iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** XI **Month of publication:** November 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XI November 2017- Available at www.ijraset.com

A Survey on Mobile Agents

Prashant R. Desai[1], Dr. Naveen Kumar Jayakumar[2]
[1, 2] *Department of Computer Engineering, Bharati Vidyapeeth Deemed University, Pune- India.*

Abstract: Mobile agents are an imminent technology for developing applications in mobile, distributed and pervasive computing. They offer a broad spectrum of features like autonomy; migrate to remote computers and process data to save remote connections. Mobile agents can be applied to a broad domain: network management, sensor network, grid computing, web services, and data mining. Numerous mobile agent systems have been developed, while some have been outdated and no longer in use others are in a continuous process of development, offering a variety of features. With the introduction of new computing platforms, to take advantage of mobile agent technology one must develop distributed applications that can cater to the needs. In this survey paper, we evaluate some of the existing mobile agent systems.
Keywords: Mobile agent, migration, mobility, mobile agent framework, distributed.

I. INTRODUCTION

With the ever-increasing growth in information and internet, the issue of how we handle the storage, retrieval, and process the data has become a significant problem. Mobile agents are such programs designed to move from one machine to another to service requests [1]. These agents can be utilized to enhance an existing architecture and perform the autonomous computation. A mobile agent executes on a machine that provides it with the necessary resources if the machine does not have the necessary resources or the agent requires resources of another machine, its state information is migrated to the appropriate machine, and the agent resumes execution on that machine[1][2]. The advantage of using mobile agent includes low bandwidth consumption since they migrate when needed and are relatively smaller. They can clone, and multiple copies exist in an instance providing parallel execution and synchronization. Mobile agents are an extension of the client-server architecture, in a typical client-server scenario, a server machine provides a set of services, and a client machine requests those services. Client and server communicate via message passing [3] [4]. The client is limited to the services provided by server these can be overcome using mobile agents to assist clients and in complete automation. In this survey paper, we will provide an up-to-date analysis of mobile agent systems

II. MOBILE AGENT SYSTEMS

A. ARA

Stands for (Agent for Remote Action) are a mobile agent framework developed at the University of Kaiserslautern. The primary goal is to provide code mobility to existing programming languages without having to disrupt the traditional concepts of algorithms, syntax, and logic [5]. The interior architecture of the ARA mobile agent is composed of core and interpreters. The mobile agents can be coded in some interpreted language and then executed by the own interpreter using particular runtime system called the cores. The core is the critical part of the ARA architecture and facilitates concepts such as migration, allowances and service points. The task of the interpreter is to segregate language-specific issues, and that of the core is to implement the language-independent functionality [5].

Currently, the ARA core is compatible with C/C++ programming languages an adaptation for Java and other programming languages such as Pascal and Lisp is under development. ARA agents are executed as parallel processes and transformed into mobile representation using a fast thread package. The core executes the threads and not the host operating system processes; the core entirely governs the agent.[6] The process of adapting the ARA core to the interpreter of any programming language requires the stubs for functions of the core API and the up calls to the core. ARA uses a distinct core call, arrange that provides the agent mobility at any point of execution. ARA had a very flexible security, [6] domains protecting the resources can be migrated to other domains and resource provisioning of these can be controlled to a fine-grain level. However, this architecture described above has a drawback; it does not allow agent interoperability. ARA mobile agents are compatible with various operating systems such as Linux, Solaris, and SunOS, and are widely used in mobile computing.

B. Agent TCL/D' Agents

Also known as D' agents is a mobile agent system developed at the Dartmouth College to overcome the limitations of existing mobile agents systems, such as inadequate security policies, inapt migration facilities, and support for diverse programming languages. The architecture of TCL agent is designed to provide support for Tool Command Language (TCL) and based on the

Typescript server model. On evolving into D'Agents, it provides support for a high-level programming language like Java [7]. The architecture can be divided into four layers; the most underlying layer is responsible for transport, the third layer is the server layer and regulates the flow of mobile agents. The second layer is the interpreter, which provides the execution environment for the supported programming languages, and at the first level, the agent itself resides. There are two types of mobiles agents: those that migrate from machine to machine accessing resources provided by the host and those that do not migrate and are stationary on a system providing services that are not provided by the host machine. Agents use the agent_jump command in TCL to migrate. The agent jump command works by saving the execution state of the agent and migrating it to the destination. At the target, the server loads the appropriate interpreter for the migrated agent restores the agent's execution state and begins execution. Other commands provided by TCL are agent_meet, agent_accept, agent send, and agent_receive. Agent_meet and agent_accept are used to establish a connection between agents.Agent_meet is used while a source connects first time with a destination. Agent_accept is used by the destination machine to accept agent_meet from the source and decide whether to establish or reject the connection.Agent_send and agent_receive are used to exchange messages between source and destination [7] [8]. Security in agent TCL is provided to various degree. Authentication, digital signatures, and encryption are provided using Pretty Good Privacy (PGP). A resource manager is assigned to resource allocation and management. A request for resources is redirected from the agent to the resource manager which verifies the agent's authentication. If the agent fails to confirm its identity, it is denied access to the resources. TCL agents are widely used information retrieval systems and information management systems.

C. Concordia

Concordia is a middleware infrastructure developed at the Mitsubishi Electric Information Technology Centre America (MEITCA). It supports the development and maintenance of applications, which rely on mobile agents for accessing information. It is written in high-level programming language, Java. The Concordia execution environment consists of a Java Virtual Machine (JVM), a Concordia server programmed in Java and a mobile agent in the network. Communication, persistent storage, security, and storage are the various services offered by the components of a Concordia system [9]. The Conduit server is responsible for agent mobility. An agent initiates a transfer mechanism by provoking the method provided by the Conduit server, which suspends the execution state of the agent and creates a persistent image to be migrated. Concordia's mobility framework offers an essential feature by supplying state information of the agent on which locations it has relocated and where it needs to go [10]. The itinerary is a type of data structure in Concordia that store information about the destinations (the places the agent has to reach) and the task it has to perform. Concordia uses Java Object Serialization (JOS) for migrating the mobile agents. The agent is serialized in the JOS format when the Conduit server transferred the agent and deserialized at the destination. Agents communicate through asynchronous distributed events or the collaboration. Agents register with the Event Manager to receive events. Concordia provides a facility for group communication, allowing multiple occasions to be sent to a group Concordia's security model [10] offers two protection modes: integrity protection of agents from tampering and protecting the server resources from unauthorized access. Security manager component provides security to server resources and authenticates each agent before it is granted access to resources.

D. Mole

It is a java based mobile agent system that provides agent mobility for distributed systems. In Mole ecosystem, the agent model is based on agents and places. It provides two type of migration, strong and weak migration. During strong migration, the system saves the complete state of the agent (execution state and data) and transfers it to the new location where it has to be restored. This technique is handy for programmers who want full transparency, but it incurs more cost on the system. In weak migration, the data state of an agent is transferred, and the programmer can control its size [11].The developer is also responsible for coding the execution state in program variables. The programmer needs to provide a start method that decides where to continue the execution of migration. Mole uses weak migration as its default because one of its objectives is to run any machine running a VM. Standard Java VM does not support recording of threads that are required for strong migration. When an agent invokes a migrate call, the threads in execution are suspended. After the suspension, an independent representation of the object is created and serialized. After that, the object is migrated to the destination location, and a success message is relayed back. The communication between the agent and the service is RPC type client-server based. In Mole system, agents communicate amongst themselves using a concept called sessions. User and agents communicate using sessions or RPC. In the session, based communication a unique identifier identifies each agent called the 'badge.' Once a session is established between two interested communication parties, they can communicate using message passing or remote calls. The primary reason to use session in Mole system is to achieve synchronization and support 'stateless'/ 'stateful' interactions. Mole's event-driven model supports asynchronous communication. Mole's security model is

based on 'Sandbox, in these model [11] agents are granted access to all the system resources, control, provisioning and secure abstractions inside the agent system. Service agents can provide access to legacy software; by using Java's native code interface since they are stationary they can be only initiated by the administrator.

E. Voyager

Voyager was developed at the Object Space Company and is a java based Object Request Broker (ORB). Its primary goal is to provide flexibility for developing distributed applications while providing extensibility and flexibility to the programmers within the system. Voyager supports distributed computing architectures such as RMI, CORBA, and DCOM for client-server applications. Voyager is developed in Java and uses java syntax to create and transfer objects amid applications. It locates the agents and sends a message via message passing transparently without interrupting the execution. Its significant advantage is it supports both conventional client-server and agent-based architectures. Objects are the primary blocks which reside in the Voyager application. The application determines the objects infrastructure for communication and services. The threads created in the Voyager application are responsible for garbage collection manage TCP/IP messages and timing. Every voyager system has a host port and communication port .Agents are remote objects in voyager systems and can exist outside the local application's address space. Applications communicate with these objects by creating virtual copies of it. The virtual objects reference the locations of the remote object when the programmer invokes them. Virtual objects are used to route messages between the remote object and the caller, including returning values through to the caller. Virtual objects can perform various tasks such as remote object creation, connection with remote object and code and object migration. Voyager supports different types of message passing methods such as synchronous, in which the message is blocked until the return value is not acknowledged. One-way message passing in which no reply is acknowledged. Garbage collection takes place by using 'ping' on the remote objects to collect details about its activity. Voyager security model is implemented using SSL adapters and firewall tunneling using HTTP or SOCKS protocol to provide a flexible and lightweight security framework.

F. Mobility-RPC

Mobility-RPC brings code mobility to any application and is developed in Java programming language. Its primary task is to move objects, tasks, execution states and the code between JVMs or nodes at runtime. The primary use of Mobility-RPC is to move code and objects between machines in a single application without having to deploy code on the remote machine. Mobility-RPC can be used write self-migrating objects within a network and for calling third party libraries remotely [12]. It is designed to take advantage of application architecture that is not possible through traditional RPC architecture along with the traditional mobile paradigms: code mobility, remote evaluation, and mobile agents. Mobility-RPC uses Kryo serialization java library over the inbuilt java library, which outperforms it in speed and data sizes. It serializes objects before sending it to the remote machine. It uses Objenesis for object deserialization without having to call its constructors. The communication protocol used in Mobility-RPC is designed independently and does not depend upon the underlying transport protocols instead; it is coupled with other transports such as TCP, UDP or SCTP [12] . The protocol has the following characteristics: message-oriented, the formats of the messages are predefined and encapsulated. Stateless, connections maybe ad-hoc or persistent but it does not require synchronization for the actual message to be sent. Multiplexing, both client-side and server-side applications are multithreaded hence the UUIDs are encoded into messages as a response. The session, the protocol although being stateless supports stateful interactions between various applications and the SessionIdentifiers are encoded into request messages.[13] Several clients can access the same session ids; the clients will be loaded into the same class and access data stored in the static fields in those classes, on the remote machine. The protocol can be implemented with the SCTP transport protocol but since it is not widely used it currently used with TCP.

III.COMPARISON OF MOBILE AGENT SYSTEMS

Based on the previous section, the following table highlights the advantages and disadvantages of each mobile agent system.

Mobile Agent System	Advantages	Disadvantages
ARA	• Concurrency is achieved using a fast thread package. • Cloning allows duplicating their internal state. • Cores act as 'service points' for agents in communication.	• No authentication process. • No protection from remote failure. • Supports only TCP protocol for transmission.

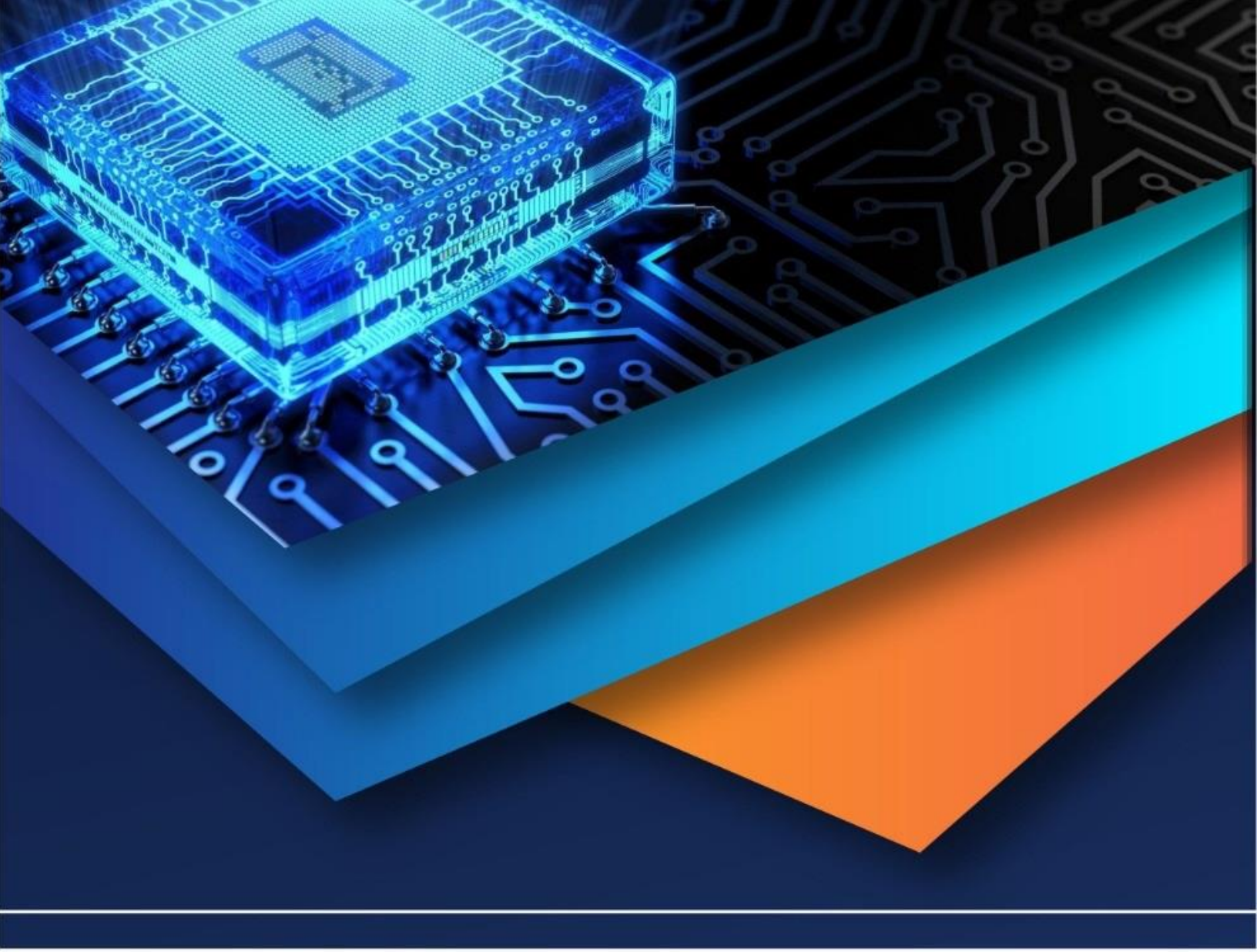
Agent TCL/D' Agent	• Extensive support for migration. • Robust security mechanism.	• TCL is highly inefficient programming language compared to C or Java. • No support for code modularization or debugging.
Concordia	• Flexible mobile agent • Support for agent persistency and collaboration. • Robust security mechanism.	• Large overhead generated for an agent since it has two images in storage. • Synchronizing name services is a problem because of a large number of agents present.
Mole	• Runs on any JVM. • Sessions allow for proper communication amongst the agent.	• Supports only Java. • Support for only strong migration. • Programming is not easy.
Voyager	• Is not syntax dependent hence, IDL is not required. • Support for distributed computing platforms such as RMI, CORBA and DCOM.	• Only supports Java programming language. • Not a strong message passing system.
Mobility-RPC	• For single, thread request 9% higher throughput than RMI [13]. • For single thread request, 8% lower latency than RMI [13].	• Only compatible with java.

IV. CONCLUSION

In this survey, we reviewed and studied several mobile agent systems; all the systems discussed focus on the framework and execution environment provided to the mobile agent, the communication techniques used for message transfer, and security mechanism. The use of mobile agents is beneficial for client-server computing for also raises the issues of flexibility, efficiency, and reliability of the system. For the majority of the systems, important issues such as agent-OS interaction, use of persistent storage for images of agents and fault tolerance was not discussed or mentioned in their research. The applications discussed did not fully utilize the agent's capabilities unless it was the mobile agent's framework. A potential application of mobile agents would be in load balancing and application migration in Big Data. The mobile agent approach is exciting and improving over time and may provide better network services in the future.

REFERENCES

[1] D. Chess, C. Harrison, and A. Kershenbaum, "Mobile agents: Are they a good idea?," Mob. Object Syst. Towar. Program. Internet, pp. 25–45, 1994.
[2] D. Johansen, "Operating system support for mobile agents," … Oper. Syst. …, no. 100413, pp. 1–6, 1995.
[3] D. Kotz and R. S. Gray, "Mobile Agents and the Future of the Internet," Search, vol. 33, no. 3, pp. 7–13, 1999.
[4] M. O. Oyediran, T. M. Fagbola, S. O. Olabiyisi, and E. O. Omidiora, "A Survey on Migration Process of Mobile Agent," vol. I, no. November, 2016.
[5] H. Peine and T. Stolpmann, "The architecture of the ara platform for mobile agents," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 1219, no. 1219, pp. 50–61, 1997.
[6] H. Peine, "Security concepts and implementation in the Ara mobile agent system," Proc. Seventh IEEE Int. Work. Enabling Technol. Infrastucture Collab. Enterp. (WET ICE '98) (Cat. No.98TB100253), pp. 236–242, 1998.
[7] R. Gray, "Agent Tcl: A transportable agent system," Proceeding ciKM Work. Intell. Inf. Agents …, 1995.
[8] R. Gray, G. Cybenko, D. Kotz, and D. Rus, "Agent Tcl 1 Overview," pp. 1–53, 1996.
[9] Mobile agent computing , a white paper. [Online] (Availabile)https://www.cis.upenn.edu/~bcpierce/courses/629/papers/Concordia-WhitePaper.html
[10] Concordia: an infrastructure for collaborating mobile agents [Online] (Available)https://www.cis.upenn.edu/~bcpierce/courses/629/papers/Concordia-MobileAgentConf.html
[11] J. Baumann, F. Hohl, K. Rothermel, and M. Straßer, "Mole–Concepts of a mobile agent system," World Wide Web, vol. 1, pp. 123–137, 1998.
[12] Mobility-RPC, [Online](Available)https://github.com/npgall/mobility-rpc.
[13] Mobility-RPC benchmark, [Online] (Available) https://github.com/npgall/mobility-rpc/blob/master/documentation/PerformanceBenchmark.md



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY