ACKNOWLEGEMENTS

I would like to thank my guide, Prof. Sridhar Iyer for his invaluable guidance and encouragement while doing this project.

My thanks are due to my classmates Rahul Jha and Srinath Perur, who provided many valuable suggestions through numerous discussions we had on the subject.

I would also like to thank my parents for their constant loving support.

Vikram Jamwal
IIT Bombay

ABSTRACT

New models for education dissemination have emerged with the growth of distributed systems, especially with the widespread penetration of Internet. This has made possible imparting education on a larger scale. Distance evaluation of students constitutes a crucial factor for success of Distance Education initiatives. Such large distributed systems also raise number of challenges in terms of design, technologies and their implementations. Most of the present day systems have client-server architectures. The client-server model though powerful, has scalability limitations for distance evaluation systems. Over the past few years the mobile agent paradigm, which has emerged as a new approach for structuring distributed applications, attempts to address many of these concerns.

In this project, we survey the existing mobile agent frameworks to understand state of the art. We then use the mobile agent approach for designing, implementing and deploying a system for distance evaluation of students. We consider the entire examination process: (i) paper-setting, where the examiners spread over the internet collaborate to produce a question paper, (ii) examination conduction, where the question papers are distributed and the answer papers are collected, and (iii) answer-paper evaluation, result compilation and publishing. In this report we detail our design, implementation and experimentations. We conclude by presenting our observations and experiences of using mobile agents for designing large distributed systems. We also list some of the challenges that still need to be tackled and indicate the future directions of our work.

TABLE OF CONTENTS

C h a p t e r 1

1 INTRODUCTION

The growth of distributed systems, especially with the widespread penetration of the Internet, has made it possible to impart education on a larger scale. This has resulted in new models for education dissemination. Distance evaluation of students constitutes a crucial factor for success of distance education initiatives.

1.1 Distance evaluation – motivation :

The motivation stems from the following two factors:

- The growth of distributed systems, with Internet being the biggest of them all, has created the possibility of education being imparted on a much larger scale. Many Universities have started their on-line courses along with their regular in house-courses[7]. As these infrastructures evolve in future, there will arise a need for assessing the remote students.
- Consider a system of examination like IIT-JEE (Indian Institute of Technology, Joint Entrance Examination), which is held on a nation-wide scale and is presently paper based. In future one would desire such an examination to be made electronic based. This will help to speed up and better manage the evaluation process.

1.2 Distance evaluation – existing schemes:

Most of the present day Internet based evaluation is web-based[4][6] and employs the *client-server* paradigm. It uses HTML-forms for user interface, with either common gateway interface(CGI)-scripts or java-servlets for back end processing. The questionnaire is downloaded by the students as a web page and the answers are submitted back to the server. This is essentially the *pull-model* of distributing the information. The second Internet based model uses java-applets as the front-end for question paper. This too follows a similar mechanism as the previous case except that using Java gives more flexibility to the examiner in choosing the type of content. With the need for providing multimedia content, multimedia support languages(e.g. flash scripting language) are too being used to provide front-ends. A component based approach, using Java-Beans, in building Internet based evaluation system is described in [5].

1.2.1 Computer based testing (CBT)

CBT has been in vogue for quite sometime now. For example, the Graduate Record Examination(GRE) has started using CBT for its evaluations. This approach presents several advantages[2][3] like provisions for instant scoring, reduced overall test timings etc. and the students can take their examinations throughout the year. Additionally the students are presented with the questions in an adaptive manner i.e. a question is picked from the question bank in a random manner and the next question that is picked from the bank is determined by the correctness of the response to the previous question by the student. Such a scheme can be used for distance evaluation too, incorporating it to existing schemes. But, as the interactions are remote, it has disadvantages in the form of slow response-times.

1.3 Extending existing distance evaluations schemes:

We will now highlight the extensions that are desirable in the distance evaluation systems:

- **Push Model** : In some cases there is a need to send the question paper to the examinee at a time as decided by the examiner. Such a scenario also arises in a case where a number of students are to be evaluated simultaneously for the same set of questions. Most of the paper-based testing methods prevalent today follow this model.

- **Variety of delivered contents** : The use of electronic media for information dissemination has made it possible to present the questions using dynamic content in form of audio, video-clips, or multimedia. It will be desirable to support such rich content in the question-paper.

- **Subjective questions**: The students may be required to provide answers that are objective, written text or involve some graphical schematics. All of these cannot be automatically evaluated and would require manual corrections. The present day on-line systems don't have a provision for these.

- **Off-line examinations**: The paradigm followed in these schemes is client-server and. the students have to remain on-line for the duration of test. For remote interactions, this can be achieved either by opening a socket connection which remains alive during the entire duration of examination, or by opening a socket connection for every request by the client.

- **Adaptive Questions**: It will be desirable to build adaptive tests wherein questions of various level of difficulty are offered to the candidates in dynamic order. This order is determined by the student's response to the previous set of questions.

We believe that it would be extremely difficult to implement the above extensions using traditional client-server technologies. In this project, we explore the use of mobile agents as an alternate implementation mechanism to implement some of the above features.

1.4 Mobile agents

A Mobile Agent (MA) is a program that can autonomously migrate between the various nodes of the network and can perform computations on behalf of the user [20]. Whenever a MA moves from one host to another, both the code and the state of the agent are transferred. Some of the benefits provided by MAs for creating distributed systems include reduction in network load, overcoming network latency and disconnected operations. We shall discuss this technology in detail in the next chapter.

In our case, mobile agents prove to be especially useful because they map and model directly into the real life situations, need a generic execution environment and can work in both - push and pull modes.

1.5 The Proposed Architecture:

We have attempted to design a scheme for implementing the complete examination process. Our scheme consists of following three stages:

1. Examination Setting, where different examiners set the question paper
2. Testing, where question papers are presented to the students
3. Evaluation and Result Compilation, where answers are collected and the results are compiled.

We have attempted to automate most of the above process, simplify the infrastructure requirements at different ends, and provide for the security and reliability of the entire system. We have used the Voyager mobile agent framework [29][30] to implement our design.

1.6 Organization of the report

In Chapter 2 we provide an overview of mobile agent technology and present the literature survey of important present day mobile-agent frameworks. Chapter 3 of this report discusses our proposed framework, Chapter 4 provides the detailed design and some implementation aspects, and in Chapter 5 we discuss the experimentation setup and results. Chapter 6 presents the future directions of our work and concludes our discussion.

Chapter 2

2 SURVEY OF MOBILE AGENT FRAMEWORKS

To realize a mobile agent in practice, we would require support from the underlying distributed network of machines. A developer should be able to write a MA using some programming language, a MA once created needs an execution environment to run, it needs to send and receive messages from its surroundings, it needs resources for storing data, it may need to persistently store itself or migrate to some other node. Its actions need to be controlled, it has a definite life-cycle, it may need to work in collaboration with other MAs and it might be of significance to trace its movements and in some cases it may require protection.

Thus we see the need for a framework that provides mechanisms to support these facilities. Such a framework for supporting mobile agents is called *mobile agent framework (MAF)*. Fig 2-1

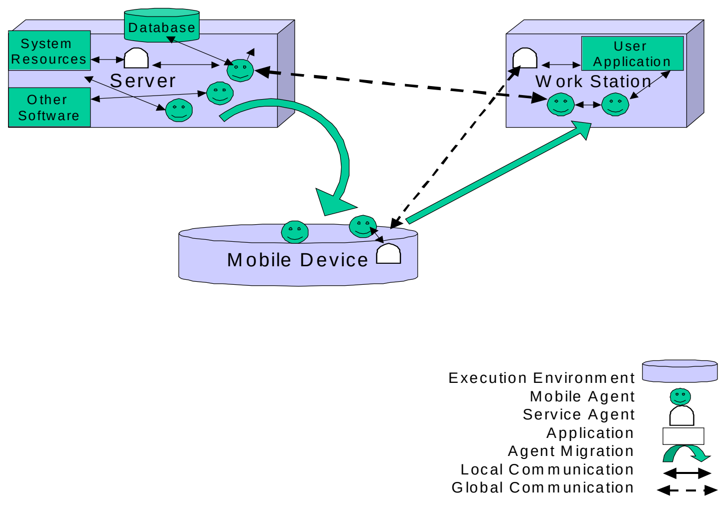


Fig 2 - 1 Different Components of a Mobile Agent Framework

We define following as the main services provided by a mobile agent framework (Green et. al [70] follow a very similar classification)

- **Life Cycle:** Services to create, destroy, start, suspend, stop etc.
- **Navigation:** Services responsible for transporting an agent (with or without state) between two computational entities residing in different locations.
- **Communication:** Communication between agents and between agents and other entities. The naming and the addressing mechanisms followed in the system
- **Security:** Ways in which agents can access network resources, as well as ways of accessing the internals of the agents from the network.

In the previous chapter we discussed the basic concept of mobile agents. In this chapter we shall discuss following aspects of mobile agent technology: (i) benefits of mobile agent technology (ii) application domains (iii) design issues in mobile agent frameworks (iv) mobile agent standardization efforts and (v) brief description of existing mobile agent frameworks

2.1 Benefits of mobile agent technology

Mobile agent technology promises to provide some very distinct advantages [9][19] compared to the other approaches. Some of them are:

2.1.1 Reduced Network Load

For the protocols that rely on heavy interactions, MAs can move to the destination host and carry on the conversations locally. This reduces the traffic on the network. A similar case exists when interactions involve large transfers of data.

2.1.2 Overcoming Network Latency

Because MAs execute locally, they can respond their environments faster. This is key requirement in some critical real-time systems.

2.1.3 Encapsulation of Protocols

Upgrading protocols in a distributed system is a cumbersome task. MAs are able to move to remote hosts and establish 'channels' based on the new or proprietary protocols.

2.1.4 Disconnected Operations

MAs can operate asynchronously and autonomously from the process that created them, after being dispatched. Mobile devices, which need continuous access of fixed network, often suffer from fragile and low bandwidth connects. In such cases they can embed their task in MAs, dispatch them, and then reconnect later to collect these agents.

2.1.5 Other benefits

The other advantages provided by mobile agents are dynamic adaptability, seamless integration, robustness and fault-tolerance.

2.2 Application Domains

Following are some the application domains [9][71] where the mobile agents can provide better solutions:

2.2.1 Distributed Information Retrieval

These applications involve collecting information from sources spread over the network based on some pre-specified criteria. MAs improve efficiency by performing the searches near the information base. This advantage will be more pronounced if the size of the information analyzed is quite huge. Also MAs can keep on carrying their work even during the times when the machines of the creators are not operational.

2.2.2 Electronic Commerce

MAs can represent a user in the network and do work on his behalf. Hence they can perform negotiations on his behalf, do purchases and perform product searches. Their ability to provide real-time responses makes them particularly suitable for these applications. Rahul et.al.[8] classify the existing Mobile Agent applications in e-commerce as Salesman Agents, Auction Agents and Buying agents.

2.2.3 Personal Assistance

Similar to the above application, a user can create an *assistant agent*, which is capable of performing tasks in network even when the user shuts off his machine. Such an agent can interact with other such agents to schedule meetings are perform other messaging or retrieval tasks for or on behalf of the user it represents.

2.2.4 Telecommunication and Networks Services

Advanced telecommunication services like videoconference, video on demand or tele-meeting can benefit from the MAs. Supporting, managing and accounting for these applications require special 'middleware' for dynamic reconfiguration and customization. As an example, for a videoconference, the application service brokers can dispatch components (implemented as MAs), which manage setup, signaling and presentation, to the users.

2.2.5 Workflow Applications

MAs can be used to implement a *workflow item* for they can then carry information as well as the behaviour. Independent of any application that created them, these MAs enable the flow of information by moving through the organization.

2.2.6 Monitoring and Notification

The autonomous and asynchronous nature of MAs enables them to be dispatched and wait for certain events and to report their status. These monitoring MAs can live beyond the lifetime of the processes that created them.

2.2.7 Information dissemination.

Agents can automatically update the software on user machines by carrying the components and deploying them. This relieves the user from botheration of upgrading his software after every new release. They can also be used to disseminate other information like news etc. Such agents in essence follow the *Internet push-model*.

2.2.8 Parallel Processing

The infrastructure for MAs presents an excellent platform for the applications that require heavy computations. This can be achieved by either a set of MAs executing in parallel or a single MA cloning itself whenever the need arises.

2.3 Basic design issues in mobile agent frameworks

A Mobile Agent Framework is an infrastructure that that implements the agent paradigm[20]. The various design issues in designing such frameworks are:

2.3.1 Mobility Model

The fundamental requirement in a Mobile Agent System is its ability to transfer an MA from one host to another. Whenever the migration occurs, the agent is deactivated, its state is captured, and this state is transferred to the new site along with the agent code. On the new site the state is again restored and the agent is reactivated. Depending upon the nature of state transmitted, mobility can be of two types:

- **Strong Mobility**: If both the data and the execution state (execution context + call stack) of the agent are transmitted, it is the case of strong mobility. The destination server can restart the execution of agent precisely from the point where the execution was stopped on the originating host. This kind of mobility is suitable for the applications like transparent load-balancing, where the processes (in form of MAs) can migrate across the servers.

- **Weak Mobility**: In this case the state of the agent is captured at a higher level, i.e. only the state of the application level data variables is gathered. This captures the execution state only at function-level in contrast the earlier case where the execution state is captured at instruction level. Since the mobile-agents are under the direct programmer's control, the kind of mobility is sufficient for most of the applications.

Java Virtual Machine (JVM) does not allow thread-level state capture. Since most of the frameworks use JVM, they only support weak mobility.

2.3.2 Code Shipping

The code of the mobile agent needs to be present at the destination host for its successful restart. This code can either be

- **carried by the agent**, in which case, the agent can migrate to any host providing the execution environment at the destination host

- **pre-installed on the destination host** This is better for security reasons as no foreign code is allowed, but this restricts the use of MAs only to pre-defined set of machines

- a**vailable on a code-base server**, from where it can be downloaded on-demand.

2.3.3 Agent Naming and Addressing Mechanisms

Agents need to be named and located to enable inter-agent communications or remote agent management. Naming of the agent can be location dependent or independent.

Different agents running parallel on different hosts may need to exchange temporary results and synchronize. One of the convenient ways of doing it is for agents on different hosts to use a common shared naming server. The other alternative is to locate the agent from the host from which it originated and which is also keeping the logging information about the current location of such agent.

Naming Servers provide location transparency for agents. Yariv and Mitsuru[77] discuss various schemes for locating mobile agents and delivery of messages between them.

- **Brute Force :** Agent is located by searching it in multiple destinations. Searching can be parallel or sequential.

- **Logging :** An agent is located by following it trial information, indicating its next destination, left in every agent server it already visited. Trail information for the disposed agents can be garbage-collected according to, for example, the expired time or explicit notification by agents

- **Registration :** An agent updates its location in a predefined directory server that allows agent to be registered, unregistered or located. Other agents use the directory to locate the agent. In practice, communicating agents need to agree in advance upon a naming server. Such agreement can be simplified by adopting an architecture in which every agent server is associated with one available naming server.

2.3.4 Agent tracking and message delivery

Similar to the above case, agents sometimes need to be tracked for sending them messages or controlling them remotely. There are two basic methods:

- **Locate-and-Transfer :** An agent is located after which the message is transferred directly to it; in this case two separate phases are used.

- **Forwarding :** Locating a receiver agent and delivery of message to it are both done in a single phase e.g. the message may be redirected by using trail information

There are two main differences between these methods of message delivery. Locate-and-Transfer may not always give the locations of agent accurately, since they may be dispatched during the second phase of the message transfer. With forwarding such cases can be eliminated, since agents are located *on-the-fly* during delivery of messages to them. Secondly, forwarding may be more efficient than Locate-and-Transfer in presence of small messages. Otherwise it might be more efficient to locate an agent and then transfer a large message directly to it.

2.3.5 Agent Communications

Agents don't exist in isolation. They need to interact with execution environments (EEs), resources/ objects, other agents or users to achieve their goals. The communication mechanisms are characterized by: (i) type of interactions, (ii)the type of mechanisms and (iii)the cardinality of the communicating partners. We discuss these below.

2.3.5.1 Types of Interactions

A MA during its life-cycle will need to interact with their EEs, resources, other agents and the users.

- **MA/ EE interaction** The MA needs to use the services provided by the EE like file-services, directory services, transport services or any other services supported by the EE. Also the EE needs to interact with MA to control and guide its movements and satisfy of check it needs. As these interactions are between a fixed entity (EE) and a roving entity-MA. Most of the interactions follow the client-server (CS) model and follow direct-method invocations.

- **MA/ MA interaction**: As two moving agents can be from different origination environments, the communication mechanisms have to be of a varied kind. The communicating MAs form peer-to-peer pattern. This forms the basis of agent collaboration.

- **MA/ User Interaction** Sometimes the agents are acting on behalf of a user and need to take instruction or report back results to the users. The interaction is usually through an GUI to the user and will include all the details of human-computer interaction.

2.3.5.2 Types of Communication Mechanisms

A MA needing to interact with its surroundings environment or other agents will use mechanisms, which are synchronous, or asynchronous. The communication partner can be either addressed directly (RPC, Streams, Message Passing) or indirectly/ anonymously (events, black-boards, tuple-spaces, synchronization objects) and all these mechanisms can be either local or remote. The different mechanisms can be described as:

- **Method Invocation** It involves an object/ agent calling the method of another object/ agent and communicating by means of passing parameters and accepting a return value. Although synchronous, asynchronous, and deferred synchronous are possible, yet it is most suitable in case of synchronous communication. It is achieved by direct reference to the method (in case the invoked object exists in the same address space) or LPC (local procedure call) and RPC (Remote Procedure Call) depending upon the local or remote presence of the invoked object.

- **Message Passing** In this case the communication takes place by passing a message to the other agent/ object. The message is passed by invoking a well-know method of the object, in asynchronous manner. The message encapsulates the protocol, which is then parsed and interpreted by the receiving object.

- **Black Board** Black board interactions occur via shared-data-spaces, which are local to each hosting EEs into which the agents store and retrieve messages. There is a need for a common message format/ identifier understood by each agent to exchange information via a blackboard. The messages need not be aware of the location of the agent or the time when the agent is going to read the message. This leads to temporal uncoupling, a desirable feature as in most applications

- **Tuple Spaces** These are the extension of blackboard model where the information is stored in tuple-space and is retrieved by associative (or pattern-matching) mechanisms.

- **Streams** The communication takes place by opening a stream connection between the two entities. In many cases this is done by opening socket connections.

2.3.5.3 Other communication features

Additionally the communication structure might provide support for the following:

- **Events Handling**. Providing an event channel helps in decoupling the system and making it more flexible and powerful

- **Group Communications** It is sometimes desired (or required) to treat a group of mobile agents in a similar manner and to address them singularly. The messages then can be classified as either unicast, broadcast, multicast or anycast depending upon whether they are meant for a single, all or a set of agents, or any one agent in a group respectively.

2.3.6 Security issues

Security is an important consideration in an open network like the Internet. It is important to safeguard both the execution environment as well as the mobile agents from any undesirable effects. The different security issues relevant to mobile agents are described in the table below.

Attacked	**Attacker**	**Attack**
Host	arriving agent	- access and corrupt the host's local files, resources - stop the server in a denial of service attack.
Host	external third party	send a huge number of agents to the host to tie up all the resources, or even crash the host
Agent	new host	access private information, e.g. a credit card number, a password, etc, for later use, or replay
Agent	another agent	access private information, or to crash the agent to stop it fulfilling its task
Agent	third party	alter exchanged messages for its own benefit, e.g. to recommend their host instead of another, or to reveal content of agent
Network	incoming agent	flood the network with copies of itself

Table 2.1 Security Issues in Mobile Agent Frameworks [78]

Mobile agent systems have to provide different kind of security mechanisms to detect and guard against these attacks. These [20] are privacy and integrity mechanisms (to protect agent code and private data), authentication mechanisms (to confirm identities of communicating parties), and authorization mechanisms (to allow agent to access server resources in a controlled manner). It may be noted that first one is most difficult to ensure and is still an unsolved research problem [74].

Some efforts have been put into the development of techniques that help *proving and detecting* the tampering done by servers. Having such techniques in place, the results of the agent can be omitted if agent code is tampered by the server. Some of the suggested approaches are, Hardware Solutions (requires the presence of a special hardware component, whose internal architecture is unknown to the public), Code Obfuscation [76], Clueless agents [73] Tracing of Execution [75].

2.4 Standardization Efforts

Mobile agent standardizations efforts have been influenced by two forums: MASIF [80] and FIPA [79]

2.4.1 Mobile Agent System Interoperability Facility (MASIF)

The mobile agent systems differ widely in architecture and implementation, thereby impending interoperability, rapid proliferation of agent technology, and growth of industry. To promote interoperability and system diversity some aspects of mobile agent technology must be standardized. MASIF is a collection of collection of definition of interfaces that provides an interoperable interface for mobile agent systems. MASIF specifies two interfaces MAFAgentSystem(for agent transfer and management) and MAFFinder(for naming and locating). MASIF is about interoperability between agent systems written in the same language expected to go through revisions. Language interoperability for active objects is difficult, and is not addressed by MASIF. Furthermore, MASIF does not standardize local agent operations such as agent interpretation, serialization/ de-serialization, and execution. In order to address interoperability concerns, the interfaces have been defined at agent system rather than the agent level. MASIF standardizes:

- **Agent Management** : One can envision a system administrator managing agent systems of different types via standard operations in a standard way: create an agent, suspend it, resume it, and terminate. It allows agent systems to control agents of other agent system. Management is addressed by interfaces for suspending, resuming, and terminating agents. Agent Transfer : It is desirable that the agent applications can freely move among agent systems of different types, resulting in a common infrastructure, and a large base of available system agents can visit.

- **Agent and Agent System Names** : Standardized syntax and semantics of agent and agent system names allow agent systems and agents to identify each other, as well as clients to identify agents and agent systems. The CORBA services are designed for static objects, CORBA naming services applied to mobile agents may not handle all cases well. Therefore MASIF defines a MAFFinder interface as a naming service.

- **Agent System Type and Location Syntax** : The agent transfer cannot happen unless the agent system type can support the agent. The location syntax is standardized so that the agent system can locate each other.

The MASIF in its current form provides the features required for the first level of interoperability, which is transport of agent information where the information format is standardized. Once the information is transferred from one system to another, how the agent system deals with the parameters internally is an implementation matter and not addressed by MASIF standard. Such information includes agent profile, which describes the language, serialization, and other requirements the agent has on the current agent system. MASIF makes it possible for an agent system to understand the requirements the agent has on its system, and it is first step in end to end interopearability.

2.4.2 Foundation For Intelligent Physical Agents (FIPA)

FIPA is a standardization effort for a complete architecture for supporting intelligent agents. The FIPA architecture consists of the following concepts and agents: Agents. Agent Platform (AP). Directory Facilitator (DF). Agent Management System (AMS). Agent Communication Channel (ACC). Agent Communication Language (ACL). FIPA has demonstrated several applications implemented using their architecture and it seems as if FIPA could be an accepted standard for agents. Example applications include personal travel assistance, personal assistant, audio/ video entertainment, and network management.

Agent Management Support for Mobility

This specification represents a normative framework for supporting software agent mobility using the FIPA agent platform. This specification is concerned with specifying the minimum requirements and technologies to allow agents to take advantage of mobility. This specification integrates closely with other FIPA specifications (especially Agent Management and Agent Security) and provides a wrapping mechanism for existing mobile agent systems to promote interoperability. Table below illustrates some of the FIPA specification features.

FIPA **DOES NOT**	FIPA **DOES**
mandate the use of mobility features	mandates how agents and APs may support mobility, if mobility is desired
mandate the use of any explicit technology for supporting mobility	it provides a wrapping mechanism for mobile agent systems
define how mobile agents and mobile agent systems operate or are implemented	however, mobility capabilities defined in this specification rely on their existence
define mobile agent security	expected in future versions

Table 2.2 FIPA mobility features

This specification defines extensions that are necessary to the AMS to support mobility. The platform profile can become a standard way for an agent to discover the mobility supported by an AP. If an AP does not support mobility, then it will refuse any mobility operation.

2.5 Mobile agent frameworks considered for study

We studied several mobile agent frameworks. In this section, we present some of the more well known mobile agent frameworks.

2.5.1 Aglets

The Aglets Software Development Kit (Aglets SDK) is the product of IBM's Research Institute in Japan [9][10][11][12][13][14]. It is also one of the pioneer mobile platforms. Aglets is a general-purpose mobile-agent platform. Recently its source-code also has been made available to the developers.

Aglets is a Java-based system in which aglets (agents) migrate between agent servers (aglet contexts). Aglets, have defined an elaborate security plan but only a limited version of this is supported.

2.5.2 Concordia

The Concordia platform [22][23] is a commercial system, developed at the Horizon Systems Laboratory of Mitsubishi Electric Information Technology Center, America. It is available for evaluation. Concordia is a framework for development and management of network-efficient mobile agent applications for accessing information anytime, anywhere and on any device supporting Java. Concordia has extensive support for agent communication, providing for asynchronous event handling as well as specialized group collaboration mechanism. It also addresses fault tolerance requirements and checkpoints for recovery, whereby it enables reliable agent transfers.

2.5.3 D'Agent

D'Agents [24,25] (formerly AgentTcl) is an experimental system being developed at Dartmouth College, USA. It is free for non-commercial use. AgentTcl was one of the first mobile agent systems. It was built on top of the Tcl language. It received worldwide attention for its support for strong mobility and for its promises to implement all security aspects of mobile code. These promises, however, have not been kept. Instead, its developers decided to shift toward multi-language support and changed its name to D'Agents. The D'Agents system, just like its ancestor AgentTcl is a general-purpose mobile agent platform, without any specified application focus.

2.5.4 Grasshopper

The Grasshopper system [27] is another commercial product. It was developed by Forschungsinstitut für offene Kommunikationsysteme (IKV++), Germany. A light edition (max. 5 agents and 2 agencies) is available for evaluation. Grasshopper is a relatively new system and one of the first platforms implementing MASIF support. Its application focus is on telecommunication applications.

2.5.5 Mole

The Mole platform [28] is another experimental system. It was developed at the University of Stuttgart, Germany and its source-code is available. The Mole platform has a relatively long history. It is also a general-purpose mobile agent platform, without expressed focus on any application area.

2.5.6 Voyager

Voyager [29][30] is a commercial product of the ObjectSpace Inc, USA. It is a general-purpose distributed middleware that is claimed to be used at more than 10,000 companies word-wide.

ObjectSpace Inc. does not advertise its system as a mobile system (that supports CORBA), but as an ORB that has mobility support. Voyager is a modular system, including security solutions, administration tools, transaction services, etc. Most of these modules, however, are only available in the commercial package.

We use the following parameters [1] for comparing above frameworks:

- Project details, supported platforms and languages, and implemented standards (See Table 2.3)

[1] http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/mal.html

- Types of migrations, agent naming and agent tracking mechanisms, and resource access control mechanisms (See Table 2.4)

- Communication mechanisms (See Table 2.5)

Table 2.3 Project details and platform / language / standards supported by mobile agent systems

Sno.	System Name	Organization	Supported Platforms	Supported Languages	Implemented Standards	Project URL
1	**Aglets**	IBM Tokyo Research	JDK 1.1.x on Win32, OS/2 Warp Version 3 and 4, AIX 4.x,Solaris for SPARC, and Solaris for x86MRJ SDK 2.0.1 on MacOS 8.x	Java 1.1	None (Interfaces for MASIF are used internally but currently not compliant to MASIF)	*http://www.trl.ibm.co.jp/aglets/*
2	**Concordia**	Mitsubishi Electric ITA	Win32, Solaris, Linux, HP/UX, AIX.	Java 1.1	Not yet, but proposed. MASIF and FIPA	*http://www.meitca.com/HSL/Projects/Concordia/*
3	**D'Agents**	Dartmouth College	Unix (nearly all variants)	Tcl, Java, Scheme	None	*http://agent.cs.dartmouth. edu/*
4	**Grasshopper**	IKV++ GmbH	Tested on: Windows NT/9x, Solaris Should run on all platforms supporting JDK 1.1 and higher.	Java 1.1	MASIF, FIPA (add on module)	*http://www.ikv.de/poducts/gra shopper/*
5	**Mole**	University of Stuttgart, IPVR	execution: all platforms supporting Java JDK 1.1development: additional make support required (experimental Java make under development)	Java 1.1	None	*http://www.informatik.unituttg rt.de/ipvr/vs/projekte/mole.html*
6	**Voyager**	ObjectSpace, Inc.	Certified 100% Pure Java	Java 1.1. and 1.2	None	*http://www.objectspace. com/products/vgrORBpro.htm*

Table 2. 4 Migration, Agent Tracking and Access Control features of mobile agent frameworks

Sno.	**System Name**	**Migration Weak/ Strong**	**Code Shippment**	**Agent Tracking**	**Directory of Services**	**Resource Access Control Mechanism**
1	**Aglets**	Weak	Necessary classes are archived and transferred to the receiver. A Jar file is supported Other classes are transferred on demand from code base server.	Logging facility is provided on the aglet Tahiti. Server.	A Finder as an experimental feature	-simple privilege configurable preferences like Java 1.1 sandbox model; trusted aglets and untrusted aglets. -fine-grained access control with security policy file like Java2. - Permission classes -access protection by each aglet - Individual aglet can set its own protection againt messages from other aglets. - Server authentication.
2	**Concordia**	Weak (but with multiple method entry points via Itinerary).	All are supported - on demand from sending host, - on demand from code server, - all classes as a whole from sending host, - all classes as a whole from code server.	Home register via mobile agent debugger	Global and local - using a string identifier. -Global directory maintained by optional Directory Manager service	- Server configured access control list. -Privileges are granted based on the identity of the user who launched the agent

3	**D'Agents**	strong (Tcl and Java), weak (Scheme)	All classes as a whole from sending host	Name service (if agent chooses to use it)	Global	Configurable policies
4	**Grasshopper**	Weak	On demand from sending host, on demand from code server	Region registry	- global for all agencies - registered to the region registry - additionally local within an agency	Configurable policies
5	**Mole**	Weak	All classes as a whole from code server	None	Local using a string identifier	None
6	**Voyager**	Weak	Flexible resource loading	Federated naming service	Federated naming service	Configurable policies

Table 2. 5 Communication Mechanisms in MA Frameworks

Sno.	**System**	**Local**	**Global**	**Addressing**
1	**Aglets**	Supports messaging. A message is an object The hook method of the receiver will be invoked with the sent message. The system defined message is also provided which is used for moving, cloning, storing and retrieving into/ from secondary storage and terminating	Proxy objects are used for the communication with an aglet. All messages are sent to the 0proxies. Proxies forward messages to the remote moved aglets The aglet server, which hosts the aglet, retrieves messages and converts system messages into system events. Security mechanisms control message passing..	A proxy object (AgletProxy) is used as a partner of a communication. And every aglet has its own identifier (AgletID). An AgletID object can be converted into an AgletProxy object in an aglet

2	**Concordia**	Distributed Events and Agent Collaboration. Data format is arbitrary object which subclasses from Concordia base class.	Distributed Events and Agent Collaboration. Data format is arbitrary object which subclasses from Concordia base class.	Publish-subscribe type model. Event may be sent directly to agent via its unique Agent ID. Also supports group-oriented events
3	**D'Agents**	Message passing (arbitrary strings, soon to be arbitrary binary data)	Message passing (arbitrary strings, soon to be arbitrary binary data)	By machine name plus unique (per-machine) integer id; Directory services provide location-independent addressing
4	**Grasshopper**	asynchronous/ synchronous messages (any java object)	asynchronous/ synchronous messages (any java object)	Combination of host name, agency name and place name
5	**Mole**	(asynchronous) messages (any java object) (synchronous) rpcs (any java object) session mechanism	(asynchronous) messages (any java object) (synchronous) rpcs (any java object) session mechanism	via the name of the agent plus name of system node via bearing a (String) badge
6	**Voyager**	local method invocation and local invocation through a remote proxy	Voyager Remote Messaging Protocol (VRMP), RMI, CORBA IIOP, DCOM	extended URLs, CORBA IORs

3 PROPOSED FRAMEWORK

We divide the examination process into three stages: (i) examination setting, (ii) distribution and testing, and (iii) evaluation and result compilation

3.1 Examination Setting

The examination setting process (Fig 3.1) takes place in a collaborative manner where the examiners sitting at different remote locations prepare their questions. Mobile Agents are then dispatched to these examiners. These MAs fetch the question papers from all of the examiners. The central controlling authority decides on the final question paper based on the inputs from different examiners.

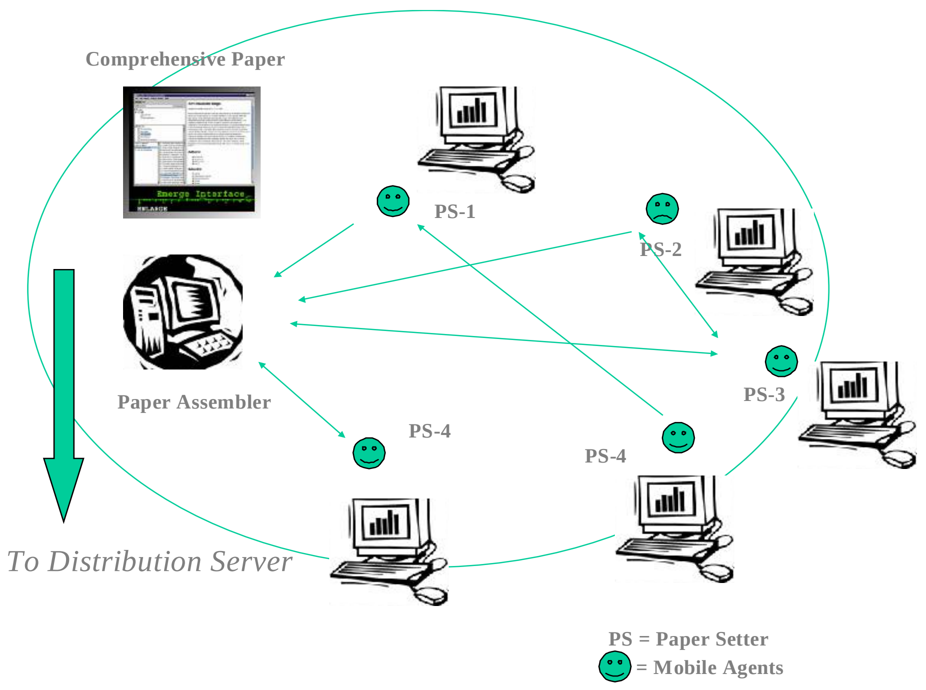


Fig 3 - 1 Examination Setting Stage

3.2 Distribution and Testing

Once a question paper is prepared, it is dispatched to the different examination centers with the help of Courier Mobile Agents (Fig 3-2). Having finished their distribution work, the Courier Agents get either terminated or they return to their place of origin. The distribution servers at these centers have a list of candidates enrolled for that center. The examination paper at each center is cloned to the number of students in each center. The examination papers can time-out themselves after a fixed interval of time. Once a student finishes answering a question or the examination paper times out, the answers are given back to the distribution center, which launches a Answer Mobile Agent for each student answer paper. These Mobile Agents then make their way to the Evaluation Center

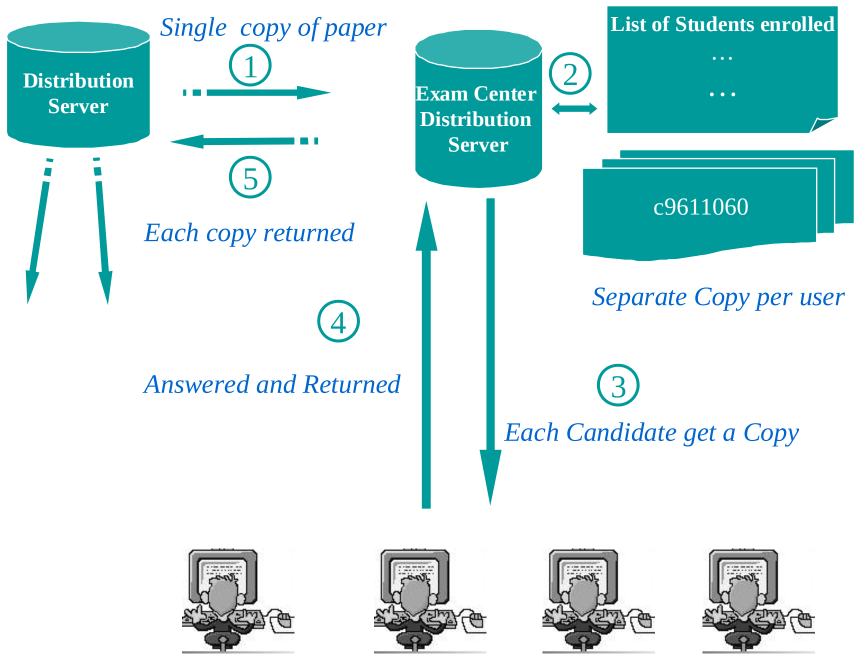


Fig 3 - 2 Distribution and Setting Stage

3.3 Evaluation and Result Compilation

Once an Answer Agent reaches the evaluation center, it is supplied with an itinerary of the examiners. The Answer Agents can also move to an Objective Question Evaluator if it possesses answers to multiple-choice questions, to automatically evaluate their answers. The Answer Agents move from one examiner to other, until all of the questions are evaluated. They then move to the Publishing Center where they supply their results and where the final comprehensive results are published.. (See Fig 3-3)

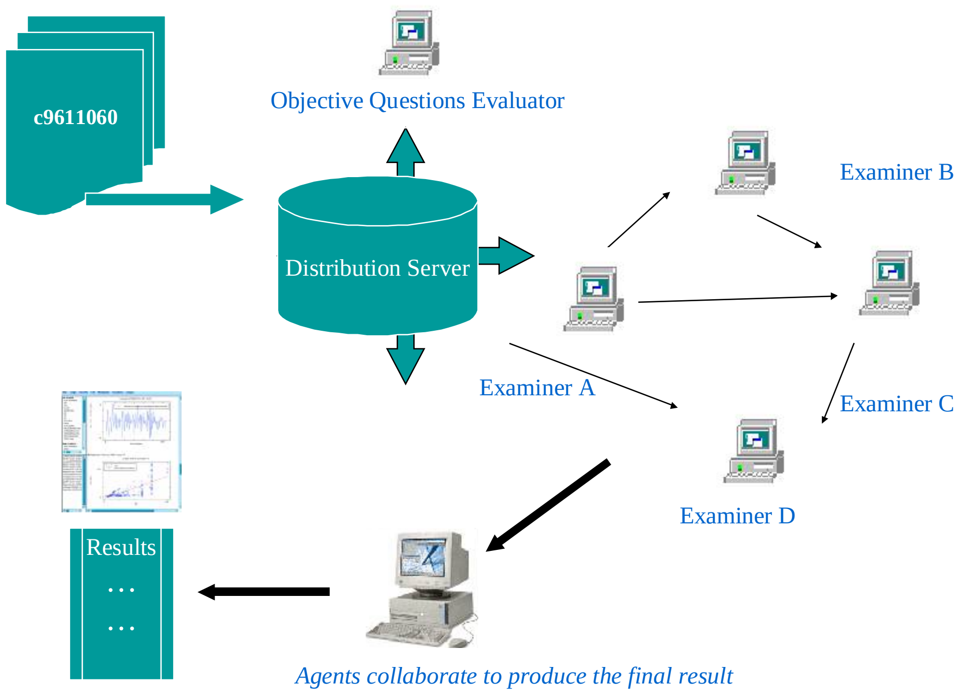


Fig 3 - 3 Evaluation and Result Compilation

3.4 Voyager: Our chosen MA platform

We have already described the Voyager framework in Section 2.6. Choosing Voyager as our application development platform was mainly influenced by the factors listed below:

- The results on performance comparison of mobile agent framework, as seen from a parallel study at IIT Bombay [72], indicate that Voyager ORB performs better for remote messaging.

- Unlike many other platforms, Voyager ORB is a generalized platform for distributed object computing. The MAs are treated as any other distributed object, with special primitives for mobile agent behavior[2]. This allows easy integration of MAs with the rest of the application structure.

- Voyager was compatible with the latest version of Java (jdk1.2) available at the time of development of application, while many systems like Aglets, Mole etc. were still not ready with the new compatible versions.

- Voyager allows the creation of remote objects. This additional feature, which exploits code-mobility, was very useful in our case as one of our goal in this application was to let the examination coordinating center have maximum control of the whole distributed examination-setup. We could thus easily install remote components like Distribution Servers on the Examining Center machines.

- Agent and objects in voyager can be moved to new location, both on the basis of absolute addressing and relative addressing. In the latter case, the object (or agent) needs to specify the reference of the object (or agent) that resides on the host that the former wants to migrate to.

[2] Voyager uses the concept of **facets,** which allow the behavior of facet object to be added to an object during runtime

- Other advantages provided by the Voyager framework were federated directory service, different kinds of messaging (one-way, synchronous, future), object and agent persistence support, distributed event handling and security provisions in the form of security manager. In addition to this it also is compatible with CORBA and DCOM, which we consider important for the possible future integration of our application with other existing software.

In the next chapter, we describe the implementation details of MADE: our system for mobile agent based distance evaluation.

4 IMPLEMENATATION ASPECTS

As discussed in the previous chapters, Voyager agents are extensions of distributed objects. This enables us to employ the object-oriented principles to the design the system. We present the important implementations aspects in all the three stages of examination process.

4.1 Examination Setting

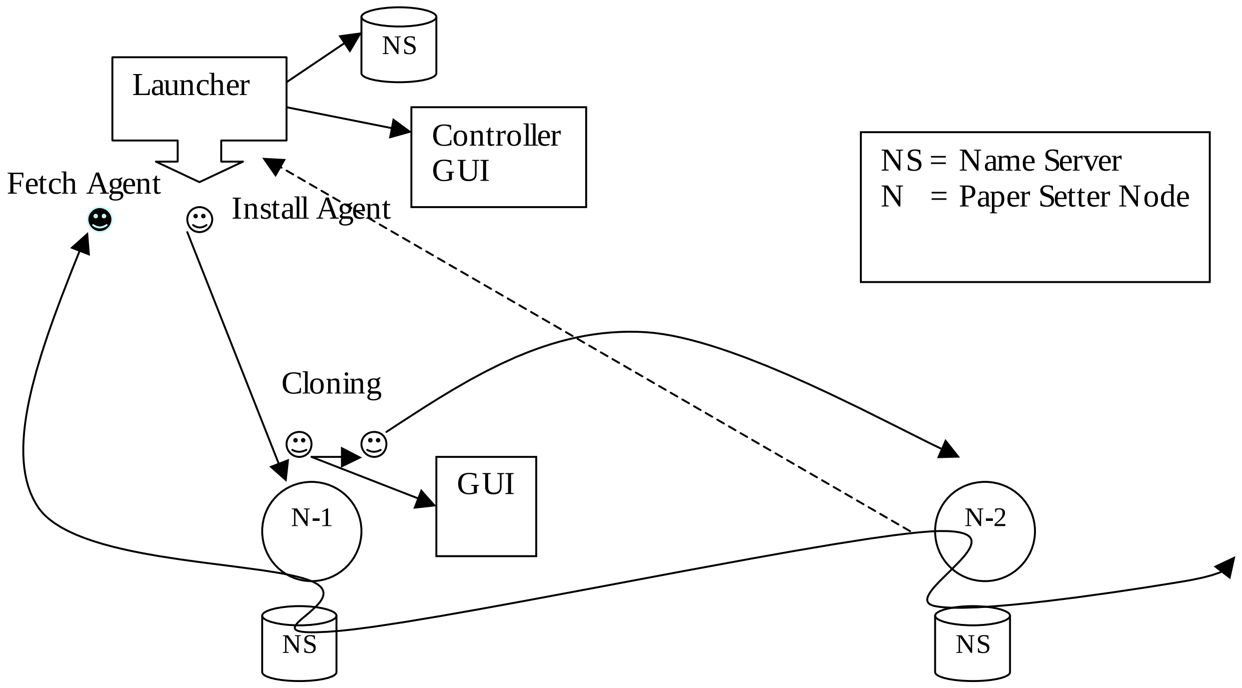


Fig 4 - 1 Details of Examination Setting process

As discussed in the previous section, the examination paper is prepared in a collaborative manner with various paper-setters setting partial question papers sitting at their remote terminals. The central coordinating authority then collects these questions and prepares a final comprehensive question paper. We use two types of mobile agents – Install and Fetch agents (See Fig 4.1). The Install Agent installs the application on different nodes. The Fetch Agent collects partial papers from the paper-setters. It also enhances the application functionality at run-time as explained in the following sections.

4.1.1 Main participants

- **Launcher**: Initializes the application, creates the paper-coordinating object, creates and launches InstallAgent and FetchAgent

- **PaperCoordinator**: Receives question objects from all the paper-setters and help the principal paper setter edit the final questions

- **InstallAgent**: Installs RemoteGUI on each machines corresponding to the remote paper-setter

- **FetchAgent**: Moves from one remote paper-setter's machine to the other until it has finished collecting question objects from all, coordinates with the Paper-coordinating object, InstallAgent interacts with the PaperCoordinator, InstallAgent, NamingService, and RemoteSetterGUI for fetching all the questions. It also provides an object that enhances the remote paper-setter's GUI dynamically at run-time.

- **RemoteSetterGUI**: Provides the GUI to remote paper-setter

- **NamingService**: Allows InstallAgents to register with it and FetchAgents to query it for reference to InstallAgent. Also facilitates FetchAgent getting a reference to the RemoteSetterGUI.

4.1.2 Collaborations (Fig 4.2)

- At the control center, the Launcher object instantiates an Install Agent. This Agent is supplied the itinerary which consists of list of paper-setters that have to be visited. The Install Agent moves to the remote paper-setter.

- IntallAgent creates the RemoteSetterGUI once it reaches a remote paper-setter machine. RemoteSetterGUI registers itself with the NamingService

- InstallAgent clones itself and the clone moves to the new paper-setter. In this way RemoteSetterGUI is installed on all the machines.

- When it is time to collect papers, Laucher instantiates a FetchAgent, and moves it to the first InstallAgent it should visit.

- FetchAgent reaches the new location, queries the NamingSerivce for a reference to the InstallAgent.

- FetchAgent also gets a reference to the RemoteSetterGUI by first querying the InstallAgent for its name and later NamingService for its reference.

- FetchAgent creates a GUI enhancing object and install it to the RemoteSetterGUI. This allows FetchAgent to directly communicate directly with the paper-setter. It prompts the paper-setter to submit the questions. Depending upon the response of paper-setter, it can go into either of these states – wait, deferred or force-fetch.

- Once the FetchAgent gets a question-paper it move to the PaperCoordinator and submits it.

- FetchAgent keeps on polling the paper-setters till they have submitted their questions or it force-fetched them.

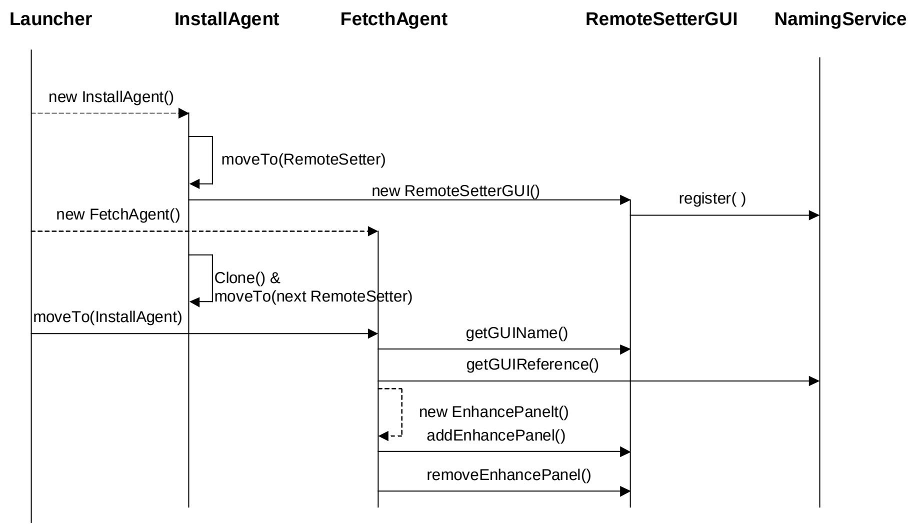


Fig 4 - 2 Interactions during Examination Setting

4.2 Distribution and Testing Stage

This stage uses two types of mobile agents – PaperCourier Agent and Answer Agent. The former's main task is delivering the question paper to all the examination centers. The latter represents a student's answer sheet and has more complex behaviour. It is to be noted that we have decided not to allow any agent to visit or be created on a student machine. This is mainly done for improving security of the system. The task of creating an answer agent is done at the distribution server, which is a more trusted host. The details for this stage are described below.

4.2.1 Main Participants

- **PaperCourierAgent**: This agent carries the question paper to all the examination centeres. It carries a single copy of a particular question paper.

- **DistributionServer**: This server distributes the question paper among all the students in an examination center my making multiple copies. It has to be supplied a list of the students enrolled in the center along with the addresses of the machines where they will be taking their tests. It also gathers the answers from the students and launches AnswerAgents to the evaluation center, one per student.

- **PaperGUI**: It is GUI made available to each student for attempting his answers.

- **AnswerAgent**: This agent represents an answer-paper of a student and is capable of moving to an evaluation-center to get its answers evaluated.

4.2.2 Collaborations (Fig 4-3)

- After being launched and supplied the itinerary for various distribution centers, the PaperCourierAgent moves to the first examination-center. Here it calls a method `atCentre( )` on itself. This causes PaperCourierAgent to begin its work at an examination center.

- After supplying the question-paper ot the DistributionServer, it moves on to the next location. After having finished it task , it terminates itself after informing the control-center that it has finished it has successfully finished its task.

- DistributionServer instantiates PaperGUI for each student enrolled on his respective host.

- Once the PaperGUI timeouts, or the students have submitted their answers, the answer object is submitted to the distribution server.

- DistriubtionServer launches the AnswerAgent, one per student answer-set.

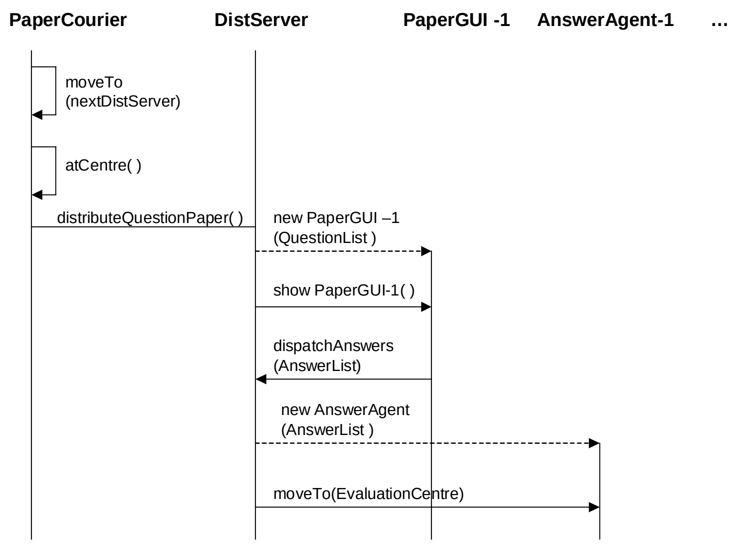


Fig 4 - 3 Interactions during Distribution and Testing

4.3 Evaluation and Result Compilation Stage

This is the final stage in the examination process. There is only one kind of agent operating here – AnswerAgent. It represents a user's answer sheet, which has the responsibility of getting its answers evaluated and its scores published. The details of this stage are described below.

4.3.1 Participants

- **Answer Agent**: This has already been described in the above section.

- **EvalCentreServer**: This server coordinates the evaluation process. It has reference to the different Examiner machines, ObjectiveEvalServer and the PublishResultGUI.

- **ObjectiveEvalServer**: This server evaluates the objective type questions. It has correct solutions, which are provided by the examination coordinator through a separate channel, i.e. the QuestionPaperAgent does not carry the question solutions with it. This is done to simplify the required security mechanisms.

- **Examiner**: Examiner are either need for auditing purpose or for evaluating subjective questions. The AnswerAgents, park at an examiners place until they get evaluated or they timeout.

- **PublishResultGUI**: Each AnswerAgent after it finishes the self-evaluation process, moves to the PublishResultGUI server and supplies its scores. When all the Answer Agents are finished with their work, the comprehensive results are compiled and published through this server.

4.3.2 Collaborations (Fig 4-4)

- AnswerAgent asks for a reference to ObjectiveEvalServer if it is carrying answers to objective type question.

- AnswerAgent supplies the student answers to the ObjectiveEvalServer which compares them to the correct solutions and evaluates the scores. These scores are returned to the AnswerAgent.

- If the AnswerAgent has answers to subjective questions, it queries the EvalCentreServer again; this time for an itinerary of examiners. Once the itinerary is available, it moves to each Examiner and gets its answers evaluated.

- After all the answers have been evaluated, the AnswerAgent enquires for PublishResultGUI reference and after getting it move and supplies its scores to PublishResultGUI server.

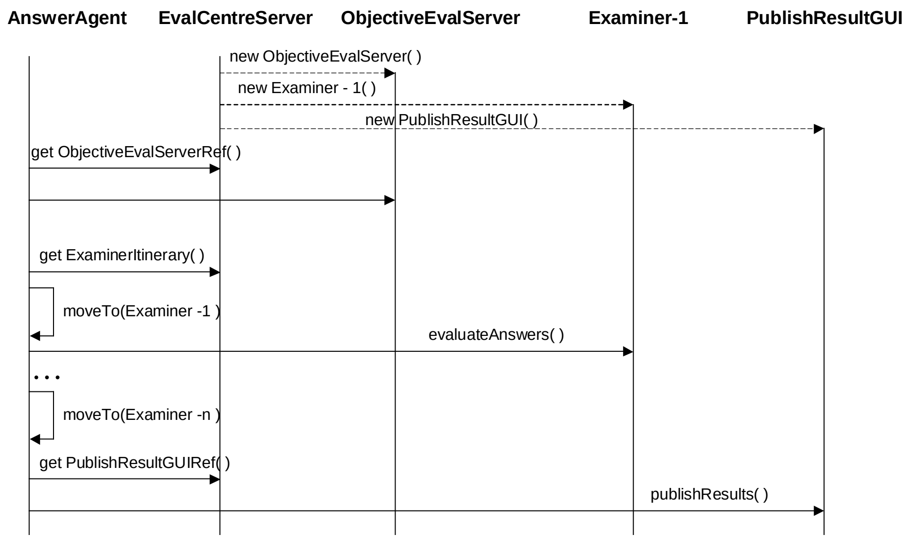


Fig 4 - 4 Interactions during Evaluation and Result Compilation

C H A P T E R   5

5 EXPERIMENTATION

The experiments were carried out on PIII, 450 MHz workstations with Windows2000 operating system. Voyager ORB was installed on all of these machines. The different configurations used, are illustrated in Fig5.1, Fig5.2 and Fig5.3, where the services running on that particular node have also been mentioned. We have simulated the set up for examination process by using different port for the different services, viz. Paper Setters and Coordinators uses IP-Port 4000, Distribution Servers at Examination Centers IP-Port 5000, Student Machines IP-Port 6000, Evaluation Center IP- Port 7000, Examiners IP-Port 8000, Objective-question evaluator IP-Port 8888, and Publishing Center IP-Port 9000.

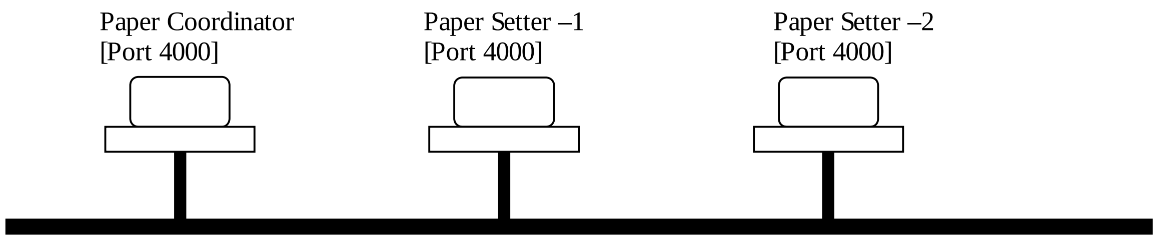


Fig 5 - 1 A typical setup for testing Examination Setting Stage

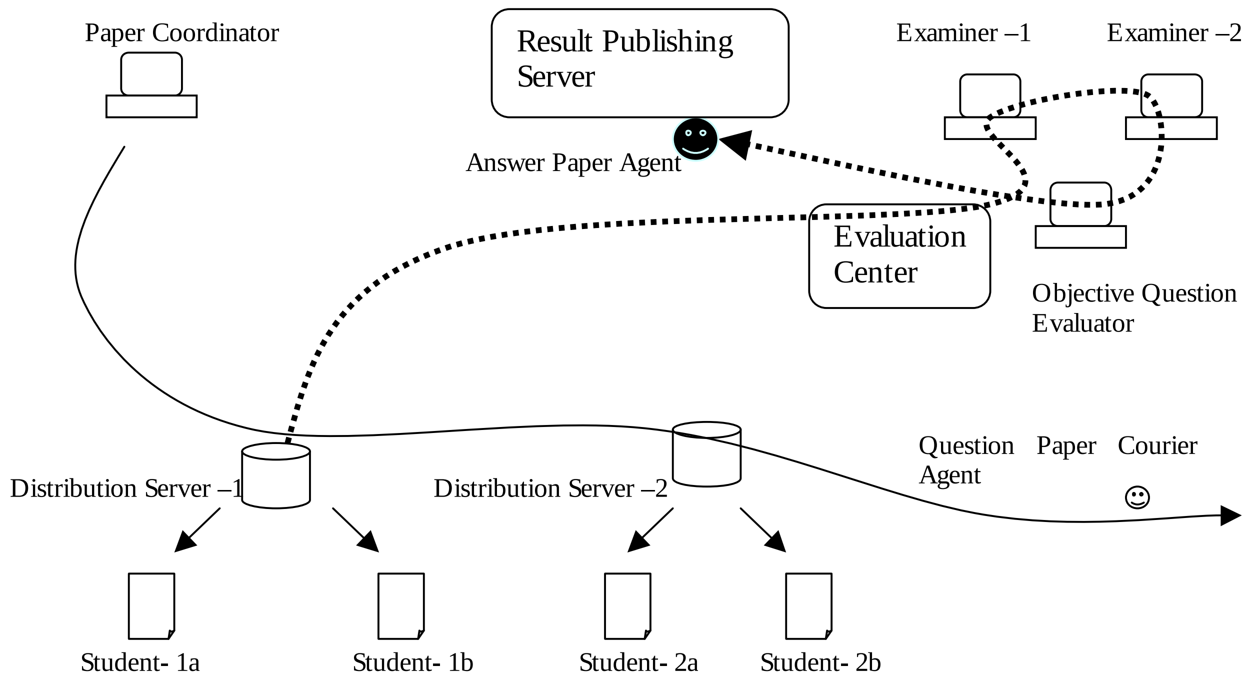


Fig 5 - 2 Schematic view of experimental-setup for Distribution-Testing and Evaluation-Result Compilation Stages

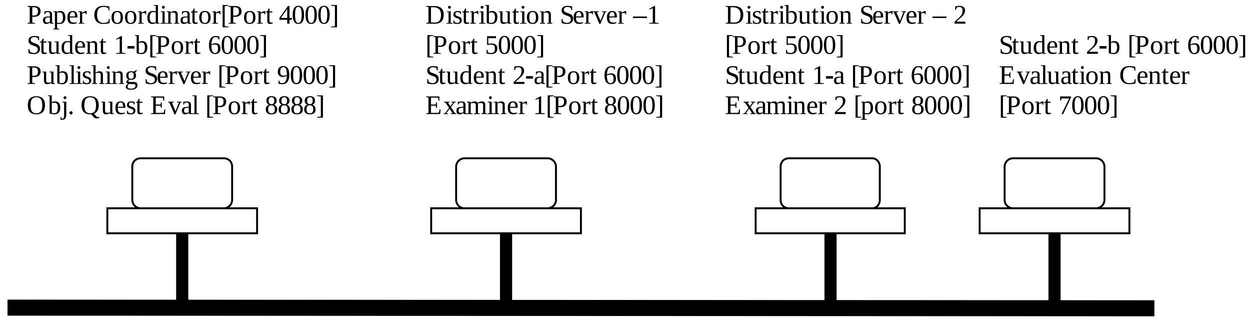


Fig 5 - 3 A typical physical setup for testing Distribution-Testing Stage and Evaluation-Result Publication Stage

5.1 Performance Evaluation

The performance criterion most relevant for our application is the *response time* for the students. We define *response time* as the time taken between a student making a request, such as, request for next question or request for next section in the question paper, and getting the appropriate response.

We have performed experiments (Fig 5-4) to make the following two set of measurements:

- Case 1: Response times in Mobile Agent Interactions
- Case 2: Response times in Client-Server Interactions

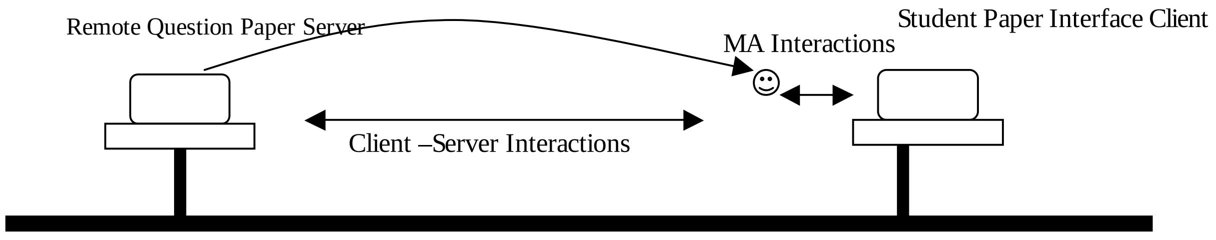
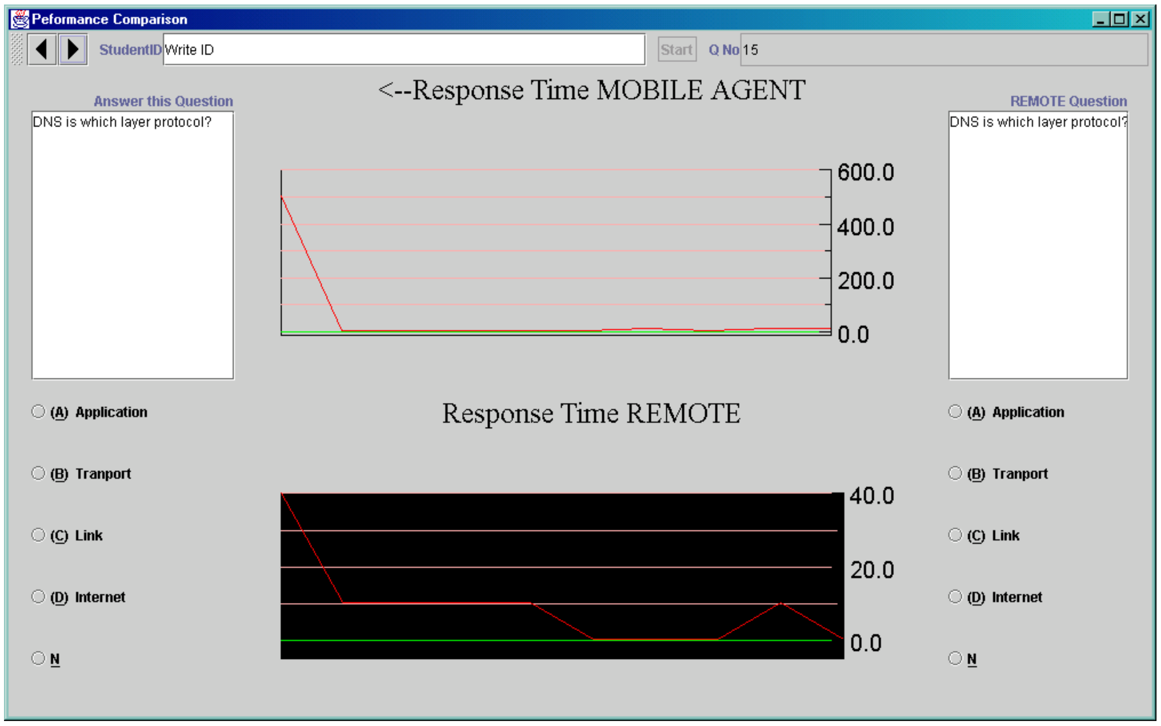

Remote Question Paper Server
Student Paper Interface Client
MA Interactions
Client –Server Interactions

Fig 5 - 4 Experimental set-up for measuring
Response Times

Fig.5.5 shows the interface for measuring these response times. The interface is an extended version of the usual objective-type question paper



DNS is which layer protocol?

DNS is which layer protocol?

Fig 5 - 5 Interface for measuring Response Time

The ‘Start’, causes a mobile agent to be launched from a remote machine, which brings in the new question paper/ section for the student in the first case. In second case the same first page of question paper/ section is fetched as data from the remote-server.

The students, browsing through the given set of questions, generate further queries. In case of MA, these questions would be been pre-fetched by the mobile agent and hence the responses will be local. In the second case every request will cause a remote request to be placed in typical client-server mode.

Fig 5-6 shows the response times in both the cases for a similar set of questions.

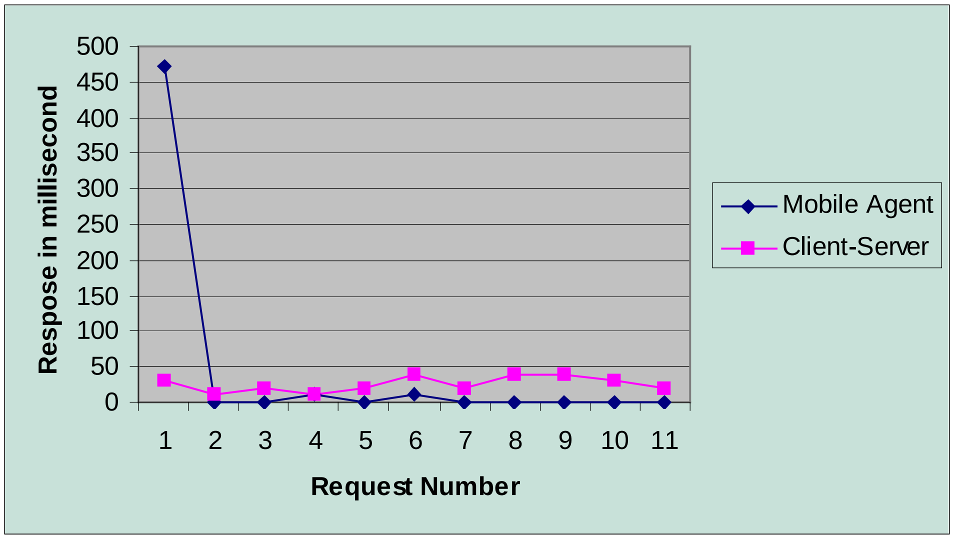


Fig 5 -6 Response times for MA and C-S implementations

5.2 Observations

We see that in case of client-server, the response times will remain more or less constant whereas in the case of MA, the initial response takes much longer while the remaining requests take negligible time as compared to client-server responses. The initial longer response in case of MA is because of the additional time taken for agent creation, dispatch and transfer. Response-time determines the user-experience and hence is critical for our application. In future with the content getting richer (graphics and multimedia support), this difference will become even more pronounced. Traditional client-server distributed programs avoid this problem by techniques like pre-fetching, caching etc. Mobile agents inherently provide these capabilities in our application.

We could achieve required ease of installation and remote management with the chosen framework though in some cases the class-loader was not able to download the required classes from the code-server. Also the framework does not provide agent tracking and control support, which is useful for our application and a critical need when we run this application on the Internet scale.

C H A P T E R 6

6 CONCLUSIONS

Most present day distributed systems are structured using the client-sever paradigm. Existing computer based evaluation systems are also necessarily client-server implementations. For local and simple objective-question evaluations, these systems prove to be sufficient. They enable features like adaptive questioning and quick compilation of results. Distance evaluations pose some new challenges. Network delays affect the student's response times and there is increased complexity of coordination and control of examination process. Other desirable features are support for push model, off-line examinations, and easy integration of different stages of examination process. However, simple extension of existing client-server systems to include these features seems impractical.

Mobile Agents provide a more flexible paradigm of structuring such systems, as they can support disconnected operation, help in better utilization of network bandwidth, and enable local interactions.

In this project, we have designed and implemented MADE: a mobile agent based system for distance evaluation. We have also implemented a similar application using traditional client-server architecture. From our implementation and experiments, we observe that mobile agents provide considerable improvements over the existing systems in the following ways: student's perceived response time, capability to handle different types of examinations (objective as well as subjective), application level multicasting which leads to better bandwidth utilization, dynamic upgradation of applications, support for heterogeneous execution environments, centralized control and management of logistics and security of the examination process.

We have shown that the mobile agent approach is viable for building next generation internet applications. However, translation of prototypes into the real world applications needs to address the following additional issues: inadequate system support for mobile agent execution, security of agent as well as host, and reliable transfer of agents.

This current work may be extended in the following directions

- **Reliability** : There is a need to provide the reliable transfer of mobile agents because we cannot afford to loose an agent carrying question or answer paper. Also we should not have multiple copies of an answer paper, during any stage of the agent transfer. Additonally, to recover from server and network outages, mobile agents should support check-pointing and recovery.

- **Persistence**: There is a need for proper archiving of the question and answer papers. It will also be important to maintain the unique identity of each answer paper; difficult thing to achieve because the digital information is easily altered and duplicated and altered.

- **Security**: Our design takes care of most of the issues so as to cause the minimum security overheads. Still we need to provide for secure transfer and authentication of questions and answer papers. Protecting the answer paper from malicious tampering will be a critical requirement for the success of the system.

BIBLIOGRAPHY

[1] Alfonso Fuggetta, Gian Pietro Picco and Giovanni Vigna. "Understanding Code Mobility ", *IEEE Transactions on Software Engineering*, vol. 24(5), 1998

[2] S. L. Wise and B.S. Plake. Research on the affects of administering tests via computers, Educational Measurement: Issues Practice, vol. 8, no. 3, pp 5-10, 1989

[3] Walworth, M., & Herrick, R. J. (1991). The use of computers for educational and testing purposes. *Proceedings Frontiers in Education Conference* (pp. 510-513),1991

[4] Y.D. Lin, C. Chou, Y.C. Lai, and W.C. Wu, WebCAT*- A Webcentric, multiserver, computer-assisted testing system," *International Journal of Educational Telecommunications*, vol. 5, no. 3, pp.171-192, 1999

[5] Jakob Hummes, Arnd Kohrs, and Bernard Merialdo. Questionnaires: a framework using mobile code for component-based tele-exams. In *Proceedings of IEEE 7th Intl. Workshops on Enabling Technologies: Infrastructure for Collaborating Enterprises (WET ICE)*, Stanford, CA, USA, June 1998

[6] Chien Chou. Constructing a Computer-Assisted Testing and Evaluation System on the World Wide Web-The CATES Experience, in *IEEE Transactions on Education*, Vol 43, No 3, Pages 266-272, August 2000

[7] Robert Ubell. Engineers turn to e-learning, *IEEE Spectrum*, October 2000

[8] Rahul Jha, Srinath Perur, Vikram Jamwal and Sridhar Iyer. Mobile Agents in e-commerce: A quantitative evaluation, in *Proceeding of 8th Int. Conference on Advanced Computing and Communication*, Kochi, Dec 2000.

[9] Danny B. Lange. Mobile Objects and Mobile Agents: The Future of Distributed Computing, In *Proceedings of The European Conference on Object-Oriented Programming '98*, 1998.

[10] Yariv Aridor and Danny B. Lange. Agent Design Patterns: Elements of Agent Application Design, In *Proceedings of Second International Conference on Autonomous Agents'98*, 1998.

[11] Danny B. Lange and Mitsuru Oshima. Mobile Agents with Java: The Aglet API, *World Wide Web Journal*, 1998.

[12]Gunter Karjoth, Danny B. Lange, and Mitsuru Oshima. A Security Model for Aglets, *IEEE Internet Computing, Vol. 1, No. 4,* July/August, 1997.

[13]Danny Lange and Mitsuru Oshima. The Aglet book "Programming and Deploying Java Mobile Agents with Aglets", *Addison-Wesley.*

[14]Yariv Aridor and Mitsuru Oshima . Infrastructure for Mobile Agents: Requirements and Design, *Proc. of 2nd International Workshop on Mobile Agents (MA '98), Springer Verlag,* September 1998.

[15]Anand Tripathi, Neeran Karnik, Manish Vora, Tanvir Ahmed, and Ram D. Singh Ajanta -- A Mobile Agent Programming System, *Technical Report # TR98-016, Department of Computer Science, University of Minnesota,* April 1999.

[16] Anand Tripathi, Neeran Karnik, Manish Vora, Tanvir Ahmed and Ram Singh. Mobile Agent Programming in Ajanta, In *Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS '99)* , 1999

[17] Neeran Karnik and Anand Tripathi . Agent Server Architecture for the Ajanta Mobile-Agent System, In *Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98),* Las Vegas, July 1998.

[18] Anand Tripathi and Neeran Karnik.. Protected Resource Access for Mobile Agent-based Distributed Computing, In *Proceedings of the ICPP workshop on Wireless Networking and Mobile Computing,* Minneapolis, August 1998.
[19] Neeran Karnik and Anand Tripathi . Design Issues in Mobile Agent Programming Systems. *IEEE Concurrency,* July-Sep 1998 pp 52-61.

[20] Neeran Karnik. Security in Mobile Agent Systems, *Ph.D. dissertation*

[21] Ladislau Bölöni and Dan C.Marinescu. A Multi-Plane State Machine Agent Framework., *CSD-TR# 99-027,* September 99
http://bond.cs.purdue.edu/papers/Multiplan.ps

[22] Tom Walsh, Noemi Paciorek, David Wong. Security and Reliability in Concordia, In *Mobility, Processes, Computers, and Agents, Addison-Wesley,* 1999, pp. 525-534.

[23] David Wong, Noemi Paciorek, Tom Walsh, Joe DiCelie, Mike Young, Bill Peet. Concordia:Infrastructure for Collaborating Mobile Agents, *First International Workshop on Mobile Agents 97 (MA '97)* in Berlin, Germany on April 7 - 8, 1997

[24] Robert S. Gray and David Kotz and George Cybenko and Daniela Rus. D'Agents: Security in a multiple-language, mobile-agent system. In Giovanni Vigna, editor, *Mobile Agents and Security, Lecture Notes in Computer Science, Springer-Verlag*, 1998.

[25] Brian Brewington and Robert Gray and Katsuhiro Moizumi and David Kotz and George Cybenko and Daniela Rus. Mobile Agents for Distributed Information Retrieval. In Matthias Klusch, editor, *Intelligent Information Agents*, chapter 15, Springer-Verlag, 1999.

[26] O. Holder, I. Ben-Shaul and H. Gazit. Dynamic Layout of Distributed Applications in FarGo. *Proceedings of the 21st International Conference on SoftwareEngineering (ICSE'99)*, Los Angeles, CA, USA, May 1999.

[27] Grasshopper Technical Overview http://www.ikv.de/products/grasshopper/

[28] Joachim Baumann, Fritz Hohl, Kurt Rothermel and Markus Straßer (1998):Mole - Concepts of a Mobile Agent System,*World Wide Web*, Vol. 1, Nr. 3, pp. 123-137

[29] G. Glass, "ObjectSpace Voyager Core Package Technical Overview ", *Mobility: process, computers and agents* , Addison-Wesley, Feb. 1999

[30] ObjectSpace Inc., ObjectSpace Voyager Core Package Technical Overview, 1997.

[31] Alberto Silva, Miguel Mira da Silva, José Delgado. AgentSpace: An Implementation of a Next-Generation Mobile Agent System, in *Proc. of MobileAgents'98, Lecture Notes in Computer Science*, 1477, Springer Verlag, 1998.

[32] Ernö Kovacs, Hong-Yon Lach, Björn Schiemann, Klaus Röhrle, Carsten Pils: Agent-based Mobile Access to Information Services, *ACTS Mobile Summit AMOS'99*, June 1999.

[33] M. Zapf, H. Müller, K. Geihs. Security Requirements for Mobile Agents in Electronic Markets, In*: Proceedings of the working conference on Trends in Distributed Systems for Electronic Commerce (TrEC'98)*, Lecture Notes in Computer Science, Springer, Hamburg (June 1998)

[34] Srilekha Mudumbai, Abdeliah Essiari, William Johnston. Anchor Toolkit(A Secure Mobile Agent System), *Technical Report*, Imaging and Computing Sciences Division Ernest Orlando Lawrence Berkeley LaboratoryUniversity of California. http://www-itg.lbl.gov/Akenti/Anchor

[35] C. Santoro. ARCA: A Framework for Mobile Agents Programming - A White Paper, *Internal Report, University of Catania* - Dec 1998

[36] Kawamura,T., Yoshioka,N., Hasegawa,T., Ohsuga,A. and Honiden,S. Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems ", *In Proceedings of the 6th Asia-Pacific Software Engneering Conference*, 1999.

[37] Werner Van Belle, Karsten Verelst, Theo D'Hondt. Location Transparent Routing in Mobile Agent Systems Merging Name Lookups with Routing, Presented at *Future Trends of Distributed Computer Systems*, December '99 PROG/DINF/VUB

[38] A. Lingnau, O. Drobnik and P. Doemel. An HTTP-based Infrastructure for Mobile Agents, *Fourth Int'l World Wide Web Conference Proceedings* (Boston, Dec1995), *World Wide Web Journal 1* (1995), pp. 461-71

[39] Mehdi Jazayeri, Wolfgang Lugmayr. Gypsy: A Component-Oriented Mobile Agent System. *8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000)* (Rhodos, Greece, January 19-21, 2000).

[40] Nelson Minor. Hive: Distributed Agents for Networking Things", *Proceedings ASA/MA '99*, 1999

[41] Walter Binder. Design and Implementation of the J-SEAL2 Mobile Agent Kernel,in *6th ECOOP Workshop on Mobile Object Systems: Operating System Support, Security, and Programming Languages*; France, June 2000;

[42] A.S. Park, M. Emmerich, D. Swertz: Service Trading for Mobile Agents with LDAP as Service Directory. *IEEE 7th Intl. Workshop on EablingTechnologies: Infrastructure for Collaborative Enterprises, WETICE'98*, Stanford University, California, USA, June, 1998.

[43] Marcus J. Huber. JAM: A BDI-theoretic Mobile Agent Architecture, in *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, Seattle, WA,May, 1999, pgs 236-243

[44] L.M.Silva, P.Simoes, G.Soares, P.Martins, V.Batista, C.Renato, L.Almeida, N.Stohr. JAMES: A Platform of Mobile Agents for the Management of TelecommunicationNetworks, *Proc. IATA'99, Intelligent Agents for Telecommuniaction Applications*, Stockholm, Sweden, August 1999

[45] Ciaran Bryce and Jan Vitek The JavaSeal Mobile Agent Kernel, in *ASA/MA Proceedings'99*, 1999

[46] Jumping Beans White Paper, Ad Astra Engineering, Inc., 1998 http://www.JumpingBeans.com/

[47] Dirk Struve. Kariboga, Mobile Agent System, Project's Documentation Web Page, http://www.projectory.de/kaariboga/index.html

[48] L. Bettini, R. De Nicola, G. Ferrari, and R. Pugliese. *Proceedings of WETICE '98, IEEE*, 1998

[49] KOE Documentation, http://www.cnri.reston.va.us/home/koe/docs/manuals/index.html

[50] Chr. Tschudin. The Messenger Environment M0 - A Condensed Description. In Vitek and Tschudin (Eds.): *Mobile Object Systems, LNCS 1222,* 1997, pp. 149-156

[51] Chr. Tschudin. On the Structuring of Computer Communications. *PhD thesis 2632*, University of Geneva, Switzerland 1994

[52] Puliafito, O. Tomarchio, and L. Vita. MAP: Design and Implementation of a Mobile Agents Platform. *Journal of System Architecture*, 46(2):145-162, 2000.

[53] Smith, Chris. *Theory and the Art of Communications Design.* State of the University Press, 1997.

[54] R Ghanea-Hercock, J Collis, D Ndumu. Co-operating Mobile Agents for distributed parallel processing, *Autonomous Agents '99*, Seattle.

[55] M. Fukuda, L. Bic, M. Dillencourt, Messages versus Messengers in Distributed Programming, *Journal of Parallel and Distributed Computing (JPDC),* Vol. 57, 199-211, 1999

[56] Mobile and Intelligent Platform for Agent Communication Environment: MIPLACE, *NEC Research & Development*, vol.40, No.3 July 1999

[57] Satoru FUjita, Suresh Jagannathan, et al. Mobile and Distributed Agents in Mobidget, Poster paper in *ASA/MA 99*, 1999

[58] G.P.Picco. µCode: A Lightweight and Flexible Mobile Code Toolkit In Mobile Agents, *Proceedings of the 2nd International Workshop on Mobile Agents 98 (MA '98),* Stuttgart (Germany), K.Rothermel and F. Hohl eds., September 1998, Springer, Lecture Notes on Computer Science vol. 1477, ISBN 3-540-64959-X, pp. 160-171

[59] Pawel Wojciechowski . Nomadic Pict: Language and Infrastructure Design for Mobile Computation. *Ph.D. thesis*, March 2000. Also as Technical Report 492, Computer Laboratory, University of Cambridge, June 2000.

[60] Pawel Wojciechowski and Peter Sewell. Nomadic Pict: Language and Infrastructure Design for Mobile Agents,. In *ASA/MA '99 (First International Symposium on Agent*

Systems and Applications/ Third International Symposium on Mobile Agents), October 1999. An extended version is to appear in IEEE Concurrency.

[61] Wen-Shyen E. Chen, C.Y. Lin and Yao-Nan Lien. A Mobile Agent Infrastructure with Mobility and Management Support. *Proceedings of the 1999 International Workshops on Parallel Processing*, Institute of Electrical and Electronics Engineers, Inc.

[62] K. Kato, Y. Someya, K. Matsubara, K. Toumura, H. Abe. An Approach to Mobile Software Robots for the WWW, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 4, 1999 July/ August.

[63] Ohsuga, A., Nagai, Y., Irie, Y., Hattori, M., and Honiden, S. PLANGENT: An Approach to Making Mobile Agents Intelligent, *IEEE Internet Computing*, Vol. 1, No. 4 (1997), pp.50-57

[64] P. Bellavista, A. Corradi and C. Stefanelli. A Secure and Open Mobile Agent Programming Environment", *Proc. Fourth International Symposium on Autonomous Decentralized Systems (ISADS '99)*, Tokyo, Japan, March 21-23,1999, pages 238-245, IEEE Computer Society Press.

[65] P.Bellavista, A.Corradi, and C.Stefanelli. An Open Secure Mobile Agent Framework for Systems Management, *Journal of Network and Systems Management (JNSM)*, Special Issue on "Mobile Agent-based Network and Service Management",September 1999.

[66] A.Corradi, R. Montanari, C. Stefanelli. Mobile Agents Protection in the Internet Environment. In *Compsac'99 Proceedings*, 1999

[67] J. White. Mobile Agents, in *Software Agents*, J. Bradshaw (ed.), AAAI Press / The MIT Press, 1996

[68] Dag Johansen, Robbert van Renesse, and Fred B. Schneider. Operating system support for mobile agents. In, *Proceedings of the 5th. IEEE Workshop on Hot Topics in Operating Systems*, Orcas Island, Wa, USA (4th-5th May, 1995), Published by: IEEE Computer Society, NY, USA,May 1995.

[69] Milojicic, D., laForge, W., Chauhan, D. Mobile Objects and Agents, Design, Implementation and Lessons Learned, in the *Proc. of the Fourth USENIX Conference on Object-Oriented Technologies and Systems (COOTS '98)*, April 27-30, 1998, Santa Fe, New Mexico

[70] Green, S. et al. Software Agents: A review, *Technical Report*, Departmentof Computer Science, Trinity College, Dublin, Ireland

[71] Gian Pietro Picoo. Understanding, Evaluating, Formalizing, and Exploiting Code Mobility, *PhD Dissertation*, Department of Automation and Informatics, Torino Polytecnico, Italy.

[72] Rahul Jha. Mobile Agents for e-commerce. *M.Tech. Thesis* 2001, IIT Bombay, India

[73] Riordan J., Schneier B.: Environmental Key Generation Towards Clueless Agents, in: Giovanni Vigna (Ed.): *Mobile Agents and Security*. pp 15-24. Springer-Verlag, 1998.

[74] Sander T., Tschudin C. F.: Protecting Mobile Agents Against Malicious Hosts, in: Giovanni Vigna (Ed.): *Mobile Agents and Security*. pp 44-60. Springer-Verlag, 1998

[75] Vigna G.: Cryptographic Traces for Mobile Agents, in: Giovanni Vigna (Ed.): *Mobile Agents and Security*. pp 137-153. Springer-Verlag, 1998.

[76] Young A.; Yung M.: Encryption Tools for Mobile Agents: Sliding Encryption, E. Biham (Ed.): Fast Software Encryption. *Proceedings of the 4th International Workshop, FSE '97*, Haifa, Israel, January 20-22, 1997., LNCS 1267, Springer-Verlag, 1997

[77] Y. Aridor, and M. Oshima. Infrastructures for Mobile Agents: Requirements and Design. In *Proceeding of Second International Workshop on Mobile Agents*, MA'98, Stuttgart, Germany, 1998

[78] Todd. Papaioannou On Structuring of Distributed Systems, The argument for mobility, *PhD Thesis*, Loughborough University University, 2000

[79] FIPA Specification Documentation *http://www.fipa.org*

[80] D. Milojicic, M. Breugst et. al. MASIF: The OMG Mobile Agent System Interoperability Facility, In *Proceeding of Second International Workshop on Mobile Agents*, MA'98, Stuttgart, Germany, 1998

.