The RETSINA MAS Infrastructure

Katia Sycara (katia@cs.cmu.edu), Massimo Paolucci (paolucci@cs.cmu.edu), Martin van Velsen (vvelsen+@cs.cmu.edu) and 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 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, to contribute towards the development of principles and practice to make the MAS infrastructure "invisible" and ubiquitous to the interacting agents[1].

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,

[1] 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.

TechReport: CMU-RI-TR-01-05.

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 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 (Thomas et al., 1998), MAS learning (Arai et al., 2000), MAS coordination (Liu and Sycara, 1996). In addition, we have been building and experimenting with MAS (Sycara et al., 1996; Decker et al., 1996; Sycara and Zeng, 1994).

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 (Gasser, 2000). 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

[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.

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 1 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 support 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 is 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

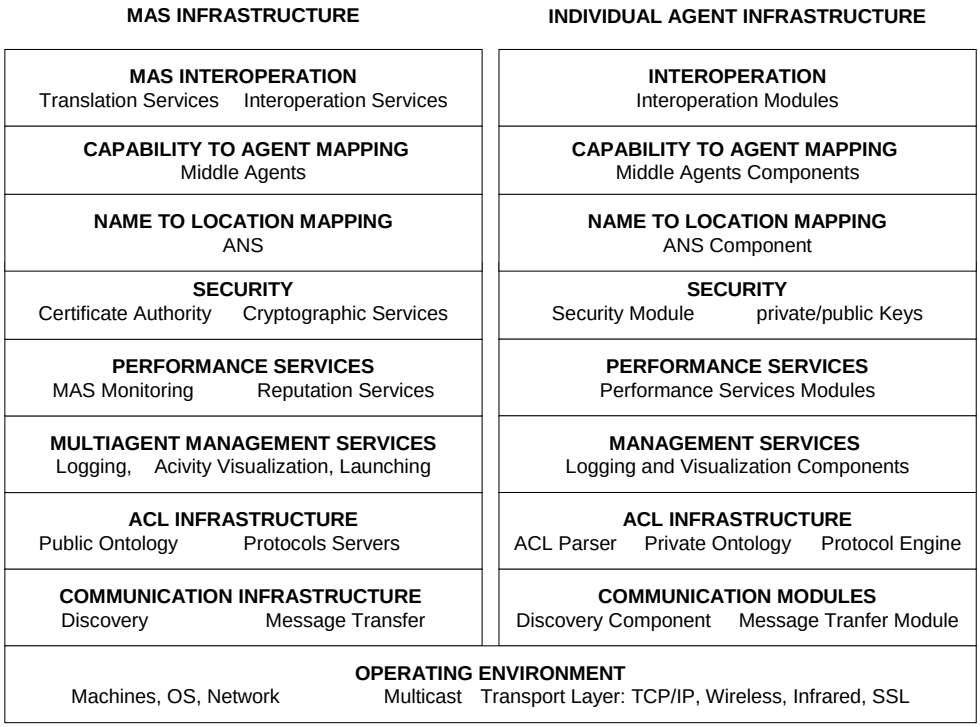

Figure 1. MAS Infrastructure and Individual Agent Infrastructure that allows an agent to be part of a MAS

"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.

[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.

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.

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 mute on the points of particular coordination regimes. We claim that the MAS infrastructure should be general enough to support various coordination schemes such as team behavior (Tambe, 1997), negotiation (Jennings et al., 1998; Sycara, 1990), Contract Nets (Smith, 1980) etc. This is why there is no coordination layer in the figure. In addition, we feel that social norms (Castelfranchi, 1998) are not part of the infrastructure but are particular to the design of a given MAS society.

In the following subsections, 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 (Shehory

[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.

and Sycara, 2000). ACL independence does not mean that the ACL can be under-specified 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 can be spoken and 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 (Greaves et al., 1999a) and protocols embody the roles and social context (Singh, 1998) 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 (Smith et al., 1998) and conversational policies (Greaves et al., 1999b) 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. In addition, 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 labelled *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 MAS 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 (Zacharia et al., 1999). 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 guarantee 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 the keys 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

[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.

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 the 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 (Corba, 2000), such as the Sensible Agent Testbed (Barber et al., 2000), or on JINI (Jini, 2000), such as the Grid (Coabs, 2000), 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* (Decker et al., 1997). Some examples of middle agents reported in the literature include the OAA

[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.

Facilitator (Martin et al., 1999), the RETSINA Matchmaker (Sycara et al., 1998) and the Infosleuth Broker (Perry et al., 1999). 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 (Decker et al., 1997; Wong and Sycara, 2000).

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

It is clear that as the number of MAS created by different groups increases, there will be an increased need for MAS interoperation. 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 .

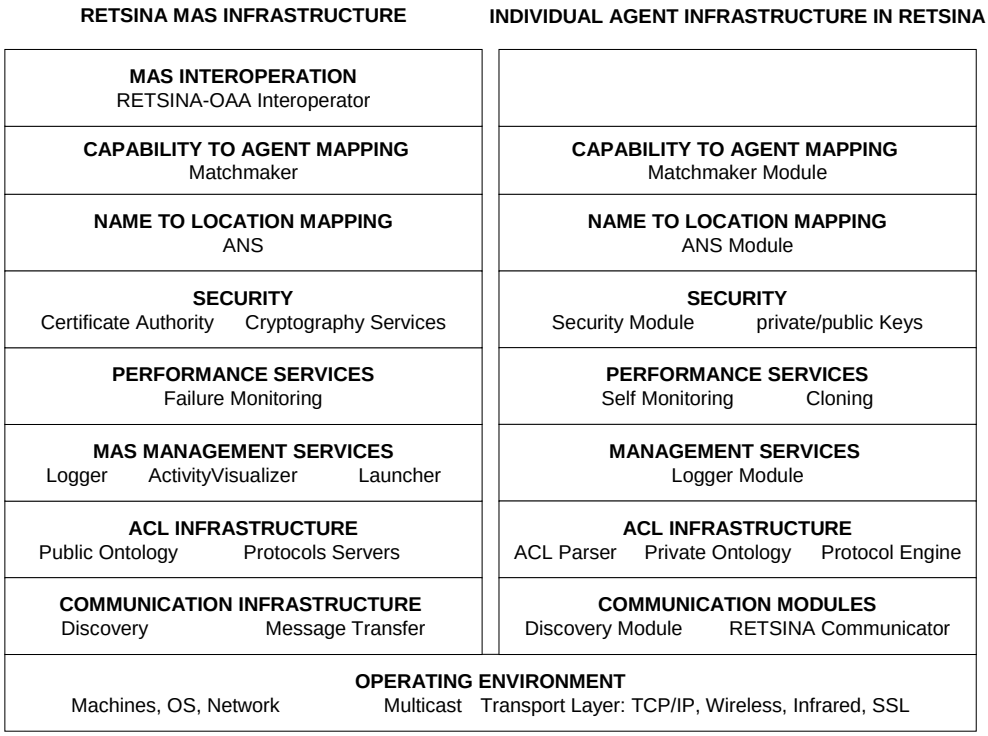

Figure 2. The RETSINA MAS Infrastructure and Individual Agent Infrastructure

We believe this area will receive increased attention, as more MAS get developed and deployed.

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 layer.

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. The agents used have been implemented in different languages 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 (Shehory and Sycara, 2000) that provides an abstraction over the physical transmission layer abstracting over the type of network used. The Communicator supports synchronous as 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.

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 (Suri et al., 2000; Funfrokcen, 1998), 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 (Finin et al., 1995). 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, 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 diverse domain-specific taxonomies of concepts derived from the Wordnet[8] (Fellbaum, 1998).

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 that allows agents to specify their roles and the messages to be exchanged and expected in the context of a protocol. The protocols employ social semantics (Singh, 1998). Currently, the implementation of the protocols is in the form of finite I/O automata.

[8] The Wordnet taxonomy could not be used directly 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 the similarity measurement; finally, we could encode the smaller taxonomies to allow more efficient concepts retrieval and a more precise similarity measurement.

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 work that has to be done by hand to start and maintain a distributed application. 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.

3.5. Performance Services

The current version of the 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 (Smith, 1980).

Some agents implement a self monitoring mechanism (Shehory et al., 1998) 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 (Wong and Sycara, 1999).

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.

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 done 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.

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.

Furthermore, 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, RETSINA uses "reference ANSs" that are visible outside the local subnetwork boundaries. 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 (Jha et al., 1998). 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 (Sycara et al., 1998) 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 (Martin et al., 1999) and Infosleuth's Broker (Perry et al., 1999) 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

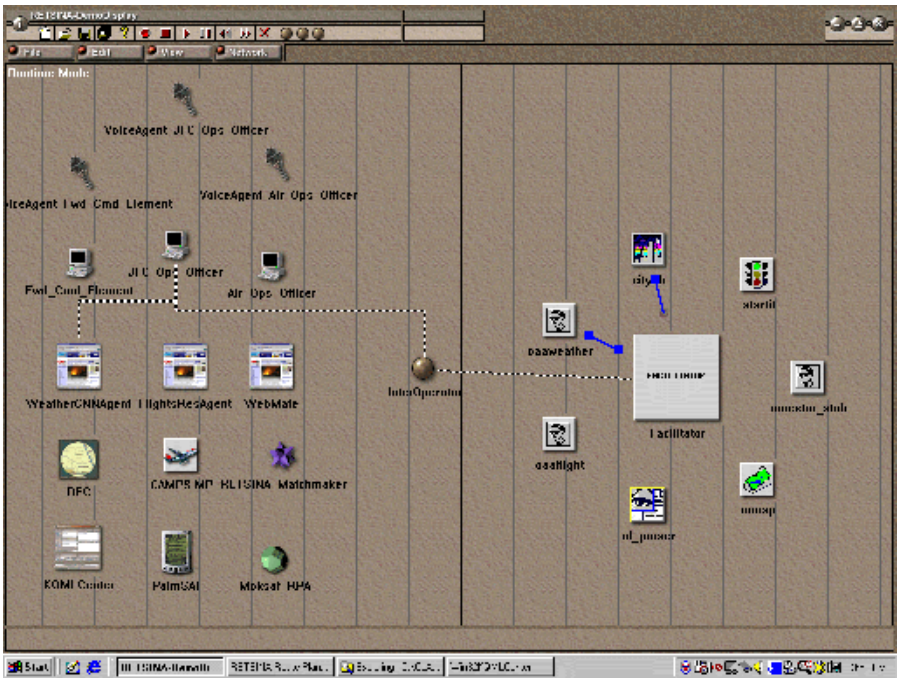

Figure 3. The RETSINA-OAA InterOperator mediates between the RETSINA MAS (on the left) and the OAA MAS (on the right)

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'. When an agent sends a single shot request to a Matchmaker, it gets back the list of providers whose advertisements match the request. When an agent sends a monitor query, besides getting the list of matching agents, it also receives notification as soon as one of the providers exits the system, or new providers enter. Agents use one or the other protocol depending on whether they need just 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 "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 MASs, 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 implemented the RETSINA-OAA InterOperator (Giampapa et al., 2000). 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 "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 financial portfolio management, E-commerce, aircraft maintenance and military logistics. In the following we discuss some of these applications.

[9] Attempts at standardizations such as FIPA(FIPA, 2000) are likely to reduce the problem, but not solve it. Differences will remain in the Ontologies used, the interaction protocols and the MAS architecture.

4.1. Warren: Financial Portfolio Management

The WARREN system (Decker et al., 1996) is an application of RETSINA to the problem of information gathering and 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 are used to gather relevant information about stocks in the portfolio (for example, stock prices, news and company financial reports.) Through the interface agent, the user can buy stocks, sell stocks, monitor the value of her own portfolio and monitor news about the stocks in the portfolio. Two task agents assist the user, the Comptroller and the Risk Critic. The Comptroller records the portfolio and could interact with stock brokers[10] to acquire stocks. The Risk Critic acts as a financial advisor and signals to the user when the acquisition of new stocks in the portfolio or the sale of some stocks modifies the risk associated with the portfolio. Information agents monitor the web to report the value of stocks and their current risk profile for the risk critic. Furthermore, some information agents monitor news casts to find news that can be of interest for the user.

4.2. Coala: E-commerce Auctions

The Coala system (Tsvetovat et al., 2000) is an application of the RETSINA infrastructure to E-commerce auctions. Coala manages collective book purchasing by bundling large groups of buyers into coalitions.

The testbed system consists of a coalition server, an auctioneer agent, a set of supplier agents, and a set of web-based interfaces, one for each end user. The system is based on a pre-negotiation protocol and a variation of sealed-bid reverse auction that allows suppliers to disclose their discount policies to the buyer coalition leader.

The buyer coalition leader uses the WWW interface to initiate reverse auctions with supplier agents. The supplier agents, in turn, decide on a step function for a volume discount schedule and make their bids accordingly to projected sizes of coalitions. When the reverse auction is complete, the coalition leader opens the coalition to new members, which can join the group if they meet the entrance requirements. After the group is formed, the coalition server proceeds to execute the transaction.

[10] Currently, Warren is not connected to a real Internet stock brokering system to perform real stock trading.

4.3. Aircraft Maintenance

Access to information is vital for mechanics doing maintenance on aircraft. Maintenance must be completed under time constraints, and a significant portion of a mechanic's time is spent looking for appropriate information from other mechanics or from paper documentation. Reports must be read and written, information sources queried and consulted, and information must be stored and organized. Not only this takes considerable time, it also results in inconsistent updates, ad hoc handwritten documentation, and lack of access to old but useful information sources. In collaboration with the Robbins Air Force Base in Georgia USA, where all F-15 aircrafts get serviced for maintenance and repair, we have developed RETSINA agents that run on wearable computers for mechanics' decision support during aircraft maintenance (Shehory et al., 1999).

In our agent supported process, a mechanic carries a wearable computer as he completes his maintenance tasks. When he encounters a discrepancy in his inspection, the mechanic fills out a form on his wearable computer. The system analyzes the form and seeks out relevant information from agents. The system then displays the repair recommendations and files the form for future use. The advantages of wearable computers with agents include automatic location and retrieval of information relevant to repairs, utilization of historical repair data, increased efficiency of access to information from manuals, and reduction in average time for repair. The overall result is timely, quality maintenance.

4.4. Logistic Domains

RETSINA has been applied to logistic domains to support multiple users in their collective decision process. In one of such domains, the agents help three decision makers plan an hypothetical evacuation of civilians out of Kuwait City. In this scenario, a US Transportation Officer, a military commander and the US Ambassador in Kuwait should decide what is the safest route out of the city. They are distributed in space: the Ambassador is in Kuwait City, the US transportation officer in some Air Force Base, and the military commander in a US base near Kuwait. Each of them uses an interface agent, called Messenger, to communicate with the others. Each Messenger eavesdrops the conversation of the humans to identify the needs of the decision makers and it anticipates information that could help them in their decision process. Each Messenger uses the MAS infrastructure to identify the agents that monitor the information sources of interest to the decision

makers such as satellites, news feeds, weather reports and intelligence reports.

This scenario was used to test and deploy the InterOperator (described above in section 3.9) between the RETSINA MAS and the OAA MAS. Agents in these two systems exchanged information regardless of the MAS boundary. So for instance, agents on the RETSINA side could gather information about flights out of Kuwait City by querying agents on the OAA side, while agents on the OAA side could gather information about weather and satellite reports from agents in the RETSINA side.

The role of agents in logistic domains is not restricted to information gathering. Agents can also help the decision makers with negotiating a shared plan freeing the decision makers from the burden of dealing with all details of constructing a common plan of action. This is done in another application of the RETSINA MAS, named Agent Storm, in which three tank commanders form a team and navigate across an area littered with mines and where enemies have been seen. Each commander is assisted by a "Mission Agent" to do information gathering, shared planning and monitoring the plan execution. The Mission Agent of one commander negotiates with the Mission Agents of the other commanders a shared plan that takes into account all the information available and the problems that can be foreseen. When the constructed plan is approved by the commanders, the agents monitor the plan execution trying to prevent failures by negotiating changes in the plan.

5. Related Work

In the previous sections 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 example of 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 Figure 1 while we analyze different MAS layer by layer.

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 (Graham and Decker, 2000), Infosleuth (Nodine et al., 1999), Jade (JADE, 2000) among others, use KQML

(Finin et al., 1997) or FIPA ACL (FIPA, 2000) as agent communication languages.

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

Starting 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. This system is in charge of starting agents on different machines in the local network. RETSINA also provides tools monitoring the activity within the MAS and management facilities.

A similar system is used by ZEUS (Nwana et al., 1999) 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 on other agents or by affecting the integrity of the system. Yenta (Foner, 1996) 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 (Greenberg et al., 1998) , 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 into agents, the ANS maps agents to locations. OAA and InfoSleuth (Nodine et al., 1999) implement brokers that map capabilities to agents. This mapping also

contains information on the location of the agents. The first difference between RETSINA and DECAF on one hand and OAA and InfoSleuth on the other is that the first two implement a distributed control in which 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 (Sycara et al., 1998) 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 predicates 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 a classification of the agent in an ontology: it specifies what the agent is about, instead of what the agent does.

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 collaborate, exchange information and services. 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.

References

Arai, S., K. Sycara, and T. R.Payne: 2000, 'Multi-agent Reinforcement Learning for Scheduling Multiple-Goals'. In: *ICMAS2000*.

Barber, K. S., D. N. Lam, C. E. Martin, and R. M. McKay: 2000, 'Sensible Agent Testbed Infrastructure for Experimentation'. In: *Agents 2000: Workshop on Infrastructure for scalable MAS*. Barcelona, Spain.

Castelfranchi, C.: 1998, 'Modelling social action for AI agents.'. *Applied Artificial Intelligence* **103**, 157–182.

Coabs: 2000, 'Grid Web Site'. http://coabs.globalinfotek.com/.

Corba: 2000, 'Corba Web Site'. http://www.corba.org/.

Decker, K., K. Sycara, and M. Williamson: 1997, 'Middle-Agents for the Internet'. In: *Proceedings of IJCAI97*.

Decker, K., K. Sycara, and D. Zeng: 1996, 'Designing a Multi-Agent Portfolio Management System.'. In: *AAAI-96 Workshop on Internet-Based Information Systems*. Portland, OR.

Fellbaum, C.: 1998, *WordNet: An Electronic Lexical Database*. MIT Press.

Finin, T., Y. Labrou, and J. Mayfield: 1995, 'KQML as an agent communication language'. In: J. Bradshaw (ed.): *Software Agents*. MIT Press.

Finin, T., Y. Labrou, and J. Mayfield: 1997, 'KQML as an agent communication language'. In: J. Bradshaw (ed.): *Software Agents*. MIT Press.

FIPA: 2000, 'Foundation For Physical Agents'. http://www.fipa.org/.

Foner, L. N.: 1996, 'A Security Architecture for Multi-Agent Matchmaking'. In: *ICMAS-96*.

Funfrokcen, S.: 1998, 'Transparent migration of Java-based mobile agents: Capturing and reestablishing state of Java programs'. In: *MA98*. Berlin, Germany.

Gasser, L.: 2000, 'MAS Infrastructure Definitions, Needs, and Prospects'. In: *Agents 2000 Workshop on Infrastructure for scalable MAS*. Barcelona, Spain.

Giampapa, J. A., M. Paolucci, and K. Sycara: 2000, 'Agent Interoperation Across Multagent System Boundaries'. In: *Proceedings of Agents 2000*. ACM Press.

Graham, J. R. and K. S. Decker: 2000, 'Towards a Distributed, Environment-Centered Agent Framework'. In: N. Jennings and Y. Lespérance (eds.): *Intelligent Agents VI — Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin.

Greaves, M., H. Holback, and J. Bradshaw: 1999a, 'What Is a Conversation Policy?'. In: *Agents 99: Workshop on Specifying and Implementing Conversation Policies*.

Greaves, M., H. Holmback, and J. M. Bradshaw: 1999b, 'What is a conversation policy?'. In: *In Agents99 Workshop on Specifying and Implementing Conversation Policies*.

Greenberg, M. S., J. C. Byington, and D. G. Harper: 1998, 'Mobile Agents and Security'. *IEEE Communications*.

JADE: 2000, 'Programmer's Guide, June 5th, 2000'. http://sharon.cselt.it/projects/jade/.

Jennings, N., K. Sycara, and M. Wooldridge: 1998, 'A roadmap of agent research and development.'. *Journal of Autonomous Agents and Multi-Agent Systems* **1**(1), 275–306.

Jha, S., P. Chalasani, O. Shehory, and K. Sycara: 1998, 'A formal treatment of distributed matchmaking'. In: *Agents 1998*.

Jini, S.: 2000, 'Jini Web Site'. http://www.sun.com/jini.

Liu, J.-S. and K. Sycara: 1996, 'Multiagent Coordination in Tightly Coupled Task Scheduling'. In: *ICMAS-96*.

Martin, D., A. Cheyer, and D. Moran: 1999, 'The Open Agent Architecture: A Framework for Building Distributed Software Systems.'. *Applied Artificial Intelligence* **13**(1-2), 92–128.

Nodine, M., W. B. amd, and A. Ngu: 1999, 'Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth(tm)'. In: *Proceedings of the 15th International Conference on Data Engineering*.

Nwana, H., D. Ndumu, L. Lee, and J. Collis: 1999, 'ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems'. *Applied Artifical Intelligence Journal* **13**(1), 129–186.

Perry, B., M. Taylor, and A. Unruh: 1999, 'Information Aggregation and Agent Interaction Patterns in InfoSleuth'. In: *cia99*. ACM Press.

Shehory, O. and K. Sycara: 2000, 'The Retsina Communicator'. In: *Agents 2000*. ACM Press.

Shehory, O., K. Sycara, Chalasani, P., and S. Jha: 1998, '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.

Shehory, O., K. Sycara, G. Sukthankar, and V. Mukherjee: 1999, 'Agent aided aircraft maintenance'. In: *Agents-99*.

Singh, M. P.: 1998, 'Agent Communication Languages: Rethinking the Principles'. *IEEE-Computer* **11**.

Smith, I., P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback: 1998, 'Designing Conversation Policies using Joint Intention Theory'. In: *ICMAS98*. IEEE Press.

Smith, R. G.: 1980, 'The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.'. *IEEE Transactions on Computers* **29**(12), 1104–1113.

Suri, N., J. M. Bradshaw, P. T. G. Maggie R. Breedy, G. A. Hill, T. S. M. Renia Jeffers, B. R. Pouliot, and D. S. Smith: 2000, 'NOMADS: toward a strong and safe mobile agent system'. In: *Agents 2000*. ACM Press.

Sycara, K.: 1990, 'Negotiation Planning: An AI Approach'. *European Journal of Operational Research* **46**, 216–234.

Sycara, K., K. Decker, A. Pannu, M. Williamson, and D. Zeng: 1996, 'Distributed Intelligent Agents'. *IEEE Expert, Intelligent Systems and their Applications* **11**(6), 36–45.

Sycara, K., J. Lu, and M. Klusch: 1998, 'Interoperability Among Heterogeneous Software Agents on the Internet'. Technical Report CMU-RI-TR-98-22, School of Computer Science, Carnegie Mellon University.

Sycara, K. and D. Zeng: 1994, 'Towards an Intelligent Electronic Secretary.'. In: *CIKM-94*.

Tambe, M.: 1997, 'Towards Flexible Teamwork'. *Journal of Artifical Intelligence Research* **7**, 83–124.

Thomas, J. D., K. Sycara, and T. R.Payne: 1998, 'Heterogeneity, Stability and Efficiency in Distributed Systems'. In: *ICMAS1998*.

Tsvetovat, M., K. Sycara, Y. Chen, and J. Ying: 2000, 'Customer Coalitions in the Electronic Marketplace'. In: *Proceedings of Workshop on Agent-Mediated Electronic Commerce, Fourth International Conference on Autonomous Agents*.

Wong, H. C. and K. Sycara: 1999, 'Adding Security and Trust to Multi-Agent Systems'. In: *Agents '99 Workshop on Deception, Fraud and Trust in Agent Societies*. Portland, OR.

Wong, H.-C. and K. Sycara: 2000, 'A Taxonomy of Middle-agents for the Internet'. In: *ICMAS'2000*.

Zacharia, G., A. Moukas, and P. Maes: 1999, 'Collaborative Reputation Mechanisms in Online Marketplaces'. In: *HICSS-32*.