Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 343 – 346 Sept. 2007 ISSN 1003—7985

Multi-broker architecture for large-scale dynamic heterogeneous information integration

Deng Huafeng Liu Yunsheng

(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: In order to realize interoperability to a large number of autonomous and heterogeneous information sources with high efficiency, an agent-based multi-broker architecture (AMA)—HustEven, is constructed. A group of broker agents are designed to provide brokering services in a peer-to-peer (P2P) manner for the non-broker agents (user agents, resource agents, query agents). Thus, the scalability and robustness of the system are enhanced. Ontology is also used by the broker agents for facilitating interoperability among all the agents in HustEven. Unlike any other AMAs, an interdomain ontology is built in this system to represent the relationships among the common concepts in the innerdomain ontologies. Therefore, a broker forwards the queries only to the other related brokers according to the interdomain ontology and the communication overhead among the brokers is reduced. Obviously, the application of the interdomain ontology enables a broker to fully take advantage of the multi-broker architecture. The experimental results show that the HustEven performs more efficiently than any other existing systems.

Key words: multi-broker architecture; semantic interoperability; ontology; information integration; open systems

How to effectively realize semantic interoperability to a large number of autonomous and heterogeneous information sources has become the key issue which must be solved[1]. Systems such as Disco, TSIMMIS, InfoMaseter and information manifold are evolved from multidatabase systems that implicitly do syntactic brokering when matching information resources[2].

The systems such as SHADE[3], LARKS[4] have attempted to address the issue of semantic brokering by providing a source input-output description to determine whether or not a semantic match occurs between a requested service and a service provider. The InfoSleuth system[2] consists of a set of collaborating agents that work together for information discovery and retrieval in a dynamic, open environment. Ontologies are always used in the information integration systems to facilitate communication among these systems automatically[5-6]. The ontology information in the InfoSleuth system is organized as several focused ontologies and ontology fragments. However, the main disadvantage of the ontology management of InfoSleuth is that it is very difficult and inefficient for each broker to maintain the whole ontology in the large systems. In our HustEven system, the ontology information in the HustEven is organized as several inner-domain ontologies and a very small interdomain ontology to take advantage of the multi-broker architecture. OWL (web ontology language), the emerging standard by the W3C[7], is used as the ontology language to describe the ontologies in our system.

1 Multi-Broker Architecture

We adopt the multi-broker architecture that is similar to the InfoSleuth system. Fig. 1 depicts the current HustEven multi-broker architecture. We use a peer-to-peer topology for inter-broker connectivity. Peer-to-peer brokering is more scalable because it allows brokers to freely advertise and unadvertise themselves to the other brokers. This topology also ensures that there is not a single point of failure that is often found in the hierarchical brokering systems.

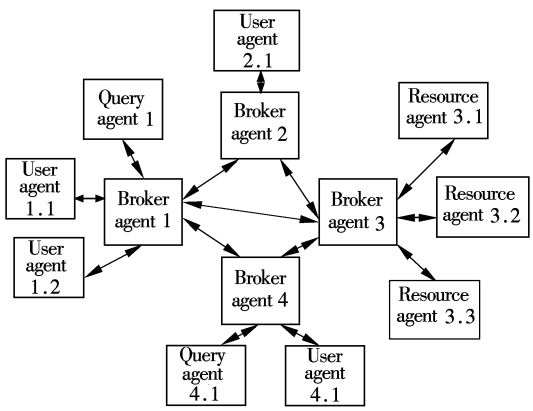

Fig. 1 Multi-broker architecture

Received 2007-05-18.

Foundation item: The National Natural Science Foundation of China (No. 60673128).

Biographies: Deng Huafeng (1974—), male, graduate; Liu Yunsheng (corresponding author), male, professor, ysliu@hust.edu.cn.

In the large-scale systems, all the agents can always be divided into several logical domains. The agents in each logical domain have common interests and characteristics. The brokers provide brokering services for the other agents in their respective domains. The biggest difference in the multi-broker architecture between InfoSleuth and HustEven is the way to organize the ontology. We will discuss this problem in section 2.

2 Single Broker Architecture

2.1 Overview of the single broker architecture

Fig. 2 depicts the single broker architecture that represents a subsystem in the HustEven multi-broker system. The single broker architecture consists of four types of agents.

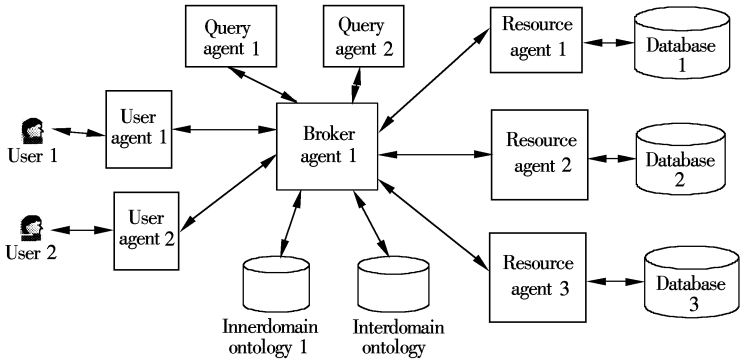

Fig. 2 Single broker architecture

- User agent

The user agent acts as the proxy for the users. It provides an interface for accepting the users' query requests and forwards the query requests to the broker agent. The HustEven system defines two query policies: the interdomain search mode and the domain search mode. The query is answered within the domains that the user agent registers itself to if the user chooses the domain search mode. Otherwise, the query will be referred to all related brokers according to the interdomain ontology.

- Broker agent

The broker agent plays a critical role in the HustEven system. The broker agent carries out the brokering service for all the other agents in its logical domain and deals with the cooperative tasks from other brokers in the system. The brokering services include receiving the advertisement that represents various agents' capabilities, updating the ontologies, dealing with the queries sent by the user agents, decomposing the query tasks to related resource agents, and returning prompt messages or results to the user agents. The brokers function as the mediators in the common integration architecture.

- Resource agent

The resource agent plays the role of the wrapper as in the mediator approach. It can answer some kinds of queries for the broker agents in its domain.

- Query agent

The query agent is a kind of special agent to participate in the brokers' primary task. The broker agents are very important and busy in the system. So when a query request arrives, the broker agent produces a query agent (a thread) responsible for decomposing the query tasks to related resource agents and synthesizing the results returned by the related resource agents. When a query is finished, the query agent will be aborted by the broker who produces it.

2.2 Organizing ontologies

Incorporating everything about ontological knowledge into a single, very large ontology makes the management of the ontology very difficult[1-2]. Therefore, we divide the whole ontology into two levels: innerdomain ontologies and an interdomain ontology. Each innerdomain ontology contains all the terms and the relationships to encompass all of the knowledge that the agents will need to describe their capabilities and queries in a subdomain. The interdomain ontology becomes very small because it only includes the terms and relationships shared in different subdomains.

The running ontology is the running version of the corresponding innerdomain ontology. Each time a resource agent comes online, it announces its capabilities to one or more brokers by sending an advertisement that uses the terms and vocabulary described in the innerdomain ontologies. Agent capabilities describe the input constraints and corresponding output of the information source about all the queries it can deal with. The broker then adds the information in the advertisement into the running ontology. So each running ontol-

ogy only includes these terms and relations which have been used currently. When an agent's capabilities change, the agent may readvertise to each broker that it is sending the original advertisement back again, so the brokers can update the information in the running ontologies. When an agent goes offline, it unregisters itself from the brokers to which it has advertised.

A query of a user agent is represented as an individual capability specified over the innerdomain ontology. According to the query policy, the queries are divided into two categories: the innerdomain query and the interdomain query. When a broker agent receives an innerdomain query, its reasoning engine checks whether the query capability matches one of the capabilities of the resource in the advertisement within its own domain. If there is a match, a query agent is created for answering the query. If the broker agent receives an innerdomain query, its reasoning engine should do an additional job. The broker agent should also check whether the query capability matches one of the capabilities of the resource agent in the other brokers' domains according to the interdomain ontology. The query should be forwarded to the related broker agents if such a match exists.

There has already been much research[8-10] on how to build the domain ontology semi-automatically or automatically and we do not discuss it here. After the ontologies are constructed, we can use the approach in Ref. [11] to build the interdomain ontology.

The interdomain ontology is very important in multi-broker architecture. According to Ref. [1], multiple ontologies that capture different terminologies but sometimes overlapping domains are independently created and managed. Because the domains are different but sometimes overlapping, we must use interdomain ontology to reflect the difference among various domains. However, the InfoSleuth has not touched the interdomain ontology that should be paid great attention to.

3 Experiments

We also conduct primary simulation-based experiments to test the performance of the approach of organizing the ontology in the InfoSleuth[2] and our approach in the HustEven system. In the experiments, there are 5 broker agents, 10 user agents and 50 resource agents. The queries in the system are all interdomain queries. The bandwidth of the network is set to be 250 kbyte/s with a set-up latency time of 0.1 s/message. Each resource agent's advertisement size is set to 1 Mbyte and it costs the broker 1 s to process 1 Mbyte of advertisements. The number of average queries received by each user agent in 3 000 s is set to 20, 25, 30, 35, 40, 45, 50 and 55, respectively.

The metric of interest here is the average response time to the query by the brokers. Fig. 3 shows the results of varying the number of average queries per user agent in the system. We observe that our approach of organizing the ontology performs better than the approach of the InfoSleuth system and the difference between the two approaches enlarges with the increase in the average queries per user agent. A broker forwards the queries only to the other related brokers according to the interdomain ontology in the HustEven and the communication overhead among the brokers is reduced. Although sending the query to related broker agents wastes some time, maintaining and searching in a very small ontology can save more time than searching in a large ontology. It explains the difference between the two approaches.

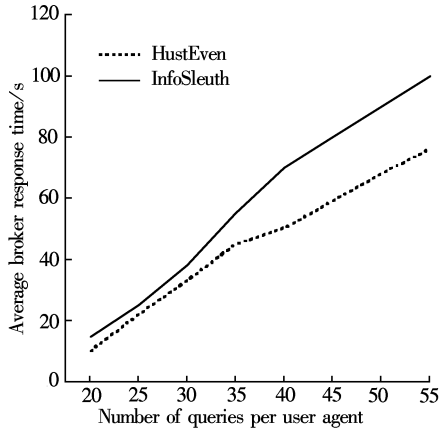

Fig. 3 Average broker response time across a range of queries per user agent

4 Conclusion

We use ontologies representing the terms and relationships of a large number of autonomous and heterogeneous information sources in the open environment to facilitate system syntax and semantic interoperability. The ontology information of HustEven is divided into two levels: innerdomain ontology and interdomain ontology to take advantage of the multi-broker architecture. Comparing to the InfoSleuth[2], we exploit the interdomain ontology to realize interoperability among the subdomains.

The overall peer-to-peer architecture and the local centralized architecture in the HustEven strike a balance between the efficiency of centralized query, and the autonomy, load balancing and robustness to at-

tacks.

A registering mechanism makes upgrading and adding the information sources to the open systems very easy. In a system running a registration mechanism, we are able to search in a small running ontology to achieve higher efficiency.

References

[1] Sheth A, Ramakrishnan C, Thomas C. Semantics for the semantic web: the implicit, the formal and the powerful [J]. *International Journal on Semantic Web and Information Systems*, 2005, **1**(1): 1 – 18.

[2] Ngu A H H, Cassandra A, Bohrer W G. Scalable semantic brokering over dynamic heterogeneous data sources in Infosleuth [J]. *IEEE Transactions on Knowledge and Data Engineering*, 2003, **15**(5): 1082 – 1098.

[3] Kuokka D, Harada L. Integrating information via matchmaking [J]. *Journal of Intelligent Information Systems*, 1996, **6**(2): 261 – 279.

[4] Sycara K, Lu J, Klusch M, et al. Matchmaking among heterogeneous agents on the Internet [C]//*AAAI Spring Symposium on Intelligent Agents in Cyberspace*. Menlo Park: AAAI Press, 1999: 40 – 46.

[5] Alonso-Calvo R, Maojo V, Billhardt H, et al. An agent-and ontology-based system for integrating public gene, protein, and disease databases [J]. *Journal of Biomedical Informatics*, 2007, **40**(1): 17 – 29.

[6] Lee J, Goodwin R. Ontology management for large-scale enterprise systems [J]. *Electronic Commerce Research and Applications*, 2006, **5**(1): 2 – 15.

[7] Bechhofer S, Harmelen F V, Hendler J, et al. Owl web ontology language reference [EB/OL]. (2004-03-25) [2007-03-28]. http://www. w3. org/TR/2004/REC-owl-ref-20040210.

[8] Silva N, Rocha J. Semantic web complex ontology mapping [C]//*IEEE/WIC* 2003 *International Conference on Web Intelligence*. Halifax, Canada, 2003: 82 – 88.

[9] Xu L, Embley D. Using domain ontologies to discover direct and indirect matches for schema elements [C]//*Proc of Semantic Integration Workshop*. Sanibel Island, 2003: 1 – 6.

[10] Dhamankar R, Lee Y, Doan A H, et al. Imap: discovering complex semantic matches between database schemas [C]//*Proc of the* 2004 *ACM SIGMOD International Conference on Management of Data*. Paris, 2004: 383 – 394.

[11] Doan A H, Madhavan J, Dhamankar R, et al. Learning to match ontologies on the semantic web [J]. *The International Journal on Very Large Data Bases*, 2003, **12**(4): 303 – 319.

一种用于大规模动态异构信息集成的多代理结构

邓华锋　　刘云生

(华中科技大学计算机科学与技术学院,武汉 430074)

摘要:为了在集成大量异构自治信息资源时实现高效率互操作性,设计了一种基于智能体(agent)的多代理结构. 用一组代理智能体以 peer-to-peer(P2P)方式为其他非代理智能体提供代理服务,从而提高系统的可扩展性与健壮性. 代理服务使用本体技术来实现系统互操作性. 与其他多代理结构不同的是,除了领域内本体,还引入了一个领域间本体来表达各个领域内本体中公共概念的联系. 因此,代理可以根据领域间本体的信息,将查询请求仅发往相关的代理,减少了网络通信量,更好地发挥多代理体系结构的优势. 实验结果表明,系统实现互操作性的效率相对于现有的系统有明显的提高.

关键词:多代理结构;语义互操作性;本体;信息集成;开放系统

中图分类号:TP311