A P2P SERVICE DISCOVERY STRATEGY BASED ON CONTENT CATALOGUES

Lican Huang

Institute of Network & Distributed Computing, Zhejiang Sci-Tech University, No.5, St.2, Xiasha Higher Education Zone, Hangzhou, P.R.China
Hangzhou e-Brain InformationTechnology Co. Ltd.
Email: huang_lican@yahoo.co.uk

ABSTRACT

This paper presents a framework for distributed service discovery based on VIRGO P2P technologies. The services are classified as multi-layer, hierarchical catalogue domains according to their contents. The service providers, which have their own service registries such as UDDIs, register the services they provide and establish a virtual tree in a VIRGO network according to the domain of their service. The service location done by the proposed strategy is effective and guaranteed. This paper also discusses the primary implementation of service discovery based on Tomcat/Axis and jUDDI.

Keywords: P2P, Service discovery, e-Science, Grid, Web service, VIRGO

1 INTRODUCTION

Service discovery is an important issue in web services (Web Services, 2006) and grid technologies (Foster & Kesselman, 1997). The current strategy for service discovery is based on centralized registry servers such as UDDIs (UDDI, 2006), which have a shortage of crushdowns due to single point failures. When there are a large number of services and a large number of service providers, service discovery becomes difficult. In this case, distributed service discovery will be required. There are many approaches for distributed service discovery based on P2P technologies (Iamnitchi & Foster, 2001). Unstructured P2P technology such as Freenet (Clarke, Sandberg, Wiley, & Hong, 2000) using the method of flooding has problems with heavy traffic and non-guaranteed searches. Structured P2P technology using DHT, such as Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001) and Pastry (Rowstron & Druschel, 2001), loses the locality property. Service discovery based on VDHA (Huang, Wu, & Pan, 2003) has a problem with high traffic loads in its root nodes. VIRGO (Huang, 2005), a P2P network, is a hybrid of structured and unstructured P2P technologies. N-tuple replicated virtual tree structured route nodes are merged with randomly cached un-structured route nodes (LRU and MinD) to solve the above problems.

This paper presents a framework for distributed service discovery based on VIRGO P2P technologies. In this framework, the services are classified as multi-layer hierarchical, catalogue domains according to their contents. The providers of services join virtual groups in the VIRGO network whose domain names are the same as the domains of their provided services. If a provider provides many services that are classified into several different domains, this provider will join these different domains in the virtual VIRGO tree. All the providers have their

own service registries and register their provided services into their own registry. Service location is effective and guaranteed. The time complexity, space complexity, and message-cost of lookup protocol of VIRGO is *O* (logN), where N is the total number of nodes in the network.

The structure of this paper is as follows: Section 2 describes the VIRGO network structure; Section 3 presents the service discovery based on VIRGO; Section 4 discusses the implementation service of VIRGO; and finally we give conclusions.

2 VIRGO NETWORK STRUCTURE

VIRGO (Huang, 2005) is a domain-related, hierarchical structure formed as a hybrid of un-structured P2P and structured P2P technology. VIRGO consists of a prerequisite virtual group tree and connections cached by least-recently used and minimum-distance replacement strategies. The virtual group tree is similar to VDHA (Huang, Wu, & Pan, 2003) but with multiple gateway nodes in every group. The virtual group tree is virtually hierarchical, with one root-layer, several middle-layers, and many leaf virtual groups. Each group has N-tuple gateway nodes. In the VIRGO network, random connections cached in a node's route table are maintained by least-recently used (LRU) and minimum distance (MinD) replacement strategies. The LRU strategy gives more weight to the frequently requested nodes. The MinD strategy gives more chance to the node with more different domains than other nodes. These cached connections make VIRGO a distributed network, not just a virtual tree network like VDHA. With random cached connections, the net-like VIRGO avoids overload in root nodes in the virtual tree topology but keeps the advantage of effective message routing in a tree-like network. As the contents change in the route table, VIRGO uses different lookup protocols and maintainence protocols from VDHA.

The user uses clients to access VIRGO via the access point node (called an entrance node). All users are managed by their owner nodes. Some nodes (called Gateway nodes) take route functions in several different layers of virtual groups. A Gateway node is a node located in both lower and upper layer groups.

Figure 1 shows two-tuple virtual hierarchical tree topology (the nodes in different layers connected with the dash line are actually the same node). In Figure 1, from the real network, three virtual overlay groups are organized. From these virtual groups, two nodes per virtual group are chosen to form the upper layer virtual group.

3 SERVICE DISCOVERY BASED ON VIRGO

The VIRGO node is a service provider as well as a service consumer. The services are classified into catalogues. The node publishes its services and their access controls in its local service repository. This node then joins with the virtual group whose domain is the same as the service catalogue’s domain. For example, if one node provides a song by Madonna, which is classified as music.popular.Madonna, it registers the service into its own repository and joins the virtual group, music.popular.Madonna, in the VIRGO network.

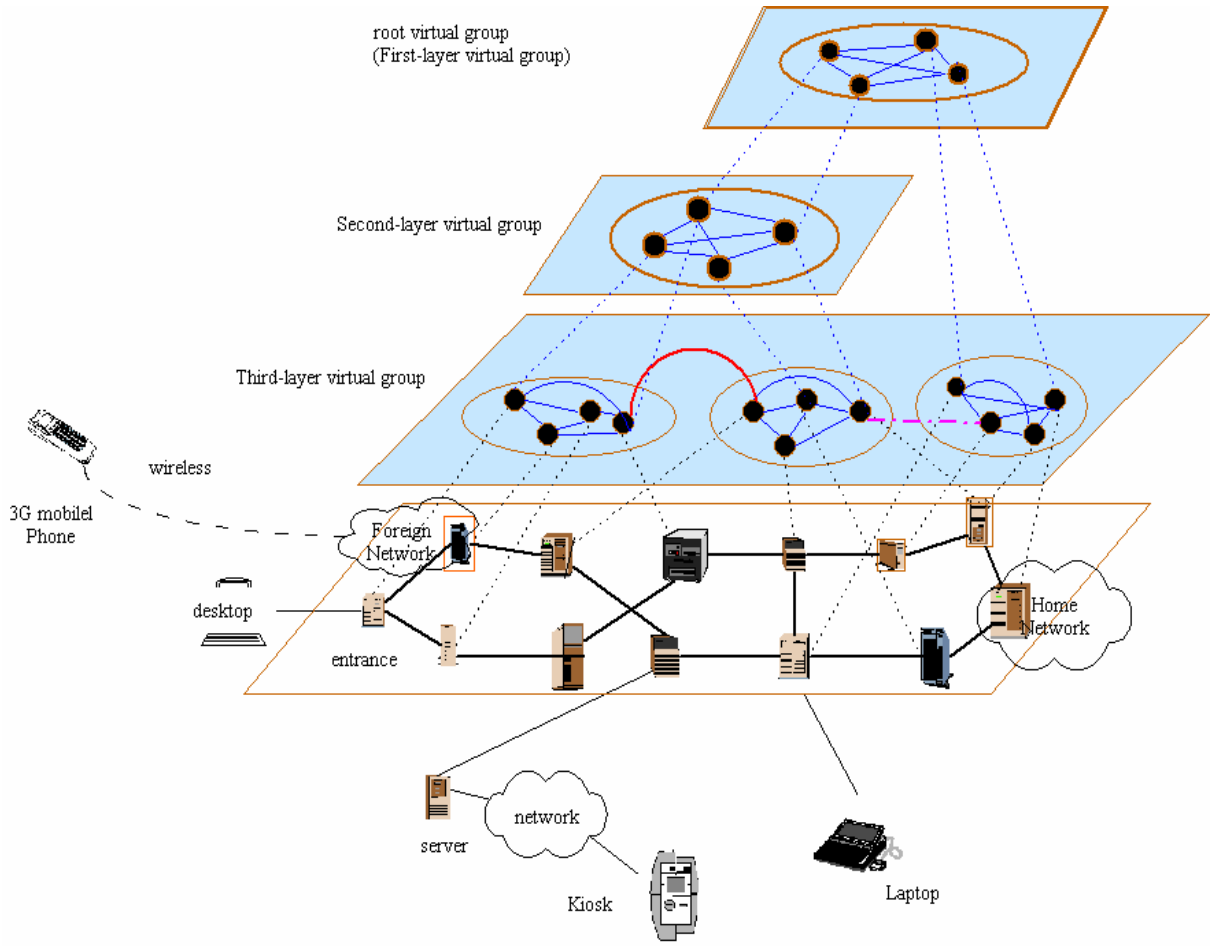

Figure 1. Two_tuple Virtual Hierarchical Tree Topology

The discovery service is based on the VIRGO lookup protocol.

The lookup protocol for VIRGO is based on the strategy that a node in a route table, which has the minimum distance from the destination node, is chosen as the next hop route. Theoretical hops from a node to a destination node are calculated by the node's location in the virtual tree and equal pre-fixed lengths between the domain of this node and the destination domain, whose details are described in Huang (2005). VIRGO uses the strategies of LRU and MinD for caching route nodes to solve the overload problems of computation and messaging existing in the tree structure.

The steps for VIRGO's lookup protocol are as follows:

Step 1 - user uses client to send QUERY MESSAGE to entrance node.

Step 2 - entrance node forwards QUERY MESSAGE to user's owner node.

Step 3 - user's owner node checks the user's authentication.

Step 4 - user's owner node routes to the node, which is closer to the destination group.

Step 5 - the route node routes to the node closer to the destination group. Step 5 is repeated until the destination group has been found.

Step 6 - the node found by step 5 broadcasts QUERY MESSAGE to all nodes belonging to the destination

group and gets the responses from all the nodes.

Step 7 - the node sends the RESULT MESSAGE to the user's owner node.

Step 8 - owner node sends RESULT MESSAGE to entrance node. The latter forwards the message to the client.

In the lookup protocol, the client first sends the request message, containing a request message ID, an authenticate-ticket, the IP address of the entrance node, the IP address of the user's owner node, the domain name of the destination, and the request context, to the entrance node. Then the entrance node forwards the request message to the user's owner node, which is the starting point for looking up the destination node. This method has two advantages. For mobile users, it can login into the VIRGO network from anywhere. The other advantage is that the owner node can maintain and authenticate its own users. The owner node will locate the destination of the group, find the requested information, and send the response message, containing the request message ID, the IP address of the entrance node, the destination virtual group node's information, and the answer context, to the entrance node. Finally the entrance node will forward the response message to the client.

Let us take a music providing service as an example (see Figure 2).

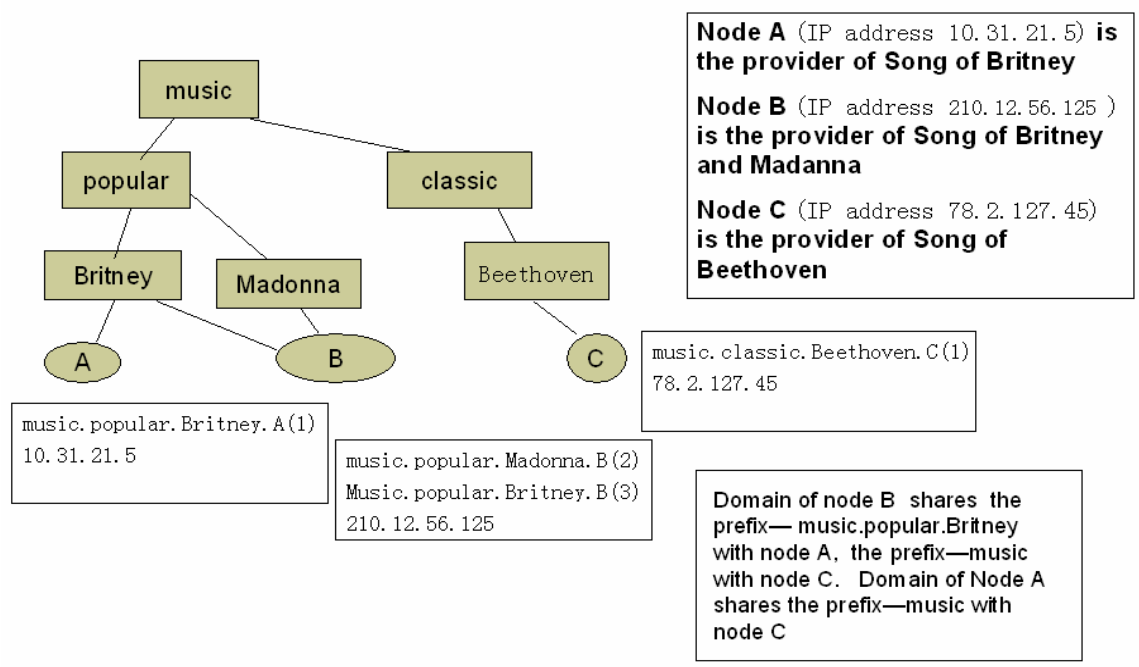

Figure 2. Example of a music providing service

In Figure 2, Node A is the provider of Song of Britney, so it is classified as the group of music.popular.Britney. Node B is the provider of Song of Britney and Madonna and joins the groups of music.popular.Britney and music.popular.Madonna. Node C is the provider of Song of Beethoven, so it is classified as the group of music.classic.Beethoven. The route tables of this example can be found in Table 1.

Table 1. Route tables of example of music providing service

(1) Route table of node A

ID	IP Address	GUL	type	Is self?
music.popular.Britney.A	10.31.21.5	1	TREE	Yes
music.popular.Britney.B	210.12.56.125	3	TREE	No
music.popular.Madonna.B	210.12.56.125	2	TREE	No
music.classic.Beethoven.C	78.2.127.45	1	TREE	No

(2) Route table of node B

ID	IP Address	GUL	type	Is self?
music.popular.Britney.A	10.31.21.5	1	TREE	No
music.popular.Britney.B	210.12.56.125	3	TREE	Yes
music.popular.Madonna.B	210.12.56.125	2	TREE	Yes

(3) Route table of node C

ID	IP Address	GUL	type	Is self?
music.popular.Britney.A	10.31.21.5	1	TREE	No
music.classic.Beethoven.C	78.2.127.45	1	TREE	Yes

A scenario for the discovery service in this example follows: A user (supposing that its owner node is B) wants to consume (download) Beethoven's Ninth Symphony from some node in VIRGO. The user first uses a client via the entrance node to access VIRGO. Then, the entrance node forwards the request message, which contains the destination group music.classic.Beethoven, to the owner node B. In node B's route table, music.popular.Britney.A (10.31.21.5, 1, tree) is min hop distance with music.classic.Beethoven (see distance formula in Huang, 2005). Therefore, it forwards the message to node A. In node A's route table, exists music.classic.Beethoven.C (78.2.127.45, 1, tree). So node C broadcasts the request message to the group of music.classic.Beethoven. In this example, only node C is in the group. If node C contains Beethoven's Ninth Symphony, node C will send the result message to owner node B. Owner node B will forward the result message to the entrance node, and the latter will forward the result message to the user. The user can download Beethoven's Ninth Symphony if the user passes node C's access control. Meanwhile, owner node B will cache node C in its route table as type LRU(music.classic.Beethoven.C, 78.2.127.45, 1, LRU). The next time a user requests Beethoven's Symphony No. 5, he is routed directly to node C rather than via node A. This will avoid heavy traffic in the root nodes of the tree structure.

4 IMPLEMENTATION

We have thus implemented a prototype of a discovery service. The discovery service is one module of VIRGO. The communication protocol for VIRGO is http. The VIRGO node uses Tomcat as Web Server and Axis as Web service container (see Figure 3). Every node implements Lookup protocol, VIRGO structure maintenance, Web Service matchmaker, and Registry. The GUI of VIRGO is in the form of web pages. The interfaces for a node's joining and leaving, service discovery, etc. are written by JSP, which sends the messages to the VIRGO core program. The VIRGO core program implements all protocols for VIRGO. The route table is stored by mySQL database. The messages are XML-formed. When a user inputs a request message for a service location, the

client will send the message to an entrance node; this entrance node will forward the message to the destination node after several intermediate route nodes. The destination node will respond to the result message from the user.

The jUDDI (jUDDI, 2006) is the service registry. The service is registered by UDDI Client--UDDI Browser version 0.2 (UDDI Browser, 2006). The name in the businessEntity of UDDI is filled by a service catalogue domain such as all.service.science.Bioinformatics. When a Query Message is forwarded to the node that belongs to the destination group, the node first locates the businessEntity using an object domain and then finds the required information. The XML format for QueryMessage is as follows:

<querymessage> *<UserID>…</UserID> <ClientID>…</ClientID><entranceNode>…</entranceNode> <ownerNode>…</ownerNode> <ObjectDomain>…</ObejectDomain><serviceMeta>…</serviceMeta> <AuthenticationTicket> …< /AuthenticationTicket>* ***</querymessage>***

The XML format for Result Message is as follows:

*<**resultmessage**><UserID>…</UserID><ClientID>…</ClientID><entranceNode>…</entranceNode> <ownerNode>…</ownerNode><ObjectDomain>…</ObejectDomain><serviceMeta> …</serviceMeta> <AuthenticationTicket> …< /AuthenticationTicket><serviceLocation>…<serviceLocation></**resultmessage**>*

The serviceLocation in resultmessage contains the details of the service published in UDDI.

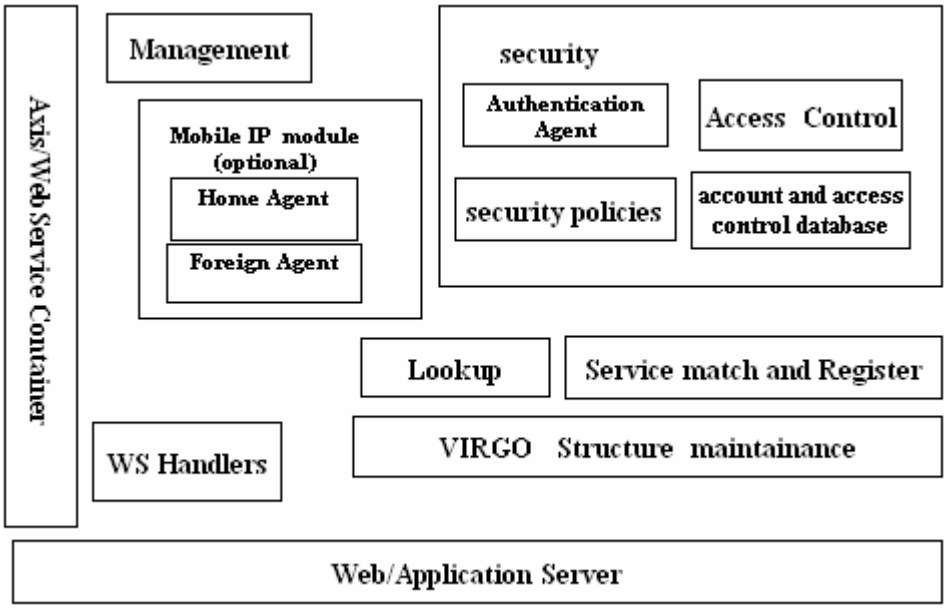

Figure 3. Architecture of VIRGO

5 CONCLUSION

This paper presents a framework for a discovery service based on VIRGO P2P technologies. VIRGO uses a hybrid of structured and unstructured P2P technologies by merging n-tuple, replicated virtual tree structured route nodes, and randomly cached un-structured route nodes. As the VIRGO network is organized,

VIRGO-based distributed service discovery becomes fully self-organized. The node in VIRGO publishes the services (classified as hierarchical domains) into its own registry (such as UDDI) and joins virtual groups of the same domains as the services are classified. The lookup protocol first locates the destination virtual group and then broadcasts the message to all the members of the virtual destination group. Because every node has its own registry, the VIRGO-based distributed discovery service is self-contained; that is, the node can work in a local network if it is not joined to a VIRGO network. The VIRGO-based distributed discovery service is effective and guaranteed. The time complexity, space complexity, and message-cost of the lookup protocol of VIRGO is *O*(logN), where N is the total number of nodes in the network.

We have started an open source project for VIRGO (VIRGO, 2006) and are about to implement VIRGO software. As UDDI is not suitable for the Ontology description of Web Services, we plan to implement an Ontology register different from UDDI. We also plan to develop a distributed web service discovery package based on VIRGO, which may be used in the WOSE project (Huang , Walker , Huang, & Rana , 2005).

6 REFERENCES

Clarke, I., Sandberg, O., Wiley, B., & Hong, T. (2000) Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pp 311–320, July 2000. ICSI, Berkeley, CA, USA.

Foster, I. & Kesselman, C. (1997) Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications,11(2)*: 115-128

Iamnitchi, A. & Foster, I. (2001) On Fully Decentralized Resource Discovery in Grid Environments, *International Workshop on Grid Computing 2001*. Denver, Colorado, USA

Rowstron, A. & Druschel, P. (2001) Pastry:Scalable, distributed object location and routing for large-scale peer-to-pear systems. *In Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware).* Retrieved from the WWW, July 31, 2007:http://research.microsoft.com/~antr/PAST/pastry.pdf

Huang, L. (2005) VIRGO: Virtual Hierarchical Overlay Network for Scalable Grid Computing *Proc. European Grid Conference (EGC2005)*. In *LNCS 3470*, pp 911-921, February 14-16, 2005, Amsterdam, Netherlands.

Huang, L. , Walker, D., Huang, Y., & Rana, O. (2005) Dynamic Web Services Selection for Workflow Optimization, *UK e-Science Programme All Hands Meeting 2005 (AHM2005)*, Nottingham, UK. Retrieved from the WWW, July 31, 2007:http://www.allhands.org.uk/2005/proceedings/papers/423.pdf

Huang, L., Wu, Z., & Pan, Y. (2003) A Scalable and Effective Architecture for Grid Services' Discovery. *in the proceedings of 1st workshop on Semantics in Peer-to-Peer and Grid Computing at the Twelfth International World Wide Web Conference*, Budapest, Hungary.

Huang, L., Wu, Z., & Pan, Y. (2003) Virtual and Dynamic Hierarchical Architecture for e-Science Grid, *International Journal of High Performance Computing Applications 17(3)*:329-347

Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., & Balakrishnan, H. (2001) Chord: a scalable peer-to-peer lookup service for internet applications, *In Proceedings of ACM SIGCOMM2001*. Retrieved from the WWW, July 31, 2007:http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

UDDI (2006) HomePage of UDDI. Retrieved from the WWW, July 31, 2007:http://www.uddi.org/

JUDDI (2006) HomePage of jUDDI. Retrieved from the WWW, July 31, 2007:http://ws.apache.org/juddi/

UDDI Browser (2006) HomePage of UDDI. Browser Retrieved from the WWW, July 31, 2007: http://uddibrowser.org/

VIRGO (2006) HomePage of VIRGO. Retrieved from the WWW, July 31, 2007: http://virgo.sourceforge.net/

Web Services(2006) HomePage of Web Services. Retrieved from the WWW, July 31, 2007: http://www.w3.org/TR/ws-arch/