A COMUNICATION PROTOCOL BETWEEN ASYNCHRONOUS PROCESSES FOR VS APL UNDER CMS

V.Casarosa - G.Faconti

Nota Interna C79-1

A COMMUNICATION PROTOCOL BETWEEN ASYNCHRONOUS PROCESSES for VS APL under CMS

V. Casarosa
IBM Scientific Center
Via Cardassi, 8
80100 - Bari, Italy
tel. 080-583346

G.Faconti
CNUCE - Institute of C.N.R.
Via S. Maria, 36
56100 - Pisa, Italy
tel. 050-45245 (X37)

CONTENTS

1. INTRODUCTION

There are many applications in which it is necessary to control different processes operating concurrently and accessing the same resources.

The shared-variable concept offers the means by which processes can communicate with each other and thereby can be made to cooperate.

In some APL systems, processes (auxiliary processors) are independent tasks whose asynchronous execution is scheduled by the host system. Under VM/370, the host system schedules independent virtual machines. Each virtual machine may contain a copy of VS APL which schedules the execution of its processors in a sequential way.

The VMCF Processor, APL106, is an auxiliary processor for VS APL under CMS which provides the user with the potential to apply new and different techniques to current applications by introducing the concept of variables shared between virtual machines.

APL106 provides APL virtual machines with the capability of send data to and receive data from any other APL virtual machine through direct storage-to-storage communication. It utilizes the services offered by the VMCF component of CP.

2. USE OF APL106

From the user point of view, the use of APL106 is quite similar to that of the auxiliary processor APL1?5 for VS APL under CMS.

Communication between the VS APL user and APL106 is made through a shared variable called the 'control' variable. If data is to be transferred in a VMCF operation, a second shared variable called the 'data' variable is also required.

An offer to share a variable with APL106 is made as follows:

106 ⎕SVO 'vars'

where:

106 is the processor identification number,

and

'vars' is the surrogate name in quotes of the variable(s) to be shared.

The control variable surrogate name must begin with the letters CTL; its total length is limited to eleven characters. The surrogate name of the data variable must begin with the letters DAT; it too is limited to eleven characters. Ignoring their first three characters, the surrogate names of the control and data variable pair must be identical; they uniquely identify a communication link.

The control variable should be offered for sharing to APL106 before the paired data variable. Offer of the data variable will be accepted by APL106, but any reference or specification of that variable will be ignored by the auxiliary processor until a control variable with the same suffix is successfully shared.

Both control and data variable can be offered together to the auxiliary processor by specifying each name as a row in a character matrix. Up to ten pairs of variables can be shared at the same time allowing communication over ten different links.

The response to a shared-variable offer is a degree of coupling indicated by an integer in the case of a single offer or by an integer vector with one element for each variable in the case of multiple offer; a response of 2 means that APL106 has matched the offer and the sharing is complete.

In order to establish a communication link with another virtual machine the control variable must be initialized, before or after the offer, with a character vector in the following way:

ctlname←'vmname protocol'

where:

'ctlname' is the surrogate name of the control variable,
'vmname' is the name of the partner virtual machine,
'protocol' is the communication protocol to be used between the two virtual machines.

The link is completed whenever the partner virtual machine offers a control variable indicating the now-offering virtual machine as partner. This information is made available to the VS APL user at the first reference of the control variable.

The data variable does not need to be initialized; any time it is specified, its content is transferred to the partner virtual machine if the communication link is active, or is simply ignored by APL106 if the communication link is not active.

A communication link is disconnected by retracting the shared variables related to this connection in the standard way:

⎕*SVR*'vars'

For the partner virtual machine the link now becomes inactive and is left outstanding.

2.1 Communication Protocols

Two communication protocols are available: *SEND* for asynchronous communication and *SENDREC* for synchronous communication.

The *SEND* protocol allows asynchronous communication between two virtual machines. When a communication link is established with this protocol, a specification of the data variable transfers the value of that variable from the VS APL user to the APL106 from where it will be transferred to the partner virtual machine. Once the data has been transferred to the APL106, control is returned to the user but no succeding specifications of the data variable are accepted until the partner has read the sent data.

The status of the transmission is indicated in the control variable; in particular, the user will be allowed to send new data (specify the data variable) when a reference of the control variable indicates a successful completion of the transmission.

A reference to the data variable makes available to the VS APL user the data sent by the partner. A new value of the data variable is obtained when a reference to the control variable indicates that new data have been successfully sent.

Sequences of references and specifications of the data variable are free from any constraint and can be done in any order by each other partner.

The *SENDREC* protocol allows synchronous communication between two users. The protocol is the same as for *SEND* except that in this case a specification of the data variable must be followed by a reference or vice versa. This means that the first user who specifies the data variable forces the partner to start with a reference.

When two users in communication specify different protocols the *SENDREC* option is forced.

2.2 General service requests

The status of the communication link and of the transmission can be controlled by specifying the control variable with one of the following keywords:

'*CANCEL*' cancels a message or a data transfer directed to the partner but not yet accepted.

'*REJECT*' cancels an incoming message or data transfer still pending.

'*QUIESCE*' temporarily sets the link as seen by the partner to inactive status.

'RESUME' resets the status set by *'QUIESCE'*.

A reference of the control variable after its specification will contain the return code for the requested service.

3. DESCRIPTION OF APL106

The transmission mechanism of APL106 involves the following three steps:

1- data are sent from the user program in the source virtual machine to its auxiliary processor,
2- data are transferred from the source to the sink virtual machine,
3- data are retrieved by the user program in the sink virtual machine through its auxiliary processor.

Because the communication is between two APL machines, no conversion is required and the data is transmitted in internal APL notation.

Steps 1 and 3 are executed by the VS APL user respectively specifying and referencing the data variable; step 2 is executed by the auxiliary processor by means of VMCF service requests to CP.

Following such an operating philosophy, APL106 can be described in two sections: the one which interfaces with VS APL through the shared-variable mechanism is called the VS APL Interface, the one which interfaces with CP though the virtual machine communication facility is called the VMCF Interface.

Both sections share common storage areas devoted to a control block for VMCF (VMCPARM), a Communication Table containing information about the existing links and the data being sent or received. Space for the VMCPARM and for the Communication Table is allocated in the processor work area during the sign-on procedure whereas the buffer space needed for the data transfer is obtained dynamically during execution from CMS free storage.

3.1 VS APL Interface

The VS APL Interface is designed so that the processor is responsive to demands regardless of the status of the VMCF Interface. This avoids possible deadlocks but makes useless the setting of an access control vector (which will be ignored).

The main function performed by the VS APL Interface is to pass to the VMCF Interface, when it is active, the service requests made to APL106 by the user and to notify the user of the results of the requested services.

The initial value of the control variable is used by the VS APL interface to ask the VMCF Interface to activate a communication link with the sink virtual machine. The communication link is completely established and active if the sink virtual machine has already extended a matching offer. In all other cases (i.e. sink not in CP directory, sink not VMCF-authorized, etc.) the link is not active and the offer is left outstanding.

Because of the design of the VS APL Interface, user requests are always accepted but performance of the requested services depends on the status of the communication link.

The VS APL Interface accesses the Communication Table to build the VMCPARM required by VMCF and specifies, at each user reference of the control variable, a five integer vector indicating:

1- the type of error encountered or successful acceptance of the request,
2- the reason for the error (if any) for the current operation,
3- the status of the communication link related to this control variable,
4- the status of the data to be sent,
5- the status of the data to be received.

3.2 VMCF Interface

The main function performed by the VMCF Interface is to execute the VMCF subfunctions as specified in the VMCPARM built by the VS APL Interface.

The VMCF Interface operates asynchronously with respect to the VS APL user; it depends on the external interrupt signals generated by CP according to VMCF transmission protocols.

The interface is activated for the first time when the VS APL user offers a control variable with an acceptable initial value. On activation, the AUTHORIZE subfunction is immediately executed in order to enable VMCF for the virtual machine.

In order to activate a communication link, the VMCF Interface executes the IDENTIFY subfunction to notify the sink virtual machine that the source is available for VMCF communication. The link will remain pending and no further requests made by the VS APL user will be executed until the sink virtual machine replies with the same identification message; at that time the link will be completed.

The VMCF Interface accesses the Communication Table to notify the VS APL Interface of the status of the communication link, the status of the data transfer and the return codes from VMCF. The Communication Table is used also to record the transmission protocol for a specific link and the address of the VMCPARM to be used during the current operation.

The deactivation of the VMCF Interface is made automatically by means of the UNAUTHORIZE subfunction when all links are disconnected by the VS APL user and hence all the entries of the Communication Table are cleared.

3.3 Communication Table

The Communication Table is used by the VS

APL and the VMCF Interfaces in order to transmit to each other the information necessary to control a communication link between two virtual machines and the related data transfers.

The Communication Table is allocated in the processor work area during the sign-on procedure of APL106 and consists of ten entries allowing a VS APL user to establish communication over ten different links. Each entry is initialized by the VS APL Interface when the VS APL user provides a valid initialization value for a control variable.

The information contained in one entry is as follows:

1- identification of the link by means of the name of the sink virtual machine,
2- identification of the pairs of variables associated with the link,
3- degree of coupling for the link,
4- type of transmission protocol,
5- address of the VMCPARM related to this link,
6- status of the data to be sent,
7- status of the data to be received,
8- return code for the current operation.

A request of service to VMCF is made by passing it a control block (VMCPARM) specifying the VMCF subfunction to be executed along with other information required by VMCF to execute that function. The VMCPARM is built by the VS APL interface utilizing the information contained in the Communication Table.

4. CONCLUSIONS

APL106 can be used to multitask virtual machines in the sense that each virtual machine becomes a parallel subtask of another virtual machine. The VMCF functions provide the serialization and communication facilities to control such an environment and the VM/370 functions provide efficient scheduling, dispatching and basic resource controls.

Because resource sharing can range anywhere from a single device to entire processes, the capabilities offered by APL106 cover a wide range of applications. In particular, APL106 is planned to be used in a multi-user data base management system where several user virtual machines communicate with a master controller in which the data base itself resides.

Many enhancements are possible for APL106 to improve its usability. In particular, three major extensions are currently being investigated or under development.

The first extension will integrate APL106 with network-control software implemented at CNUCE (RPCNET) so that the two partners of the communication can reside in different nodes of a communication network.

The second extension will allow a communication link to be established between processes instead of users, so that APL106 can be run concurrently with other auxiliary processors which also make use of the VMCF services on the same virtual machine.
All the requests for VMCF services will be made by the processors to a routine added to CMS nucleus which will act as a common interface toward VMCF.

The third extension will allow storage-to-storage communication between APL and FORTRAN and/or Assembler virtual machines. For this purpose a modified version of APL106 will be a stand-alone routine callable from those languages. It will provide the means of communicating between each other following one of the protocols as defined in paragraph 2.1 and the conversion routines for the translation of the data from its internal APL notation and the standard EBCDIC code.

5. REFERENCES

1- VS APL Program Logic, Program Number 5748-AP1, IBM Form No. LY20-8032-1
2- VS APL for CMS: Writing Auxiliary Processors, Program Number 5748-AP1, IBM

Form No. SH20-9068-1

3- VS APL for CMS: Terminal User's Guide, Program Number 5748-AP1, IBM Form No. SH20-9067-1

4- IBM Virtual Machine Facility/370: System Programmer's Guide, IBM Form No. GC20-1807-6

5- F.Caneschi, E.Ferro, L.Lenzini, M.Martelli, C.Menchi, M.Sommani, F.Tarini - 'Architecture of and the Service Facilities Provided by RCPNET-The Italian Computer Network for Education and Research Institution.' - Proceedings of the Fourth International Conference on Computer Communication, 26-29 September 1978, Kyoto, Japan

6- F.Antonacci, P.Dell'orco, V.Spadavecchia, A.Turtur - 'AQL: A Problem-solving Query Language for Relational Data Bases.' - IBM Journal of research and development - Vol.22, No.5, September 1978.

Usage Notes

1. NETREL is a CMS module generated in the transient area ([11]); as such it can be invoked from within a program, as many other CMS commands.

REFERENCES

[1] F. Caneschi, E. Ferro, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, F. Tarini, "Architecture of and the Service Facilities Provided by RPCNET - The Italian Computer Network for Education and Research Institutions", Proceedings of the ICCC 4th International Conference for Computer Communication, (September 1978).

[2] F. Caneschi, E. Ferro, L. Lazzeri, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, F. Tarini, G. Torrigiani, "RPCNET: Architektura i Servis", Avtomatika i Vycislitel'naja Technika, 6 (1978) 44-55.

[3] F. Caneschi, E. Ferro, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, F. Tarini, "The RPCNET Applications User Manual", Technical Report CSN-031, IIASA, (1978).

[4] F. Caneschi, "A Computer Network: Structure and Protocols of the RPCNET", Professional Paper PP.78-12, IIASA, (December 1978).

[5] A. Fusi, "RNAM - Macro Instructions for Application Programs", RPCNET Internal Document US012-00, (July 1975).

[6] P. Bertaina, M. Magini, C. Paoli, F. Tarini, "Spool To Spool Protocol: Initial Design", RPCNET Internal Document IS007-01, (October 1974).

[7] F. Caneschi, M. Sommani, "Remote File Access: a Data Communication Protocol for a Computer Network", Research Report, IIASA, (to be published).

[8] "IBM Virtual Machine Facility /370: Introduction", File No. S370-20, Order No. GC20-1800.

[9] "IBM Virtual Machine Facility /370: CMS User's Guide", File No. S370-39, Order No. GC20-1819.

[10] "IBM Virtual Machine Facility /370: CMS Command and Macro Reference", File No. S370-36, Order No. GC20-1818.

[11] "IBM Virtual Machine Facility /370: System Logic and Problem Determination Guide Volume 2", File No. S370-37, Order No. SY20-0887.

[12] "IBM Virtual Machine Facility /370: CP Command Reference for General Users", File No. S370-36, Order No. GC20-1820.

[13] P. Schicker, A. Duenki, W. Baechi, "Bulk Transfer Function (Proposal)", Forschungsproject COST-11, European Informatics Network, EIN/ZHR/75/20, (September 1975).

STAMPATO PRESSO IL
SERVIZIO TECNOGRAFICO
DEL CNUCE