IMPLEMENTATION OF THE SERVICE SEARCH IN THE WOSNET

MARKUS WULFF
FB Informatik, Universität Rostock, D-18051 Rostock, Germany
mwulff@informatik.uni-rostock.de

Abstract
The WOSNet builds a large distributed system without a central resource management. Therefore, a suitable strategy is presented to locate services and resources in the WOSNet. This paper gives the background for the WOS Service Search Unit and describes the used mechanisms as well as the implementation of the service search and its integration in the WOS.

Keywords: Communication, Distributed Systems, Search methods, World Wide Web, WOS

1 INTRODUCTION

The WOSTM [11] approach to global distributed computing [7] aims to provide service mechanisms which meet the requirements of the net-centric view of services and processes. Using this concept, each WOS node operates as a server as well as a client. Therefore, the WOSNet consists of a series of versioned servers or nodes which can provide a set of services and resources [9].

There is no central resource management in the WOS [6, 7, 9]. That means, no machine has information about all available services and resources in the WOSNet. Each WOS node saves its own parameters and the parameters of other known nodes, used for remote service execution, in its local warehouse. If a new machine is added to the WOSNet, it obtains information about other nodes and provides information about itself using a suitable initialisation routine [1].

Using this decentral principle, the system achieves a high flexibility. There is no bottleneck as in systems with a central information management. The disadvantage of the decentral method is the limited set of information, which each node has at its disposal. In addition it is not useful to perform a high frequent update. So it is necessary to develop a well directed search in the WOSNet for services and information on demand [2, 8].

In fact, the WOS needs two different kinds of search. First, the search for a node which speaks the appropriate version of WOSP [1] and second the actual search for the node which can provide the service or resources needed by the user. The essential difference is, that the search for a suitable version must run at the low communication level (WOSRP) and the service search mechanism is provides by the WOS Service Search Unit at the WOSP protocol level. This article only describes the WOS service search.
2 BASIC SEARCH MECHANISMS

Each WOS node stores information about local and remote services in the local warehouse. Assumed, a certain service is requested by the user and there is no entry for this service in the machines warehouse, a search will be initiated. A list of addresses of remote nodes from the local warehouse is created for that [4].

There are two basic ways to do this search [3, 7]. The first one is the broadcast. That means, that the requesting machine sends its request to all machines in the list at the same time (ignoring the time for generating and sending of each message). The advantage is a fast reply, but the resulting network load is high. The second way to find a service in the WOSNet uses the serial chains. In that case only one message containing the list of all nodes is sent to the first address in the list. The first machine then passes the message to the next node and so on. Here the problems are exactly the opposite — a low network load but a long average response time.

A combination of both strategies should solve these problems by improving the relation between network load and response time.

2.1 Sequential Search Chains

The basic idea of the “sequential search chains” is the combination of the broadcast and the serial concept [1], shown in Fig. 1.

![Sequential Search Chains Diagram](image)

**Figure 1: Sequential Search Chains**

First, a list of $n$ addresses is splitted into $a$ parts. Each of these parts now contains $b$ (or maybe less in chain $a$) addresses and will become one search chain. In [8] a number of six to ten chains is recommended. All $a$ chains are sent out parallel, but the elements (addresses) of each chain are processed in serial manner. If the message (search token) reaches the first node in list, the node checks its local warehouse for the requested service. Now there are three possible results:

1. The service is available. In this case the node sends the positive answer directly back to the requesting machine.
2. The service is not available on this machine. But there is an entry of a remote machine providing the requested service. The address of this node is added to the service request message (for details see Sec. 3).

3. The service is not available.

In every case the current node removes its address from the list and sends the request message to the next address in list, if any. If not, the last node in chain sends a termination message to the requesting machine.

To improve the response time of a service request, another combination of different strategies is possible—the search tree. If a node can not provide a required service, it can initiate a search by itself using the described principle. This approach creates a tree-like search graph, but it has three major disadvantages.

- The network load could become very high.
- The original node has no control about the subchains and their termination.
- Many machines would possibly being ask for the same service more than once.

2.2 Fault Tolerance

Since WOS is developed for wide area networks, a fault tolerance a mechanism is necessary (see Fig. 2). Two types of failure may occur [1]:

1. The network breaks down or times out while the message is being transmitted.
2. A WOS server cannot serve the request after receiving it.

![Figure 2: Fault tolerance by using acknowledgement messages](image)

The use of two acknowledgement messages should overcome these problems. The reception acknowledgement (RACK) confirms the proper reception of the message by the next machine in chain. The termination acknowledgement (TACK) indicates that all WOS nodes in the chain were visited.
3 IMPLEMENTATION

To implement the search mechanisms summarised in Sec. 2, the functionality of a WOS node must be enlarged. A new module is needed, which handles the incoming search tokens and which can also create and send out search chains if the Remote Resource Control Unit (RRCU) want it to do. So, the RRCU initiated a service request and updates also the local warehouse with results of a search. Because all actions need basic communication routines, at first the used communication interface shall be discussed in the following part.

3.1 The communication

For all communication which comes up to realize a service request, the WOS communication layer interface, the Triplets [10], are used. The communication in the WOS provides two possible modes — the connectionless mode and the connection-oriented mode. In order to ensure an efficient search, the connectionless mode is used. That means that there will be no WOSP connection established to sent a request. But the use of TCP communication mode for a save delivery on the lower communication layers is still possible. Fig. 3 shows the flow of the connectionless communication.

![Connectionless communication mode](image)

**Figure 3: Connectionless communication mode**

If WOSP Analyzer 1 at node N1 wants to send a message it has to call the WOSP Parser with the message to be sent as parameter (1). The WOSP Parser generates a message send that message via WOSRP (2,3) to the remote node. WOSRP sends the message through the network (3) to the WOSRP at the remote node N2. The remote WOSRP searches its local warehouse
(4,5) for the version ID and server name. If compatible, WOSRP launches the corresponding WOSP Analyzer 2 server with the parameters below and sends it the message (6). The WOSP Analyzer 2 (at node N2) interprets its protocol commands and converts the message into Triplet format using WOSP Parser (7,8). If the message contains a service request, the message is passed to the Service Search Unit (SSU), shown in Fig. 4.

Figure 4: The flow of a service request inside a node

The service request is processed as follows: The incoming request is passed by the communication layer to the Service Search Unit (1). Then, after analysing the message, the local RRCU is required to check its Resource Warehouse (2) for an appropriate entry (3). If the service is available on this machine (4), the RRCU gives all parameters and restrictions for this service back to the SSU (5) and a reply message is sent to the requesting node. If the service is only "known" (4'), the return from the RRCU (5) is a list of WOS server addresses, known by this node, which provides the service. This list of addresses is not directly replied, but appended to the original service request message. This helps to minimise the network load and prevents the multiple reply of one address by removing addresses which are already appended by other servers. Last (6), the search token is passed to the next node in the list or, if there is no further address, the TACK is sent out. Before passing on the message, the address of the next server is removed from the list.

The reply (7) is not handled by the SSU. The message handler of the WOS communication layer passes a reply directly to the RRCU. The RRCU can now update the local resource warehouse with the reply data (7').

3.2 The message format

Therefore, the message format of a service request must contain the following information:

- the name of the required service
- possible execution parameters
- the address of the requesting WOS node
- the list of addresses of nodes to be visited
- an initially empty list for the reply of addresses
As said above, the communication interface of the WOS are Triplets. The format of the Triplet-list is as follows:

```
query data *[metadata]* data data data
```

The service request message contains at least five Triplets (see also Fig. 5). The first one contains the message type (QUERY). Together with the identifier _ServiceSearch_ these Triplet marks the message as WOS service request. The QUERY-Triplet is followed by four DATA-Triplets. One for the name of the service requested an other one for the address of the requesting machine. After the Triplet with the name of the service may follow one or more Triplets describing the service, if necessary (not shown in Fig. 5). The fourth Triplet of the request contains a space separated list of all nodes to be visited. The last Triplet is initially empty. Each node can fill in there addresses of other nodes providing the service from their own local warehouse.

![Figure 5: Triplet message for a service request](image)

If a node can offer the required service, it sends a reply message back to the requesting machine. This reply contains:

- the name of the required service
- parameters and restrictions for this service

Parameters of a service are all details concerning the service itself (eg. environment, resources etc.) and statements about the availability of the service (eg. time windows, prices etc.). The message contains at least three Triplets. So the message format is:

```
reply data *[metadata]* *[data *[metadata]* ]*
```

The first one is the REPLY-Triplet followed by a DATA-Triplet with the name of the service and then follows one or more METADATA-Triplets containing all parameters for the service. If there is more than one match for the request, all services can appended to the reply using the same format. The resulting list of Triplets is shown in Fig. 6.

![Figure 6: Triplet message for a service reply](image)

The interface to the Service Search Unit is the method WOS_Search(), which is part of the class WOS_ServiceSearch. It is normally called by the RRCU if a certain service is not available on the local machine.
public String[] WOS_Search(WOSP_ListOfTriplets serviceName, String wosPID, String versionID, InputStream addrList, boolean search)

returns: MessageID's of all chains if successful, null otherwise

where

serviceName: name of the required service and parameters
wosPID: ID of the requesting process
versionID: WOSP version ID
addrList: list of WOS servers to be visited
search: search if true, execution request otherwise (see below)

This search mechanism is used in two cases. First to locate WOS server, which can provide a special service and on the other hand to find the best server if the service actually should be used. Best server means for instance the machine with the currently lowest workload or with the lowest price and so on. The only modification for the second type of request is the identification string of the QUERY-Triplet. It is changed to __ExecutionRequest___. All other steps are from the search point of view the same.

4 CONCLUSIONS

In this paper a reliable and efficient search method for the service search in the Web Operating System and other large distributed systems was introduced. The specified system builds a compromise between speed and resource consumption. In such a manner the implementation fulfills the requirements of a search engine with no centralised data catalogues. While the implementation is almost finished, further mechanisms to improve security and performance must still be discussed and included.
REFERENCES


Author
Markus Wulff is a student of Computer Science at the University of Rostock since 1996. His interest includes distributed and parallel computing as well as communication. He is working in the WOS project group since 1998. As a guest of the Laval University of Québec (Canada) in 1998 he was substantial involved in the development of the WOS communication layer.