COMMUNICATION MONITOR OF SERVICE BASE SYSTEM

1U-7

P. FELLNER, T. FUKAZAWA, H. TANAKA

Department of Engineering, University of Tokyo

INTRODUCTION

A Service Base System (SBS) is a concept for an easy extensible computer network system, which offers to the user a convenient interface to access to all available services of the network [1].Each node of such network is regarded as a Service Base (SB), which holds some of the services of the whole system. When the user, connected to one of the nodes (fig. 1) requests a service, his front-end node will respond, if it can treat the service, or, if not, will give another request to the back-end. Since requests and answers are very frequent in an SBS, a powerful communication facility has to be associated, which handles all desired message exchanges. Beside that, error - and subcommand handling (especially for the startup procedures of the system) is necessary. All this is performed by the so called MONITOR, which is associated to each SB (fig. 2). Though in previous versions of SBS sequential monitor implementations had been used, in this paper we introduce a monitor with a parallel working communication facility, which means, that the monitor after writing a request to any of its interfaces doesn't wait for a respond on this interface, but continues to work on other problems. A parallel monitor is desirable, because the nodes in a distributed system also work parallel to each other.

REQUEST AND ANSWER FORMAT

In a SBS, requests and answers from one node to another occur very frequently. For the co-operation of the communication facility and the SB and to have the system easy extensible, a standardized format for requests and answers had been chosen: Thus, if a SB gives a request to another one, it will do it in the format

r(pn,rn,text)Prompt ,

where 'r' is an indicator, that the message is a request, 'pn' is the number of the SB, where the message should be sent to, 'rn' is a unique number (request number), given by the SB to be able to distinguish different requests, 'text' is the request itself and 'Prompt' indicates the end of the message. An answer, given by the SB, has the format

a(pn,rn,text)Prompt

where 'a' is the indicator for answer, 'pn' is the SB, to which the answer is given to, 'rn' is the request number of the corresponding request, to which this answer belongs to, and 'text' is the answer itself. If the answer is for the user, the SB writes the format

u(Ø,Ø,text)Prompt ,

where 'u' indicates, that this answer is for the user; a SB also accepts a request from the user in the format

u(Ø,Ø,text)

STRUCTURE OF THE PARALLEL MONITOR

Our implementation is based on UNIX operating system, which in our case is running on a VAX 73Ø and a VAX 78Ø. By using UNIX multiprocess facility the parallel processing environment is realized. A SB is implemented through a prolog process with a changed toplevel in order to accept the standardized request and answer formats. The monitor consists mainly of 2 parts:

A parallel working communication part performs all message exchanges of all interfaces of the node (to the user, to prolog or to other nodes of the network). To realize this there are 2 approaches: To check, if there is any message on an interface of the node, either interrupt or polling mechanism can be used. Since interrupt mechanism is extremely depending on the used UNIX version, polling was chosen with the disadvantage of overhead and inefficiency.

The second part handles subcommands and error conditions. Subcommands are necessary at startup procedures (to change the status of the system), to get control information and to redefine the system after an error occured. Beside that, a tracefile facility is available, to show the message interaction of the system. Important for the performance is a table associated with each monitor, in which the information about the status of the monitor and its logical location within the network is hold. The performance of the monitor is according to the contents of this table, so the monitor of each node is actually quite the same, only the content of each table is a different one.

Fig. 3 shows the structure of the parallel working monitor.

STRUCTURE OF THE COMMUNICATION PART

Fig. 4 shows the structure of the parallel communication part. The interface to the SB (prolog process) is accomplished by the UNIX pipe mechanism. Interfaces to the user or to other nodes are implemented by UNIX device drivers. To each interface a read buffer is associated, in which the contents of the interface is read. 2 major procedures handle the message exchange:

rpr :This procedure reads the contents of a pipe- or a device driver file descriptor to the associated buffer.

wpr:These procedures write the contents of a read buffer to any other file descriptor (to a prolog pipe or to any other device driver).

For the logical function of the SBS it is necessary, that the communication part changes the content of a messages in special cases, which is also performed by those 3 procedures: When the communication part of node x receives a request from SB x for SB y, (in the format r(y,rn,text)Prompt), it will write it to the interface of node y, and the message will be sent to node y. The communication part of node y will recognize the message as a request for SB y, coming from node x, and in order not to loose the information about the source of the request, it has to change the received format from

r(y,rn,request) to r(x,rn,request),

and will write it to its interface of SB y. When after a while the answer is given by SB y,it will do it in the format

a(x,rn,answer)Prompt.

The communication part will now write this message to the interface of node x, the communication part of node x will recognize the message as an answer for SB x, coming from node y, will change the format from

a(x,rn,answer) to a(y,rn,answer),

and will write it to the SB x . Because of the combination 'y' and 'rn' SB x will now recognize this message as the answer to its previous request with the same code 'y' and 'rn' .

CONCLUSION

Until now an SBS simulator had been implemented, it is a simulator in regard to the aspect, that the SB's are simulated on one machine by different prolog processes. The used parallel monitor in this model works satisfactory, but it appeared, that some further research on prolog's toplevel is necessary, since it cannot interpret the request number 'rn' in a satisfactory way. A monitor for a real distributed computer system is in the stage of development.

REFERENCES

[1] Fukazawa, Tanaka, Motooka, 'Service Base System', Distributed Processing System - Symposium, 1984.10

[2] Fukazawa, Tanaka, Motooka, 'Service Base System with Logic Programming', 29th National Conference of Information Processing Society,6H-6,1984.9

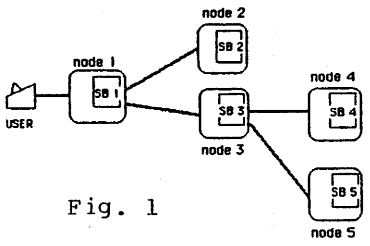

Fig. 1

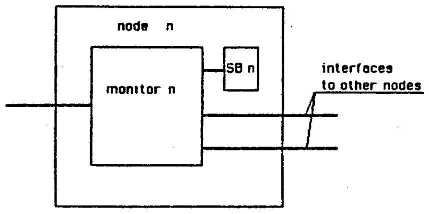

Fig. 2

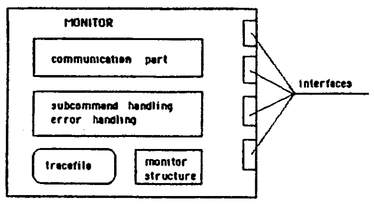

Fig. 3

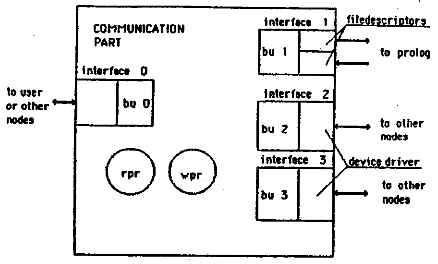

Fig. 4