A P2P Service Discovery Strategy
Based on Content Catalogues

Dr. Lican Huang, Director
Institute of Network & Distributed Computing
Zhejiang Sci-Tech University
Hangzhou e-Brain Information Company, LTD
Outline

- Introduction
- Overview of VIRGO
- P2P Service Discovery Based on VIRGO
- Implementation
- Conclusion
Introduction

- The current strategy for service discovery is based on centralized registers such as UDDI.
- More distributed service discoveries based on P2P technologies such as Chord lose locality.
- Distributed service discovery based on VIRGO may solve the above problems.
Overview of VIRGO

- **Decentralization**: VIRGO is fully distributed, robust, easy-managed.
- **Load balance**: Cached LRU and MinD nodes in route table help to solve the problem of the load balance in the tree structure.
- **Scalability**: Time complexity, space complexity and message-cost of lookup protocol of VIRGO is $O(\log N)$, where $N$ is the total number of nodes in the network.
- **Availability**: There is at least one path between every two nodes.
VIRGO - two_tuple Virtual Hierarchical Overlay Network-1
VIRGO - two_tuple Virtual Hierarchical Overlay Network-2

Taking a look at the diagram, it illustrates the hierarchical structure of the VIRGO network. The network is divided into layers, with each layer representing a different level of abstraction.

- **Root Layer**: This is the highest level and represents the overall network structure.
- **Second Layer**: This layer includes categories such as Science, Biology, Botany, Science, Chemistry, and Science, Computer, Network.
- **Third Layer**: This layer provides more detailed categories like BT3, BT2, BT1, A12, A13, A14, and Net1, Net2, Net3.

For example, the node BT3 has a type of TREE and a route node BT1, BT2, BT3, BT1, BT2, BT3, A12, A13, Net2. Similarly, the node Net1 has a type of TREE and a route node Net2, Net3, A12, A13, A14, Net1, Net2, Net3.

The network architecture is designed to allow nodes to communicate efficiently by using the appropriate routing algorithms, such as TREE, LRU, and MinD, to find the best path to their destination.
One_tuple Virtual Hierarchical Overlay Network—Music Example

- Music Catalogue
  - music.popular
  - music.classic
- There are 3 nodes--Node A, B, C
- A node can join more than one group
New VIRGO Network Creation

(1) Node A (IP address 10.31.21.5) Set up new VIRGO network

Node A is the provider of Song of Britney, so it is classified as the group of music.popular.Britney
Node Join-1

Node C (IP address 78.2.127.45) joins VIRGO network

Node C is the provider of Song of Beethoven, so it is classified as the group of music.classic.Beethoven

Domains of Node A and C share the prefix--music

music.popular.Britney.A(1)
10.31.21.5

music.classic.Beethoven.C
(1)
78.2.127.45

Node B is the provider of Song of Britney and Madonna, so it is classified as the groups of music.popular.Britney and music.popular.Madonna.

Domain of node B shares the prefix—music.popular.Britney with node A, the prefix—music with node C.


music.classic.Beethoven.C (1) 78.2.127.45
Node’s Join Protocol

1. $P_{\text{join}}$. send(JOINMESSAGE, $P_{\text{groupToJoin}}$)
2. $P_{\text{groupToJoin}}$. send(JOINMESSAGE, $\forall p_i \{p_i \in \text{joinGroup}\}$)
3. $\forall p_i \{p_i \in \text{joinGroup}\}$. send($p_i$. APPROVEMESSAGE, $P_{\text{join}}$);
   $\forall p_i \{p_i \in \text{joinGroup}\}$. RouteTableAdd($P_{\text{join}}$. NE, TREE)
4. $P_{\text{join}}$. RouteTableAdd($\forall p_i \{p_i \in \text{joinGroup}\}$. NE, TREE)
5. Repeat step 2 to 4 in upper layer groups until replicated nodes no less than n-tuple or root group.
Node’s Leave Protocol

1. $P_{dpt}. \text{send}(\text{LEFTMESSAGE, } \forall p_i \{p_i \in \text{leftgroup}\})$
2. $\forall p_i \{p_i \in \text{leftgroup}\}. \text{RouteTableDelete}(P_{dpt}. \text{NE}, \text{TREE})$
3. Choose $p_{\text{replacenode}}$ to replace the left node’s role
4. Repeat step 1 and 3 in lower layer groups until to the bottom layer group of $P_{dpt}$. 
Node’s failure Protocol

1. \( P_{\text{notice}}. \text{send(FAILUREMESSAGE, } \forall p_i \{ p_i \in \text{failgroup} \} \) 
2. \( \forall p_i \{ p_i \in \text{failgroup} \}. \text{RouteTableDelete}(P_{\text{fail}}. \text{NE, TREE}) \)
3. Choose \( p_{\text{replacenode}} \) to replace the left node’s role
4. Repeat step 1 and 3 in lower layer groups until to the bottom layer group of left node \( P_{\text{fail}} \).
Software Architecture of VIRGO
VIRGO based Distributed Service Discovery

- Web Services are classified into catalogues according to their functions or disciplines such as (all.service.science.bioinformatics).
- Service providers registry their services into their own Registers such as UDDIs.
- Service providers join VIRGO network according to their domains which are the same as catalogues of web services they provided.
UDDI-classification

Use yellow page as Domain Catalogue

Web Service
Lookup Protocol

- 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 closer node to the destination group. Repeat process step 5 until the destination group has been found.
- Step 6 the node found by step 5 broadcasts QUERY MESSAGE to all nodes belongs 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.
Service Lookup-QueryMessage

XML Format for QueryMessage:

<querymessage>
<UserID>…</UserID>
<ClientID>…</ClientID>
<entranceNode>…</entranceNode>
<ownerNode>…</ownerNode>
<ObjectDomain>…</ObjectDomain>
<serviceMeta> …</serviceMeta>
<AuthenticationTicket> …</AuthenticationTicket>
</querymessage>
XML Format for ResultMessage:

<resultmessage>
<UserID>…</UserID>
<ClientID>…</ClientID>
<entranceNode>…</entranceNode>
<ownerNode>…</ownerNode>
<ObjectDomain>…</ObjectDomain>
<serviceMeta> …</serviceMeta>
<AuthenticationTicket> …</AuthenticationTicket>
<serviceLocation>…</serviceLocation>
</resultmessage>
Implementation-- Enviroment

*Program languages:* Java, Jsp

*Operation systems:* Linux, Windows

*Web service container:* Tomcat + AXIS

*Database:* mySQL

*UDDI server:* JUDDI

*UDDI Client:* UDDI Browser version 0.2
Implementation-Package
Conclusion

1. VIRGO-based distributed service discovery is fully self-organized.

2. VIRGO-based distributed service discovery is self-contained.

3. VIRGO-based distributed service discovery is effective.

4. All service providers have their own service registers (such as UDDI)

5. All messages are XML-formed
More material

http://virgo.sourceforge.net/
Question?

Thanks