International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in; editorijoer@gmail.com

Vol.7., Issue.5, 2019
Sept-Oct

RESEARCH ARTICLE

ISSN: 2321-7758

SYSTEM AND COMPUTER NETWORKS: FRONT-END FUNCTIONS IN A DISTRIBUTED ENVIRONMENT ON A NETWORK

ABDULLA MAHMOUD ALSHIBANI MOUSBAH[1], OUSAMA A SHAABAN[2]

[1]Faculty of Science, Bni_Waleed University, Bani_Waleed, Libya
Alshibani55555@hotmail.com
[2]The higher institute of Engineering Techonology, Tripoli
Email: Eng.osama9@gmail.com

https://doi.org/10.33329/ijoer.75.12

ABDULLA MAHMOUD ALSHIBANI MOUSBAH

OUSAMA A SHAABAN

ABSTRACT

In this work, we proposed to deal with some aspects of computer science seen from a functional angle. It remains for us to examine the possible extensions of our work. We will proceed to the inverse of the one used in this paper. We will give the state and the possible consequences of our work, then we will broaden our vision to address more general aspects concerning some ideas developed in this paper. A number of The External Order Language (EOL) commands have been implemented, others may meet the requirements: the structure of the EOL interpreter is indeed adapted to a modular extension of language commands. An important aspect of the later developments lies in the co-operation of validating the ideas expressed regarding of applications. This is the subject of current vaults whose illustration is to set up a system for cooperating file management systems, and this study is one of the most important extensions of distributed computing, namely management of distributed databases.

Introduction

The study of networks has so far been addressed by the telecommunication aspect by pushing the system and external points of view in the longer term. The external aspect concerns the user who is confronted with a multitude of different operating systems. However, very few users can claim to know perfectly two or more systems; what was thought to be an advantage of the general networks thus becomes an inconvenience. With their growing development and especially entering the era of telematics, it seems that the moment has come to seek to facilitate the use of the computer tool in the context of distributed computing. As for the system aspect, we notes, in spite of the operational character of the networks and the experience already acquired in this field, that a significant amount of system programming must be invested whenever it is desired[1, 2]:

- connect a new computer to a network,

- Must allow wider use of remote computer resources.

The objectives set out lead to a number of guiding principles in the design of the proposed architecture. The first of these principles consists of a functional approach that aims to distinguish entities characterized by their function (s) and then determine the interactions between them. It is important the autonomy of the entities, some visautonomy means the possibility of the deposition of their activities, these synchronizing, through interactions. This notion is to underline others. This

in parallel, it is necessary, all the more important as these entities are defined in a separate environment and no prior hypothesis must be made about their location[3]. This approach leads us to distinguish and separate very clearly the two aspects:

- Data processing to produce a planned action in the description of the functions.
- Interaction or communication to allow the routing information to other entities.

The application of this principle aims at the modularity of the proposed architecture thus allowing for expeditions that do not constantly question the overall package

The Initial Environment of the Study

The initial environment is that of a centre of calculation with a certain number of isolated mini-calculators, each of which is specialized in the processing of a small number of functions. The fact of assigning to one computer a small number of functions has both economic and technical justifications[4]. Indeed, the very rapid progress of the technology of the integration brings on the market of the equipment more and more powerful at prices less and less high. This phenomenon involves a questioning of the computer tool. Replace a large computer , able to do everything, by several specialized minicomputers become financially interesting, on the one hand because the sum of the costs of minicomputers does not generally reach the price of a large computer, on the other hand because the investment can be progressive and be made as and when measurement of needs. From a technical point of view, the specialization aspect is attractive insofar as writing specific software is easier to develop and maintain than specialized software. In the context of a computing centre, the specialization of calculators' results in their heterogeneity since each of them is adapted to the functions it processes. A calculator designed for real-time applications is indeed very different from a calculator specialized in numerical computation.

What is a network system?

The vision of the computer centre given to the user by the communication network presented in the previous chapter is that of a collection of computers or systems offering various services and resources accessible from unmarked terminals. However, the user must know the command languages of all the systems he wants to access as well as the location of the resources on the different computers. Indeed the logical connection procedures for the various applications and their mode of use are dependent on the host systems.

The need for a network system[5] is felt to not only support the transmission and homogenization aspects of hardware but also play a more active role as intermediaries between users and applications. This role is subordinated to the implementation of a language allowing "entire network system".

In this case, the user enters the network by connecting to the network system rather than to a particular system on the network. It has access to the services of all systems without having to carry out a "LOGIN" of each of them.

A network control language is a unifying instrument that, from the user's point of view, replaces the control languages of accessible systems. The functions that it expresses are very dependent on the chosen option as to the vision that one wishes to offer the network to the users. Two choices are possible:

- We want to leave the user with the idea of the existence of a network; the command language is one in which the user can talk to the network. In this case it is much more than a command language common to all computers because it expresses additional network queries functions.

Examples of network requests:

- -file transfer from one site to another
- -request to perform work on a remote site

Knowledge of the network is expressed by identifying the sites involved that appear in a command as parameters. This knowledge can be restricted by making these parameters implicit in some cases, or by giving the "network system" the responsibility of their choice.

We want to ensure the complete transparency of the network for the user; the command language seems

very close to a "local" command language. In this case, it is the system that fully supports the network aspect:

- -localization of the elements put into play by a command,
- -determination of the sites involved,
- -decision of transfers to be made, etc.,

Whatever choice is made, the network system manages a network environment; depending on the case, the interpretation of the command language involves more or less complex actions.

SOME EXAMPLES OF NETWORK SYSTEMS

Multi-Mini-computers[6]

The aim of this project is to create a multi-mini-computer system that is transparent for the user and in which the various functions of the software are entrusted to specialized mini-computers. We thus distinguish:

- -the "minis performants",
- -the "mini master" to which all I / O devices are connected and which supervises the operation overall system,
- the "mini file".

All minis are connected to each other by a single bus.

The minis can be of various origins and on the other hand a user work (described in a scenario language) must be able to be executed on any mini executor. To allow this possibility, any programme user, written in advanced language is the subject of a compilation for which the object language is a language common LOC (Common Object Language). There is, on each mini, a transpose a translator of LOC language machine.

The SOC Project (Connected Computer Systems)

This project[7] focuses more particularly on software-related problems and the use of the network by traditional computer users. Its ambition is nevertheless limited by the homogeneous aspect of the network: connected computers are identical from the point of view hardware (IBM 360 or 370) as software (OS 360).

The network is managed by the Network System a set of distributed programs running on each of the network computers, separate from the local "operating system" and how to get on top of all the principles and principles functions are the same. We will not deal here with the integration aspect of the constituents of the Network System an "operating system"; rather, let us clear the functions of the Network System: we-communication between the computers of the network, - implementation of a command language called external language (LE) for the users. A set of commands expressed in LE constitutes a Network Work whose execution implies the compilation of LE into a more basic language called internal language (LI). The LI is also used for dialogues between different computers. The compiler generates as many blocks of LI as there are computers involved in the network work.

The characteristics of LI are as follows:

• Each LI command is executed on a single machine.

- -if exchanges are necessary between two computers to execute the commands of a network work, the compiler generates orders that are executed on each of the two machines.
- -the execution of each block of LI is provided on the machine concerned by an interpreter.

The possibilities offered to the user are:

- -the transfer of data from one site to another,
- The execution of remote works.

When preparing a network job, the user is required to specify the computers involved; there is a lack of network transference.

The project IGOR (Network-oriented General Interpreter)

This study is part of the development of a network system for the Cyclades network. As for SOC, two languages have been developed[8]:

- an external language which allows the description of the resources and the

actions desired by the user which ones the site (s).

- an intermediate language produced by the LE compilation. On each computer in the network, there is a LI interpreter that executes the LI code.

UNIX

In this example[9], the language of the queries is an extension of a JCL from a local machine by addition of network commands; this solution remains linked to choice of machine and its change involves the change of the command language.

APPROACH TO THE DESIGN OF THE JRC NETWORK SYSTEM

From the user's point of view the term "Distributed Calculation Center" is inappropriate in that it is desired that the computer center should appear as a single calculator with its command language and resources; the network aspect is completely ignored by the user.

On the other hand, from a system point of view, it evokes a number of concepts necessary to enable this transparency.

In order to finally lead to the highlighting of functions within the JRC network system, our approach will be in two convergent directions:

1. Take into account the needs and desires of the users and define functions to solve them, the approach is then downward.
2. Take advantage of the experience gained in the field of telecommunications and try to expand the concepts already developed to reach and improve the usability of a network system; in this case, the approach is excessive.

This phase of work will make it possible to specify the initial objectives.

Downward approach

It starts with the aspirations of the users and the way they want the computer center to be perceived by them.

The users:

They want to access ... applications using possibly files according to a procedure that is the simplest possible. For this purpose, we will put to their available a command language that substitutes, in their eyes, all the languages that existed until then in the computer center, mainly in their functions of management of resources (files, devices) and call processor (launching an application). In addition, this language will completely mask the presence of a network.

The application will be the only notion accessible from outside the system; the role of the network system is to deliver the user specific constraints to the hosts constituting the system.

The user-application exchanges are carried out through this interlocutor and under his control. This very superficial description is a first element to apprehend the system; it introduces five roles; each role is held by an entity independent of the others, the geographical location of each being any (distributed, located on a site) and their operation being asynchronous. These roles are:

1. User
2. Adapter between the supporting operating system user access and the rest of the system
3. Interpreter of the command language
4. application
5. adapter between the operating system supporting the application and the rest of the system; in fact, the applications located on the different interconnected hosts are accessible via the operating system of these hosts and the associated control language, one of the hypotheses of our study being to keep the software existing specialty on each host.

Therefore no need to introduce conventions of Compendium of Communication. It is at this level that it is experience and lessons learned from research into network matter.

Schematically, the defined entities will have the relations indicated by the lines below.

International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in; editorijoer@gmail.com
Vol.7., Issue.5, 2019
Sept-Oct

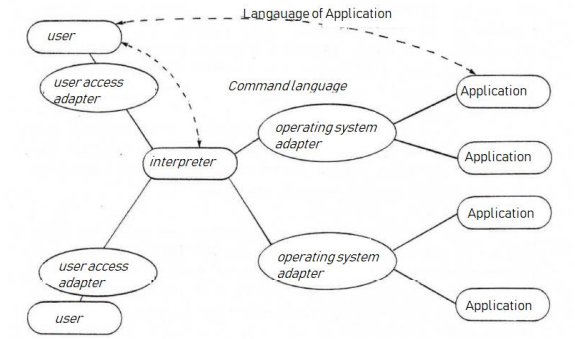

Figure 1

Upward approach

This approach is a network approach; it is about managing a distributed environment. The peculiarity of a distributed system is that within this system there is no entity which is aware of the overall state of the system: at other times, we do not distinguish between "master" and a distributed system. In particular, two entities that wish to communicate cannot rely on a "master" who controls their exchanges. As intermediaries in their reports, these entities, which also have complete autonomy, must define rules of mutual behaviour, which define a protocol. In this context, information is not only data intended for applications, but also a "label" that defines its source, destination, etc.

The communication between the entities that have to exchange information uses the mechanisms developed for computer networks; all interactions will be based on messages. As for the communication conventions they will express themselves in terms of language and protocol. On the other hand, it is important to standardize the methods of communication; even if some entities are local to each other, they will communicate according to the same rules as the remote entities.

It is difficult to discern a boundary between language and protocol. A protocol, which is already a language in itself, is a support for a higher level language, ensuring, under the best conditions, the transport of this language and the synchronization of the interlocutors who use it.

The content of the messages conveyed can be schematized as follows:

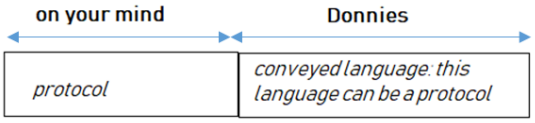

Figure 2

In the following, we will reserve the terms:

Language for dialogues between terminal entities: user, application, interpreter, and protocol for exchange conventions used to transport these languages.

THE EXTERNAL ORDER LANGUAGE

Users have access to applications through the network system through language that enables them to formulate queries. This command language represents the vision that users have of the CCR; for this reason it is called external control language (ECL)

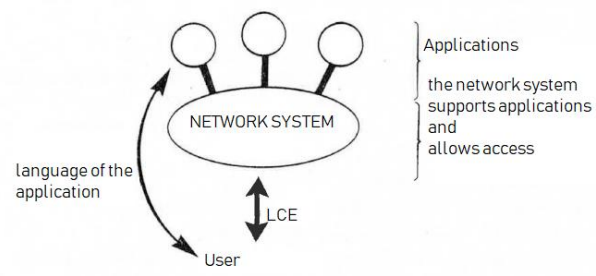

Figure 3

The EOL therefore acts as a dialogue language between the users and the network system. Its purpose is to hide from the users the variety and esotericism of the control languages existing on the network computers and to offer them a more pleasant vision of the computing center thanks to a gain of simplicity and homogeneity.

To facilitate the understanding of the system and allow the user to easily use it, it is important to give it a simple and easy to understand structure. It is not a question of realizing a super-command language convertible in the proper languages to each computer but rather essential elements to allow the user to access the application of his choice.

This language must be as close as possible to a 1 year old 1 for 1 learning and learning curve.

On the other hand, as it is a dialogue tool, it must be both interactive and interactive[10]

Its main objective being to allow the access to applications, it must offer the requests of:

- connection to an application
- disconnection
- It must also allow to express simply the most frequent sequences (compilation followed by an execution)

• In addition, it is necessary to contain utilities such as:

- time request,
- request for information about a user,
- sending messages between users,
- -manufacture and use of catalogued procedures,
- use of resources (files, devices), etc.,

Interpretation of the EOL

The interpreter of the EOL is the real interlocutor of the users. All orders from users come to him because he is the only one to know the actions that they induce.

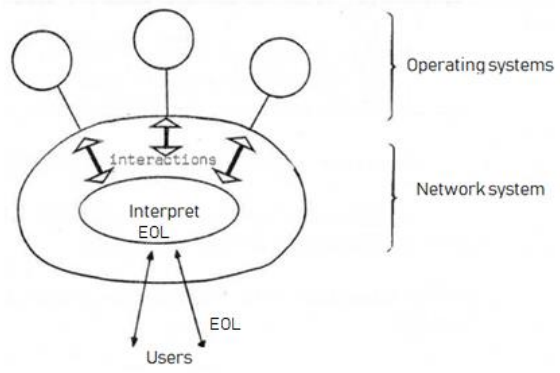

Figure 4

It plays a centralizing role vis-à-vis the users. He holds the list of all potential users of the computer center. To each of them is associated. An environment that personalizes it with respect to the performer; any command is interpreted according to the user who emitted it, that is to say according to its environment. In other words, the syntax of the ICL is general but its semantics are personalized for each user, who thus has EOL of its own and who defines his vision of the computing center (the applications to which he has access, his prerogatives, etc.). User environments are therefore essential data for the interpreter.

It is also necessary for the interpreter to know a minimum of information concerning the applications; an important aspect of its role is indeed to interpret access requests of applications. At its level, are therefore identified at any given time all applications available in the center.

These two types of information are very good.

Mediator played by the interpreter in the initialization of the user dialogs with applications.

After sketching the context of · the interpreter's work we functions.

We will not dwell on the syntactical analysis that does not occur in the first place, but rather in the syntax of the text. Language must be fully known to the interpreter.

On the other hand, at the level of the semantic analysis one can choose very different orientations:

1. The interpreter controls all the semantics of the language.

It therefore has all the useful information relating to the entities (users or applications) implicated by the command. For example, consider a service connection command with a file name parameter. With the chosen option, the file name must be controlled by the interpreter before the execution of the command, namely the connection itself. This implies that the interpreter knows, at least in part, the file management used by the application. This solution seems disproportionate and not very extensible because the introduction of a new application causes deep modifications of the interpreter.

2. The interpreter 'partly controls only the semantics of the language.

Some parameters, although part of the EOL, are transparent to him, their semantic analysis being carried over to other parts of the system. The idea is to locate the interpretation of these parameters. To the applications themselves, since it is they who use them and who, by any means better than anyone else, know their meaning. It is then necessary to

introduce additional elements into the system, these elements being loaded with further interpretation taking into account both the applications and the operating systems supporting these applications; these elements constitute envelopes for applications; these envelopes serve as an interface between the network system, the local operating system, and the applications, hence their term "network envelopes"

We chose the second solution because it limits the intrusion of the interpreter into what can be considered as the "private domain" of the applications, and it is easily extensible. In addition, it allows some decentralization of control at the network system level.

The functional diagram is precise: it is represented in Figure 5

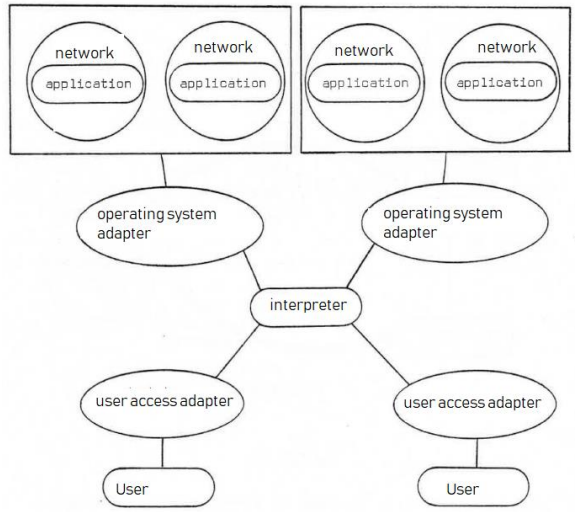

Figure 5

The last part of the interpretation is the actual execution of the order; two cases may occur:

1. Execution does not require the collaboration of others entities.

An example is given by the HELP command which allows the user to obtain information such as: available applications, reminders on the syntax of the EOL, etc. All this information is contained in the environment of the user; the interpreter who is available from this environment does not require the assistance of any other entity to respond to the user's request.

2. Execution requires interactions between certain entities. The typical case is the request for connection to an application.

INTERNAL CONTROL LANGUAGES

Application Cooperation

The existence of internal command languages is related to the implementation of distributed applications. A distributed application consists of a set of components (modules or software elements) interacting through communication means for the purpose of cooing, each component being associated with an execution site. In our study, the term "application co-operation" seems to be better chosen because it suggests that it is "already existing and autonomous applications that interact while for a distributed application each component does not necessarily offer a service that is available without it. The competition of other components.

Application cooperation is another step in the design of the JRC. It proves essential, in certain cases, to assure to the user the perfect transparency of location of the resources. The most anticipated need is at the file management level. If one does not want to limit the use of a file to its creation site, it becomes necessary to introduce a cooperation between different file management systems to achieve global file management. This implies interactions between the different file management systems, and this leads to the existence of communication conventions that is to say of a dialogue language. The debate, already initiated on the subject of the EOL, reappears it as:

-that each file management system knows the JCL of the other systems, to be able to express its exchanges with them in their own language immediately understood by them. Each system conforms to the interface that others impose.

The second solution is more elegant because it is general and allows the addition or modification of one of the file management systems without requiring modification of each other. On the other hand, it goes in the direction of a homogenization at the level of the languages conveyed by the network. Whenever applications want to cooperate, they use the intermediate language defined between them. This campaign may be unique and general at the level

of the CCR or particular and therefore adapted to each case of cooperation. It is formalized and can be as complex as one wants since it is invisible to the user. It is used for dialogue between internal entities, hence its name as an internal command language (ICL).

THE PROTOCOL "DISTRIBUTED CALCULATION CENTER"

To govern the dialogues between the different interlocutors of the JRC, we saw that it was necessary to introduce an exchange protocol; this protocol called "protocol distributed computing centre "allows both the transfer information and synchronization between all entities. Any entity that wants to communicate with others inside the JRC must respect this protocol.

ICL Interpretation

This interpretation translates the internal command language into the various operating system command languages that support the cooperating applications. Conversely, it is also about translating the local JCL into the common language.

Functionality of the CCR protocol

The services offered by the CCR protocol cover many aspects:

- Addressing mechanism of entities that interact
- establishment and rupture of dialogue
- data transfer on established routes
- integrity check
- security check
- administrative aspect

Addressing mechanism of interacting entities

The entities that interact are, we recall: users (with the interpreter or with the applications)

- the interpreter of the EOL
- network applications

The CCR protocol provides a way of naming these entities: this means makes it possible to specify the interlocutors involved in the different dialogs.

LOGICAL ARCHITECTURE

Architecture by levels: This technique consists of a hierarchical assembly of levels as shown in Figure 6.

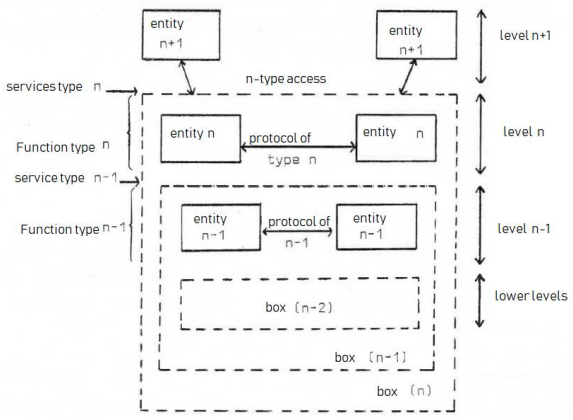

Figure 6: Hierarchical Assembly

We can give the following formal definition:

- The level n of the structure can use the services of type n-1 offered by level n-1 through an access type n-1.
- The structure of the lower levels is not known at level n which only considers the services rendered by the type n-1 box.
- level n is composed of n-type entities that cooperate under an n-type protocol
- The n-type entities perform n-type functions using the n-1 type services to provide n-type services at the n + 1 level.
- The architecture specifications of a level refer to the set of services offered by the lower levels, through the access functions. This set of access functions that defines an interface can be seen as the way of describing the logical structure of a network.

This definition does not imply a one-to-one correspondence between entities of type net entities of type n-1. An n-1 type entity can very well be used by several n-type entities.

The model of Figure 6 is sufficient to represent simple configurations such as two systems connected by an

autographed line. In the case of more complex configurations, it is possible to combine a chaining of boxes as shown in the following figure 6.1

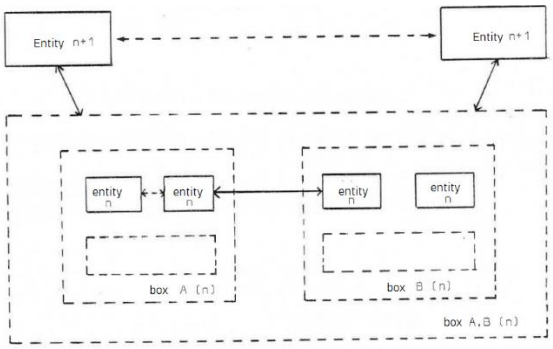

Figure 6.1

The basis[11] for structuring a tiered system responds to a desire to partition the functions that must ensure:

- Sequentially in time from level to level.
- The independence of the activities of the different levels: a change within a level must affect only this level.
- The transparency of the implementation of each level for its users.
- Sharing of common services by different users.

REMINDER OF THE ROLE OF THE FRONTAL FUNCTION

We will proceed to a brief reminder of the various functional elements highlighted in the previous sections

- Transport service implementing logical channels of communication.
- Virtual terminal manager.
- CCR protocol manager with a particular role
- For the corresponding entities.
- Interpreter of the EOL who ensures the referral between
- Network envelope that adapts the interfaces "local system operating system application CCR" and in which the ICL interpretation is included.
- The actual application the implementation of a logical architecture is intended to provide a better understanding of the functional elements and then to specify the interactions between these levels and also the relationships between elements of the same level.

We consider three levels that appeared naturally during the definition of functions.

The first is the implementation of the transport service; we will not go into the details of its structure but we will insist on its external aspect, i.e., on the aspect tool that it offers. The second level is constituted by the establishment of a dialogue structure using the transport service, which corresponds to the management of the CCR protocol. We will assign at this level the trade dynamics function.

Finally, the higher level realizes the actual dialogue, that is to say a way of using the different languages to get to the top of the page.

THE LOGIC WAYS OF COMMUNICATION

These are logical paths connecting either virtual devices to the correspondent or applications (ausens network application) to the correspondent. In fact, the nature of the entity (apparatus or application) at the end of a logical communication channel (VLC) does not matter. Through these VLCs, the correspondent has a global vision of user access and applications.

These links are implemented by the transport service taking into account the location of the two entities to be connected.

A logical channel of communication breaks down into one or multiple sets, each comprising TOSAN: CCR-2:

1. Transmission medium: line, bus, modem-line modem set.

2. Transmission management, ie physical management of the medium and management of the transmission procedure.

3 · The management of communication protocols

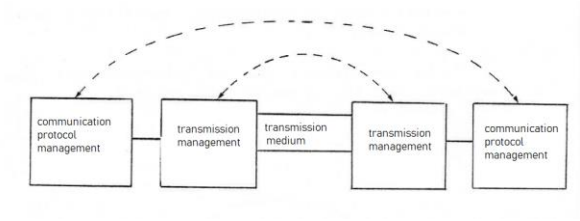

Figure 7: General outline of a VLC

It is through the VLCs that communications between the CCR entity can be performed. The s i n f o rm a tions they allow to exchange them are transparent. Their role is to ensure that this information arrives at its destination with a minimum error rate and to ensure a flow control of the exchanged data.

VLCs are objects on which a number of operations are possible. These operations can be of two types:

- use
- maintenance

DYNAMIC EXCHANGES

The media of the exchanges, namely the logical channels of communication, having been defined, it remains to specify the dynamics of these exchanges. This is based on the CCR protocol, which serves as a sort of vehicle for the information that is exchanged between the entities that dialogue. The elements concerned by this aspect are those who have to support and manage the protocol. It's about:

1. The virtual terminal emulator, which handles the exchanges between the virtual terminal and the system,
2. The top level of the access method of each connected computer that will handle the protocol for counting all the entities located on that host,
3. Correspondent who plays the role of intermediary between these entities.

These eiliments will ensure a central dialogue between all entitis. This central is supervised by the correspondent who, for each user, takes into account the two types of dialogue:

- user-interpreter,
- user - application.

NATURE OF DIALOGUES

The dialogues we are talking about here are the end-to-end dialogues that are in between. There are three types of dialogues within the CCR:

- user-interpreter dialogue,
- user-application dialog,
- application - application dialog

Each of these dialogues is realized in a particular language, respectively:

- EOL,
- Application language; it will be, for example, APL, the language of a text editor, etc.,
- ICL.

The functions of the network envelope depend on the choice made as to the dialogue language used between applications that cooperate:

1. Each network envelope that wants to transmit data to an application must comply with the interface that it imposes. This implies that the envelope could be the JC L of all the applications with which it wishes to cooperate.
2. Network applications that want to cooperate define a common language that they use in their interactions. The envelope is only responsible for matching the local JCL and the defined common language[12].

Conclusion

The architecture, there is no doubt that the design of future operating systems will be influenced by network requirements. In this work, we have these requirements to interface with a few of the existing operating systems with a network, taking into account other aspects of the transmission aspect. Increasingly, physical systems architectures are replacing functional structures with heterogeneous computing equipment. However, a single concept has to be implemented to make these structures transparent to users. The languages, at the same time as: x: developments regarding the architecture of the systems also take place in distributed, domain and evolution must languages. It identifies both the command languages and the programming languages that are the means users perceive these systems. One

of the goals we have pursued throughout our work has been to simplify access to distributed computer systems with services that have logic and physical equipment. As for the programming languages, they must be adapted to the distributed structure set up and have considerable opportunities for parallelism and transferability.

REFERENCES

[1] METCALF R.M., BOGGS D.R. "Ethernet: distributed packet switching for local computer networks" CACN vol. 19 No 7., 1976

[2] METCALF R M., "Strategies for operating systems in computer networks" Froc. ACM Nat. Conf. 1972 pp 278-281

[3] M. Di Santo, et al.,, in Advances in Parallel Computing, 1998

[4] FARBER D.J. "An overview· of distributed processing aims" Froc. 8th IEEE Computer Society International Conference 1974

[5] ELOVITZ H.S., HEITMEYER C.L. "What is a computer network?" Compcori Fall pp 5-3 5-10, 1977

[6] HEBENSTREIT J. "Project M2: design and realization of a system Multi-Mini-Computers" Congress AFCET, Gyf-sur-Yvette, 1976

[7] SOC Project , Report1 , Project department, IBM, 1971

[8] SERGEANT G., FARZA M.N. "Interpretive machine for the implementation of a command language on the Cyclades network" Thesis of Doctor Engineer, Toulouse Paul-Sabatier (Oct. 74)

[9] MANNING E., HOWARD R., O'DONNELL C., PAMMETT K., CHANGE. "A UNIX-based local processor and network? Ccess machine" Computer Networks, Vol. 1, No · 2, pp 139-142, 1976

[10] RAYMOND J. "NJCL a command language for a computer network" CRI contract, Grenoble I (July 73)

[11] PARNAS D.L., SIEWIOREK D.P. "Use of the concept of transparency in the design of hierarchically structured systems" Communications of the ACM 18.7 pp 401-408, 1975

[12] Larry L. Peterson, Bruce S. Davie, in Computer Networks (Fifth Edition), 2012