Internet Multi-Tier Information Retrieval Agent System Architecture

N. A. EL-RAMLY\textsuperscript{1}, H. M. HAR\textsuperscript{2}, M. A. ABD-ELWAHED\textsuperscript{1}, A. M. TOLBA\textsuperscript{1}

\textsuperscript{1} Computer Science and Math Dept, Faculty of Science, Menoufia University, EGYPT
\textsuperscript{2} Systems Engineering and computers, Dept. Faculty of Engineering, Azhar University, EGYPT

Abstract- The explosive growth in the volume of information available on the Internet computing environments requires new architectures for information retrieval systems. Typically, in these environments, the information resources are dynamic, heterogeneous and distributed. In addition, these computing environments are open, where information resources may be connected or disconnected at any time. This paper presents the design and implementation details of N-Tier Agent-Based Architecture For Web Information Retrieval. It will also explained that how these agents communicate with each other in order to accomplish information retrieval through an AMR (Agent Messaging Router) which is a part of a software package called JAT (Java Agent Template). JAT provides the basic infrastructure for creation of agents and their communication.

Keywords: Software Agents, KQML, JAT, Information Retrieval, Mobile Agent.

1 Introduction

Each year, Internet-oriented applications become more popular. To fully use the Web's advantages, new design and programming paradigms need to be developed. Traditional distributed applications assign a set of processes to a given execution environment that, acting as local resource managers, cooperate in a network unaware fashion. In contrast, the agent-based programming paradigm defines applications as network-aware cooperating clusters of agents. Several information retrieval systems are widely used on the web. Many of them use a central database. When a browser sends a request, a Common Gateway Interface (CGI) or Active Server Page (ASP) program on the server side executes a search on the database to process the request. This approach has several inadequacies which can easily be overcome by a design that conforms to the agent-based programming paradigm [10]. Utilization of a distributed database architecture, extended user facilities, reduced network consumption are some of the advantages gained by an agent-based design.

In this paper we explain in detail the implementation issues of a multi-agent information retrieval system, which is developed using the JAT package as the basic agent template to create agents and an AMR for agent communication. The main objective of this work is to build an information retrieval system that is capable of helping users to locate and retrieve information from distributed resources in an open environment. The main design principles of the system are that the system should: 1) provide the users with an integrated view of information, 2) provide relevant information to the users interest, 3) pro-actively search for information, 4) monitor any possible changes to the information resources, 5) provide information to the users within a bounded-time, 6) be aware of the new information sources that may enter and old information sources that may exist the environment. This paper presents a multi-tier multi-agent system architecture that is build to accommodate the design principles.
2 Software Agents

It is difficult to find a simple and universally accepted definition for an agent that includes all its attributes [2]. We can define an agent as a software module, eventually equipped with AI mechanisms, capable of solving - autonomously or in cooperation with other agents - a certain problem or carrying out a particular task [1]. Attempts to characterize agents have resulted in a list of attributes.

- Autonomous - acts independently
- Adaptive/Learning - adapt to evolving environment
- Mobility - across machine boundaries
- Intelligence - can reason
- Persistent - over time
- Goal oriented
- Communicative / Collaborative - interacts with other agents
- Flexible

Agents are able to perform numerous functions or activities without external intervention over extended periods of time. They also tend to be small in size. They do not constitute a complete application, instead they form a part of the system by working with other agents [1,2].

2.1 Application Domains of Agents

Their interesting features make agents an attractive topic in computer science. Objects and distributed object architectures, adaptive learning systems, artificial intelligence, expert systems, genetic algorithms, distributed processing, distributed algorithms, security and social environments are just a few areas in computer science that agent can be integrated to easily. There are basically three different domains where mobile agents have potential deployment [3]. One is data-intensive application where the data is remotely located, is owned by the remote service provider, and the user has specialized needs. Here, the user sends an agent to the server storing the data. The second domain is where agents are launched by an appliance. The third is for extensible servers, where a user can ship and install an agent representing him more permanently on a remote server.

2.2 Problems Agents Solve

An agent architecture solves many classical problems. Mobile agents solve Client/Server network bandwidth problem. When performing a Client/Server query it may be necessary to create many transactions and each create network traffic and decrease available network bandwidth [3]. By creating an agent and sending the agent to the server side, network consumption is reduced. The agent will do all the queries at the server side and will come back with the results.

An agent architecture also solves unreliable network connections. In most applications today, the connection must be alive during the process. If the connection goes down, the client has to start the transaction again. Agent technology allows a client to dispatch a transaction when it is online. The client then goes offline while agents process the job for the client, and when the client gets online again, an agent sends the results to the client.

3 Java Agent Template

JAT (Java Agent Template) [4] is a package of programs written in the Java language that allow users to quickly create new software agents that communicate robustly over the Internet. JAT provides a basic infrastructure in which agents register with an AMR facilitator using a name and password, connect/disconnect from the Internet, send and receive messages, transfer files, and invoke other programs or actions on the various computers where they are running. Figure 1. depicts the JAT approach.
JAT facilitates especially construction of agents that send and receive messages using the emerging standard communications language, Knowledge Querying and Manipulation Language (KQML) [5]. The communications are built on open Internet standards, TCP/IP, SMTP, and FTP. However, developers may easily build agent systems using other agent languages using JAT.

3.1 The JAT Architecture

The JAT architecture is organized as a hierarchy of increasingly specialized layers shown in Figure 2., so that developers can select the appropriate layer from which to start building their systems. Thus, a developer who wants to utilize TCP/IP communications but does not want to use KQML can use only the Abstract and Base layers as described below [4].

**Abstract Layer** provides the collection of abstract classes necessary for JAT implementation. Although JAT assumes all connections to be made using TCP/IP, one can implement different protocols such as UDP by extending the Abstract Layer.

**Base Layer** provides basic communication based on TCP/IP and the abstract layer. There is no restriction on the message language or protocol. The Base Layer can be extended, for example, to allow inputs from sockets and output to files. The Base Layer can also be extended to provide agents with multiple message ports, etc.

**KQML Layer** provides for storage and parsing of KQML messages. Extensions to the KQML standard, proposed by the Center for Design Research [6], are implemented to provide a standard protocol for registering, connecting, disconnecting, etc.

**Router Layer** provides name registration and message routing and queuing for agents. All agents send and receive messages via the Router, which forwards them to their named destinations. When an agent intentionally disconnects, or accidentally crashes, the Router stores incoming messages until the agent is reconnected. The Router is particularly important for applet agents, which can only initiate socket connections with the host that spawned them, due to WWW and Java security restrictions.

**Protocol Layer** on top of Router Layer will support diverse standard internet services such as SMTP, FTP, POP3, HTTP, etc both for stand alone applications and applets. Current version supports SMTP and FTP but other protocols can be easily extended from Protocol Layer. If an agents is expecting to transfer non-sentential, lengthy data or needs to send KQML message through email, Protocol Layer will be a good starting point.

4 Information Retrieval Model

Our model presented in this paper focuses on viewing information retrieval as problem independent of its structure, for which a single or a group of agents can participate in an open environment. Our system architecture proposed in this work is a multi-agent multi-tier system architecture, in which each agent is autonomous, goal-driven, cooperative, coordinated, rational and able to communicate with other agents to fulfill the users’ needs. The system architecture includes five types of agents: Interface, Broker, Domain, Mobile and Information agents as shown in Figure 3. At the front end of the system Interface agents provide flexible mediation services between users and the information environment. They interact directly with the users, accept queries and provide results. At the back end of the system, Information agents act as information providers for the interface agents, keep track of the distribution of the information resources and the dynamic nature of the information resources in terms of their contents. At the middle-tier of the system, Broker agents keep track of the existing agents of all tiers.
and their capabilities to facilitate the appropriate cooperation among them. Domain agent supervises the activities that occur within a domain, Mobile agents are the components within the architecture which can migrate between network nodes. The user interacts with the system through a Graphical User Interface (GUI) to submit queries and get results. The agents communicate with each other by means of high-level agent query language, using Knowledge Query Manipulation Language (KQML) [5]. This approach provides a high system performance when agents retrieve similar information or achieve more than one goal simultaneously, through the interface agents. Through the broker agent, the copes well with new information sources that may join or disjoint the environment. In addition, it is responsible for the scalability of the system as the number of agents increases. This approach also facilitates monitoring any possible changes to the information resources and proactively searches for information, through the information agents.

5 Agents Functionalities

This section describes the functionalities of each type of the agents in the proposed system architecture.

**Interface Agent:** The interface agent is the user’s intelligent interface to the system and allows the user to interact with the information systems environment [7]. The interface agent provides graphical interfaces for the user to submit both queries and the desired constraints, to provide feedback, and to display results. The interface agent accepts queries from the user or from other interface agents. The query is described by a set of words that include the topic of interest representing the goal to be achieved and a set of constraints. The user’s interest includes information ‘quality’ and the desired response time. A goal might be decomposable into sub-goals; each might be locally achievable or require interaction with the user and/or other agents. The interface agent generates the solutions to each goal and chooses a solution that best fits the user’s needs. The interface agent learns about the user’s topic of interest and preferences. It also learns about other agents’ capabilities in relation to the user’s topic of interest, the desired information’ quality and the expected response time, each time they involve in interaction. The interface agent is capable of accessing a local database, storing information for future queries, and maintaining models of the other agents (user and software). Therefore, the interface agent may act as an information provider to other interface agents. The interface agent is able to interact directly with the Information agents or indirectly through the broker agents, for which the efficiency of the system can be enhanced when retrieving similar queries, and preventing the system from collapsing if the broker agent in the middle tier is malfunctioning or disappears.

**Broker Agent:** The broker agent acts as a service provider that pairs interface agents seeking information about specific topics with interface and/or Information agents that are able to provide...
that information. As the interface and the information agents come online, they advertise their capabilities to the broker agent, using KQML. The broker agent accepts advertisements from other agents to confirm their existence and capabilities, and organizes them into groups based on their topic of interest. The broker agent knows the unavailable agents either by receiving notification messages or by setting an expiration time for acknowledging their existence. The broker agent responsibility has a very advantageous impact on the system’s functionality and performance. Firstly, users don’t need to interact with the broker agents. This supports the transparency aspect of the system. Secondly, the agents are grouped based on their topics of interest. This allows the interface agents to direct requests and messages to the interested agents only; therefore, the interface agent’s knowledge about the network structure and the network traffic required to achieve its goals are reduced. Thirdly, the architecture allows new agents of different types to join the system and existing ones to disjoin. This makes the system operates in an open environment and achieves the scalability of the system by a seamless integration as the number of agents grows.

**Domain Agent:** A domain agent is a static agent that supervises the activities that occur within a domain; a domain is a logical boundary used to delimit nodes, agents and resources into manageable and distinct entities. The domain agent has a number of responsibilities to information resources, users and agents within a given domain:

- Providing a migration service to mobile agents wishing to leave the domain.
- Providing an authentication and validation check on mobile agents wishing to enter the domain.
- Mediating global access to information resources; before a mobile agent can access an information resource it must obtain permission from the domain agent.
- Ensuring that the domain does not become swamped by mobile agents.

The domain agent, then, is the key force within a domain and is ultimately responsible for ensuring that security is enforced within the domain and that agents can communicate with information resources and other agents.

**Mobile agent:** Mobile agent is the component within the model which can migrate between network nodes [3]. It is the mechanism by which the user exercises control over their own distributed information resources and gain access to other, shared information resources. Mobile agents are equipped with a set of user-defined goals which describe the nature and limits of their functionality, for example, a resource discovery agent is a mobile agent with a different goal set than, say, a navigation assistant agent. It is probable that, in addition to the limits on the functionality that users place on their mobile agents, the mobile agents themselves will encounter limits (in the form of security, authentication and validation) that exist within domains [3]. In some cases, these limits will compromise the goals that have been given to the mobile agent.

**Information Agent:** The information agent acts as an information provider that has a direct access to the information resources [8]. The responsibility of the information agent is to accept queries submitted by the interface agents and returns the results to them. Due to the dynamic nature of the environment, information resources are constantly changing by, adding new information, deleting old information or modifying existing information. The information agent is able to handle event notifications on the information resource updates. In addition, it monitors any changes in the information resources based on a predefined timeframe. This capability allows the information agent to provide the interface agent with an appropriate set of information resources that carry up-to-date information. The Information agent receives queries during interaction from other agents expressed in a common query language (KQML). The information agent is able to construct the received query from a common query language into a language of the information source. Once the query is constructed, the information agent sends the query to the information source for execution, and constructs the answers back into the common query language. The Information agent accepts a query similar to that of the interface agent, which consists of a set of words describing the interface agent’s topic of interest and the associated constraints. The constraints might include the quality of information and the response time at which the information should be available.
6 System Characteristics

The basic characteristics of the agent-based information retrieval system are as follows [9].

- **Autonomous and Intelligent**: An autonomous agent is within and part of an environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future [9]. We easily see that the broker agent satisfies these requirements. We can also say that it is intelligent since it has the ability to decide, can think on behalf of the user. Its actions can change over time as agents in the system change their behavior.

- **Learning**: We shortly discussed the questions broker agent can respond. This could be possible with learning. When the broker agent first gets on-line, it knows nothing but with control messages it sends to other agents, it begins to retrieve information and in a short time it becomes fully informed about the status of the entire system. Its information base is up to date since these control messages are sent periodically [2].

- **Communicative/Collaborative**: The system consists of multiple agents that try to respond to user requests by communicating and working together [4].

- **Flexible**: Besides the flexibilities provided to the user, the system can easily be expanded with the integration of new information agents [8].

7 Advantages Over Classical Information Retrieval Systems

The system has many advantages over classical information retrieval systems. Traditional system rely on a central database while this system has a distributed database architecture [7]. Utilization of category information minimizes network usage and search time. Even if a user line is broken before he gets the results, he does not need to make the search again because system holds the result for him. When he is online again he will receive the results. The user can also register the request so that at a later time, he can retrieve the results. When he issues a request, he does not need to wait for the results and can start a new search before his prior requests are finalized. This is not possible in classical systems. Because of this system architecture arranged as a multi-tier model, it takes all advantages of multi-tier application such as scalability, load balance and if any agent in any tier fail the system remains intact.

8 Conclusion

This paper has presented the design and implementation details of N-tier information retrieval system based on a multi-agent architecture. The system has several advantages over similar classical systems, providing the user various flexibilities. The system is also expandable and tolerant to failures.

9 References