R. Payne, "Heterogeneity, stability and efficiency in distributed systems," in *ICMAS1998*, 1998.
43. M. Tsvetovat, K. Sycara, Y. Chen, and J. Ying, "Customer coalitions in the electronic marketplace," in *Proceedings of Workshop on Agent-Mediated Electronic Commerce, Fourth International Conference on Autonomous Agents*, 2000.
44. H. C. Wong and K. Sycara, "Adding security and trust to multi-agent systems," in *Agents '99 Workshop on Deception, Fraud and Trust in Agent Societies*, Portland, OR, 1999.
45. H.-C. Wong and K. Sycara, "A taxonomy of middle-agents for the internet," in *ICMAS'2000*, 2000.
46. G. Zacharia, A. Moukas, and P. Maes, "Collaborative reputation mechanisms in online marketplaces," in *HICSS-32*, 1999.