DRAFT 24 Aug 1998

Grid Meta Service considerations for the Control of Agent Based Systems

WNxxxxxxx
3 Sep 1998

Alan Piszcz
e-mail: apiszcz@mitre.org

MITRE
McLean, Virginia

Table of Contents

ABSTRACT

The Defense Advanced Research Projects Agency (DARPA) Information Systems Office (ISO) has a new project area named Control of Agent Based Systems (CoABS). One of the goals of this project is to create an agent-based systems infrastructure with collaboration across other research projects to standardize on an infrastructure. Services needed by agent-based systems utilizing this infrastructure include naming, directory (white/yellow pages) and facilitation. Services such as white pages and/or yellow pages to advertise capabilities of an agent system and individual agents. For CoABS the notion of a bootstrap service which allows discovery of agent-based capabilities without using an Agent Communication Language (ACL) may also be an important capability for integration of the agent-based systems with the information grid envisioned for the future. This paper describes an Grid Meta Service (GMS) which provides a potential architecture addressing naming, white/yellow pages and directory services.

1.0 Introduction

A service is needed which will convey the following information among agents: agent capabilities, locations (geographic and logical), agent systems in the grid, pointers and lookup information where agents may be persisted. Retrieval performance should be high for known/indexed attributes, and the service should still allow searches on non-indexed information, at the expense of performance. This service should have the following properties: replication, distributed and lightweight (CPU resources/disk storage) and accessible in multiple languages and ideally based on open standard. Clients should be able to access the service using C, JAVA, LISP, PERL, and shell scripts. The service should allow insert/delete/modify of objects, and additionally provide security for these operations. This object structure should allow simple storage data items (strings) and provide storage of binary attributes. Binary attributes may be used to store small items which may be executable code objects, images, and audio. This service should address a collection of capabilities normally found in services such as: Yellow Pages, White Pages, Broker, Meta-protocols (protocols that agents use), Directory Facilitator and Matchmaker (data source only).

Agent creation in the grid environment entails registering geographic, logical location of owner, Global unique identifier, services or capabilities it provides, ACL, ontology and other attributes. Development of this service would require a lightweight, simple and extensible schema, and basic database query logic support. It should support hundreds of thousands and potentially millions of entries.

The GMS would store, index, replicate and provide access to all directory type information. This resource could be made available to the CoABS community for evaluation and potential use. The ability to access this service should include FIPA ACL, Agent Name Service FIPA, and a non-agent standards based protocol.

This document will define the rational and functionality of a possible solution for the Grid Meta Service. Section descriptions follow:

Section 1.0 Introduction
Section 2.0 Requirements Agent Name Servers, Directory Facilitators
Section 3.0 Requirements Matchmakers, Brokers
Section 4.0 Considerations
Section 5.0 Development and Transition Plan
Section 6.0 Performance and Scalability

2.0 Requirements

In order to address the requirements two functional areas are considered. First, the existing specifications, systems and software were reviewed for directory services. Second, unique attributes have been defined for further discussion and refinement with CoABS. Since there are a number of related services from multiple implementations and specifications, an overview is needed to understand the unique characteristics (if any) for each. This document contains a review of services that map to white pages, yellow pages, directory facilitators or name services, and an overview of matchmaking or broker services will be described. Table 2.0-1 provides a summarization of the sections that follow, it breaks out the topmost level of the Application Program Interface (API) for each service reviewed. Since similarities exist between the agent name services and the Internet Domain Name Server a brief review of its characteristics are provided in this section.

Andrew Sears at the Massachusetts Institute of Technology has proposed similar services for collaboration services. There are parallels between GMS and the Integrated Conferencing

Services [Sears1996] proposes. Sections of his analysis will be included to strengthen certain discoveries of his work and its association to GMS.

White/Yellow Pages, Directory Facilitators		
Organization	Service	Actions
CMU	RETSINA Client	exists lookup register unregister
Dartmouth	D'Agents (none)	NA
FIPA97	Agent Name Service	deregister modify search
IETF	Domain Name Server	nslookup (eg.)
MAF	MAFFinder	lookup_agent lookup_agent_system lookup_place register_agent register_agent_system register_place unregister_agent unregister_agent_system unregister_place
MCC	InfoSleuth Agent Name Server	

Table 2.0-1 Directory Facilitators Overview

2.1 FIPA Directory Facilitator Specification

"The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association registered in Geneva, Switzerland. FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment. This goal is pursued by making available in a timely manner, internationally agreed specifications that maximise interoperability across agent-based applications, services and equipment. This is realised through the open international collaboration of member organisations, which are companies and universities active in the agent field. FIPA intends to make the results of its activities available to all interested parties and to contribute the results of its activities to appropriate formal standards bodies." [FIPA1997]

FIPA describes the following actions for directory facilitation: search, modify, deregister. Agent Management actions specify additional services we will note for potential inclusion at a later date, include register-agent, deregister-agent, modify-agent, authenticate, and forward.

2.2 Object Management Group Mobile Agent Facility (MAF) Specification

The OMG, MAF has a related service for naming agents. Included below is an overview of the MAF Naming Service and MAFFinder.

Background on MAF

"Mobile agents (also called transportable agents) are a relatively new technology that is fueling a new industry. Because the technology and the industry are new, mobile agent systems (for example Crystaliz's MuBot, Dartmouth College's AgentTcl, IBM's Aglets, the Open Group's MOA, GMD FOKUS's JMAF/Magna, and General Magic's Odyssey) differ widely in architecture and implementation. The differences among the mobile agent systems prevent interoperability and rapid proliferation of agent technology, which could in turn impede the growth of the industry. To promote both interoperability and system diversity, we need to standardize some aspects of mobile agent technology. This chapter introduces the terminology we use in this submission. It then assesses what current mobile agent systems have in common in the areas of agent interaction, agent transfer, and security. After we define the common model, we extract a set of features to standardize that will promote interoperability and enhance most existing mobile agent systems. Together, the definitions and the assessment provide a common view of mobile agent technology." [MAF1997]

2.2.1 OMG MAF Naming Service and MAFFinder [MAF1997]

This is section [2.1] from [MAF1997].

"The CORBA Naming Service binds names (represented as strings) to CORBA objects. Applications use this service to publish named objects, or to find an object given only the name. To obtain a reference to a naming service, an application typically bootstraps a reference to a Naming Context using the ORB::resolve_initial_references operation. This MAF submission describes two CORBA object interfaces: MAFAgentSystem and MAFFinder. These objects may be published in the Name Service if the agent system implementor desires. It is not mandatory that the user do this, but it may offer some programming convenience. For example, an agent entering a region may use the Name Service to get a reference to the MAFFinder. Agents that wish to act as CORBA objects may also choose to publish themselves using the Name Service. Doing so gives applications a way to dynamically get object references to remote agents. Using this reference, an application can interact with the agent using CORBA RPC. The MAFFinder interface provides naming and trader services for agents. This interface is defined because CORBA Naming Service alone does not address the following issues of locating a mobile agent:

- Agents may not be CORBA objects
- Tracking down a mobile agent is likely to require intimate knowledge of the agent system implementation. (e.g. knowing the log format of the agent system to trace an agent if the agent system requires an agent to leaves logs)
- mobility of objects is currently not addressed by the CORBA Naming Service Objects such as stationary agents, places, and agent systems, which are stationary by nature, are more likely to manifest themselves as CORBA objects, and can be located by the CORBA Naming Service. Objects such as mobile agents, which are mobile by nature, are better served by the MAFFinder." [MAF1997]

2.2.2 OMG MAF Agent Naming [MAF1997]

This is section 2.4.1 from [MAF1997]. It is important for the destination agent system to be able to identify the principal on whose behalf an agent is acting. This is true even when that principal is not authenticated, because certain applications may find it acceptable to use application-defined heuristics to evaluate authenticity. An agent system can provide the following information to an authorized user about an agent that it is hosting:

- The agent's name (principal and identity)
- Whether or not the principal has been authenticated (authenticity)
- The authenticator (algorithm) used to evaluate the agent's authenticity

CORBA security uses the term principal instead of authority. Secure ORBs exchange security information about principals when remote operations are invoked. This information is available to an application, such as an agent system, as a Credential object. If an ORB does not support security services, however, or a principal is not authenticated, the principal identity information is not available (if the Credential is available, the only identity will be the Public identity). It is necessary for agent systems to exchange principal information when agents are transferred. The information in the Credential, if available, may be used to evaluate the authenticity of the information exchanged. If the Credential is not available, the agents authenticity is automatically false.

2.2.3 OMG MAF Finder Interface [MAF1997]

This is taken from section 3.6 from [MAF1997]. The MAFFinder interface provides methods for maintaining a dynamic name and location database of agents, places, and agent systems. The interface does not dictate what method a client uses to find an agent. Instead, it provides ways to locate agents,
agent systems, and places that supports a wide range of location techniques. There are many possible ways of locating an agent. Here are four possibilities:

- Brute force search

Find every agent system in the region, then send an agent to travel through every agent system to find the agent.

- Logging

Whenever an agent leaves an agent system, it leaves a mark that says where it is going. Therefore, an agent system can always follow the logs to locate that agent. There should also be a way to garbage collect the logs after the agent dies.

- Agent registration

Every agent registers its current location in a database. This database always has the latest information available about an agent's location. Note that registering the new location of an agent does add overhead to the agent go() operation. Therefore, database operations can be a bottleneck.

- Agent advertisement

Register all the stationary places only. An agent's location is registered only when the agent advertises itself. To find a non-advertised agent, the agent system can use a brute force search or logging.

2.3 D'Agents 2.0

"Agent Tcl is a mobile-agent system that is under development at Darmouth College. Agent Tcl uses the flexible scripting language Tcl as its main language but provides an framework for incorporation additional languages. Agent Tcl provides migration and communication primitives that do not require the programmer to explicitly capture state information and that hide the actual transport mechanisms but that are low-level enough to be used as building blocks for a range of protocols." [GRAY1996]

D'Agents does not utilize, define or depend on an agent name service, facilitator information or directory service. Each agent implementation maintains its own internal list of agents and systems to contact.

2.4 CMU RETSINA Agent Name Server

"The Reusable Environment for Task Structured Intelligent Network Agents (RETSINA) approach relies on well-known agents and some basic interactions with them. It uses middle agents such as, matchmakers and Agent Name Servers. Agents request matchmakers for the names of agents that can provide the required service and use Agent Name Servers to send TCP/IP based messages. The RETSINA Agent Message Communication architecture is totally independent of the RETSINA agent system. The RETSINA Agent Name Server is a set of Java programs that allows your software agents to communicate over the internet." [RETSINA1997]

2.5 Microelectronics and Computer Technology Corporation (MCC) Infosleuth Agent Name Server

TBD

2.6 Domain Name Server (DNS)

The DNS capabilities have been in service for over 10 years and provide a reliable, distributed and scalable name service that should be understood when building another naming service. RFC 1034 and 1035 describe the goals and implementation of DNS.

"The goal of domain names is to provide a mechanism for naming resources in such a way that the names are usable in different hosts, networks, protocol families, internets, and administrative organizations. From the user's point of view, domain names are useful as arguments to a local agent, called a resolver, which retrieves information associated with the domain name. Thus a user might ask for the host address or mail information associated with a particular domain name. To enable the user to request a particular type of information, an appropriate query type is passed to the resolver with the domain name. To the user, the domain tree is a single information space; the resolver is responsible for hiding the distribution of data among name servers from the user. From the resolver's point of view, the database that makes up the domain space is distributed among various name servers. Different parts of the domain space are stored in different name servers, although a particular data item will be stored redundantly in two or more name servers. The resolver starts with knowledge of at least one name server. When the resolver processes a user query it asks a known name server for the information; in return, the resolver either receives the desired information or a referral to another name server. Using these referrals, resolvers learn the identities and contents of other name servers. Resolvers are responsible for dealing with the distribution of the domain space and dealing with the effects of name server failure by consulting redundant databases in other servers. Name servers manage two kinds of data. The first kind of data held in sets called zones; each zone is the complete database for a particular "pruned" subtree of the domain space. This data is called authoritative. A name server periodically checks to make sure that its zones are up to date, and if not, obtains a new copy of updated zones from master files stored locally or in another name server. The second kind of data is cached data which was acquired by a local resolver. This data may be incomplete, but improves the performance of the retrieval process when non-local data is repeatedly accessed. Cached data is eventually discarded by a timeout mechanism. This functional structure isolates the problems of user interface, failure recovery, and distribution in the resolvers and isolates the database update and refresh problems in the name servers." [Mockapetris1987]

Andrew Sears offers a review of Dyanmic DNS.
“One possible directory access protocol is the DNS. Changes to the DNS have been suggested that would allow for dynamic updates to the directory. These changes actually are designed to allow users to have write access to their DNS entries, rather than to optimize the directory for dynamic data, as is the case with dynamic LDAP. The goal of these changes is to allow laptop computers to change their Internet Protocol (IP) address as they are relocated. It might be possible to use the DNS as a general-purpose dynamic directory, without changes to the specification. The primary disadvantage of DNS is that it is not based on records with attribute/value pairs. This makes it much more difficult to update single parts of the system. In addition, it limits the extensibility of the system. Even with all the servers it has, DNS's advantage is reduced by the fact that each would require a software upgrade before it could be used for dynamic data. The main problem with DNS is that it was never designed to serve as a general-purpose directory as was LDAP/X.500, and there are currently no plans to use DNS to store white pages information as there is with LDAP. Plans to modify DNS protocols and develop new servers do not yet include optimizing the protocol and server for dynamic data, as is being done for LDAP.” [Sears1996]

2.7 Hypertext Transfer Protocol (HTTP)

HTTP is another possibility for an access protocol to conferencing directories. Its main advantage is its ubiquity, which allows easy viewing of directories by anyone with a browser. The main disadvantage of HTTP is that it was designed as a document access protocol rather than a directory access protocol. It does not include features such as type-specific access to attribute information, nor does it allow simple tasks like server-based searches. Although it provides authentication for reads, authentication for writes requires the setup of individual user accounts for most server designs, which makes implementation difficult. Because HTTP is not designed as a directory access protocol and lacks much of the functionality needed, it cannot be considered an appropriate choice for a conferencing directory access protocol, although it could provide read access to the directory.

3.0 Matchmakers, Brokers and related services

Matchmakers or brokers are services that utilize services like name servers, white and yellow page directory servers. Brokers have the role of determining which service may be satisfy an agent request. Figure 3.0.1 provides an overview list of some existing specifications or implementations for broker and matchmakers. This section is meant to start discussion and thinking about these services and possible inclusion into the Grid Meta Service.

Brokers, Matchmakers		
Organization	Service	Actions
CMU	RETSINA Matchmaker	getAgent getAgents monitor unregister update
FIPA98 TC10	Matchmaker	ADVERTISE PROXY PUBLISH RECOMMEND SUBSCRIBE UNADVERTISE UNSUBSCRIBE
MCC	Infosleuth Broker	
OMG	Trading Object Service 1.0	Admin Link Lookup Proxy Register

Table 3.0-1 Matchmakers, Brokers and related services Overview

3.1 FIPA 10TH Meeting Annex on Brokerage through a MatchMaker

"Intelligent brokerage is an important functionality for FIPA agent environments to share information resources in highly distributed and dynamic environment such as the Internet. In multi-agent environment, a matchmaker facilitates coordination between agents by various communication services. In this document, we shall introduce a matchmaker agent and show how four basic ways of brokerage, *subscribing*, *recommending*, *brokering*, and *recruiting*, introduced in [1] can be realized by a matchmaker with FIPA agent environments. These brokerage ways are well known as basic ways of brokerage within multiple agents and is also useful even for software brokerage through wrapper agents. By defining matchmaker's several actions, FIPA agent community can have these brokerage ways, not only based on current information, but also being able to cope with dynamic changes of a situation." [FIPA1998]

3.2 Object Management Group CORBA Trading Object Service

"The OMG trading object service facilitates the offering and the discovery of instances of services of particular types. A trader is an object that supports the trading object service in a distributed environment. It can be viewed as an object through which other objects can advertise their capabilities and match their needs against advertised capabilities. Advertising a capability or offering a service is called "export." Matching against needs or discovering services is called "import." Export and import facilitate dynamic discovery of, and late binding to, services. To export, an object gives the trader a description of a service and the location of an interface where that service is available. To import, an object asks the trader for a service having certain characteristics. The trader checks against the service descriptions it holds and responds to the importer with the location of the selected service's interface." [CORBATrader1997]

"16.2.1 Exporter

An exporter advertises a service with a trader. An exporter can be the service provider

or it can advertise a service on behalf of another.

16.2.2 Importer

An importer uses a trader to search for services matching some criteria. An importer can be the potential service client or it can import a service on behalf of another.

16.2.3 Service Types

A service type, which represents the information needed to describe a service, is associated with each traded service. It comprises:

- an interface type which defines the computational signature of the service interface, and
- zero or more named property types. Typically these represent behavioral, non-functional, and non-computational aspects that are not captured by the computational signature." [CORBATrader1997]

3.3 CMU RETSINA MatchMaker

"A matchmaker is abstract database of agents and their advertisement. (An advertisement is a description of a service that an agent provides). In other words, matchmaker is a like yellow pages where agents are listed by service (or advertisement) they provide. Once a matchmaker sends the name of the agent that can provide a certain service (in response to the getAgent operation), the matchmaker's job is finished. The actual negotiation and task delegation is handled by the agents themselves. We are in the process of implementing a broker that would allow an agent to submit a query or a task and would find the right agent to perform that task and forward the results to the agent. Hence a broker provides complete anonymity to the agents." [RETSINAMatchmaker1997]

3.4 Microelectronics and Computer Technology Corporation (MCC) Infosleuth Broker

TBD

4.0 Considerations

The following recommendations are meant to invoke conversation early in the project development cycle. These recommendations may need consensus discussion and modification in the CoABS architecture working group.

Naming, and directory services are going to be a core capability of many of the agent based systems used in the grid. The author believes it will be one of the most frequently used services. CoABS will need an early implementation which can grow as developers make future demands on this service. Choosing a defacto-standard for the service should provide the CoABS team the ability to influence research issues while leveraging commercial capabilities in this area.

Current commercial directory servers include Microsoft Active Directory, Novell Network Directory Server (NDS), X.500, Lightweight Directory Access Protocol (LDAP).

If there is an implementation which already addresses all of the requirements, provides source code of the entire implementation, and is freely available to CoABS we need to identify that source as soon as possible. Defining the schema of objects for the services would be the next step.

Documentation of the service with respect to the developers of the systems in CoABS needs to be available early so they can adapt their agent interfaces to the new services.

This service should not interfere any internal service already in use by an agent platform.

This service should allow querying, and control, update, modify, etc., with a non-agent protocol as well as KQML-Lite and FIPA-ACL.

The implementation plan in section 5.0 should be reviewed for potential development and implementation into CoABS.

Figure 4.0-1 indicates that GMS should initially provide directory facilitation (DF) services it also has a close relationship with broker type services. Combining these services may be worth future consideration.

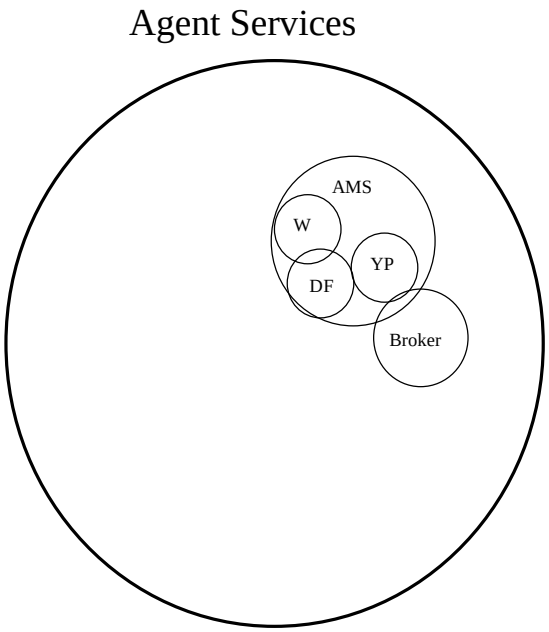

Figure 4.0-1 Grid Meta Service Relationship to Agent Services

4.1 Call for requirements

ID	Type	Description	CoABS Area	Need Date	OS	Comments

5.0 Development and Transition Plan

TBD

5.1 GMS Strawman

The GMS architecture should be extensible and support a progressive development plan. Starting with initial capabilities being available very early in the project. This will allow existing research to begin use and determine future requirements for objects that the service must handle. As future protocols, data models and information becomes mature the service should allow flexible updates

by the individual research teams for the objects they are responsible for. Access through via the Internet may prove essential due the geographical and organizational distribution of the researchers involved.

The core engine being considered for the GMS is the Lightweight Directory Access Protocol (LDAP) [Howes1997], a de-facto Internet standard with multiple Request For Comments (RFC) describing it. It also has a freely distributed implementation from the University of Michigan as well as a commercial product available from Netscape. In the near future other COTS LDAP implementations are likely to be available due to its features, and Internet acceptance.

Figure 5.1-1 shows the highest level interface channels available for accessing the GMS. LDAP will be the standard LDAP protocol available through most languages to learn about system resources. KQML-LITE and FIPA-ACL will be agent compliant interfaces to the same information, this will allow future systems and non-domain platform agents to communicate with the GMS.

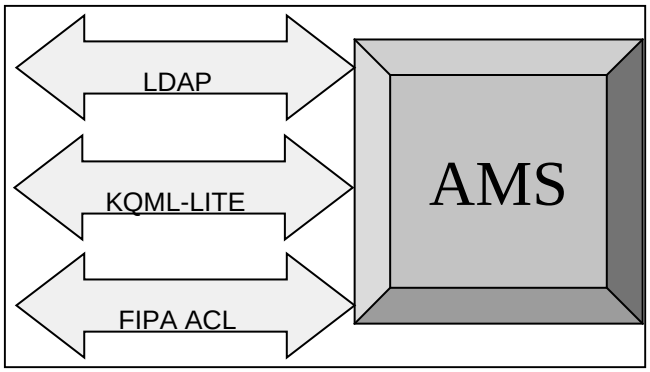

Figure 5-1.1: Grid Meta Services Communication Channels

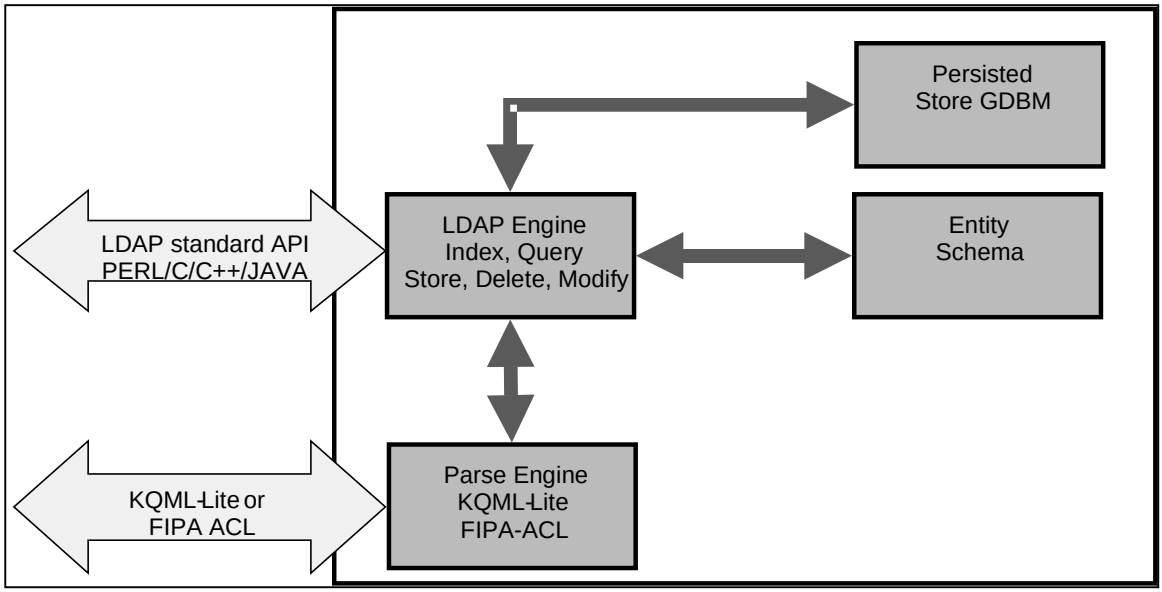

Figure 5.1-2 Grid Meta Services Strawman Architecture

Figure 5-1.2 shows the base components for the strawman vision of the Grid MetaData Service. The LDAPv3 engine to provide entity storage and query, a parse engine for support of the Agent Communication languages. The Entity schema and persistence mechanisms are functions of LDAP.

Figure 5-1.3 illustrates the details of GMS. Shaded portions are components which are available for use. The remaining items are areas of development and further research.

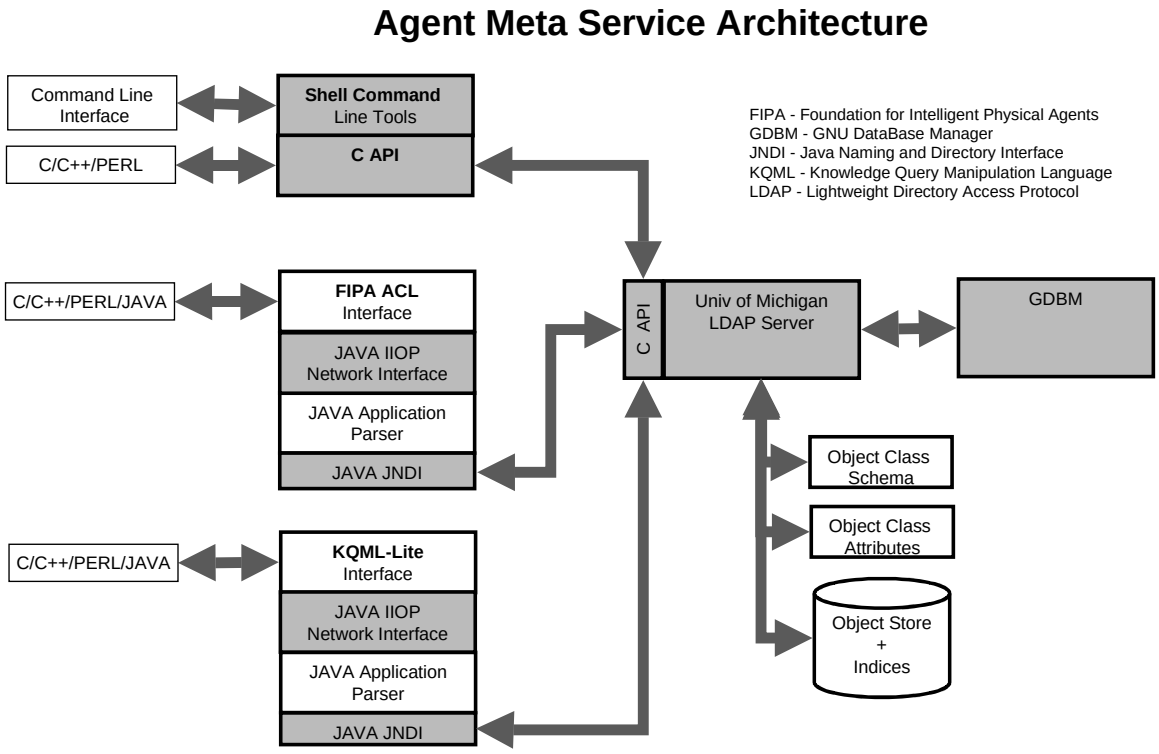

Figure 5-1.3: Grid Meta Services Architecture

5.2 LDAP Characteristics

LDAP characteristics that make it a viable core engine for GMS.

LDAP (Lightweight Directory Access Protocol)

- Open standards: RFC 1558, 1777, 1778, 1779, 1798, 1823, 1959, 2251,2252, 2253,2554, 2255, 2256.
- Multithreaded
- Multi-platform (Source CODE available)
- Supports add/delete/modify
- Availability: Freely (Univ of Michigan), COTS- Netscape, …
- Support multiple back end databases
- Supports basic logic queries
- Replication support built in (master/slave)
- Referral to other servers built in
- X.500 support

- Schema definition support including schema checking
- Hierarchical infrastructure inherent in design
- Should be able to segment services under multiple LDAP domains, (Distinguished Names)
- Provides basic server statistics, # accesses, bytes transferred
- Provides load and dump files in ASCII format for human readability and transport.

LDAP Models: (adapted from page 24 of LDAP [Howes1997])

Information Model: Defines the type of information that can be stored in an LDAP directory.

Naming Model: Defines how information in the LDAP directory can be organized and referenced.

Functional Model: Defines what can be done with the information in an LDAP directory, and how it can be accessed and updated.

Security Model: Defines how the information in an LDAP directory can be protected from unauthorized access for modification.

5.3 GMS Entity Candidate Descriptions

The following items are candidates for CoABS GMS objects. Objects for LDAP are data only items that describe a resource or the characteristics of an agent system.

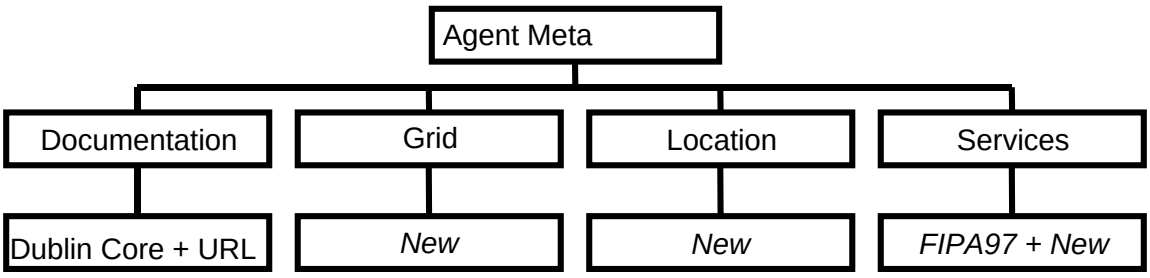

Figure 5.3-1 Grid Meta Services Entities

5.3.1 Documentation, Dublin Core

One candidate for describing documents and creation information is the Dublin Core MetaData Element Set [DublinCore1995]. Additionally the GMS should store locations of the documentation for access.

The metadata elements fall into three groups which roughly indicate the class or scope of information stored in them:
1] elements related mainly to the Content of the resource,
2] elements related mainly to the resource when viewed as Intellectual Property, and
3] elements related mainly to the Instantiation of the resource.

Content	Intellectual Property	Instantiation
Title	Creator	Date
Subject	Publisher	Type
Description	Contributor	Format
Source	Rights	Identifier
Language		
Relation		
Coverage		

5.3.2 Grid Management

This information is TBD. It may be the location for storing information using Grid specific language or syntax which is yet to defined.

5.3.3 Location

Global Unique Identifier {agent|agentSystem}
Logical Creator
Logical Location
Physical Location (x,y,z) of creator
Physical Location (x,y,z) of agent if mobile
CreationTime
Time Location Registered

5.3.4 FIPA Directory Facilitator Attributes [FIPA1997]

agent-name
agent-type
agent-services
interaction-protocols
ontology
agent-address
ownership
df-state {active|retired|suspended}

Somewhere the following attributes should be available for each agent in the system if required. This type of information would be key to building the 'master process equivalent' in UNIX for the Agent Based Grid.

Disk Requirements {transient|persistent}
Memory Requirement
Platform Requirement
Network Requirement
Geographical location (X,Y,Z) (agent|service)
Expected CPU resource required

5.3.5 Service Description

Service Descriptions should identify an agent service and an address for the service.

[FIPA1997] fipa-man-service-description 9.3.3

service-name
service-type
service-ontology
fixed-properties
negotiable-properties
negotiable-properties
communicaiton-properties

other…
Service maps, (name remapping for common services)
Modalities of agent service, (speech, text, vision…)
POC information for agent (who,where,phone,etc)
Pointers or URLs for storage mechanisms (persistence)

5.4 Proposed Directory Information Tree for GMS

The information structure of the GMS is needed in order to access and utilize GMS efficiently. A brief overview of design concepts is given followed by the proposed tree for GMS.

5.4.1 Data Hierarchy Background

The following section is described in an excerpt below taken from the Netscape Directory Server Administration Manual.

"The Netscape Directory Server uses a simple database that responds quickly to high-volume lookup or search operations. Because the database is read much more often than it is written, the database is tuned for this type of access.

Data hierarchy

> The string representation of an entry's location in an LDAP database is known as a distinguished name, or DN.

Data in the directory is arranged in a tree hierarchy. That is, the hierarchy begins at a single point known as the root and branches down to the location of the directory entries.

Because LDAP is intended to be a global directory service, the top of the Directory Server tree is traditionally represented by country name, followed by a series of geographic and physical location information, followed by a common name. For example, a person named Pat Hanson who works in shipping and receiving for A1 Fishing Supplies in the state of Minnesota, US, would be located with the following entry:

Country:
    US
State:
    Minnesota
Organization:
    A1 Fishing Supplies
Organizational unit:
    Shipping and Receiving
Common name:
    Pat Hanson

A different Pat Hanson who works for the same company but in the accounting department would be uniquely represented with the following entry:

Country:
US
State:
Minnesota
Organization:
A1 Fishing Supplies
Organizational unit:
Accounting
Common name:
Pat Hanson

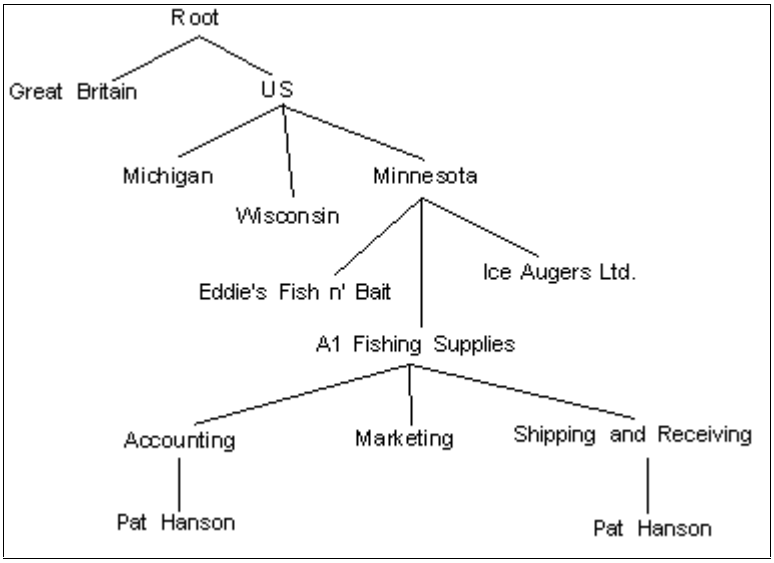

Figure 5.4.1-1 Example LDAP directory hierarchy.

Data model

Because LDAP's data model, known as the schema, is based on the X.500 standard, LDAP databases can contain a virtually unlimited range of information. X.500 is an international standard for the global directory structure. Part of the standard is a definition of the kinds of information that can be included in the directory database.

Types of entries

The type of data that an entry is defined to contain is known as the entry's object class.

Every entry in the directory is defined to be of a certain type, or object class. Some object classes that a directory can hold are as follows:

Organizational Person--an entry representing a person who is employed by or in some way associated with the organization.

Residential Person--an entry representing a person who is in the residential environment (that is, a person who is not a member of the organization).
Organizational Role--an entry representing a position or role within an organization. Examples might be the postmaster, system administrator, or the help desk.
Device--an entry representing a physical unit that can communicate, such as a modem or a disk drive.

5.4.2 Proposed GMS Hierarchy

The following is a candidate hierarchy for GMS. Figure 5.4.2-1 illustrates a potential hierarchy for GMS. The alternate section describes more detail about the top level organization. This information may be added at a later time, this determination can be deferred since GMS is expected to initially run as a proprietary service on its own well known TCP/IP port. If worldwide integration with standard LDAP servers is desired the alternate organizational unit definitions may be needed.

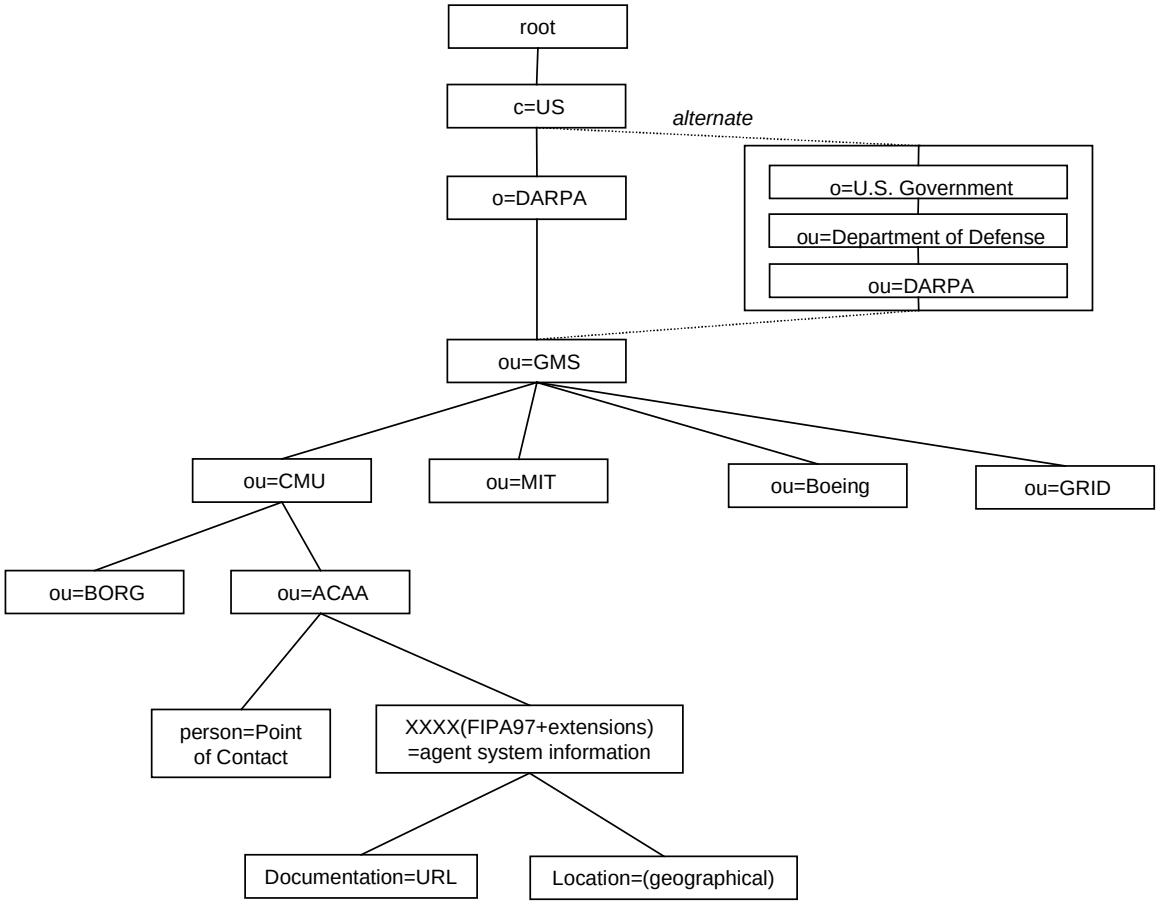

Figure 5.4-2-1 Grid Meta Services Entities

6 Performance and Scalability

Some development efforts defer performance and scalability until the system integration and test phase of the effort. Usual reasons include tradeoffs for functionality vs. performance and scalability. Grid services need metrics to scope the performance envelope as well as define time

phased development plans to address these concerns earlier in the process. One case of performance testing exists and is included for review. The configuration used in the test case presented by Jeff Hodges is not the typical use pattern. For GMS all typically searches will be based on indexed attributes, in cases where new searches are causing performance issues index analysis and implementation would be expected. In case cases where a client needs to perform multiple lookups, a connection (bind) will be performed once, and the query set applied. A significant overhead for any small transaction is the setup and connection time, which can be minimized for multiple queries.

Subject: updated: slapd performance test results
To: ldap@umich.edu
Cc: hodges
From: Jeff.Hodges@stanford.edu
Date: Wed, 10 Jul 96 12:04:36 -0700 (this copy updated 16-Jul-97)

I performed a "let's max slapd out as much as possible" test yesterday, and have added the info to the results below. Also, I'd left out that the dbase technology we're presently using beneath slapd is gdbm-1.7.3.

Bottom line: I got slapd on a 64mb Sparc 2/Solaris 2.4 to run at a sustained rate of about 1,434,240 "bind/search/unbind" (aka "heavyweight) connections per 24 hr period (i.e. about 16.6 hvywhtConn/sec). It was servicing 44 concurrent clients, each querying the directory as quickly as possible (given what was going on on them at the time).

Jeff

Slapd Testing Results Jeff.Hodges@Stanford.edu
Last updated: 16-Jul-97

Contents:

Basic server-side test results:

sustained rate of "heavy weight" query responses generated by slapd: 12..16.6/sec

"heavy weight" queries are composed of...

bind
search
unbind

...protocol operations. Each "bind/search/unbind" sequence comprised one query (and "connection"). Each search was for a known exact match. Search space is described below in

"Test setup". A sustained rate of 12 queries/sec translates to answering 1,036,800 queries per 24 hr period. 16.6/sec translates to about 1,434,240/24hr.

Total high-water mark of queries handled by the test server w/o failure: 4,692,130

(it never failed, btw. We had to take the server down due to moving our machine room layout around, else I would've left it all running for several more days)

Note that the "sustained rate" climbed to the high end of the range as I added more clients, and fell as number of clients decreased.

The "max sustained rate" that this particular slapd server setup can generate seems to be somewhat less than 17 connections/sec (for this particular style of simple unauthenticated bind and simple exact-match search for a known name, and number of clients I was able to rustle up).

With about 22 clients, the server was handling connections at the rate of 16.3/sec. I added 22 more clients for a total of 44 and got a rate of 16.6/sec. I ran out of "easily obtained" clients at that point and so didn't add any more. Note that this rate was sufficient to, in the vast majority of connections, service these clients within the range of their TCP timeouts. I got << 100 timeouts among the 4.5M queries. It is interesting to note that the timeouts were exclusively on "slow" (e.g. other sparc2) machines and/or heavily-loaded ones. I obtained no timeouts on "fast" machines (e.g. Dec Alpha 400/233).

Note that this is a "search" test. Doing a similar test where each entry is modified would be very interesting, especially considering the "dynamic directory service extensions" internet draft.

Test setup:

slapd server config...

slapd 3.3 running on a Sparcstation 2, 64 mb, 1 gig run-of-the-mill disk, Solaris 2.4.

slapd compiled with SunPro compiler, no optimization, but w/ debugging info (-g)

Slapd's default base DN is "o=Stanford University, c=US".

slapd utilized gdbm-1.7.3 as the underlying database technology. gdbm was compiled with the SunPro C compiler with "-O".

slapd was run via the SunPro debugger, with memory access & leak checking OFF.

Slapd and the debugger were the only active processes of note on the machine.

NOTE: slapd had the concurrency patches applies (the same ones I posted to ldap@umich.edu a couple weeks ago)

Clients config...

Clients varied from a DEC Alpha (64mb, Digital Unix 3.2), to a fair pile (39 in total) of Sparcstation 20's (Solaris 2.4).

Clients utilized the "ldapsearch" tool (part of the umich ldap release). It had been compiled with the gnu c compiler with -O2.

The search space was 20,797 unique names. Each client had access to a flat ascii list of all the names in the search space. Each client iterated through the entire list over and over, querying the directory for each name.

There were between 10..44 concurrent clients at any particular time. Each client was querying as fast as it could given load imposed by other processes on the client machine (some were heavily loaded with users, a few had effectively no interactive users and querying slapd was all they were doing) and network delays in contacting the directory server (some clients were on fairly busy nets, others on lightly-loaded nets).

This client-side was driven by a simple perl script running on each client. (see below).

Directory structure...

The directory structure being queried was...

```
c=US
  |
o=Stanford University
  |
ou=People
  |
<~21K individual entries of objectclass=person>
```

The actual ldap query utilized in the test is expressed in the perl script shown below.

Client-side test results:

I kept client-side logs on a few of the client machines during the above test. These logs provide query timing information, system load, # of users, and note timeout occurances.

I processed these logs and obtained the histograms shown below.

During the period the test was running, SlowClient (a fairly-heavily loaded Sparc2) had...

2..13 users
load varied roughly between 3..5

To me, the short answer that the histograms below provide is that a simple directory lookup (note that it is a "heavyweight" query in that it is a bind/query/unbind sequence) will typically cost a direct user (i.e. a human or a program) 1 sec or less to accomplish in around 90% of the cases on a machine with operating characteristics similar to SlowClient. 8% of the time the query might take 1..10 secs. 2% of the time it will take longer. It will very seldom timeout.

And the numbers probably only get better (in general). Take FastClient (see report below). It sez that 99% of the time the query will take 1 sec or less.

SlowClient:

Sparc 20, Solaris 2.4, 64mb, multi-user, 2..13 users during test period, load varied roughly between 3..5

*** Timing Log Analysis ***

```
Analysis of log performed on: Fri Jun 14 12:54:32 1996

First logged timestamp: 13-Jun-1996 08:14:23
Last logged timestamp:  14-Jun-1996 12:53:22

secs    # of occurances      % of total queries
  0          23706                  29
  1          48224                  59
  2           1897                   2
  3            372                   0
  4             56                   0
  5           1428                   1
  6           3336                   4
  7           1337                   1
  8            109                   0
  9             19                   0
 10            109                   0
 11            200                   0
 12             86                   0
 13              3                   0
 14              2                   0
 15              1                   0
 16              3                   0
 17              1                   0
 18            136                   0
 19            116                   0
 20             13                   0
 21              2                   0
 23              8                   0
 24             11                   0
 42              7                   0
 43             11                   0

Total number of queries: 81193

Average response time, to the user, for a query was: 1.40 seconds.

[note that this *simple average* is highly skewed by a few of the
queries, approx 2%, taking longer than 10 sec.]

Number of queries that timed out for whatever reason: 1
```

FastClient:

DEC Alpha 400/233, 64mb, OSF/1 3.2a, single user, load average was consistently less than 1.0 during the test.

*** Timing Log Analysis ***

```
Analysis of log performed on: Fri Jun 14 14:51:06 1996

First logged timestamp: 12-Jun-1996 11:12:59
Last logged timestamp:  14-Jun-1996 14:50:55

secs    # of occurances      % of total queries
  0          84067                 43
  1         104604                 54
  2           2764                  1
  3             25                  0
  4            405                  0
  5            268                  0
  6             49                  0
  7             10                  0
  8              3                  0
  9              8                  0
 10            120                  0
 11             56                  0
 15              1                  0
 16              1                  0
 22              8                  0
 23              2                  0
 25              1                  0
 27              1                  0

Total number of queries: 192393

Average response time, to the user, for a query was: 0.82 seconds.

Number of queries that timed out for whatever reason: 0
```

Client-side perl script:

```
#!/usr/local/bin/perl
#
# QueryForNames.pl
#
# a quick hack. takes a list of names on STDIN, and queries
# the directory via ldapsearch for each name. The filter is "cn=<name>".
# The DN of each entry is printed to stdout, and the query return time
# is noted.
#
# Example usage:
#
#  >  cat  list.of.names  |  QueryForNames.pl  >  query.log  &;  tail  -f
query.log
#

while($name = <STDIN>) {
  chop $name;
  open(LDAP, "/usr/pubsw/bin/ldapsearch -h <host> 'cn=$name' dn |");
```

```
  while($result = <LDAP>) {
    chop $result;
    ($result,$rest) = split(/,/,$result);  # extract just the CN
    $whenItIs = localtime;
    print "$whenItIs;  $result\n";
  }
  close(LDAP);
}

# end of QueryForNames.pl
```

End of Slapd Testing Results

7.0 Acronyms

API	Application Program Interface
CoABS	Control of Agent Based Systems
CORBA	Common Object Request Broker Architecture
CMU	Carnegie Mellon University
FIPA	Foundation for Intelligent Physical Agents
IETF	Internet Engineering Task Force
LDAP	Lightweight Directory Access Protocol
MCC	Microelectronics and Computer Technology Corporation
OMG	Object Management Group
RETSINA	Reusable Environment for Task Structured Intelligent Network Agents
RFC	Request For Comment

8.0 References

[CORBATrader1997] Object Management Group, "CORBAservices Trading Object Service: v1.0", Section 16, March 1997, ftp://www.omg.org/pub/docs/formal/97-12-23.pdf

[DublinCore1998] Stuart Weibel, Jean Godby, Eric Miller, "OCLC/NCSA Metadata Workshop Report", Office of Research, OCLC Online Computer Library Center, Inc., Ron Daniel, Advanced Computing Lab, Los Alamos National Laboratory, 1995, http://purl.oclc.org/metadata/dublin_core/

[FIPA1997] Foundation for Intelligent Physical Agents, SWITZERLAND, 1998, http://drogo.cselt.stet.it/fipa/

[FIPA1998] Foundation for Intelligent Physical Agents, "FIPA97's Developer Guide, Annex 4 Case Study", 10TH Meeting Working Annex, SWITZERLAND, 1998, http://drogo.cselt.stet.it/fipa/spec/fipa98/fipa8717.zip

[Gray1996] Gray. R., "Agent TCL: A flexible and secure mobile-agent system system", Proceedings of Fourth Annual Usenix Tcl/Tk Workshop,

pp. 9-23, 1996. (This document is listed the Overview 2.0 document for D'Agents 2.0)

[Howes1997] Howes T., Smith M, Programming Directory-Enabled Applications with Lightweight Directory Access Protocol, Macmillian Technical Publishing, 1997

[MAF1997] OMG's Common Facility Task Force RFP3, Crystaliz, GMD FOKUS, General Magic, IBM and The Open Group, "Mobile Agent Facility Specification", June 1997, http://www.camb.opengroup.org/RI/MAF/

[Netscape1998] Netscape, Netscape Directory Server Administration Guide, Netscape, 1997, http://help.netscape.com/products/server/directory/1x/UNIX/contents.html

[RETSINA1997] Carnegie Mellon University, "RETSINA Middle Agent Software", http://www.cs.cmu.edu/~softagents/retsina/ans/java/docs/ansHTML/index.html, 1997

[RETSINAMatchmaker1997] Carnegie Mellon University, "RETSINA Middle Agent Software, Matchmaker", 1997, http://www.cs.cmu.edu/~softagents/retsina/ans/java/docs/ansHTML/matchmaker.html

[Mockapetris1987] Mockapetris P., "DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION ", RFC 1035, ISI Network Working Group, 1987

[Sears1996] Sears A., "A Scaleable Directory Schema in LDAP for Integrated Conferencing Services", MIT Internet Telephony Consortium, 1996, http://itel.mit.edu/icons/compare.html, http://itel.mit.edu:/itel/directory.html