"AN INDIRECT COMPUTER CONNECTION: THE UNIVAC ACCESS"

Fausto Caneschi

November 1978 WP-78-47

2361
Laxenburg
Austria

International Institute for Applied Systems Analysis

ABSTRACT

This paper deals with a method of gathering an access to a computer/operating system using standard tools (emulators and user programs), but with an indirect connection. The difference between the networking approach and this one is underlined, then the method is described.

CONTENTS

1. INTRODUCTION

The networking dream seems to be forced to remain a dream for the near future. The main reason of this is that, although there exists the technical possibility, more and more often there's no political willing. On the other hand, an access to remote computers and/or data bases is more and more requested by the users, and it has proved to be an economical tool too.

An economical tool for two main reasons:

a) a user accustomed with a computer/operating system often dislikes to learn to use another computer, perhaps for a limited period of time, simply to do the same kind of work he (she) did before. Moreover, the wasted time for the skilling of that person is a waste of money too.

b) it is certainly uneconomical to have the same programs and/or the same data bases in two or more different computers, if it is possible (and it is) to connect them together in some ways.

In these conditions one is forced to use unconventional tricks and unconventional hardware/software configurations, in order to obtain a sort of "computer communication".

It is worthwile noting at this point that such a "computer communication" will never be a computer network in its classical sense, because it cannot be considered as the general approach, and for this reason doesn't solve all the problems, but only a very restricted set of them.

Sometimes, however, one can obtain unexpected and useful results, at least from a user's point of view, especially for batch access. In fact, almost all the operating systems are used to receive job streams from "remote stations", execute them, and send the results back to their origin. A "star" configuration, with the mainframe in the middle, and the remote stations at the end of the "arms" (telecommunication lines) is quite usual for most of the computing centers.

The problem becomes to be more complicated if the connection is not a direct one, but, to say, the bits have to pass through some "switcher".

This is the case in the Laxenburg- Milano connection, which is considered in detail in this paper.

This connection is actually a two-pieces connection, because the physical line goes from Laxenburg to Pisa, and from Pisa to Milano.

The reason for this fact is purely economical: in fact, as the Laxenburg-Pisa connection is a leased line, the more we use it, the less we pay (in percentage). Moreover, another connection (leased line) already existed between Pisa and Milano. It seemed quite reasonable to use these two lines to have the possibility of submitting jobs to the CILEA computers in Milano.

There are two reasons for the usage of the Milano computers: the first one is that some scientists at IIASA are used to work with UNIVAC computers, therefore it is easier for them to continue their work here in the same way as in their home institutes. The second reason is that the prices for CPU time, disk memory, etc. at CILEA are incomparably less than at Vienna.

So, the problem seems to be quite defined: there are good reasons to try, there is the technical possibility too; the only thing we have to do is to solve it.

2. PROBLEM ANALYSIS

Let's look at the Pisa computer which acts as a "switcher" in this connection.

The VM/370 operating system running on the 370/168 at CNUCE, Pisa, handles the lines used for the remote job entry via a subsystem, called RSCS (Remote Spooling Communications Subsystem). RSCS, which runs on a dedicated virtual machine as an independent operating system, "speaks" with the system at the other end of the line using one of the standard IBM BSC protocols (3780, 2780, 3270, ...), so the remote station should appear to RSCS as it was an IBM device, that means that it should "emulate" some IBM device.

The IIASA remote station consists of a WANG minicomputer which emulates an IBM 3780 remote terminal, which, in turn, consists of a card reader, a line printer, and (optionally and not implemented in the WANG) a card puncher. With this configuration, one can send card decks (actually diskette files) to the mainframe, and receive the results back as printouts (Figure 1).

The CILEA (Milano) remote station configuration is quite similar, with some little differences:

a) the IBM emulated device is a 2780 (similar to 3780, but with less facilities);

b) the emulator is a program which runs on the UNIVAC computer. This is the main difference: as the UNIVAC computer is a mainframe like the IBM in Pisa, every computer looks at the other one as a remote station (non-intelligent terminal). This is the reason why this connection is not a computer network: every computer considers the other one as a device: there is no possibility of a program-to-program communication.

For these reasons, every job to be sent to CILEA has to be considered as a printout by the CNUCE computer (the emulated 2780 remote station has a card reader and a line printer), and a program is needed in Milano to convert from the printer format (133 characters for line) to the card format (80 characters per line). Moreover, we cannot send to Milano a card deck, because that remote station has no card puncher (neither physical, nor emulated), so that every punched file directed to the CILEA remote station will be purged by the system in Pisa. On the other hand, the remote station at

IIASA consists of a card reader (as an input device): every file we send through this station is considered as a card deck, and cannot be switched by the system (even if the possibility does exist) onto the CILEA line, which accepts only line printer files.

The problem is the same on the other side: when a file has arrived in Milano -- supposed it is converted in card-image format, and supposed the user program written on the card deck was executed -- a printout will be produced. This file cannot be sent back to the origin of the previous input file, because the emulated device is a card-reader (record length: 80 characters) and the file to be sent back is a 133 characters-per-line file. Moreover, supposed we have a program in Milano which converts the print-outs into card images, those virtual card decks will not be sent into the IIASA line, which accepts only printer formats: they will be purged by the system in Pisa as soon as they will arrive there.

For the above mentioned reasons, it is clear that the Pisa computer cannot act as a merely switching node; it has to perform some operations to convert the files into appropriate formats, that is, it has to "add value".

3. SOLUTION ANALYSIS

An optimal solution would be the "networking" solution, that is packet switching, use of layered protocols, complete text transparency and program-to-program communication. This solution is not so unrealistic as one could think: in fact, if we are really planning to make two computers "talking" together (that leads to infinite possibilities), we must choose it. On the other hand, some standards have already been developed, both by IIASA and other national and international institutes, so that the actual work would consist in choosing and implementing some of these standards.

The only fault in this solution is that it requires some formal agreements, and some allocation of resources before the project starts.

So, if we think to a way of getting "some results" in a very short time, we should consider a more narrow, and more pragmatical approach, which will always be "dirty" from a networking point of view, and will always have well defined constraints.

We considered two different approaches to solve the problem:

a) modification of the RSCS system, in order to enable it to accept and/or convert formats which don't match with the configuration of the lines.

b) use of standard VM facilities and writing of user programs to receive files, convert them, and send them to the appropriate destination.

The first approach has the following advantages:

a1) no computing time is requested for the conversions: in fact, the RSCS a1) no computing time is requested for the conversions: in fact, the RSCS computing time is charged to the line costs, which are fixed.

a2) the solution is "clean" from a system's point of view;

and the following disadvantages:

a3) a modification of an operating system is requested: this implies a period of tests which have to be very accurate (RSCS is a "public" service).

a4) the availability of the CNUCE systemists is not sure, both for the time (man-power) requested for such a modification, and for the real willingness of modifying a system which "works", on request of only one user.

The second approach has the following advantages:

b1) a user program is requested: that is, something which can be maintained and tested without involving any person in the "switching centre" (CNUCE), but simply by some persons at the opposite sides of the connection (IIASA and CILEA);

and the following disadvantages:

b2) some computing time is needed, to perform the translation operations on the files.

b3) the solution is not "clean" from a theoretical point of view: in fact, it is actually something like a trick, consisting in "parking" the input file in a temporary disk storage, modifying it according to the appropriate format, and then "printing" it to its destination.

As we can see, the two approaches are quite complimentary: one's dis advantages are other's advantages, and vice versa. For this reason, the chosen solution was that one with less disadvantages, which obviously led to the major advantages.

It has been chosen the solution b) because:

1) the computing time, needed for the format translations, is very short, and does not affect in a significant way the actual computing costs.

2) Although it is not "clean" from a theoretical point of view, this is a quick and efficient solution, and that is just what was needed.

4. SYSTEM CONFIGURATION

We will describe the solution in two separate parts: the first one for the Laxenburg-Milano direction, the second one for the opposite direction (Figure 2).

4.1. Laxenburg-Milano

Let's suppose that a file is stored in a WANG diskette, which consists of a program to be executed on the UNIVAC computer and all the UNIVAC control cards needed. One can write this file using the PDP/UNIX editor, and then send it into the WANG diskette, or edit the file directly in the WANG system.

It is necessary to add some control cards on the top of the file, and some control cards to the end of the file. Then, the file can be sent to a "public" virtual machine in Pisa (Appendix A).

That's all. The printouts, if any, will be printed on the WANG printer.

Let's describe now what happens to that file. The file (in card format) is read from the reader of a public virtual machine into a temporary disk space; then, some other CILEA control cards are added, and finally the file is "printed" onto the line to Milano. When the file arrives at Milano, it is re-converted into card format by a program running under UNIVAC operating system, and enqueued for execution.

4.2 Milano-Laxenburg

The output file in Milano (record length: 133 characters) is translated into card format, to be "read" by the emulated card reader connected with Pisa.

If the text does not exceed 80 characters, the record remains untouched; else, a continuation code (exadecimal 'FE') is written in the 80th position of the "card", and the rest of the record is written onto the next card. Some control cards are added to the top and the bottom of the file (Appendix B). This virtual card deck is sent to the same "public" virtual machine in Pisa as the previous card deck from Laxenburg.

The procedure is similar to the previous case, but a little bit more complicated. In fact, the file is read onto a temporary disk space, then it is examined record by record to check for the "continuation" character. A printer file is then built, and the printing is directed to the Laxenburg printer.

5. FINAL REMARKS

The kind of connection described above is a quick and efficient way to access in batch mode remote computers. With a little effort, this method could, within its limits, be extended and improved.

As it has been mentioned earlier, this solution will never give us the ability of overpassing its natural limitations, but it poses the basis for going further, in the sense that a "human communication" has been established between IIASA and CILEA, which could be used for longer term projects in the field of computer communication. But this is another story.

APPENDIX A

Sending a file from Laxenburg to Milano.

a1) the file should be stored in a WANG mini-diskette, with the following structure:

```
/JOB OPERATOR A052 SMISTA
/SET PASS WURSTCHE
LEGGI FILE UNIVAC
.....
.....      CARD DECK FOR THE UNIVAC
.....      COMPUTER
.....
/*
CP SP CONS STOP PURGE
CP LINK REELCA 191 192 RR NET
AC 192 B/A
SENDUNI FILE UNIVAC
/*
```

a2) This file has to be sent to BATMON virtual machine.

APPENDIX B

Sending a file from Milano to Laxenburg.

b1) The file, prepared by a program running under UNIVAC operating system, should have the following structure:

```
/JOB OPERATOR A052 SMISTA = REMLAX
/SET PASS WURSTCHE
LEGGI PRINTING REMLAX
.....
.....        } CARD DECK PRODUCED BY UNIVAC
.....        } WITH CONTINUATION CHARACTERS
.....        } (IF NECESSARY)
/*
CP SP CONS STOP PURGE
FI IN DISK PRINTING REMLAX
FI OUT PRINTER
CP LINK REELCA 191 192 RR NET
AC 192 B/A
SWITCH
/*
```

b2) this file has to be sent to BATMON virtual machine.

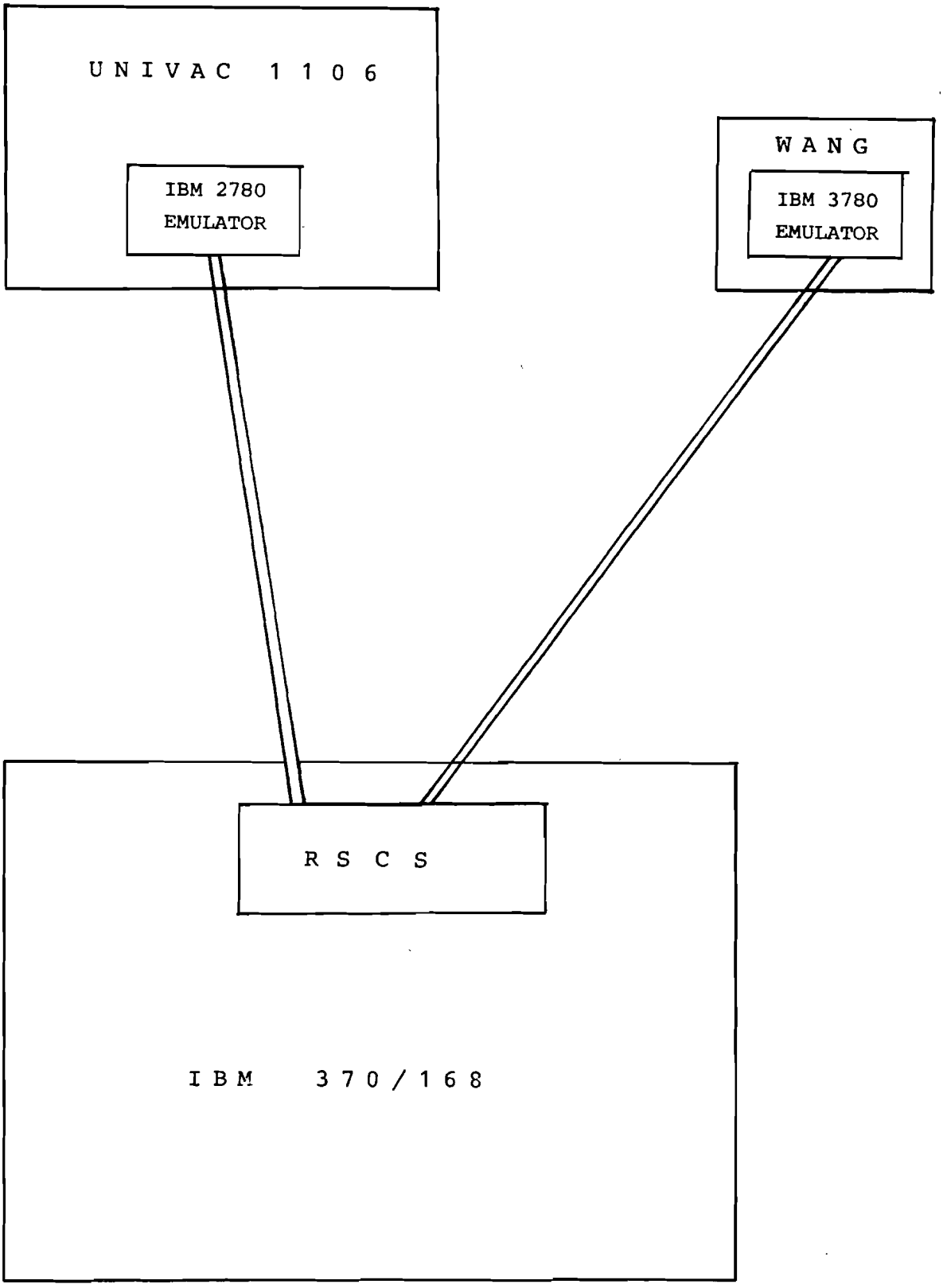

Figure 1: The Batch Connections.

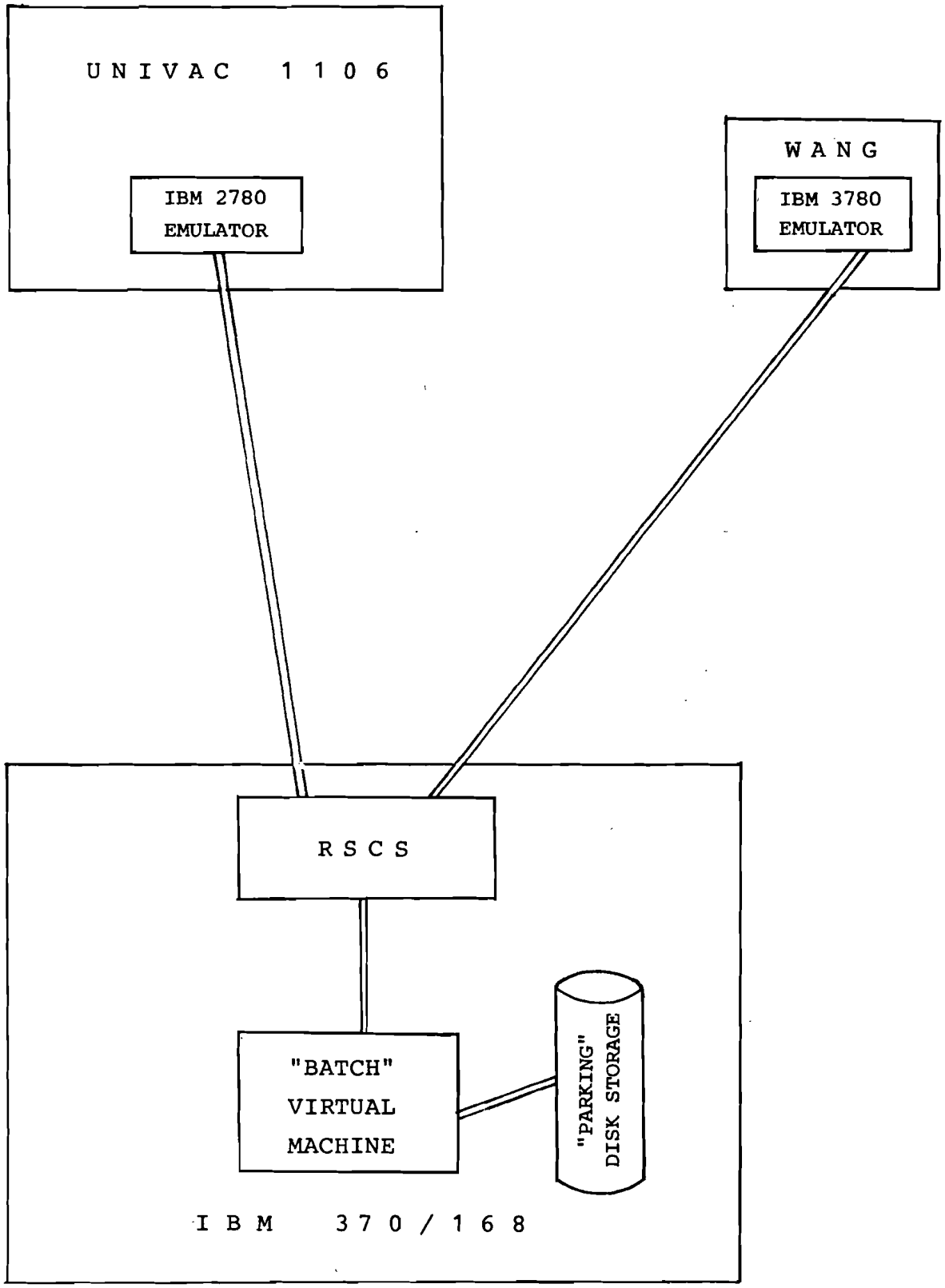

Figure 2: System Configuration.