A Mediator-based Multi-Agent Architecture for Information Gathering from the Web

Hongen Lu, Leon Sterling
Intelligent Agent Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Parkville, VIC 3052, AUSTRALIA
{helu, leon}@cs.mu.oz.au

Abstract

Due to the explosive growth of the World Wide Web, finding specific information is becoming extremely difficult, sometimes even frustrating. To address this problem, several new technologies have been developed, such as search engines and information agents. These systems aim to assist people to find specific information from the Web. The Web is a dynamic changing open environment with more and more information providers become available each day. How to find information providers and how to integrate information agents in such an open environment is a new challenge.

In this paper, we present a mediator-based multi-agent architecture for information gathering from the World Wide Web. A mediator in our architecture serves as middleware that not only solves the connection problem, but also infers the cooperation relationships among information agents. A multi-agent prototype, SportsAgents, has been built using this architecture to find sports results from the Web.

1 Introduction

Due to the explosive growth of the World Wide Web, finding specific information is becoming extremely difficult, sometimes even frustrating. To address this problem, several exciting new technologies have been developed. Search engines, such as Yahoo [1], AltaVista [2], Lycos [3], are the most popular ones. They compile an index of the Web manually or automatically using "spiders". The index is used to answer users' queries based on keywords. The information providers based on search engines are built distributively over the Web using different languages and architectures. Most of them aim to provide users an assistant for searching for interesting information.

However, the Web is a dynamic changing open environment with many new kinds of information providers becoming available daily. In such a dynamic environment, information agents may appear and disappear at any time. Complete indexing of the Web is even difficult, sometimes impossible [10]. So how to find the relevant information providers and how to integrate information agents are new challenges.

2 Why Mediators?

Intelligent information agents, such as Ahoy [4] [14], ShopBot [5] and SportsFinder [12], are programs that assist people to find specific information from the Web. We can view these agents as

[1] at http://www.yahoo.com
[2] at http://www.altavista.com
[3] at http://www.lycos.com
[4] at http://ahoy.cs.washington.edu:6060

information service providers, which have the capabilities to find information for users, for example locating a person's homepage, finding the cheapest available prices for music CDs, or finding sports results of a team or a player.

However, for a novice user how to find these services, and for an information agent how to locate the service providers and communicate with them to solve its tasks cooperatively is a challenge.

One of the basic problems facing designers of open, multi-agent systems for the Internet is the connection problem — finding the other agents who might have the information or other capabilities that you need [3].

A possible solution is a software mediator. A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications [17]. In [2], a mediator is an information producing, or serving, entity in a large-scale internetworked environment.

3 System Architecture

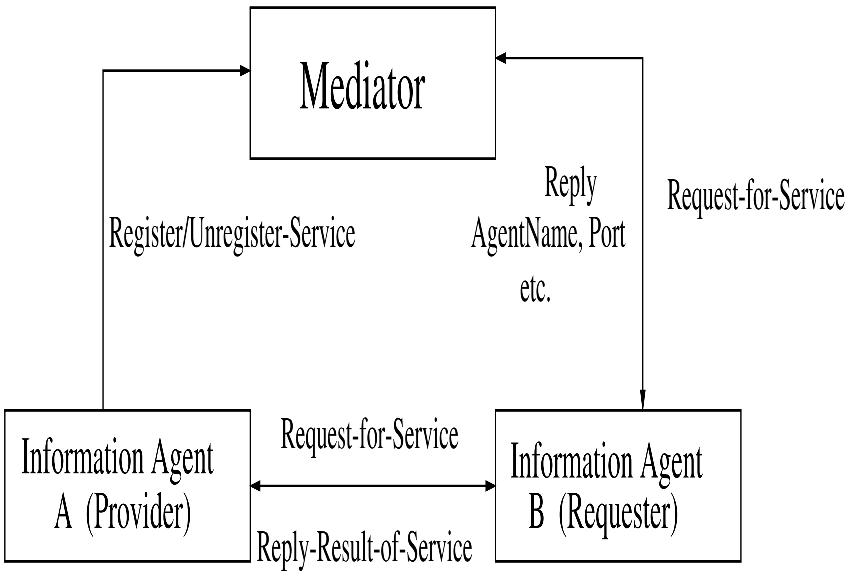

Figure 1: SportsAgents System Architecture

In our architecture mediators are introduced to help information agents interact with each other. When an information agent starts, it can register its capability(what kind of service it can provide) with a mediator. When an information agent receives a query or a subtask within a query that can not be solved by itself, it can request the mediator to find out other agents that have the capability or a set of agents who can work cooperatively to provide that service. In Figure 1 the system architecture of SportsAgents is presented.

3.1 Mediators

A mediator is a special kind of information agent acting as middleware to take as input, a request to find an agent that provides a service, and returns as output, a list of such agents and their cooperation relationships. Mediator also stores the services offered by different agents in the existing environment, and when a new agent is introduced into the environment it can register its capability to the mediator, if this agent wants its service to be used by others. Information agents also can unregister their services to the mediator when they want to quit the cooperation or exit. We present a mediator in SportsAgents in Figure 2.

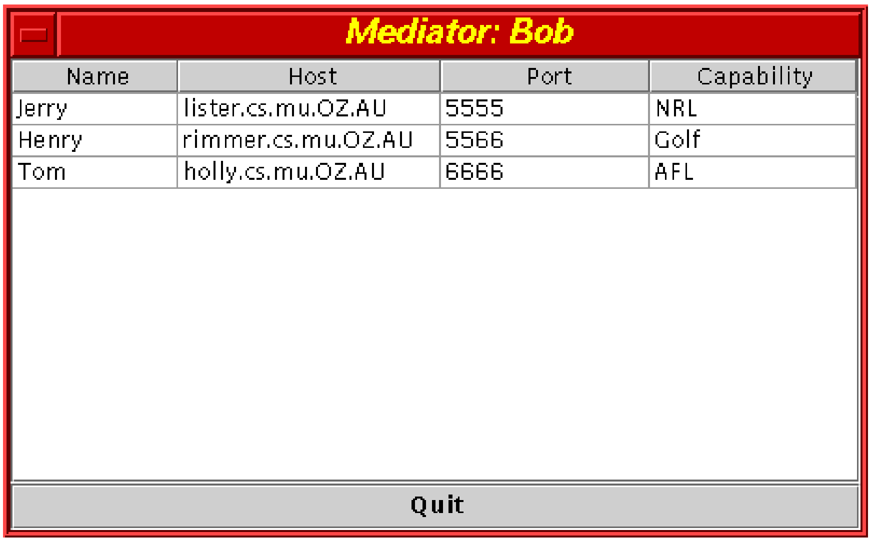

Figure 2: Mediator

3.2 Information Agents

Agent is one of the "hot" topics in the information age. The last ten years have seen a remarkable interest in agent oriented technology, spanning applications as diverse as information extraction, user interface and network management. Information agents are software programs that inhabit the world of computer events and networks to act on behalf of their human users to perform laborious information gathering tasks. Briefly an information agent is a program that has the ability to:

- express its capability to other agents, e.g. mediator.
- find certain information from the Web to answer user's query using its knowledge about a domain.
- communicate with other agents using an agent communication language, e.g. KQML.

The recent explosive growth of the World Wide Web provides a wealth of on-line information. This has made critical the need for some sort of intelligent assistant to a user who is searching for interesting information. Information agents are becoming popular due to this dramatic need.

3.3 Interface Agents

A interface agent acts as an interface between the user and other information agents. It receives a user's query and makes a plan to answer this query, and then communicates with the mediator and other information agents to execute the plan. The result of the query is also presented to the user by the interface agent. In Figure 3, a GUI interface is presented. Users can start and quit an information agent via this GUI interface. The relevant debugging and receiving messages are also displayed to users on this interface.

4 Example

Exploiting the above mediator architecture, we built a multi-agent prototype system, SportsAgents, using JACK. SportsAgents can answer a user's query like "which is the best sports city in Australia?". JACK is a Java-based multi-agent framework developed by Agent Oriented

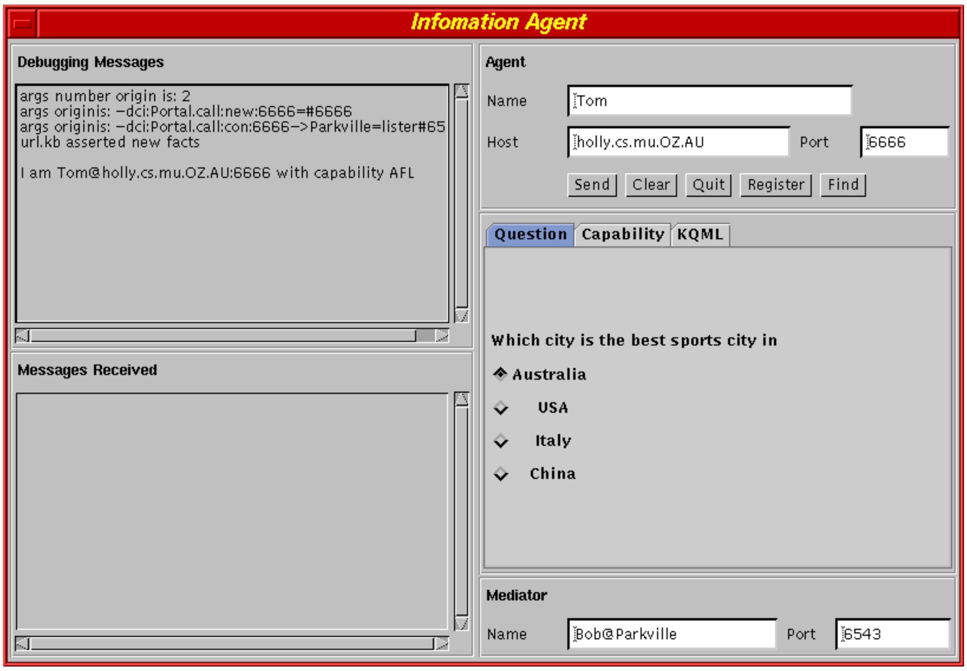

Figure 3: Interface Agent

Software Pty. Ltd. It includes all components of the Java development environment as well as offering specific extensions to implement agent behaviour.

In SportsAgents, three types of agents, mediator, information agents and interface agents, are implemented.

- Mediator keeps a table of the details of each information agent available at current time;
- Information agent can extract sports results for a sport or a competition, like golf, football, it also communicates with the mediator to advertise its capability;
- Interface agent provides the GUI interface for the user to ask questions and view the query result.

To find the best sports city in a country, we consider several most popular sports in that country and extract the result of each sport and evaluate each city's performance. Interface agent presents the result to users.

4.1 Service Hierarchy

Since each information agent is developed distributively, their capabilities are different from each other. SportsFinder [12] can find the sports result of golf, cycling, football and basketball etc. for users, while Ahoy [14] is good at locating people's homepage. However considering an application domain, such as sports, there exists a hierarchy relationship among these information agents. For example, information agent $\mathcal{A}$ can find all the results for Australian football teams, while agent $\mathcal{B}$ can only find the results of AFL(Australian Football League), in this case the service agent $\mathcal{B}$ can provide is a subset of agent $\mathcal{A}$, i.e. Service($\mathcal{B}$) $\subset$ Service($\mathcal{A}$).

In this section, we characterise agent service relations. We use S_i to denote the service of agent$_i$ and a set of sports names and/or competitions names to express what the service is about. For the above example, we have Service($\mathcal{B}$)=$\{$AFL$\}$.

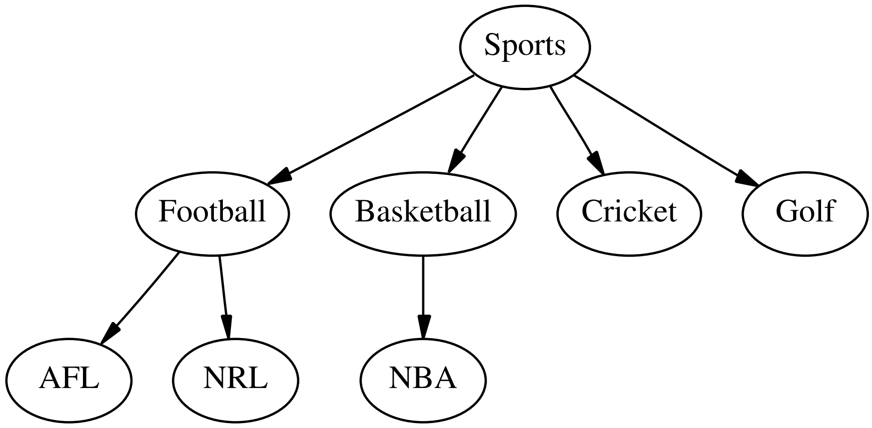

Figure 4: An Example of Sports Service Hierarchy

- **Identical Service:** $S_1 = S_2$.
- **Subservice:** $S_1 \subset S$.
- **Substitute Service:** a service S_1 can be substituted by service S_2, $S_1 \leftrightarrow S_2$.
- **Partial Substitute Service:** $S_1 \cap S_2 \neq \phi$.
- **Reciprocal Service:** $\exists S = S_1 \cup S_2 \ \wedge \ S_1 \cap S_2 = \phi$, then S_1 and S_2 are reciprocal with S.

To find the service relationship between two information agents, we need a data structure to describe the sports service hierarchy. We use a directed cyclic graph (DCG) to present the relations between agent services. The nodes in the graph present the services, and the edges are labelled with the service relationships. Following in Figure 4, we give a part of the sports service hierarchy.

4.2 Matchmaking Algorithm

One function of a mediator is to provide information, such as agent name, port, capability to query agent, so that the query agent knows which agents to cooperate with to solve its problem. We call this process "*matchmaking*".

Matchmaking is based on a cooperative partnership between information providers and consumers. In the SportsAgents system, a mediator is introduced to solve the connection problem, it finds the current available information agents who have the capability that the query agent(information consumer) is asking for. In case no available agent can fulfill the query service itself, the mediator will infer the available services to find a set of available information agents that can cooperate in some way to complete the query service.

We present the algorithm for service matchmaking in **Algorithm 4.1**.

5 Discussion

In [1], a platform called IMPACT to support multi-agent interactions is described. The platform provides a set of servers that facilitate agents interoperability in an application independent manner. In IMPACT, these servers match the services within a given distance. The matchmaking algorithm does not consider the relationships among services.

The mediator in our architecture requires an arbitrary amount of deduction or knowledge to match any given service and request. Consider the example we give in Figure 2. Information

Algorithm 4.1 matchmaking(S:Service, head:Hierarchy)

Given S, the service an information agent requests, agent table $AgentTable$, which contains the contact information of currently available information agents, and the service hierarchy $Head$. Each node in the service hierarchy has the following structure:

```
public class Service extends Vector {
    String name;
    Service up;
    Service down;
    Service next;
        ⋮
}
```

this algorithm returns the agents contact information and their relationships.

find = false; head = Head;
Agent-found =null; relation = null;

if *AgentTable.index(S) = true* **then** {
    while *(AgentTable.query(S) != null)* {
        Agent-found = Agent-found ADD AgentTable.query(S);
    }
    relation = find-relation(S, head);
    find = true;
}
else {
    if *head.name* $\simeq$ *S.name* **then** {
        service = head.down;
        while *(service != null AND !find)* {
            matchmaking(S, service);
            service = service.next;
        }
    }
    else {
        matchmaking(S, head.down);
        matchmaking(S, head.next);
    }
}
return *Agent-found, relation.* □

agents Tom and Jerry can provide the service of getting the results of AFL and NRL(National Rugby League) respectively. However to answer the user's query "which is the best sports city in Australia?", agent must know the result of one of the most popular sports, Football, of Australia. Unfortunately, none of the current available information agents has that capability. Using our matchmaking algorithm, we can infer that Tom and Jerry can work cooperatively to get the information. So the mediator will reply the request information agent with Tom and Jerry's contact information as well as their cooperative relationship. This can not be achieved in IMPACT platform. We believe inference and some reasoning abilities is necessary for mediators to provide intelligent matchmaking. We can build many domain specific mediators to help find other information agents in the specific domain.

At this stage, we embed the agents contact details, such as agent names, port, etc., and their cooperation relationships into agent communication language, which we use in SportsAgents is KQML. In the future, we will give a more complex and expressive agent capability description language for service advertising and requesting.

6 Conclusion

SportsAgents is a mediator-based multi-agent system. In Figure 5, a sample result of SportsAgents is given. Mediator-based architecture is useful to build open Web-based applications, since in such dynamic domain, applications are developed distributively and they become available or exist at any time. There is an obvious need for a standardised, meaningful communication among agents to enable them to perform collaborative task execution. Mediator-based architecture for Web applications is a step towards this goal. In such architecture, new information agents can be introduced easily. The developers do not need to care too much about how to find other information agents to collaborate with, since mediator can provider the partners the agents want. This makes the system open to new applications and flexible to organise agent cooperation.

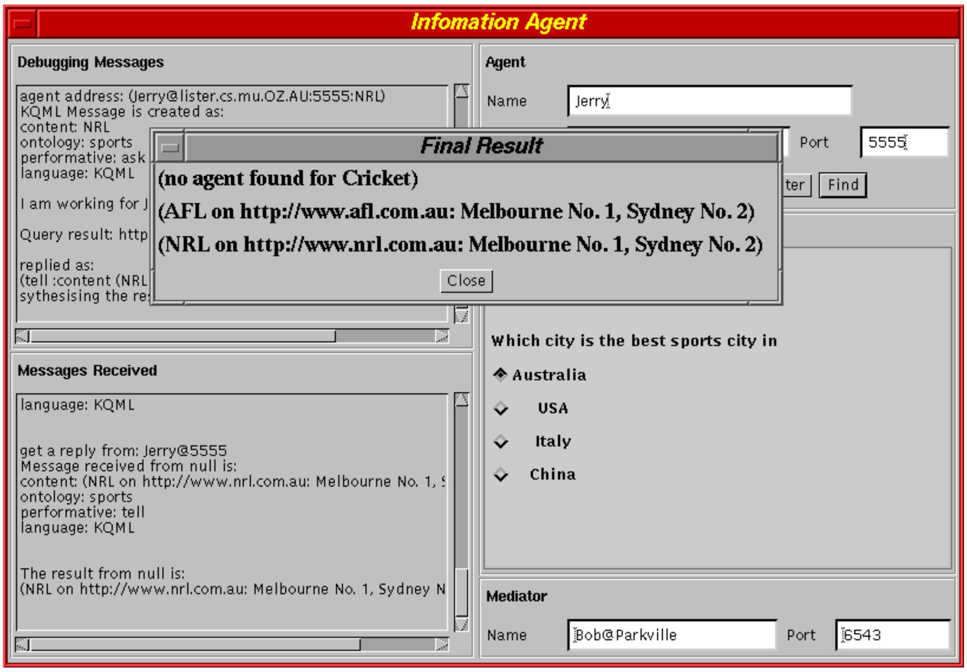

Figure 5: A Result of SportsAgents

Mediator in our architecture serves as middleware that not only solves the connection problem, but also inferences the cooperation relationships among information agents, this will direct information agents to work in a cooperative way to answer user's query. In such a way, information agents can improve its capability, and information gathering from the Web become more scalable. Our future works are giving a more expressive agent capability description language, and improving the reasoning ability of mediator.

References

[1] Khaled Arisha, Sarit Kraus, V. S. Subrahmanian, and etal. IMPACT: The interactive maryland platform for agents collaborating together. Technical report, Dept. of Computer Science, University of Maryland, College Park, Maryland 20742, 1998.

[2] Son Dao and Brad Perry. Information mediation in cyberspace: Scalable methods for declarative information networks. *Journal of Intelligent Information Systems*, 6(2/3):131–150, May 1996.

[3] K. Decker, K. Sycara, and M. Williamson. Matchmaking and brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, December 1996. available at `http://www.cs.cmu.edu/ softagents/papers/icmas96-matchbroker.ps.gz`.

[4] K. Decker, K. Sycara, and M. Williamson. Middle-Agents for the Internet. In *Proceedings of 15th IJCAI*, pages 578–583, Nagoya, Japan, August 1997.

[5] R. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the World Wide Web. In *Proceedings of the First International Conference on Automonous Agents*, 1997. Available at: `ftp://ftp.cs.washington.edu/pub/ai/shopbot.ps`.

[6] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3):309–314, 1997.

[7] Eugene C. Freuder and Richard J. Wallace. Suggestion strategies for constraint-based matchmaker agents. In *The AAAI-97 Workshop on Constraints and Agents*, 1997. available at `http://www.cs.unh.edu/ccc/ca/papers.html`.

[8] Daniel Kuokka and Larry Harada. Supporting information retrieval via matchmaking, 1995. available at `ftp://venera.isi.edu/sims/sss95/kuokka.ps.Z`.

[9] Daniel Kuokka and Larry Harada. Integrating information via matchmaking. *Journal of Intelligent Information Systems*, 6(2/3):261–279, May 1996.

[10] Steve Lawrence and C. Lee Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.

[11] Agent Oriented Software Pty. Ltd. *JACK Intelligent Agents User Guide*, 1999.

[12] Hongen Lu, Leon Sterling, and Alex Wyatt. SportsFinder: An information agent to extract sports results from the World Wide Web. In *Proceedings of the Fourth Internatinal Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology(PAAM99)*, pages 255–265, London, UK, April 1999. The Practical Applicaiton Company Ltd.

[13] Tuomas Sandholm. eMediator: a next generation electronic commerce server. Technical Report WUCS-99-02, Department of Computer Science, Washington University, St. Louis, MO 63130, January 1999. available at: `http://www.cs.wustl.edu/ sandholm/eMediator.tr.ps`.

[14] Jonathan Shakes, Marc Langheinrich, and Oren Etzioni. Dynamic reference sifting: A case study in the homepage domain. In *Proceedings of the Sixth International World Wide Web Conference*, pages 189–200, 1997. Available at: `http://ahoy.cs.washington.edu:6060/doc/paper.html`.

[15] K. Sycara, A. Pannu, M. Williamson, D. Zeng, and K. Decker. Distributed intelligent agents. *IEEE Expert, Intelligent Systems and their Applications*, 1996.

[16] Katia Sycara, Jianguo Lu, and Matthias Klusch. Interoperability among heterogeneous software agents on the Internet. Technical Report CMU-RI-TR-98-22, the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, October 1998. Available at: `http://www.cs.cmu.edu/ klusch/TechRep2.ps`.

[17] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), March 1992.

[18] M. Wooldridge and N. R. Jennings. Cooperative problem-solving. *Journal of Logic and Computation*, 9(4):563–592, 1999.