MASS-Mobile Agent Systems for Smart Applications with CBD

Haeng-Kon Kim

Department of Computer Information & Communication Engineering,
Catholic University of Deagu, Korea.

Abstract - Mobile agents are software nomads that act as your personal representative, working autonomously through networks. They are able to visit network nodes directly using available computing power and are not limited by platform. This emerging field is now poised to become a cornerstone for new Mobile -based ubiquitous computing environments. Mobile agents provide a new abstraction for deploying functionality over the existing. In this paper, we present an architecture that allows currently available mobile servers to become capable of sending and receiving agents in an easy way. We design and implement the MASS (Mobile Agent Systems for Smart applications with CBD). By using this approach, existing mobile infrastructure can be maintained, while gaining a whole new potential by being able to make use of mobility agent technology. Our approach involves wrapping the components inside a Java servlet that can be included in any mobile server supporting the Servlet Specification. This servlet enables the servers to receive and send agents that can query local information, and also enables the agents to behave as servlets themselves. We currently have used the framework with several existing commercial mobile servers, inclusively having the security mechanisms of the framework correctly running and integrated with the security architecture of the smart mobile devices.

Keywords: Agent communication networks, Mobile agents, Web and Mobile device agents, Component based development, Smart applications

1. Introduction

www.edlib.asdf.res.in

Mobile agents are small threads of execution that are able to migrate from machine to machine, performing operations locally [1]. One very interesting application area for mobile agents is mobile device computing. Mobile agents provide a very attractive paradigm for this area. The agents can be launched from a machine, navigate from device to device, collecting information or performing transactions, finally returning home with the goods or results. This scenario is especially attractive when we consider the proliferation of wireless mobile devices that is currently taking place. A user can launch an agent into the web, shutdown the device and reconnect hours later, collecting the agent with the results. Some key applications for agents in mobile computing include:

- Information gathering agents, which collect information from different web sites or distributed databases, finally presenting it to its owner.
- Shopping agents, which look for the best deals for their owners, perform commercial transactions, and present the best results found, so that their owners can make a decision.
- Management agents, which carry information into selected web sites or databases and make sure that all the distributed web-infrastructure is up-to-date.
- Monitor agents, which migrate into selected web sites and monitor some information (like stock options), Warning the owner when certain events happen or even performing some actions on those events.

Although mobile agents provide an attractive conceptual framework for mobile device-based computing – small threads migrating from server to server, performing their functions, there are still many difficulties that must be addressed. These difficulties are currently preventing the widespread adoption of the technology. Some of the key problems include: security, user and provider psychological resistance, infrastructure integration, interoperability and reliability.

Security: If one wants to deploy mobile agents into the world-wide-web on wireless mobile devices security is a critical issue that must be carefully considered. There are many points to examine when it comes to mobile agent security. Because the agents are going to arrive at a host that probably knows nothing about them, there must be mechanisms that prevent the agents from damaging the host or access information that they do not have permissions to. Also, because the agents are going to execute in an open environment, on machines that they may not know them very well, they are extremely vulnerable to attacks from those. The hosts can steal information from the agents: make them perform actions they did not voluntarily wanted to; or even misguide them into give false information to other entities. Finally, the agents must be protected from attacks of other agents running in the same host [2, 3].

Currently, the mechanisms needed for protecting the hosts from the misbehaved agents and the agents from attacks of other agents are well known. These mechanisms are mostly based on proper authentication and authorization. Even so, today there is a major technical problem on protecting the hosts from the mobile agents. Protecting the agents from the hosts is technically very difficult. Because the agents are executing on a host, the host has access to all the state and code of the agent. Although there are some promising approaches for solving this problem, like computation with encrypted functions and code obfuscation, the problem is still far from being solved.

Infrastructure Integration: Another relevant issue is how a web site should integrate a mobile agent platform into its infrastructure. Currently available systems follow basically two approaches. The first one involves installing an agent platform that is completely unaware of the web server. The agents migrate to and from the agent platform and interact with the web server as if they were just normal clients, with the difference that they are local. Although this approach is appropriate for operations like querying information on the host, or monitoring when certain changes happen, it quite limits the functionality that can be implemented on the agents. For instance, it is quite hard for the agents to publish information on the site, or to extend the functionality of the servers by migrating agents into them, or even having the agents represented in a web page of the server for their users to remotely interact with them.

The second available approach consists in developing a custom-made web server that is also able to host agents. The problem is that typically the web sites are already up and running, and do not want to replace their existing infrastructure. Also, typically these agent-enhanced web servers do not have the robustness or scalability needed for production-running sites. Thus, it would be quite difficult for a web site to accept replacing its industrial-strength web infrastructure for a technology that follows one of the above approaches [4, 5, 6].

Reliability: Another very important question is reliability. If a user is going to send an agent into the web, many things can go wrong that can make the agent to be lost. A simple server crash may kill all the agents that are running there. Even a routine operation like a server shutdown may lead to the loss of the agents running on the server.

There are currently may mechanisms that can be applied for ensuring that the agents do not get lost, like persistent storage and fault-tolerance techniques. Nevertheless, it is important to carefully consider the reliability requirements when deploying an agent infrastructure on the web [6].

In this paper we present an architecture that allows currently available web servers and mobile devices to become capable of sending and receiving agents in an easy way. We also design and implement the MASS (Smart Mobile Agent Systems for Smart applications with CBD). By using this approach, existing web infrastructure can be maintained, while gaining a whole new potential by being able to make use of agent technology. Our approach involves wrapping the components inside a Java servlet that can be included in any mobile agent for supporting the smart applications and specification. This mobile agent enables the servers to receive and send agents that can query local information, and also enables the agents to behave as mobile agent themselves. We currently have used the framework with several existing commercial web servers, inclusively having the security mechanisms of the framework correctly running and integrated with the security architecture of the server.

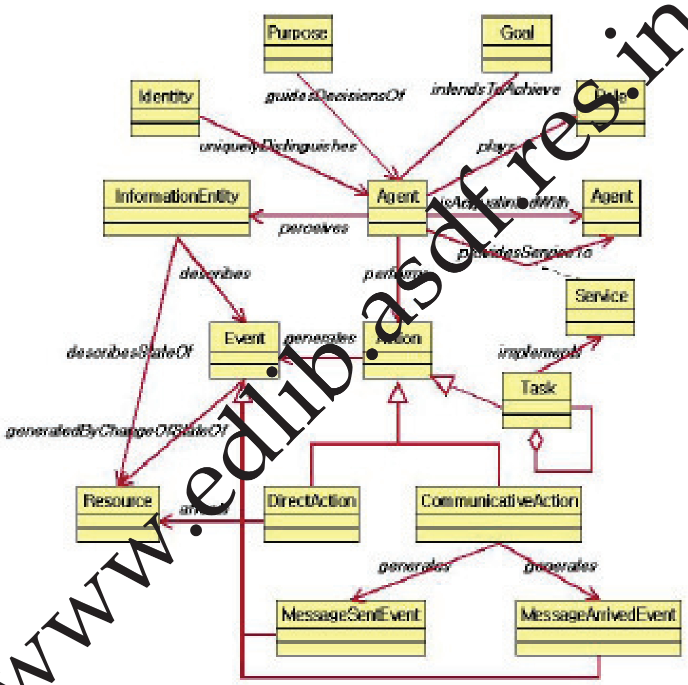
www.edlib.asdf.res.in

2. Related Works

2.1 Agent Concept Model

An agent is an atomic autonomous entity that is capable of performing some useful function. The functional capability is captured as the agent's services. A service is the knowledge level analogue of an object's operation. The quality of autonomy means that an agent's actions are not solely dictated by external events or interactions, but also by its own motivation. We capture this motivation in an attribute named purpose. The purpose will, for example, influence whether an agent agrees to a request to perform a service and also the way it provides the service. Software Agent and Human Agent are specialization of agent [7, 8].

Figure 1 gives an informal agent-centric overview of how these concepts are inter-related. The role concept allows the part played by an agent to be separated logically from the identity of the agent itself. The distinction between role and agent is analogous to that between interface and class: a role describes the external characteristics of an agent in a particular context. An agent may be capable of playing several roles, and multiple agents may be able to play the same role. Roles can also be used as indirect references to agents. This is useful in defining re-usable patterns. Resource is used to represent non-autonomous entities such as databases or external programs used by agents. Standard object-oriented concepts are adequate for modeling resources.



www.edlib.asdf.res.in

Fig 1. Agent Concept Model

2.2 Reference Matrix of MA-CBD

In order to construct component reference architecture, agent is classified in general agent type and mobile agent (MA) function attribute in our research as the figure 2 that is a component and Meta matrix of based on all above described for Mobile Agent [9].

Reference architecture is consisted of dimension, which has 14 general types and 11 concrete business agent types with domain oriented component architecture. These two classification areas tend to be independent for each cross-referenced. Each area has its own horizontal and vertical characteristics. General agent types are corresponding to agent platform and application. It is possible to develop agent system or application by the referencing architecture. The technology of agent can be applied to business domain.

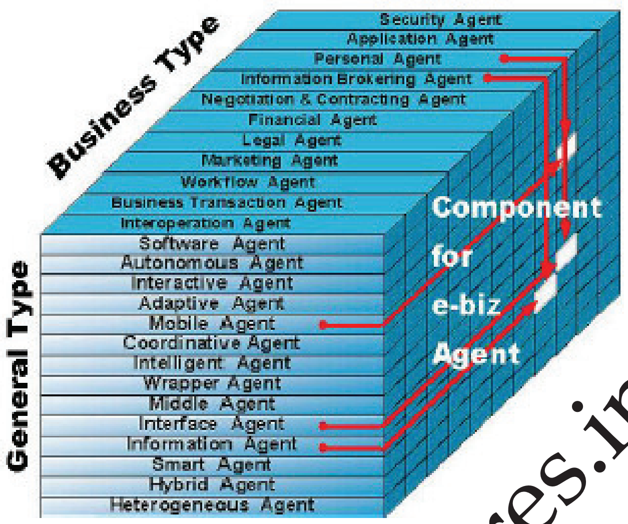


Fig. 2 Reference Matrix of Mobile Agent on Mobile Device with CBD

Developed component is classified by the reference architecture and is placed according to general agent type and business attribute. In case agent is applied to the agent system or business domain, system is possibly to build up by identifying component related to business domain and combining it.
In our opinion, the mobile agent paradigm provides a very good conceptual model for developing distributed Internet applications. Nevertheless, there are some very important problems that must be addressed.

3.2 MA Overview

The current industry best-practice software methods and become agent-enabled by integrating the mobility components. We call this approach SACMAS Smart Application Centric Mobile Agent Systems – since the applications are central and mobile agents are just a part of the system playing specific roles. The consequences of SACMAS are as follows:

- It is not necessary to design the whole application around agents. Agents are sent back to middleware, in pair with other distributed programming technologies.
- Security is integrated with the application security framework, rather than being completely generic.
- Agents interact directly with the application from the inside. This eliminates the need to setup interface agents and configure/manage their security policies.
- There is no agent platform to install and maintain. Although there are still distributed applications to install and manage, this is much simpler than managing a separate infrastructure shared by a large number of distributed applications with different policies and requirements.
- The end-user sees applications, not agents. In this way, the acceptance of applications that use mobile agents is increased since what the end user sees is the added value functionality, not the agents.

www.edlib.asdf.res.in

Our framework was implemented using the JavaBeans component framework, and is centered on the so-called Mobility Component. This component provides the basic support for agent migration and management, and an extensibility mechanism that allows other components to connect to it. These other components may implement functionalities like different inter-agent communication mechanisms, security, persistence and others.

One very important aspect of the extensibility mechanism is that it is based on an event model – Agent Lifecycle Events. After a high-level service component has registered with the Mobility Component, it is notified whenever some state transition occurs in an agent. As an example, consider the security component. On being notified that an agent is arriving, it can verify its credentials and examine its state. If it finds the agent not to be trusted, it can veto the event, prohibiting the agent from arriving.

We believe that security, from the point of view of the host, is solvable in the near future, depending mostly on the adoption of basic resource control methods in the Java platform. Protecting the agents from the hosts, it is a complicated problem in the general case, but there are approaches that can be used in web agents. These approaches give some security guaranties for the agents running on the web, by carefully considering requirements of those agents in the Internet domain.

Regarding the resistance of the users on using mobile agents, it is necessary to provide a stronger focus on the applications that use agents, and not on the agents themselves. We believe that the user doesn't even need to be aware of the agents or the agent platforms. What he needs to see and interact with are the applications. On the other hand, convincing the web hosts to introduce the technology into their sites is basically a question of market, and maturation of the agent technology. When the technology comes to a point where the providers can be assured that there is no danger in deploying such a framework on their infrastructure, they will give such functionality to their users. This will allow them to differentiate from the competition, providing a better service to the customers.

Infrastructure integration, interoperability and reliability are serious technical problems that must be addressed. This is just part of maturing process of the technology. In this paper we address the infrastructure integration problem.

3. Mobile Agent with Existing Mobile Devices Infrastructure on CBD

3.1 Motivation

Our interest in building support for mobile agents in web serves arouse from the necessity of validating how easy or not was to agent-enable existing applications by using the MA framework. Web servers, and in particular the creation of web-agents, appeared to be an interesting application field because the paradigm seams so fit for using on the web.

In the case of our framework, this seemed to be perfect since this adapter could be used for housing the Mobility Component. For this particular system, we had three requirements:

- It should be possible for the agents to behave as a web resource (i.e. publish information). A user should be able to use a web browser to access and interact with the agents that would be dynamically generating the web pages.
- The agents should be able to query local information present on the web server.
- If possible, the agents should be able to perform management operations on the server. This last requirement was based on our interest in studding the usefulness of mobile agents for distributed network and application management.

3.2 Architecture

To meet the above requirements, we came up with the architecture, named MASS architectures depicted as in figure 3. The *Mobility Wrapper* is a class that overrides the *HTTP Frame* and houses the Mobility Component. This wrapper also listens to the *Agent Lifecycle Events*. Thus, at any given time it knows the state of every agent in the system, and publishes this information into a URI. This means that a user accessing the web site is able of seeing which agents are presently running on the server. The

www.edlib.asdf.res.in

information published by Mobility Wrapper about each agent is accompanied by a link, which includes the identity of the agent. The wrapper is able to do this because when an agent arrives, it receives the corresponding event and saves a reference to it. Also, when an agent migrates or dies, the wrapper also receives an event and is able to garbage-collect that reference. The bottom line is that it is possible for a user to interact with the agents currently running on the web server by simply accessing a starting page. Another interesting point of this approach is that because in the MA framework the agents arrive and interact with the applications from the inside, the agents have access to the internal objects of the applications.

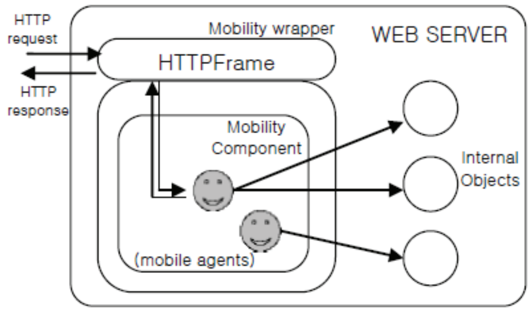


Fig 3. MASS Architectures

Fig 3. MASS Architectures

3.3 MASS-CBD Process

As we suggested MASS-CBD reference architecture in previous our research[10], component development process based architecture is a set of activities and associated results, which lead to the production of a component as shown in figure 4. These may involve the development of component from MASS specification by using UML model. Here, our main concern is the specification workflow. The domain analysis specification, design model, implemented component, which are produced though the process, are stored in the repository [11]. The requirement of agent should be first identified in desired business system. The primary property of agent is able to analyze after that the description for specific agent platform and the sorts of essential properties should be understood. At the same time, it is very important to consider whether the requirement, which is already defined, is corresponding to agent type in reference architecture and what business concept is focused on. For the mobile-business domain analysis, UML approach is used. Diagrams used in problem domain analysis are use case diagram. Use case diagram is a diagram that shows a set of use cases and actors and their relationships. It supports the behavior of a system by modeling static aspects of a system.

www.edlib.asdf.res.in

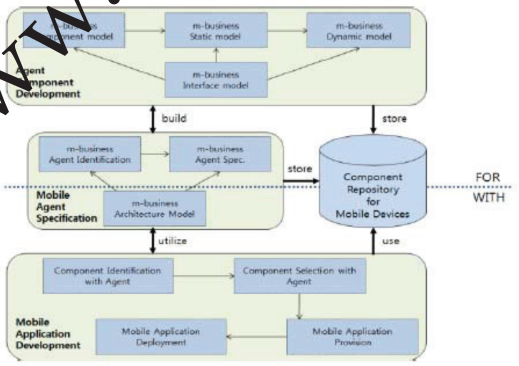

m-business Component model
m-business Static model
m-business Dynamic model
m-business Interface model
Agent Component Development
build
store
m-business Agent Identification
m-business Agent Spec.
m-business Architecture Model
Mobile Agent Specification
store
Component Repository for Mobile Devices
FOR
WITH
utilize
use
Component Identification with Agent
Component Selection with Agent
Mobile Application Development
Mobile Application Deployment
Mobile Application Provision

Fig 4. MASS-CBD process

3.4 Mata Information Description for Mobility Smart Servlet Container

The key idea to build server-independent support for mobile agents was that the HTTP Frame interface was not providing much more than what could be provided by the Servlet Specification. The HTTP Frame is only providing a hook for mapping an URI to an object inside of the web server that can respond to the HTTP requests. This can be accomplished with a servlet. The servlet technology provides a simple mechanism for extending the functionality of a web server, allowing URIs to be associated with object.

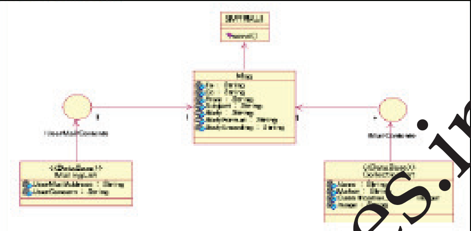
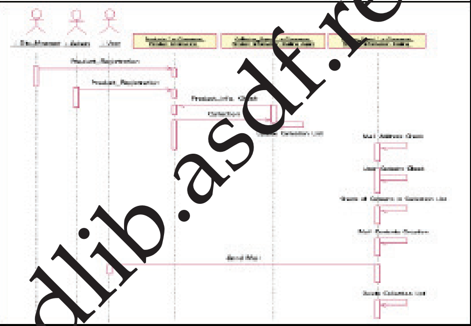
Item	Description
Agent Name	**E-Mailing Agent**
E-Business Type	**Information Broking Agent**
General Agent Type	**Information Agent**
Identification Code	**IMA-15-01**
Access Information	**Product DB, User DB**
Produce Information	**NewProductInfoContents**
Related Agent	**Collection Agent**
Information Model	
Operation Model	

Fig 5. MASS-CBD Mata Description for Servlet Container

www.edlib.asdf.res.in

In our research, we apply the MASS-CBD mata description for servlet container and its Mass repository as shown in figure5. These instances are called servlets, and are able to process requests sent to them. Currently there are many web servers supporting the Servlet Specification, and there are many stand-alone servlet engines that can be connected to the web servers for providing servlet functionality. Thus, migrating the wrapper into a servlet container would allow us to run the framework in any web server or servlet engine that supported the specification. Several issues were brought up considering this migration.

Our first concern was if it would not be too heavy to run the framework on a servlet engine. Our expectation was that it would not be, since the main component, which was the one being used, has a very small footprint and while running is also very lightweight, offering good scalability. The second point that we considered was that currently there is no uniform manner by which the agents can access the information on the web server. Making the agents behave as data sources associated with, since the servlet can forward the requests to the appropriate agent. The problem arises when an agent has to access the information stored locally on the web server. The most straightforward approach, and the one that we adopted, was to have the agents read the information as just an ordinary client. Although there is a small performance penalty, it is not very significant since the agents and the data source are in the same machine, and the loop back interface provides very large bandwidth. Figure 6 is a static and dynamic model for MASS modeling with UML.

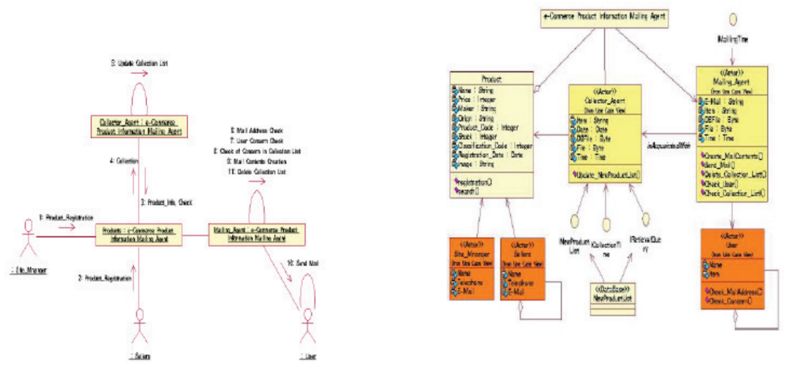


Fig 6. static and dynamic model for MASS modeling with UML

3.5 Implementation Architecture

The implementation of the servlet container follows the same base guidelines of the Mobility wrapper, but with same important changes.

First, the web server may be decoupled from the servlet engine, and from the servlet itself. In this case, the function of the web server is to provide a mapping between URLs and the resources, forwarding the requests to the appropriate servlets. Each request that corresponds to an interaction with an agent is forward to the Mobility Servlet Container, which then passes it to the appropriate agent. Secondly, the Security Component of our framework is instantiated and running, providing security features for the running agents and for the host. Figure 7 shows the approach. It should be noted that it is not necessary to decouple the web server from the servlet engine. If the web server supports the Servlet Specification by itself, then the container may be installed and configured on the web server itself. The biggest change on the architecture is not visible on the forwarding process taking place.

www.edlib.asdf.res.in

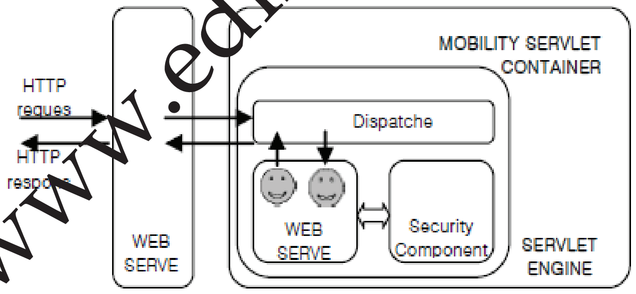


Fig 7. The mobility servlet container

The MASS framework provides the concept of services for agents. What this means is that an agent on arriving at a host can query which are the currently available services, and request an object implementing that service interface. That idea was used in our implementation. When an agent arrives at a web server, it may not only query the local web server, but it can also ask for a service instance that allows it to behave as a smart mobile device. Servlet, and publish information.

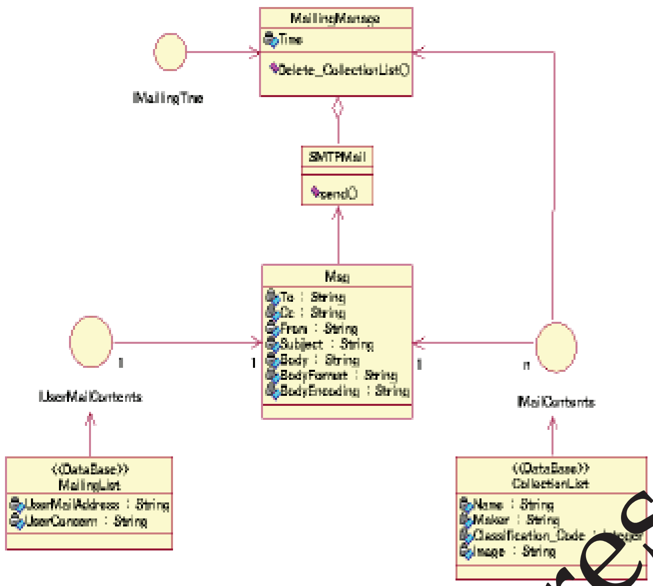


Fig 8. Implement the MASS servlet interface

When an agent requires an object that allows it to publish information, the object that is passed actually requires that the agent to implement the MASS servlet interface as in Figure 8.

This is important since allows the agents to distinguish between different clients, and act accordingly. Figure 9 shows the implementation of the MASS agents as servlet.

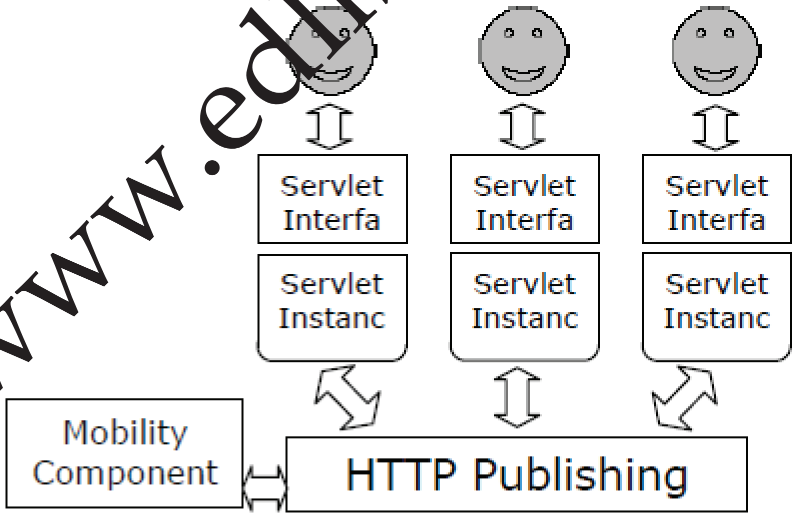


Fig 9. Implementation of the MASS Agents as servlets

3.6 Security

In our container, the Security Component is instantiated and provides several protection services for the agents and the host. This component allows the agents to migrate between hosts using SSL, which prevents tampering and eavesdropping on the contents of the agents [2]. It also implements a fine-gained

www.edlib.asdf.res.in

authorization mechanism that guaranties that only the agents with the correct permissions can perform certain operations, like reading directly from disk, or connect to other hosts in the network. Finally, the component implements cryptographic primitives that allow secure protocols for information gathering and comparison-shopping using out smart agent, that use mobile agents, to be implemented in an easy way. In our approach, the main problem to be solved concerning security is resource control. Since the Java complicated issue. We are currently investigating the possibility of using third-party resource control libraries with our system, and its implications in terms of runtime penalty.

3.7 Adding New Service

On interesting aspect of our system is that it allows different services to be instantiated and made available for the agents at runtime. It's makes the framework very flexible for implementing different features on different web sites.

For instance, let's suppose that a programmer wants to implement a marketplace for agents, where the agents negotiate between themselves, and consult and publish information on the web site. The programmer can use the basic architecture described here and configure the Mobility Component to load one or more of the different available components that implement several inter-agent communication mechanisms.

Our experience is that the MASS framework provides a nice approach for integrating mobile agents into existing web infrastructures. One of the most fascinating characteristics of the approach is that after having the basic infrastructure deployed (the container servlet), any new functionality can be easily introduced into the existing web infrastructure. The main limitation found with the framework has to do with resource control. Currently the framework is only usable in a secure way, on an intranet or on an extranet.

www.edlib.asdf.res.in

4. Evaluations

We currently have experimented with the framework in several web and mobile devices servers and servlet engines, with very positive results. For experimenting with the framework, we have built some prototype applications. Two of the most interesting smart applications are: We will now examine some performance results of the distributed agent-based crawler application. Although these are not extensive tests, they are useful to show the performance gains that can be expected by using a mobile agent-based approach while building distributed applications.

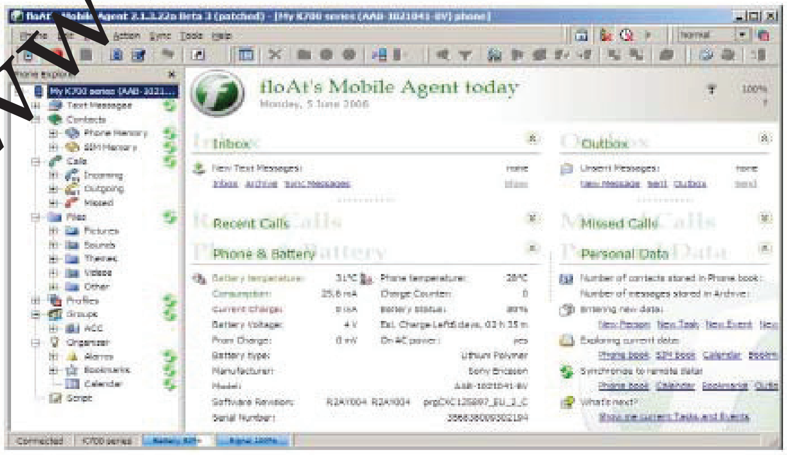
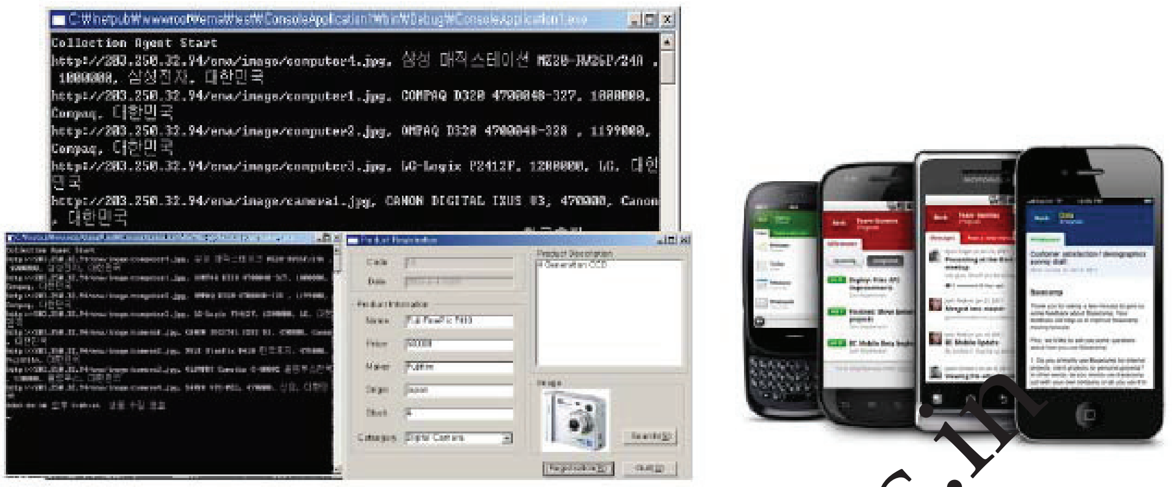


Fig 10. MASS Systems structure based on .net Framework



```
Collection Agent Start
http://203.250.32.94/ena/image/computer1.jpg, 삼성 매직스테이션 M820-JM26P/240 ,
 1000000, 삼성전자, 대한민국
http://203.250.32.94/ena/image/computer1.jpg, COMPAQ D320 4700048-327, 1080000,
Compaq, 대한민국
http://203.250.32.94/ena/image/computer2.jpg, OMPAQ D320 4700048-328 , 1199000,
Compaq, 대한민국
http://203.250.32.94/ena/image/computer3.jpg, LG-Logix P2412F, 1200000, LG, 대한
민국
http://203.250.32.94/ena/image/camera1.jpg, CANON DIGITAL IXUS U3, 470000, Canon
, 대한민국
```

Fig.11 Client MASS Execution to Find Cost-Effective Digital Camera

Figure 10 shows the MASS systems structures based on .Net frameworks. Figure 11 shows the clients execution sample on MASS. It is a sample to find the cost effective digital camera with MASS. We decided to setup an experimental framework that allowed us to test the agent-based approach vs. the client/server approach from indexing a web site. Another application that was developed was a distributed site-indexing system (a mobile agent-based crawler). In this application, the user specifies a site that he wishes to be indexed, and an agent jumps to that site performing the indexation locally. Because building the complete index of a site requires that all the pages of the site to be accessed, using an agent that performs the operations locally is much less expensive in terms of time and bandwidth than bringing the complete site to the local machine. The performance results of this application are discussed in the next section.

5. Conclusion

In this paper we have presented our experiences on using the MASS component framework for developing smart mobile agents. The MASS component framework in our work can be added into existing applications for agent-enabling them, providing the support needed for receiving and sending agents in an easy way. In this work, we have built an architecture that allows any web and mobile device server that supports the servlet specification to receive agents. The main features of the architecture are:

- Any web and mobile device server that supports the Servlet Specification is able to receive and send agents.
- The execution of the agents is restricted by proper authentication and fine-gain authorization mechanisms, so long as the existing security manager has not been modified in a way that is not compatible with the Java 2 security delegation mechanism.
- The agents are able of processing HTTP requests, having session information, as well as acting as regular servlets.
- It is possible to dynamically load new services, adding new features at run time. This makes the approach very configurable and capable of addressing different requirements of different sites.
- It has a small footprint and a lightweight execution environment.

In this paper we also discuss our experiences on integrating the framework components into off-the-shelf web and mobile device servers, enabling them to receive and send agents. Our approach does not involve deploying a stand-alone agent server that is not integrated with the web and mobile device server, nor does it require a specialized costume-made web and mobile device server. We provide a framework that is able

www.edlib.asdf.res.in

of using existing web and mobile device infrastructures, giving them the capability of using agents in their operation. Our framework also provides a very strong security model, with authentication and authorization mechanisms that control incoming agents, and cryptographic primitives that are useful to protect the integrity and confidentiality of the agents. This is essential if an infrastructure is going to be deployed on an open environment.

References

1) Chunlin Li and LaYuan Li, "A multi-agent-based model for service-oriented interaction in a mobile grid computing environment", Pervasive and Mobile Computing, Volume 4, Issue 5, October 2008, Pages 755-774
2) Tai-hoon Kim and Haeng-Kon Kim, "A Relationship between security enginerring and security evaluation" LNCS Volume 3046, May 2004, pp. 717-724
3) Giancarlo Fortino, Alfredo Garro and Wilma Russo, " Achieving Mobile Agent Systems interoperability through software layering" , Information and Software Technology, Volume 50, Issue 4, March 2008, Pages 322-341
4) Ronald Ashri, Michael Luck and Mark d'Inverno, "From SMART to agent systems development " , Engineering Applications of Artificial Intelligence, Volume 18, Issue 2, March 2005, Pages 129-140, Agent-oriented Software Development
5) Johnny Wong, Guy Helmer, Venkatraman Naganathan, Sriniwas Polavarapu, Vasant Honavar and Les Miller , " SMART mobile agent facility " , Journal of Systems and Software, Volume 56, Issue 1, 1 February 2001, Pages 9-22
6) Hassan Artail, and Elie Kahale, " MAWS: A platform-independent framework for mobile agents using Web services" , Journal of Parallel and Distributed Computing Volume 66, Issue 3, March 2006, Pages 428-443
7) Haeng-kon Kim, "A Study on the Agent Component Development support to PDA", Kips journal Volum 13, IssusD-1, pp37-50
8) Haeng-kon Kim and Eun-ju Park, "Component Specification Model for the Web Services", LNCS3983 pp.927-936
9) Sergio Ilarri, Eduardo Mena and Arantza Illarramendi, "A system based on mobile agents to test mobile computing applications", Journal of Network and Computer Applications. Volume 32, Issue 4, July 2009, Pages 846-865
10) Damianos Gavalas, George E. Tsekouras, Christos Anagnostopoulos, " A mobile agent platform for distributed network and systems management" , Journal of Systems and Software. Volume 82, Issue 2, February 2009, Pages 355-371.

www.edlib.asdf.res.in