Service Base System : A framework for distributed utilities

Tadashi Ogino **Hidehiko Tanaka**

Department of Electrical Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan

June 20, 1987

1 Introduction

In recent years, computer communication technology has been developing rapidly and it becomes very natural to connect multi computers with network.

However, the major part of recent studies about computer networks seems to be limited to communication technology. It is sure that we have to consider a good method to manage distributed computer resources such as program and data.

We have been studying a system to accomplish purposes below,

1. A user can combine and make use of distributed programs and data with no difficulties, even if he doesn't know where they really exist or how to use remote machines.

2. New facilities can be developed and added to the system easily regardless of other nodes in the network.

We don't think to develop a new network operating system from beginning. We are to propose a framework to manage distributed utilities and we think our system can be constructed on an existing NOS.

Section 2 of this paper describes an overview of Service Base System (SBS), and next we explain a structure of SBS. After we mention about experimental systems and discuss about SBS, section 6 provides a summary of status and trials.

2 Overview of Service Base System

We want to construct a system which is independent of some fixed architecture. So we have to establish a model for computer resources as simple as

possible. About communication protocol, only high level protocol is defined. A concrete method to establish actual system and low level protocol is left to system creators. First, we will propose a model of Service Base System (SBS). And next we will explain request and answer model and three layered views. In the last part of this section, we also refer to service description.

2.1 A model for Service Base System

We call all the utilities which is offered by computer systems 'service'. We are to consider a system which deals with services. So we call our system "Service Base System (SBS)".

A service is actualized as some data and some operation on the data (Figure 1). For example, if you want to compile a fortran program, you

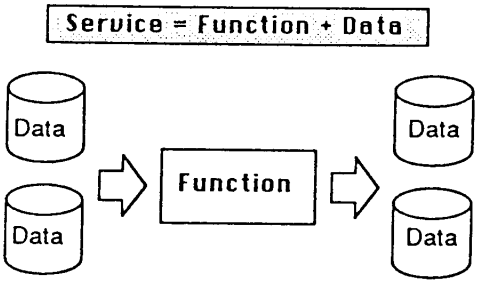

Figure 1: Service model

can regard a source text written in fortran as input data, an object code as output data and fortran compiler as operation on them. It follows that resources which SBS have to manage are divided into three classes such as data, function (operation) and services which are actualized as some data and function.

As data model for SBS, we adopt relational data model which is very flexible and popular. But in order to make it possible to deal with data which it is difficult to describe with relational model, we can use stream type data. Stream type data is used when we deal with input data from keyboard and output data to display.

Relational type data is defined to fix attribute, domain and so. Stream type data is a data with which you can read and write character one by one. It is defined to fix character set, its order rule and so. Two types of data are not divided exactly and some data can be dealt with as relational type data and as stream type data.

Function is a facility which operates on no or more than one input data and output more than one data. Function is defined to fix allowable input

Service is defined to fix input data, output data and function. We can register a new service which is combined from existing services. A user can get desirous facilities to combine primitive data, primitive function and existing services.

2.2 Request and Answer model

In the SBS, all of the communication between user and machines, between machine and machine are standardized by the method which we call "service request and answer".

Service request is to designate a combination of data, function or services. Service answer is to return service execution result such as if service is executed normally or not.

Next we consider logical network in the SBS. A scope which is managed by single management system is called a 'node'. Normally, one node corresponds to one machine. It is possible that one node corresponds to multi machines which are connected to one local area network.

If node A can request a service to node B, node A is called front node (FN) of node B, node B is called back node (BN) of node A (Figure 2). Logically,

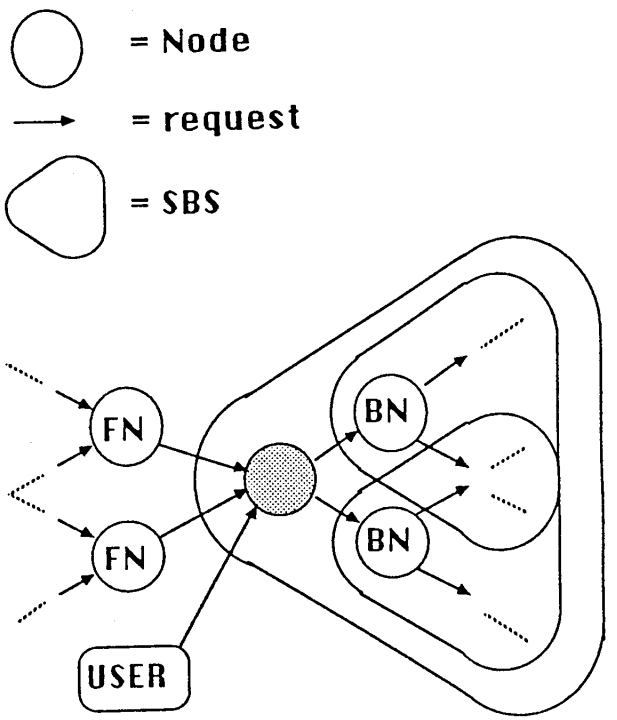

Figure 2: Network model of SBS

FNs of one node and BNs of the node are distinguished. But they can be the

same node physically. It is not necessary that network organization must be tree structure.

A service request is not sent to a node where the service actually exists, but it is sent to a node which knows the information about the service. Of course, the service may exists at the latter node. But if not, the node has to know where to request the service. Any way the service request is sent to the next node in succession and finally the service request arrives at the node where the service really exists.

As you can reallize from above explanation, the relation between any two nodes is equivalent. So, it does not happen that load power is centralized to specific nodes.

One node has every information concerning some part of the services offered by BNs. It doesn't know some part of that services. Only an administrator of the node can select necessary services. As a result, it is not true that all the services that the network offers can be accessed from any node in the network. But it may save a lot of trouble and we can get much merit totally. The computer facilities are quite stationary and it is not necessary to manage the services in real time. It may be enough that a system manager can register the service at any time when you need some specific service.

2.3 Three layered views

In the SBS, as mentioned above, each node has information about the services the node can offer. In order to manage this information about services, we adopt a management method using three layered views. We apply the concept of three layered schema in the DB field to this method. The information about services is managed with three hierarchical views, such as internal view, conceptual view and external view (Figure 3).

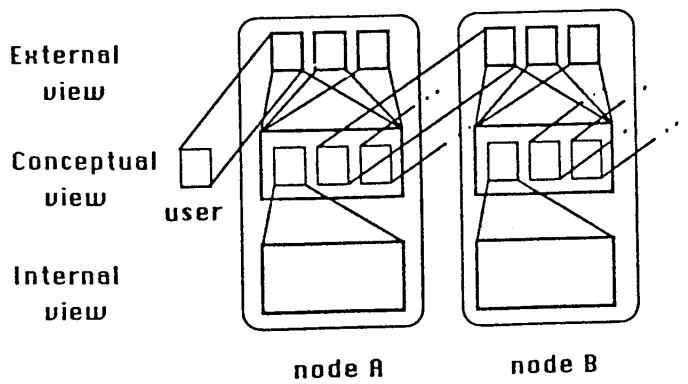

Figure 3: Three layered views

Each node has one internal view and it is the set of description about the services which exist in the local node.

Each node has one conceptual view. It unites the view about the services which exist in the local system and the view about the services offered by BNs, and absorbs the difference of distribution.

External view is made for each FN and each user. It is the description about the services which are offered to FNs and users.

2.4 Service description

In the SBS, the information about each data and function are described and stored. The information that have to be stored are service name, classification if it is data or function or service, existing node, mapping information to the next view, dictionary description and directory description. Dictionary description and directory description are description about contents of the service.

To understand the meaning of one service and to execute it, it is necessary to have description about contents of the service. Let's think about data. When a user want to use some data, the description he needs is about what kind of data that is. For example, one data is "an address book of the working staff" and the other is "sort program written in pascal". But for machines, the necessary information is such like the access method of the data or the data format and so on.

As mentioned above, the necessary information about one data is divided into the one for user and the one for machine. It may be the same result about function or service. In the SBS, the information for user is called 'dictionary' and that for machine is called 'directory'. To say strictly, two kinds of information may not be the different information. But we are to classify the information to two type.

3 An organization of SBS

An organization of a node which composes SBS is shown in Figure 4. It is assumed that each node has its own Operating System and Data Base System. They are called LOS (local OS) and LDBS (local DBS) respectively.

It is possible to construct LOS and LDBS for SBS from beginning. However we assume that LOS and LDBS have already existed and we make SBS on them.

The necessary facility of LOS is communication facility, to say nothing of general OS facilities like process management, memory management and so on. Communication facilities mean, at least, message communication function between other node and file transfer function.

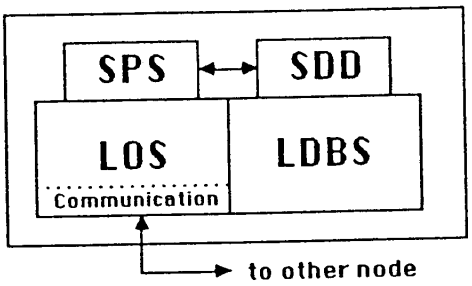

Figure 4: An organization of a node

As for LDBS, it is sufficient that it can deal with Relational Data Base System.

The new parts we have to implement in order to construct SBS are SPS (Service Processing System) and SDD (Service Dictionary/Directory).

SPS manages the service request and answer message. It receives a service request sent by FN or user and interprets the request. If the service can be processed in the local node, SPS executes it using LOS facility. If it can't be processed locally, SPS sends a new service request to BN using communication facility.

SDD manages the service descriptions. SDD stores the service descriptions using LDBS facility. A query about some service is sent from SPS to SDD. SDD arranges corresponding data properly and transfers it to SPS. Moreover, if a new service definition is given from SPS, after SDD performs consistency check, security check and other checks, SDD stores the new definition in LDBS. SDD also deals with network information.

4 Experimental system

We have been developing a SBS experimental system. The available experimental system now make it clear that the service request and answer method goes well even in the environment where different kinds of machines are connected. In this section, we will explain the organization of experimental system and some examples.

4.1 An organization of the experimental system

The experimental system was made on the environment where three machines (Hitachi M680H, Dec Vax8600, Vax-11/730) are connected (Figure 5). M680H and Vax8600 are the machines in the Computer Center of University of Tokyo, and Vax-11/730 is in our laboratory.

OS of M680H is VOS3 and that of two Vaxs is UNIX. Communication program between M680H and Vax8600 is called 'cvos' and is offered to Vax8600

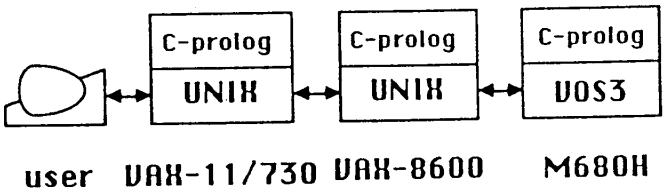

Figure 5: The experimental system

users to remote login to M680H from Vax8600 with a speed of 2400 bps. It also offers a file transfer facility. Communication program between Vax8600 and Vax-11/730 was developed by out project team. But it only offers a remote login facility from Vax-11/730 to Vax8600 with the speed of 9600 bps or less. I am sorry that two Vax are not connected with UNIX network facility.

As you can understand from above explanation, network ability in the experimental system is very poor. But in such conditions, we can show you good result.

System description language is C-prolog. SBS has to deal with the service description and we can regard the service description as the knowledge about the system. It is said that logic programming language is suited to deal with knowledge. So we have selected C-prolog to describe SBS experimental system.

4.2 Service execution

In this experimental system, all the services are dealt with like c-prolog predicates. If you want to execute the UNIX command "date", you can execute it as below.

```
| ?- date.
Thu May 7 17:25:06 GMT+9:00 1987
yes
```

4.3 Service definition

In this experimental system, only service name and existing node are stored as service description. We have made some service definition predicates below.

A predicate to register a service existing in local is assert_int(S_name, I_name). S_name is a service name to be used by users. I_name is a service name in the internal view.

A predicate to register a service existing in remote is **assert_ext(S_name, E_name, Node)**. **S_name** is a service name to be used by users. **E_name** is a service name in the external view of remote machine. **Node** is a node identifier where the service exists.

All the information about the services are stored in the style of C-prolog's fact.

For example, if you want to register the local UNIX command "date" and you want to use it with the name "udate", you may do it as below.

```
| ?- assert_int(udate, date).
yes
| ?- listing(service).
service(udate, name, udate).
service(udate, map, date).
service(udate, place, int).
yes
```

You can also register remote services like above.

4.4 Example

We will show you a little complicated example. To make it easy to understand what happens, we use only two machines, M680H and Vax8600. We can combine the two command in different machines.

At first we register a VOS3 command "lists", which will list the file name and necessary information about the file. We have to define "lists" command both in Vax8600 and M680H.

In M680H,

```
| ?- assert_int(lists(X), lists(X)).
yes
```

In Vax8600,

```
| ?- assert_ext(lists(X), lists(X), 1).
yes
```

1 means M680H in this example. In this case, output of the command "lists" goes to display. In order to save the output, we made "mfile" predicate. Using this predicate, we register a new service "listsf".

In Vax8600,

listsf(X, File) :- mfile(lists(X), File).

Then the output of the "lists" command is stored in File.
And next we register a UNIX command "grep", which will search a string and print all lines which contain the string.
In Vax8600,

| ?- assert_int(grep(String, File), grep(String, File)).
yes

Now we can use the combination of above two services. Example is shown in Figure 6. In this case, three layered view is displayed like Figure 7.

```
| ?- listsf('%',temp),grep('VDATA',temp).

PO        114        99     ARCHIV   A80595.LISPLIB.COMP.VDATA
PO         95        38     ARCHIV   A80595.LISPLIB.VDATA
PO         95        28     LD0005   A80595.PROLOG.VDATA
PO         95        71     LD0024   A80595.PROLOG.VDATA.OLD
PO        171       113     ARCHIV   A80595.SB.VDATA
PS         19        18     ARCHIV   A80595.SBLIB.VDATA
PO         19        10     ARCHIV   A80595.TEST.VDATA
PO         76         3     LD0023   A80595.UTIL.NEW.VDATA
```

Figure 6: Example of execution

In addition to above example, we have tried some examples such as editor, compiler and application programs. We have confirmed that this method goes well.

5 Discussion

In the experimental system, we have confirmed that the request and answer model goes well even in the environment that different machines are connected with poor communication facilities.

The system performance was not measured correctly, but response time didn't increase so excessively.

The experimental system was written in C-prolog and C. Program size in C-prolog is about 500 lines, and program size in C is about 2500 lines. This is very small in comparison with the function that can be provided by SBS.

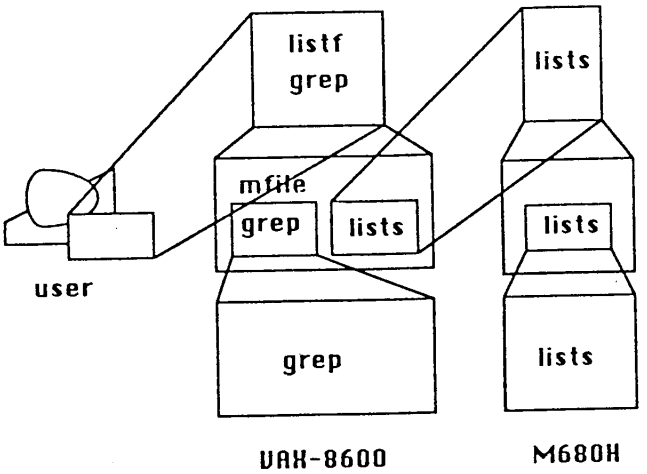

Figure 7: Example of three layered views

The service description power of the experimental system is too weak, because only service name and existing nodes are stored. We are now considering a new method to describe service description.

6 Conclusion

In this paper, we have proposed Service Base System. SBS makes it possible for users to use distributed services with no difficulties. We have explained the model of distributed services, the request and answer model and organization of SBS. An experimental system written in C-prolog and C was made, and we confirmed our request and answer model goes well.

About the service description, now we have been developing a new description method and constructing a new experimental system based on the description method. We will announce a paper about it in the near future.

Though there are some problems left to be solved, SBS is expected to be the basic key framework to realize the expandable distributed information processing network.