Distributed Search in Peer-to-Peer Networks

Most Web content is invisible to current search engines. JXTA Search addresses this problem, providing a unique query routing protocol that makes content visible and facilitates its use.

Steve Waterhouse, David M. Doolin, Gene Kan, and Yaroslav Faybishenko
Sun Microsystems

As the number of documents on the Web grows into the billions, the proportion of those documents available through indexing search engines shrinks.[1,2] The amount of content available through the "hidden web" of searchable online databases might be 400 times larger.[3] Application servers and network-enabled business systems have the potential to increase this content by many more orders of magnitude, and the growth of peer-to-peer networks, such as JXTA,[4] promises even more.

The JXTA Search discovery and access model exposes content unavailable through traditional indexing and caching search engines using a query routing protocol (QRP) for distributed information retrieval. JXTA Search is programming language, platform, and network transport independent, and occupies a middle ground between the decentralized Gnutella and centralized Napster geometries.

In this article, we review search problems addressed by JXTA Search and summarize services necessary for implementation. Other works describe JXTA Search in detail and compare it to other approaches.[5,6]

A Peer-to-Peer Search Engine

JXTA is a suite of protocols that facilitates P2P communication. It provides a common network architecture layer for a wide variety of network services and applications. An XML-derived syntax defines JXTA protocols, so JXTA implementations don't depend on programming language, platform, or network transport. Peers on a JXTA network are arbitrary devices, and can be anything with an electronic heartbeat.[4]

JXTA Search is a decentralized peer-to-peer search engine. It provides a common distributed query interface for peer devices, exposing services and content for a network of information providers and consumers. JXTA Search encourages information providers to publish a description of queries they are willing to answer for information consumers. The nature and capabilities of information consumers and providers are irrelevant to JXTA Search; it models them simply as peers on an arbitrary network. For example, the information consumer could be a person using a Web browser and the provider a database-driven Web site. Or, both consumer and provider could be cell phones on a JXTA network.[4]

In the fully decentralized Gnutella model, every query is propagated to every peer.[7] In the Napster model, a central server routes all queries.[7] In contrast, for JXTA Search, peers can act as *hubs* and can register with other hubs as information providers to field queries from other peers based on arbitrary content description registrations. JXTA Search registrations allow searches to proceed along two complementary axes: wide and deep.

JXTA Search provides an efficient mechanism for distributing queries across a *wide network* of peers using specialized peers (the hubs). Hubs can specialize by geography, peer content similarity, or application. Queries can be forwarded from hub to hub, as necessary. Figure 1 gives an overview of peer relationships in JXTA Search.

Deep search is based on the notion that information available on a Web site can be accessed only through an interface to the Web server. A typical example is a browser-based search engine, where users issue queries interactively to a back-end server, which accesses an index, or database, or some other information source not available as a static Web page. This approach increases result relevance, information access, and search efficiency.

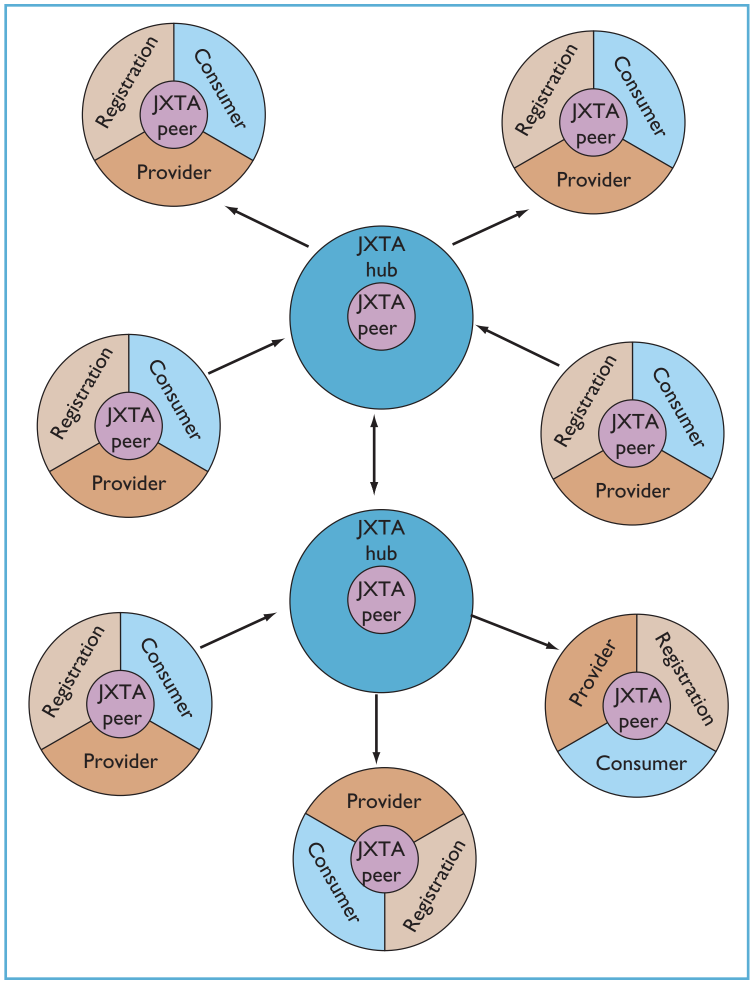

Figure 1. A JXTA Search network. Each peer on a JXTA Search network interacts with one hub. Each hub forwards requests to registered providers, which may be other hubs. Each peer on the network may act as a consumer, provider, and hub.

- The *relevance* of search results is directly affected by the search engine's updating capability. Crawlers can take days to weeks to crawl and index the Web, imposing a prohibitive cost given the transient nature of many deep Web resources. JXTA Search defines a registration protocol allowing information providers to rapidly update or change registrations defining queries they are interested in handling.
- *Access* to information on a Web site might require user interaction. For example, database-driven Web shopping sites can contain millions of databased product descriptions. Crawling entails querying the site's database repeatedly to extract product descriptions. Because many such pages are generated dynamically from information about the consumer or the query's context (time, purchasing behavior, location, and so on), crawling can provide biased results. This problem is solved using the registration protocol mentioned above.
- Search *efficiency* improves by cutting out layers of irrelevant work. Consider a crawler (for example, HiWE[3]) accessing dynamic content from a database-driven Web site, gleaning content from available pages to store in its own database. Because the page is locally stored, a stale link could result from a query. In contrast, the approach taken by JXTA Search specifies a response protocol giving both parties more control over the exchange of data.

Query Routing Protocol

JXTA Search's QRP design promotes structured, lightweight, and efficient query message exchange. The QRP consists of three components expressed in XML syntax: registration, request,

```
<?xml version='1.0'?>
<request id="1C8DAC3036A811D584AEC2C23"
       query-space="http://bigbookseller.com/js"
       xmlns="http://jxta.org/search">
       <query>
              <author> Joy </author>
              <title> Java </title>
       </query>
</request>
```

Figure 2. A JXTA Search query request. Predicates limit response, so the provider gets the desired request.

```
<?xml version='1.0'?>
<register xmlns="http://search.jxta.org">
    <query-server>
       http://bigbookseller.com/exec/jxtasearch
    </query-server>
    <query-space
uri="http://bigbookseller.com/jxtasearch">
       <predicate>
       <query>
          <author> Joy Stephenson </author>
          <title> Java JXTA XML Cryptography </title>
       </query>
       </predicate>
    </query-space>
</register>
```

Figure 3. A JXTA Search provider registration. Queries matching these conditions are directed to the specified query server.

and response. These components exist in *query spaces*, unique identifiers similar to XML namespaces, which coordinate consumers and providers.

- providers *register* an XML-based encoding of a query space and predicates with a hub on the JXTA Search network,
- users and end applications present *queries* to the JXTA Search network in arbitrary XML as defined for a specific query space,
- *responses* are returned from the provider in the appropriate XML defined by the query space.

Predicates applied to each request in the given query space limit response; only requests matching the predicate are sent to the provider. For example, Figure 2 illustrates a query request using a product query space for books matching certain authors and titles. In this example, the query space is defined as http://bigbookseller.com/jxtasearch. The query specifies that the "author" should be "Joy" or that the title should contain "Java." The response message consists of a body of arbitrary XML wrapped in <responses>...</responses> tags. The response's body can be arbitrary XML as long as it corresponds to the specified query space. Providers can be any peer in the network, either JXTA peers running a JXTA Search service or HTTP peers adapted to the QRP.

Provider registrations advertise a providers' metadata and structure. Registrations are stored on hubs and specify the server, one or more query spaces, one or more predicates, and free form XML describing the types of information provided. In a Web implementation, the server might be a CGI script capable of processing the QRP request messages and responding with a QRP response. Consider the registration illustrated in Figure 3: http://bigbookseller.com/jxtasearch specifies the query space text. This provider registers for any query containing "Joy" or "Stephenson" in the <author> field and containing "Java," "JXTA," "XML," or "Cryptography" in the <title> field. Queries matching these conditions are directed to the query server, http://bigbookseller.com/exec/jxtasearch.

Query Predicates

A query predicate is a Boolean expression composed of query node patterns. Predicates must be in *conjunctive normal form* (a conjunction of disjunctions). Then, each QNP matches a node of an XML query when the QNP can be constructed by either deleting a node in the query or replacing the query with a subnode of itself. Consider the query predicate shown in Figure 4. The first two conjunctions are implicit disjunctions. When an <or>...</or> tag contains only a single QNP, the <or>...</or> may be dropped. Similarly, if the top-level only has one element, the <and>...</and> may be dropped. Thus, at its simplest, a predicate's form can be: <predicate>foo bar</predicate>, which would match any query containing the word "foo" or the word "bar." Predicates are also used by the resolver to rank provider capabilities based on comparison between the query and QNPs.

Query Resolution and Routing

Because routing all queries to all peers is inefficient, the JXTA Search resolver indexes all tags and text in the registration files for mapping query terms to providers, creating separate indices for each query space. To be matched to a query, the provider must at least be in the same query space. The resolver first selects providers satisfying all clauses of at least one predicate. Selected next are providers without a matching predicate, but similar in their responses and having the same query space as providers with a matching predicate. If the number of selected providers is still less than some predetermined maximum, the resolver randomly selects a provider from the same query space as the query.

The JXTA Search router performs several functions. Initially, the router receives the queries from the peer. Next, the resolver selects peers on the network registered to receive matching queries. The router sends queries to this set of providers, which could be other hubs as well as peers. Finally, the router collects, sorts by relevance, and presents responses to the end user application. The JXTA Search protocol specification includes an in-depth discussion of the JXTA Search resolver, router, QRP, and predicates.[8]

```
<predicate>
   <and>
      <object type=file>
      <object><format>jpeg</format></object>
      <or>
         <desc>foo</desc>
         <desc>bar</desc>
      </or>
   </and>
</predicate>
```

Figure 4. A JXTA Search provider registration query predicate. The resolver ranks provider capabilities.

Implementing JXTA Search

Because QRP is text-based (XML), JXTA Search implementations are platform and language independent. A JXTA Search network instance requires peers running a number of services. The consumer service sends QRP queries to peer hubs and awaits QRP responses. The consumer peer is ultimately responsible for presenting the results to the end user. The provider service accepts QRP queries routed from the hub, handles the query appropriately, and returns a QRP reponse to the requestor. Hubs can be both consumers and providers, but must also implement registration, resolver, and router services. The registration service maintains an index of QRP registrations for the hub. Registration caching, indexing, and other chores are strictly implementation dependent. The resolver service matches QRP queries to providers, and the router service routes and manages query connections, and collates and returns results to consumer or client using a QRP response.

Hubs can be set up in arbitrary configurations. For example, a Web site might run a hub as a vertical aggregator of content pertaining to Java programming, registering sites with content focused on Java. In a different instance, a hub as a horizontal integrator could register news Web sites from a wide variety of media outlets.

Opportunities for extending JXTA Search are manifold. For example, the JXTA Search platform supports arbitrary XML so it can integrate emerging standards (such as the resource description framework [RDF] for describing metadata) and security (through public key infrastructure). You can plug categorization, classification, or other relevance approaches for refining searches and preventing spam into the basic framework. A number of application domains lend themselves well to the JXTA Search model:

- *Consumer Web searching.* JXTA Search could be used with standard caching search engines such as Google.[9]
- *Business-to-business exchanges and supply chain networks.* The conventional approach to data synchronization in exchanges is to replicate buyer and seller data at the exchange, but a P2P approach is potentially much more efficient.[10]
- *Personal electronic devices such as cell phones.*[4] Peer network growth will explode over the next few years. Linking them will require a consistent discovery framework.

JXTA Search is available as open source software under the Sun JXTA license (BSD-style) at http://www.jxta.org/project/www/license.html. A Web client demonstration using the Java servlet API lets users query providers and presents search results ranked according to a simple term frequency/inverse document frequency method.

Acknowledgments

JXTA Search is based on the Infrasearch engine written by Gene Kan, Yaroslav Faybishenko, and Doug Cutting. The authors thank the three anonymous reviewers for greatly improving this article.

References

1. S. Lawrence and C.L. Giles, “Searching the World Wide Web,” *Science*, vol. 280, no. 5360, 1998, pp. 98–100; available online at http//citeseer.nj.nec.com/lawrence98searching.html (current 30 Nov. 2001).
2. S. Lawrence and C. L. Giles, “Accessibility of Information on the Web,” *Nature*, vol. 400, 1999, pp. 107–109.
3. S. Raghavan and H. Garcia-Molina, “Crawling the Hidden Web,” tech. Report 2000-36, Computer Science Dept., Stanford Univ., Palo Alto, Calif., 2000; available online at http://dbpubs.stanford.edu/pub/2000-36 (current 30 Nov. 2001).
4. L. Gong, “JXTA: A Network Programming Environment,” *IEEE Internet Computing*, vol. 5, no. 3, May/June 2001,

pp. 88-95.
5. S. Waterhouse, "JXTA Search: Distributed Search for Distributed Networks," white paper, Sun Microsystems, Palo Alto, Calif., 2001; available online at http://search.jxta.org/protocol.html (current Nov. 2001).
6. S. Botros and S. Waterhouse, "Search in JXTA and other Distributed Networks," *Proc. 2001 Int'l Conf. Peer-to-Peer Computing*, 2001; available online at http://people.jxta.org/stevew/BotrosWaterhouse2001.pdf, (current Nov. 2001).
7. B. Yang and H. Gracia-Molina, "Comparing Hybrid Peer-to-Peer Systems," tech. report 2000-35, Computer Science Dept., Stanford Univ., Palo Alto, Calif., 2000; available online at http://dbpubs. stanford.edu:8090/pub/2000-35 (current Nov. 2001).
8. S. Waterhouse, "JXTA Search Protocol Specification," tech. report, Sun Microsystems, Palo Alto, Calif., 2001; available online at http://search. jxta.org/JXTAsearch.pdf (current Nov. 2001).
9. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, , 1998, pp. 1-7.
10. J. Bond, ed., "P2P in B2B," *P2P in B2B Mindshare Workshop*, Netmarkets Europe, Dec. 2000; available online at http://www. netmarketseurope.com (current Nov. 2001).

Steve Waterhouse is Director of Engineering for Sun Microsystems' JXTA group. Waterhouse publishes in the fields of distributed search, speech recognition, neural networks, and data mining. He received a PhD in engineering from Jesus College, Cambridge University, and the Isaac Newton prize for outstanding PhD research.

David M. Doolin is a programmer with the JXTA group, with an interest in discrete element methods. Doolin holds BS and MS degrees from the University of Tennessee, Knoxville, and is a PhD candidate at the University of California, Berkeley. He is an associate member of the American Society of Civil Engineers and a member of the Society for Industrial and Applied Mathematics.

Gene Kan is an engineer with the JXTA group, whose interests include peer-to-peer computing and distributed storage. He holds a BS in computer science from UC Berkeley. Kan was among the first to produce an open source version of Gnutella software after its release. In his spare moments, he enjoys racing his cars, LTLENDN and BIGENDN.

Yaroslav Faybishenko is an engineer with the JXTA group interested in decentralized network services. Faybishenko holds a BS in computer science from UC Berkeley.

Readers may contact the authors at Sun Microsystems, 901 San Antonio Road, Palo Alto, CA; {steve.waterhouse, david.doolin, gene.kan, yaroslav.faybishenko}@sun.com.