24

Implementing advanced internet search engines

G. Lorenz, S. Dangi, D. Jones, P. Carpenter and S. Shenoi
Department of Computer Science
University of Tulsa, Tulsa, Oklahoma 74104, USA
sujeet@utulsa.edu

Keywords
Internet search engines, metadata, persistent agents, mobile agents

PROJECT DESCRIPTION

Current Internet search tools, e.g., Yahoo! and AltaVista, are relatively simple. Their reliance on indexed files containing keyword-to-IP-address mappings limits them to handling low-level keyword queries. Future Internet search tools will be much more sophisticated (see, e.g., Arens *et al.*, 1993; Iuaba *et al.*, 1995; Liu and Pu, 1997). They will employ metadata repositories to support content-based querying and distributed, persistent agents performing a variety of functions, including data gathering, metadata extraction, data mining and information fusion. Users could create swarms of persistent search agents that would range the Internet in response to sophisticated queries, keeping them informed about updates and terminating only on explicit user directives. Clearly, such search engines will pose serious threats to security and privacy.

Figure 1 shows the architecture of an advanced search engine being developed at the University of Tulsa to evaluate security and privacy threats. The server houses a metadata repository, a base agent and various search agents.

The metadata repository maintains schema information about information repositories, including structured, semi-structured and unstructured sources. It is continually refreshed by metadata daemons, persistent agents that search for new information sources, old sources that are no longer accessible and those whose schemas have been modified.

Database Security XI T.Y. Lin and Shelly Qian (Eds)

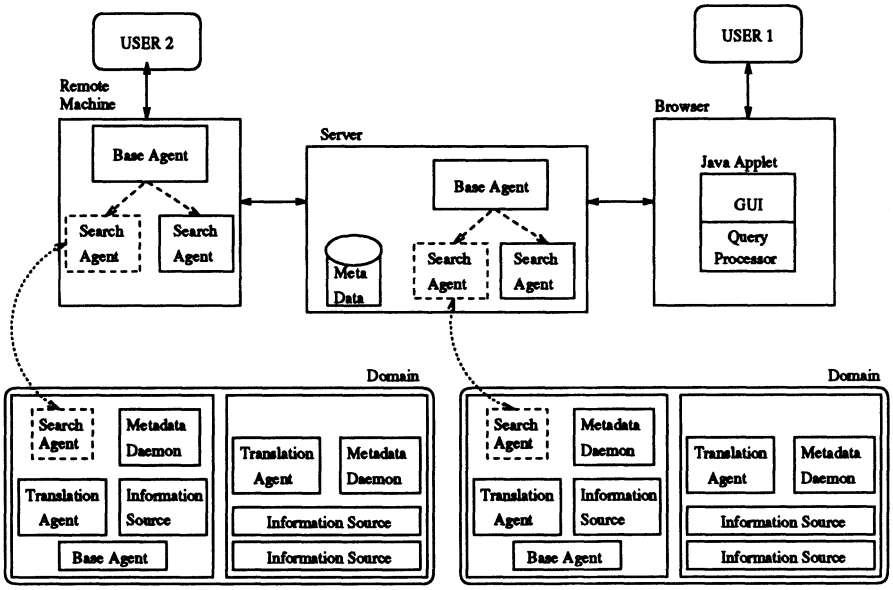

Figure 1 Search engine architecture.

The base agent manages all queries submitted to the search engine. It analyzes queries and determines the appropriate number and type of search agents needed. The base agent can start new search agent threads or spawn persistent search agents. A new thread is created for each non-persistent query. A persistent query, e.g., one involving a changing or evolving information source, requires the creation of a persistent agent. All search agents report to their base agent which forwards information to the user.

Search agents are the primary information gathering entities. They access the metadata repository for intensional (schema) information about information sources. Separate translation agents, one for each type of information source, are used to obtain actual data. Search agents transmit results to their base agent. In addition, they provide progress reports conveying the number of sources searched, the number remaining to be searched and the estimated completion time. Data access failures are reported and inaccessible sources are flagged for possible elimination from the metadata repository. A user maintains control of search agents through their base agent, enabling searches to be tuned, suspended or terminated.

Translation agents enable search agents to access heterogeneous information sources. Requests from search agents, expressed in a KQML-like language (Finin *et al.*, 1993), are translated into the native languages of information sources. Information transmitted between agents is represented in the Knowledge Interchange Format (KIF) (Genesereth and Fikes, 1992) and embedded in agent communication language wrappers. Similar information sources, e.g., databases using the Java/Open Database Connectivity Interface, use the same translation agent. Translation agents can reside on the server or on machines

hosting information sources. Database administrators wishing to facilitate or control access by search agents may provide their own local translation agents.

The Java Development Kit (Sun Microsystems) and the Java Aglet Applications Programming Interface (IBM) (Lange and Oshima, 1997) are used for the implementation. Java's architecture-neutrality and platform-independence are well suited to implementing the search engine. The Java Aglet Applications Programming Interface (J-AAPI) is designed for the rapid development of distributed persistent agents. It retains all the benefits of Java, including code persistence through object serialization and remote method invocation. J-AAPI also provides (data and execution) state persistence required by mobile agents. Using J-AAPI, a Java object (agent) can halt execution, relocate to another host and resume execution there. Object code and data are serialized and transferred to the remote machine, where they are unserialized and then restarted in the correct state.

Most search engine users will use a web browser to access a GUI and query processor implemented by Java applets, similar to current search engines. However, a copy of the base agent may be downloaded and run locally on remote machines to enhance local and global performance. Note that the implementation of the base agent as a Java application allows full search engine functionality unhindered by the Java applet security model.

Acknowledgement This research was supported by MPO Grants MDA904-94-C-6117 and MDA904-96-1-0114 and OCAST Grant AR2-002.

REFERENCES

Arens, Y., Chee, C.Y., Hsu C. and Knoblock, C.A. (1993) Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, **2**, 127–158.

Finin, T. *et al.* (1993) Specification of the KQML agent communication language. Technical Report EIT TR 92-04 (updated July 1993), Enterprise Integration Technologies, Palo Alto, California.

Genesereth, M.R. and Fikes, R.E. (1992) Knowledge Interchange Format (KIF) Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Palo Alto, California.

Iuaba, M. (1995) Internet Consultant: An integrated conversational agent for Internet exploration, *Proceedings of the Global Information and Software Society Internet Conference*, 65–78.

Lange, D.B. and Oshima, M. (1997) Programming mobile agents in Java with the Java aglet API. Technical Report, IBM Tokyo Research Laboratory, Tokyo.

Liu, L. and Pu, C. (1997) DIOM: Object-oriented approach to integration and access of heterogeneous information sources. ***Distributed and Parallel Databases*, 5(2)**, 167–205.