Scalable fault tolerant Agent Grooming Environment - SAGE

H. Farooq Ahmad, Hiroki Suguri
Multi Agent Systems Group
Communication Technologies (Comtec)
2-15-28 Omachi, Aoba-ku, Sendai, 980-0804, Japan
{farooq, suguri}@comtec.co.jp

Arshad Ali, Sarmad Malik, Muazzam Mugal,
M. Omair Shafiq, Amina Tariq ,Amna Basharat
NUST Institute of Information Technology (NIIT)
National University of Sciences and Technology
(NUST) ,Rawalpindi, Pakistan
{arshad.ali,sarmad,omair.shafiq}@niit.edu.pk
muazzam_mugal@yahoo.com
{aam_naa, aamna15}@hotmail.com

Abstract

Scalable fault tolerant Agent Grooming Environment (SAGE) is first open source initiative in South-Asia. It is a multi-agent system which has been developed according to FIPA (Foundation for Intelligent Physical Agents) 2002 specifications. SAGE has been designed with a distributed and decentralized architecture to achieve fault tolerance and scalability as its key features. Due to these characteristics, SAGE is not only regarded as 2nd generation Multi Agent System but also provides a competitive edge over other platforms.

Keywords

Multi-agent System, Agents, Fault tolerance, Scalable

1. Introduction

The agent platform provides an environment in which agents can execute and perform their tasks. Agents have the ability to perceive their environment, maintain knowledge, reason about and execute particular actions to solve specified tasks and achieve their goals. The design of Multi-Agent System (MAS) is considerably more complicated than the single agent system. Multi-Agent System requires additional considerations including communication mechanism, an environmental knowledge maintenance and sociability to support inter-operations. These agent systems are becoming necessary component in semantic web, ubiquitous and grid computing especially for the management of information and data.

2. Features

SAGE achieves the aim of a fault tolerant Agent Platform by offering a decentralized architecture based on the notion of Virtual Agent Cluster, which provides fault tolerance capability by using separate communication layers among different machines. The Virtual Agent Cluster works autonomously, regardless of the external environment events, providing a self healing, proactive abstraction on top of all instances of multi-agent systems. Also the architecture ensures high assurance using peer to peer architecture which brings scalability, fault tolerance and load balancing among distributed peers.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

The main components are, AMS (Agent Management System) that manages the platform, DF (Directory Facilitator) which provides yellow pages service and MTS (Message Transport Service) for messages delivery in agent platforms. Also, Agents communicate by using Agent Communication Language (ACL) messages. The SAGE core architecture is shown in Figure 1. The decentralized architecture of SAGE also embeds the capability of self-monitoring at the system level by allowing the agents to internally monitor themselves as well as externally monitor other agents. The external monitoring

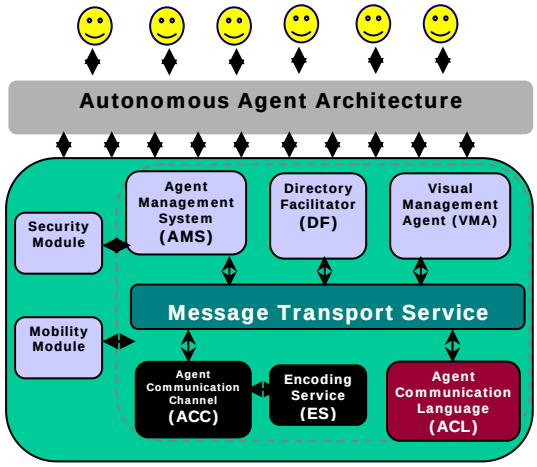

Figure 1. Main Architecture

capability has been incorporated in SAGE by allowing all the instances within the Virtual Agent Cluster to send heart beats (Hello messages) to each other to check the liveliness of peer instances of Multi-Agent System. One of the features of SAGE is the ability of agents to be self descriptive as each sage-agent keeps its own descriptive information as attributes, and makes it available through system agents of SAGE e.g. the Directory Facilitator (DF) or the Agent Management System (AMS). The system framework then makes the agents dynamically discover and interact with each other. The core components of SAGE may span on multiple machines and acting like a single virtual agent cluster. The failure of one machine does not affect the agent system working on peer machines. One of the most important aspects of the architecture is its independence or autonomy. All peers of multi agent system run on separate machines. These peers autonomously provide services to their local application agents. All peers keep their registry information local and in case of failure of any remote peer, all remaining peers will keep working, providing illusion of autonomy inside VAC.

Agent Management System (AMS) is the mandatory component of an Agent Platform. It exerts supervisory control over the platform. AMS can either request or can forcefully enforce functions on other agents. It performs management functions like create, kill, suspend and resume agents as well as controlling the various agent platform parameters. Directory Facilitator (DF) is responsible to provide yellow-pages directory service. Agents may register their services to the DF or query the DF to find out what services are offered by other agents. Agent is responsible to provide information related to service parameters like servie_type, service_name etc. Furthermore, an agent can also deregister or modify its service. VMA is an agent that offers a graphical interface to platform administration and monitoring. The agent offers many services that show the state of the Agent Platform as well as it offers various tools that are used to perform administrative interaction with the AMS, DF and to test application agents. It also shows the details of the agents that reside inside the platform. Message Transport Service is the backbone of Multi-Agent System. It supports the sending and receiving of ACL messages between system and application agents. The agents involved may be local to a single Agent Platform or on different Agent Platforms. Two modes of communication are involved for message transportation in SAGE which includes Inter-platform and Intra-platform communication. The ACL module is responsible for creation of a message that is understandable by all entities involved in the multi-agent system. All agents create ACL messages using some pre-defined rules and it is sent to the required destination. At the reception end, the agent will take its decision based on the ACL Message. Agent Communication Languages provides agents with a means of exchanging information and knowledge, which is the essence of all forms of interaction in multi-agent systems. ACL is a language that specifies message format and include descriptions of their pragmatics i.e. the communicative acts or intentions of agents. Furthermore, every agent has common semantics to talk with each other which is based on a shared ontology.

3. User-base

Users may employ and customize SAGE multi-agent system according to their application areas. For example, if it is required to be deployed in medicine field, users need to create ontology, filling the contents and use these files in creating Doctor/Patient agents. These sender and receiver agents are created after inheriting from abstract agent file. It can be used in various areas from meeting scheduler to research and people management etc.

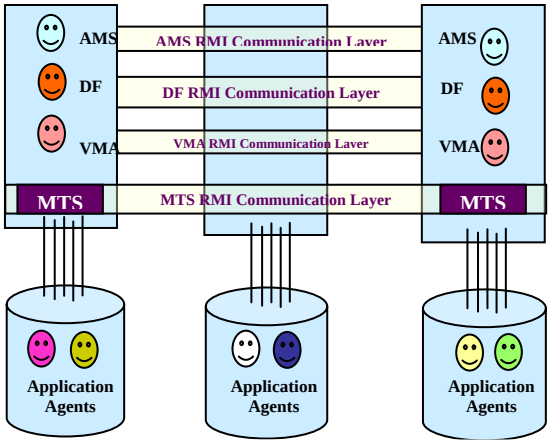

Figure 2. Decentralized and Distributed view of SAGE

SAGE as decentralized multi-agent system gives an organizational advantage as well. Users need no complicated setup, they can simply develop and run their agents without knowing any central coordination. This lowers the barrier for users to develop and deploy agents. Keeping these advantages in mind, the SAGE Agents are designed not to assume any central entities on the multi agent system framework.

4. Strong and Weak Points

SAGE has been made to run stand-alone. Its packages include the utilities of HSQL along with an option to use Microsoft Access for database support. The strong points of SAGE includes its property of decentralized architecture, priority based queues in message communication to achieve fault tolerant and scalable behavior. The weak points are to include more autonomous and social behavior in agents. Also, there is a need to make it more user-friendly by allowing graphical ontology creation and programming communicating agents that execute on platform with utilizing minimum resources.

5. Related Work

The most widely used FIPA compliant platforms include JADE, FIPA-OS and Zeus. SAGE can be compared with these agent platforms as they have a centralized architecture and in case of failure of the main container, whole system will come down. In order to provide fault tolerant and scalable behavior, SAGE is decentralized and distributed as shown in Figure 2. Also, priority based queues have been used in message communications to avoid congestion and failure problems. The SAGE architecture provides tools for decentralized runtime agent management, directory facilitation monitoring and editing, message exchange debugging and agent life cycle control. It overcomes the problems inherent in first generation multi-agent systems by providing support for a decentralized architecture.

6. Results

We have achieved fault tolerance and scalability as shown in Figure 3(a) and 3(b).

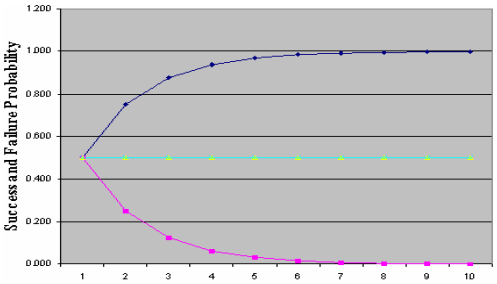

Figure 3(a). AMS Failure probability

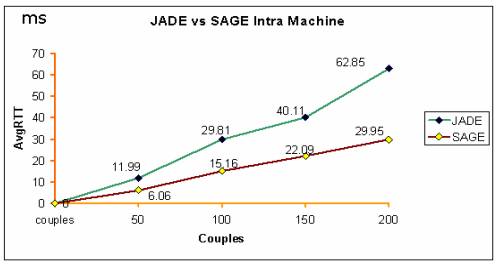

Figure 3(b). MTS Performance

7. Future Work

The future work includes the customization of SAGE as a lightweight agent platform for PDAs and other WAP users over mobile phones. Also the work for agent behavior layer on top of SAGE is in progress to facilitate agent programming and inheriting fundamental agent behaviors in application agents. Lastly, the possibilities of performing tasks by making groups of agents with varying rationalities is under-way to explore and integrate the execution of agents in a team-work.