DÉPARTEMENT
D'INFORMATIQUE

FACULTÉ DES SCIENCES
ET DE GÉNIE

UNIVERSITÉ LAVAL

2.

A STUDY OF NEW SERVICES ON INTEGRATED COMPUTER-TELECOMMUNICATIONS NETWORKS : (final report)

1.
Marion R. Finley, Jr.
Thien Vo-Dai

Rapport de recherche RR-8202

2.

A STUDY OF NEW SERVICES ON INTEGRATED COMPUTER-TELECOMMUNICATIONS NETWORKS (final report)

Marion R. Finley, Jr.
Thien Vo-Dai

Rapport de recherche RR-8202
June 1982

FINAL REPORT

CONTRACT NO.: 20SU-36100-0-9511

MSS SERIAL NO.: OSU80-00171

A STUDY OF NEW SERVICES ON INTEGRATED COMPUTER-TELECOMMUNICATIONS NETWORKS

by

Marion R. Finley, Jr.

Thien Vo-Dai

Departement d'informatique

UNIVERSITE LAVAL

Cité universitaire, Qué.

CANADA G1K 7P4

TABLE OF CONTENTS

LISTS OF FIGURES

CHAPTER 3

CHAPTER 4

CHAPTER 5

* LAN---Local Area Network

LIST OF LISTS AND TABLES

ABSTRACT

The purpose of the study reported herein was to determine the impact of new telecommunications services on the nation's telecommunitions networks. To make this study tractable, attention was focused on videotex as a new service prototype and upon the switched local telephone network as the most crucial access means for new services. Both analytical and simulation methods were developped and applied to a simplified local network derived from data furnished by Bell Canada. In the case of this simplified network, it was found that relatively modest increases in videotex traffic produce serious degradation in network performance, which could be partially offset by increasing the capacity of the class-5 to class-4 office trunks. Some considerations were given to future developments in the local subscriber loop networks in which far greater capacity would exist, including integrated services digital networks (ISDN's) and local area networks.

RESUME

Le but de l'étude dont les résultats sont présentés dans le présent document était de déterminer l'impact de nouveaux services de télécommunication sur les réseaux nationaux de télécommunication. Afin de rendre cette étude faisable, on a choisi le vidéotex comme service prototype et le réseau commuté téléphonique comme le moyen d'accès le plus sensible aux effets des nouveaux services. On a développé une méthode analytique et une méthode de simulation qu'on a appliquées à un modèle d'un réseau téléphonique simplifié, dérivé de données fournies par Bell Canada. Dans le cas de ce réseau simplifié, on a trouvé que même une augmentation modeste du trafic de vidéotex descendrait sensiblement le niveau de performance du réseau. On pourrait contrecarrer partiellement cet effet en augmentant la capacité des tronçons reliant les bureaux de classe-5 aux bureaux de classe-4. On a également considéré les développements éventuels dans le futur des réseaux locaux des abonnés qui auraient une capacité de transmission accrue. Comme exemple, on a traité des réseaux numériques de services intégrés et des réseaux locaux d'informatique répartie.

ACKNOWLEDGMENTS

The authors would like to acknowledge the collaboration encouragement, and help received from their colleagues and contacts throughout the Canadian telecommunications community: Howard "L". Macumber, Government Telecommunications Agency, Department of Communications; Charles Terreault, vice-président adjoint, Bell Canada; John Coyne of Coyne and Associates; Michel Dufresne of Télécâble-Videotron, Montréal, amoung many others. Finally, they wish to recognize the positive contributions made to this project by their graduate assistants, Mario Lavoie and Guy Rochette. Mr. Lavoie was responsible for implementing the simulator, and Mr. Rochette the analytical method. They authored respectively Annexes III and II.

1. Introduction

In this report, the authors give the final results for the project entitled "A Study of New Services on Integrated Computer - Telecommunications Networks", Contract Number 20SU-36100-0-9511, for the Government Telecommunications Agency, Department of Communications, Government of Canada, for 1980-1981. This project was based in turn upon a project for the same Agency carried out the preceeding year (1979-1980) and entitled "A Technical and Economic Study on the Impact of Introduction of New Services on Existing Telecommunications Networks" [FINL'80a].

The basic purpose of this project was to develop telecommunication network performance evaluation tools, both analytical as well as simulation, permitting evaluation of the impact of various proposed new services upon the Canadian telecommunications network system. To reduce this project to manageable proportions, the authors centered their attention upon one class of new service that may be a kind of prototype for the majority of proposed new services, namely videotex. They likewise restricted their attention to the circuit-switched telephone networks POTS as it was this subsystem of the overall national telecommunications system that seemed to be the most critical bottleneck in the implementation of new services.

To this end, the authors developed an analytical as well as a simulation model for the study of circuit-switched networks. The basis for the analytical model is to be found in an article by Lin [LIN'78]. We have, however, considerably modified Lin's method.

The next problem was to develop a reasonable approximation to a "typical" local telephone network. This meant obtaining traffic intensity data from a telephone operating company. Here Bell Canada most graciously provided us with data from which we abstracted such a "typical" local network.

Finally, we had to estimate holding times for videotex traffic. Information gleaned from conversations with people at the pioneering Prestel project gave us a reasonable initial estimate.

The principal result of this study, given in detail in the last section of Chapter 4, is that, to the extent that the data abstracted from those provided to us by Bell Canada were correctly interpreted, the local telephone network is surprisingly fragile with respect to relatively modest increases in the number of videotex users. Thus, an increase from 1% of the total subscriber population to 5% yields extremely serious degradation of network performance as reflected in a dramatic increase

in the average network blocking probability.

This report breaks down as follows: in Chapter 2, we give some contemporary definitions of videotex and an overview of videotex activities throughout the world. The COSTPRO project deserves more than the passing mention it received, but we felt videotex merited our first concern. In Chapter 3, we summarize the International Standards Organization's approach to network architecture, the Open System Interconnection reference model. This concept is then applied to develop the concept of videotex networks. Then, in Chapter 4, we outline our modeling approaches, present an idealized "typical" local network, called by us BELL, and review the results of our two basic methods in Sections 4.4 and 4.5. Details of the theory, the program for the analytical method and the program for the simulation method are relegated to Annexes I, II and III respectively. Finally, in Chapter 5, we summarize trends in the local subscriber loop network.

We wish to stress that the conclusions stated in this report in no wise reflect either any policy or official opinion of the Department of Communications, Government of Canada or of Bell Canada. Rather, they are merely results of an academic study that are presented for the wisest use by the Canadian Telecommunications community.

2. NEW SERVICES

2.1 Introduction

In a previous report [FINL'80a] the authors gave a rapid survey of the then currently emerging new telecommunications services. Included in this survey were the following:

a- videotex or "still image" services;

b- interactive television;

c- electronic funds transfer systems (EFTS);

d- CTIS - Canadian Trade Information System, realized by the Canadian Organization for the Simplification of Trade Procedures or COSTPRO:

e- electronic mail.

After briefly characterizing these services and after having taken some pains to define the subclass of services dubbed by us "visual information system", the authors restricted their attention to the videotex service and oriented their basic study around the potential impact of this service on the

circuit-switched public telephone network. Restricting attention to the videotex service was justified as this service attains a certain universality in that, in principle at least, it contains all the other services, except interactive television, as special cases. Thus, electronic funds transfer involves transmission of specially formatted messages from remote entry points to and from a set of distributed databases. This would appear to be compatible with videotex. Likewise, electronic mail involves transmission of messages to electronic mail "post-office boxes", i.e., user databases, and again would appear to be consistent with videotex. Finally, the COSTPRO Canadian Trade Information System or CTIS again involves transmission of specially formatted messages to destributed databases, hence within the purview of videotex.

Indeed, according to the scheme presented by the authors in their previous report, from the point of view of the circuit-switched telephone network, the above-mentionned services all involve access to a (class-5) telephone office, finding a path through the network to another (class-5) telephone office to which the databases used may be connected just as any subscriber would be. Hence, taking as measure of impact the node-to-node grade-of-service as function of traffic intensity, it makes little difference whether the service be videotex, CTIS, or whatever. The quantitative results of our impact

study are given at the end of Chapter IV of the present report.

To return for a moment to the authors hypothesis concerning the universality of videotex as a prototype of new (narrowband) services, one must take notice of the current high degree of interest in this service as reflected in the numerous conferences, seminars, and other kinds of meeting devoted to various aspects of videotex. Such assemblies seem to be heavily loaded on the marketing, implementation, and theoretical aspects of videotex, but very sparse as far as traffic data are concerned. Among the functioning videotex systems, however, one observes the following phenomena:

a- videotex services seem to appeal more to special interest business and industrial groups than to the purely home market;

b- the applications of videotex seem to fall into the categories hypothesized by the authors, i.e., EFTS, electronic mail, and other specialized applications;

c- there is serious doubt about the viability of the home market.

The authors' hypothesis thus has been at least partially vindicated by experience thus far obtained. In the remainder of this chapter, the authors give the currently accepted definitions of videotex, summarize the major trends in the application of this new service and resume briefly the COSTPRO system.

2.2 Definition of Videotex

According to the CCITT, videotex may be defined as follows:

"Videotex systems are text communication systems having in addition the capability of a given level of pictorial representation and a repertoire of display attributes. The text and the pictures obtained are intended to be displayed using the current television (TV) raster standards of the different countries." [CCIT'80]

This definition makes of videotex a true communications medium rather than an elaborate storage and retrieval mechanism. The messages to be communicated may have both a textual as well as a graphic nature. The definition is silent on the question of interactivity. Thus, the British Prestel group gives the further definitions [FEDI'79]:

Videotex - "A communication system in which digitally encoded frames are transmitted for reception by a modified TV set where a limited number of frames are stored and displayed. Most systems have a colour capacity employing the fundamentals red, green, blue (on or off) in any combination thereby giving 8 colours (black, white, red, green, yellow, blue, magenta, cyan)."

Broadcast Videotex - "The generic name for Videotex systems employing one way communication. The entire set of frames is transmitted repeatedly, the user specifies and the receiver selects, stores and displays the required information. Most systems at present are inserting the information in the interframe blanking of a TV signal transmitted over the air."

Interactive Videotex - "The generic name for Videotex systems employing two way communication. The user is able to communicate with the system to specify his requirements. Single frames are transmitted to the receiver, where they are stored and displayed..."

These definitions, which reflect the concepts of British Telecom (the former British Post Office) and some other

European PTT's, are not universally accepted. For example, broadcast videotex is more frequently referred to as teletext and in North America videotex is normally taken to be at least potentially interactive, although a number of implementations initially are of the broadcast or teletext type.

Historical Note:

Videotex seems to be a concept that originated with the British Post Office: initially referred to as "viewdata", videotex was seen as a means of augmenting revenue on domestic telephone services, the domestic telephone being notoriously underused. The British called this system Prestel. Through the energetic efforts of the British, the concept appears to have penetrated the continent and North America, in particular Canada. In France, the ANTIOPE project, designed apparently to use the newly introduced packet-switched network, offered initially a broadcast or teletext form of videotex, followed by the ambitious Télétel project which offers true interactive videotex. In Germany, a form of the British videotex was adopted and called Bildschirmtext. In Canada, the Department of Communications advanced a videotex scheme, namely Telidon, which was adopted by Bell Canada for the latter's Vista project. The Japanese developped in 1979 their CAPTAINS system (Character and Pattern Telephone Access Information Network System) which must deal with the

thorny problems of the Japanese writing system (kanji and the kana alphabets). Finally, in a dramatic move whose consequences are yet to be fathomed, the American giant, American-Telephone and Telegraph announced in May 1981 its proposal for a presentation level protocol (see Chapter 3) as a North American videotex standard [AT&T'81a] This proposal appears to be largely consistent with the Telidon scheme, but divergent from the British Prestel, less so from the French Télétel. It would seem to favor the introduction of a modified form of Telidon on a mass scale in the North American market.

2.3 Recent Trends in Videotex [VIDE'81]

For the purposes of this study, the authors limited themselves to the impact of videotex on the circuit-switched telephone network. The quantitative results of their study are presented in Chapter 4. In this section, they wish to summarize current trends in videotex in an attempt to give a qualitative image of future utilization of both the telephone networks as well as of other telecommunications networks such as that dark horse, cable television.

Concept of Videotex

Several definitions of videotex were advanced in the previous section in which the communications aspect was stressed, thus making of videotex a new communications medium, much more than a classical retrieval system or computer messaging system. In actual practice, it is surprising how much confusion exists as to what videotex really is or is not. Indeed, one might crudely say that videotex is naught but a friendly means of tapping computer services, at this moment primarily of a retrieval or messaging nature in which some pictorial elements may be present. In the long run, videotex may turn out to be the means of bringing computer power within reach of large segments of the commercial, industrial, governmental and home constituents of our society. Since the telephone network will

undoubtedly carry a heavy portion of this traffic, our impact results presented in Chapter 4 take on an additional importance.

Technical Trends

- Standardization - There is currently a heated contest raging in order to define internationally acceptable standards for videotex. Clearly, such a step is necessary if we wish to avoid a fragmentation of the user communities - or to put it more bluntly, of the potentially enormous market-place-along the lines of different manufacturers individual standards. A universally accepted standard would permit users to access any database, whatever his terminal equipment might be. The CCITT proposed standard [CCIT'80] seems to include the Prestel, Telidon and Télétel standards. Pursuing goals best known to itself, the American Telephone and Telegraph company recently announced it own standard [AT&T'81 a, b] which seems to have a high degree of compatibility with the Telidon system and much less with Prestel. The AT&T standard takes the form of a presentation level protocol as the latter is defined by the International Standards Organization (ISO) in the latter's Open Systems Interconnection (OSI) Reference Model (see [ISO'79] and the discussion of this model presented in Chapter 3 of this report). This sixth-level protocol in the seven-level layered architecture proposed by the ISO is primarily

responsible for the encoding of text, graphics, and display control information (see Figure 3 in Chapter 3). (Indeed, from the point-of-view of the OSI model, videotex might be considered as defined by such protocol.) Key features of the AT&T proposed protocol include:

- allowing for both 7- and 8-bit coding;

- dynamically redefinable character sets (DRCS). (to be described in more detail below);

- color-mapping to yield a wide range of colors;

- possibility of continous scaling of text size;

- possibility of creating macro PDI's (picture description instructions);

- instructions for creating images such as signatures and logos;

- possibility for simple animation.

It appears that the Canadian Telidon standard (viewed as a sixth level protocol) should be easily rendered conformable to the AT&T proposed standard. In fact, AT&T

claims [BERK'80] to have consulted closely with both the Telidon group as well as the ISO subcommittee on videotex.

- Third Party Databases - The German videotex Bildschirmtext appears to be the first public videotex system with gateways into other systems, notably a computerized banking service. This concept of videotex as a gateway to database systems outside the videotex system proper is referred to as a "third-party database " and is eleborated in more detail in Chapter 3.

- Dynamically Redefinable Character Sets (DRCS) A DRCS is a set of characters whose encoded descriptions, contained in the host databases, may be downloaded to the user terminal. This is a powerful option that all the major videotex competitors are attempting to offer [AT&T'81, CCIT'80, VIDE'81].

- Picture Videotex - This term is used to describe the possibility of transmitting digital compressed photographic images downstream to the subscriber whose terminal logic then reconstructs the original image. Prestel seems to be the first system to offer this service [PRES'80].

- Videotex on Cable Television Systems - A number of cable television operators are offering or preparing to

offer videotex on their networks (see the following section on user trends). Video channels are divided into a number of videotex and other lower bit rate channels. Access to these channels uses carrier-sense multiple access methods (CSMA) making of the cable television plant a potentially serious competitor to the telephone system as a carrier for computer communications traffic.

- Videodisc - Videodiscs are being considered as a convenient storage and retrieval medium for videotex [SCHU'81]. It is to be noted that this usage was forseen by one of the authors four years ago [FINL'77]. Videodisc offers great advantages as a storage medium due to its enormous information storing capacity and compactness. Design parameters for such applications are different from those of intertainment videodiscs, however. There are thus incompatibility problems to be overcome. For further discussion of this medium and additional references see the article of Finley on visual information systems cited in the Bibliography [FINL'80b].

User Trends

A vast spectrum of videotex applications are being planned or already operational, both in testing situations as well as in hard-headed commercial ventures. Applications include banking, teleshopping, mail order companies, travel

agency functions, publishing (magazines and newspapers), theater or cinema reservations, stock market information, agricultural and educational systems. An excellent cross section of these applications may be found in Proceedings fo the Videotex'81 conference recently held in Toronto [VIDE'81]. The following section will also give the reader an excellent idea of the spread of videotex activities thoughout the world.

Overview of Videotex Activities Worldwide

Given below is a list of videotex activities worldwide. This list was compounded from various sources, notably [VIDE'81] and [LINK'81]: given the fluid situation, one cannot claim that it is up-to-date or exhaustive. For each project are given the name of the country and if possible, the city wherein the project is located, the project name, the service name and the number of users.

List of Videotex Acitvities Worldwide

Country	City/Region	Company/Agency	Project/Service	Number of Terminate	Starting Date
Australia	-	Source Telecomputing	The source	(8,000?)	June 1979
Austria	-	Austrian PTT	Bildschirmtext	300	March 1981
Belgium	-	French Television & Telecommunications Research Center	Perceval	?	?
Brazil	Sao Paolo	Telesp (Telecomunicaçodes de Sao Paolo)	French Télétel	1,000-1,500	?
Canada	South Headingly, Manitoba	Manitoba Telephone System	Project IDA	30	Operational
	Ontario	Ontario Education Communications Authority	Project Telidon	55	Operational
	Calgary, Alberta	Alberta Governnment Telephones	Vidon (Telidon)	30	mid 1981
	Saint-John, New Brunswick	New Brunswick Telephone Company	Project Mercury (Telidon)	20	early 1981
	Toronto & Quebec City	Bell Canada	Vista (Telidon)	500	fall 1981?
	Vancouver	British Columbia Telephone	Videotex Project (Telidon)	150	mid 1981
	Montreal	Télécâble-Vidéotron	Télidon-2	250	1982

List fo Videotex Activities Worldwide (continued)

Country	City/Region	Company/Agency	Project/Service	Number of Terminate	Starting Date
	Elie, Manitoba	Manitoba Telephone System	Elie Project (Telidon)	150	fall 1981
Denmark	-	Danish PTT	Teledat	200	January 198
Finland	Helsinki	Helsingin Telset Oy	Teleset	200	1978
	-	Teletieto Oy	Teleset	?	1981
France	Vélizy	French PTT	Télétel	3,000	June 1981
	Ille-et-Vilaine	French PTT	Electronic Directory	250,000	1982
Germany	Berlin, Düsseldorf	Deutsche Bundes-post	Bildschirmtext	6,000	1980
Holland	Amsterdam	VNU/TVS	TVS	50	1980
	-	Dutch PTT	Viditel	4,000	1980
Hong Kong	Hong Kong	Hong Kong Telephone	Viewdata (Prestel)	?	1980
Italy	-	Societa Italiana per l'Eservizio Telefonica PA (SIP)	Prestel	?	?
Japan	Tokyo	Ministry of Posts & Telecommunications	CAPTAINS	1,000	1979

List of Videotex Activities Worldwide (continued)

Country	City/Region	Company/Agency	Project/Service	Number of Terminate	Starting Date
	Higashi-Ikoma	Ministry of International Trade and Industry	HI-OVIS	165	1979
Sweden	Stockholm	Swedish Telecommunications	DataVision	30	1980-1981
Switzerland	Berne	Swiss PTT	Prestel	100-150	1979
United Kingdom	-	British Telecom	Prestel	8,000	1978
United States	Kentucky	Department of Agriculture	Green Thumb	200	1980
	Texas	Dow Jones	?	8	1979
	Coral Gables, Florida	ATET with Knight-Ridder	Viewtron	260	1980
	Dallas, Texas	Belo/Dow Jones	Park Cities, Texas	35	1980
	Columbus, Ohio	OCLC	Viewtel	200	1980
	San Diego, California	Home Serv/Cox Cable	?	50	1981
	US + Canada	Dow Jones	News/Retrieval Service	15,000	1977
	US, Canada, Australia	Source Telecomputing	The Source	8,000	1979

List of Videotex Activities Worldwide (Continued)

Country	City/Region	Compangy/Agency	Project/Service	Number of Terminate	Starting Date
	-	CompuServ	CompuServ Information Service	7,000	1979
	New-York City	Citibank	?	100	1981
	New-York City	Chemical Bank	Project Pronto	200	1981
Venezuela	Caracas	Central Office of Statistics & Information of the Presidency of the Republic	Orientation & Information System (uses Telidon)	?	1980

2.4 COSTPRO [COST'80]

"The Canadian Trade Information System (CTIS) is a set of voluntary standards designed to assist businesses to be competitive by reducing the overhead costs of managing trade transaction information"[COST'80]. This represents no more and no less than an attempt to rationalize trade information documents, the machinery for handling such documents, the exchange of such documents via electronic media, in particular the packet-switched networks available in Canada, namely Datapac and Infoswitch. It is extensively described in a set of documents published by COSTPRO.

As of this writing, the autors have no additional data on this system than was presented in their previous report. However, to the extent that CTIS terminals access Datapac or Infoswitch via the local telephone network, all the arguments applied to videotex apply equally well to CTIS mutatis mutandis. Therefore, for initial modelling purposes, the impact of this service will be similar to that of videotex.

3. NETWORK ARCHITECTURE FOR NEW SERVICES

We can define a telecommunication service as a set of resources, e.g., data bases, information processing systems and programming languages, to which users have access, either locally or remotely. These resources form a processing system which we shall refer to as service providers. The service provider for a telecommunication service may be distributed into different locations as are the users. The means by which users have access to the service provider is via one or more telecommunication networks. The means by which different resources in the service provider communicate in order to respond to users' requests is also via a telecommunication network. Thus, telecommunication networks are the backbone for present and future telecommunication services. This section aims at exploring the relationship between the telecommunication services and the telecommunication networks within the framework of layered architecture, a concept which has emerged in the last decade as a solid framework for computer network organisation.

3.1 Layered Architecture and Telecommunication Services

A network is a collection of nodes interconnected by links. A node is characterized by a set of functions it

must perform to assure communication between users of the network. The users may be, as we have previously pointed out, the users of a telecommunication service as well as the service providers. Some of the basic functions are the following [GREE'80]:

1. Guarantee that a transmission path exists using the common carrier;

2. Ensure that the message delivered is in bit form using Modems;

3. Move individual messages using datalink control;

4. Provide economic intermittant use using dial-up, multidropping, multiplexing, packet and fast circuit-switching;

5. Send messages to the correct node and the correct subaddress within the node and to bypass failed lines or stations;

6. Accomodate buffer size: avoid need to resend long messages using packetizing-depacketizing;

7. Resolve missmatches between actual and accomodatable

flow rates using flow control and buffering;

8. Accomodate end-user intermittency patterns using Datagram, transactions or session dialogue management;

9. Accomodate end-user format, code, and language requirements using protocol conversions.

Once the functions just listed are provided, communication between the users and the service provider is possible. These functions are shown in Figure 3.1 as different layers. A user message would go from layer to layer as it is routed in the network from origin to destination. Character streams typed in by the terminal user undergo a protocol conversion, then have sequence numbers and other information added for managing the dialogue, are then provided with addresses, arranged in packets, and so forth. Two interesting things are obvious: the functions in the same layer occur in pairs, and the two members of each pair talk essentially only to each other. For example, one modem talks to the other, ignoring both details of the transmission link and the meaning of bits it is handling. As another example, a DLC (data link control) element ignores what its modem is doing about modulation and demodulation and also what the information field within a

frame contains. A DLC interacts only with the DLC at the other end to convey the frame successfully from one sending node to the next receiving node on the same line, and so forth.

In systems terminology, the service provider and service users are open systems. The need for interconnecting open systems led the ISO (International Organisation for Standardization) to decide on creating the SC 16 subcommitte with the objective to define a standard for "open system interconnection". SC 16 decided to give highest priority to the development of a standard model of architecture which would constitute the framework for development of standard network communication protocols.

The basic objective of SC 16 is to standardize the rules of interaction between interconnected systems. Thus, only the external behavior of the open systems must conform to the OSI (Open System Interconnection) model proposed by the SC 16, while the internal organization and functioning of each individual Open System is out of the scope of the OSI standard, since these are not visible from other systems with which it is interconnected [ZIMM'80].

The OSI model is based on the general principle of layering. Layering is a structuring technique which permits the network of Open Systems to be viewed as logically composed

of a succession of layers, each wrapping the lower layers and isolating them from the higher layers as examplified in Figure 3.1. The OSI uses the alternative but equivalant representation of layers as shown in Figure 3.2 where successive layers are represented in a vertical sequence, with the physical media for Open System Interconnection at the bottom.

Each individual system is viewed as being logically composed of a succession of subsystems, each of which corresponds to the intersection of the system with a layer. In other words, a layer is viewed as being logically composed of subsystems of the same rank for all interconnected systems. Each subsystem is, in turn, viewed as being made of one or several entities. In other words, each layer is made of entities, each of which belongs to one system. Entities in the same layer are termed peer entities.

The basic idea of layering is that each layer adds value to the service provided by the set of lower layers in such a way that the highest layer is offered the set of services needed to run a distributed system. Except for the highest layer which operates for its own purpose, entities in a given layer distributed among the interconnected Open Systems work collectively to provide the service to the entities in the next higher level. In other words, the entities in a given layer add value to the service they receive from the next lower level and offer

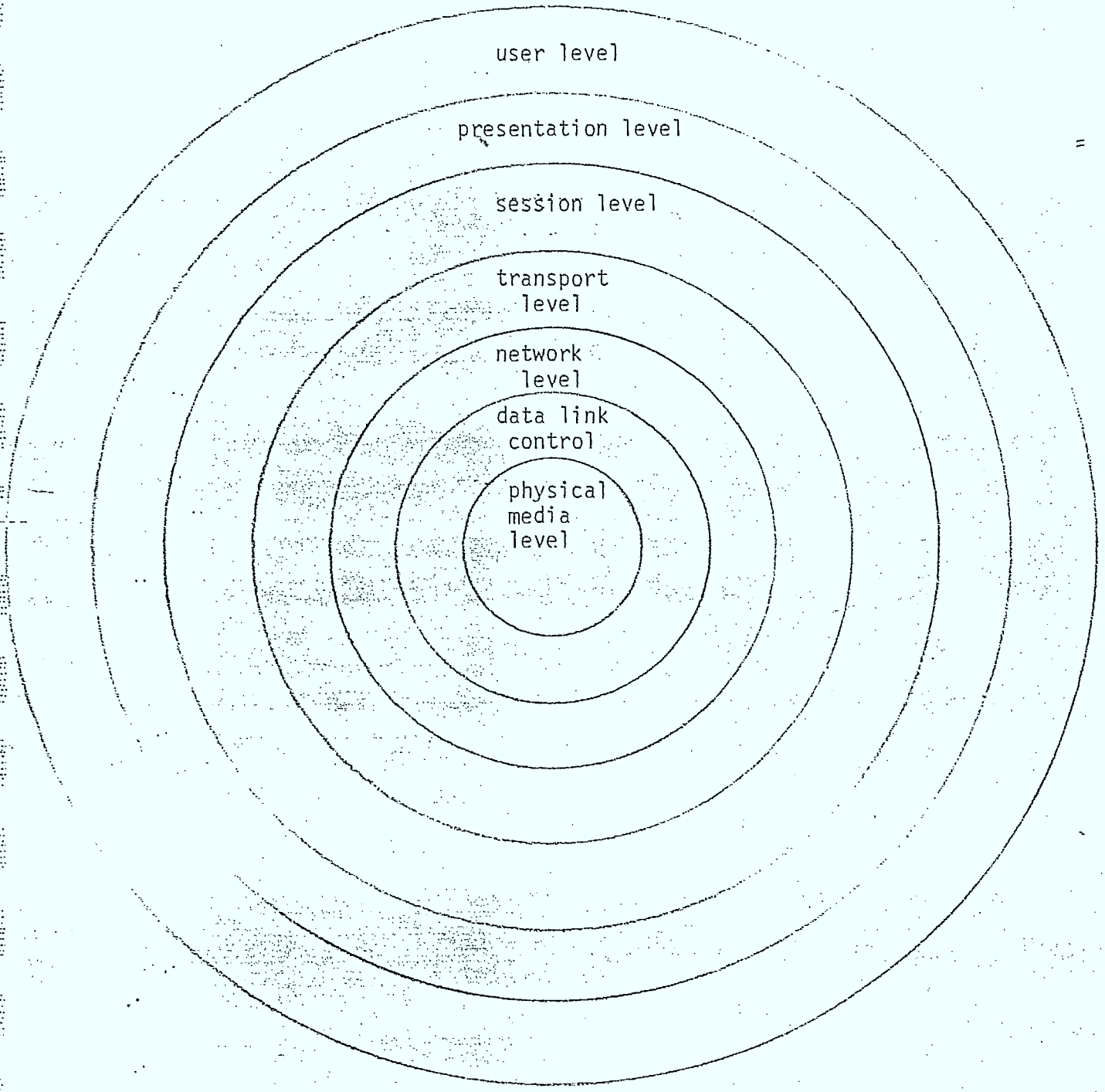
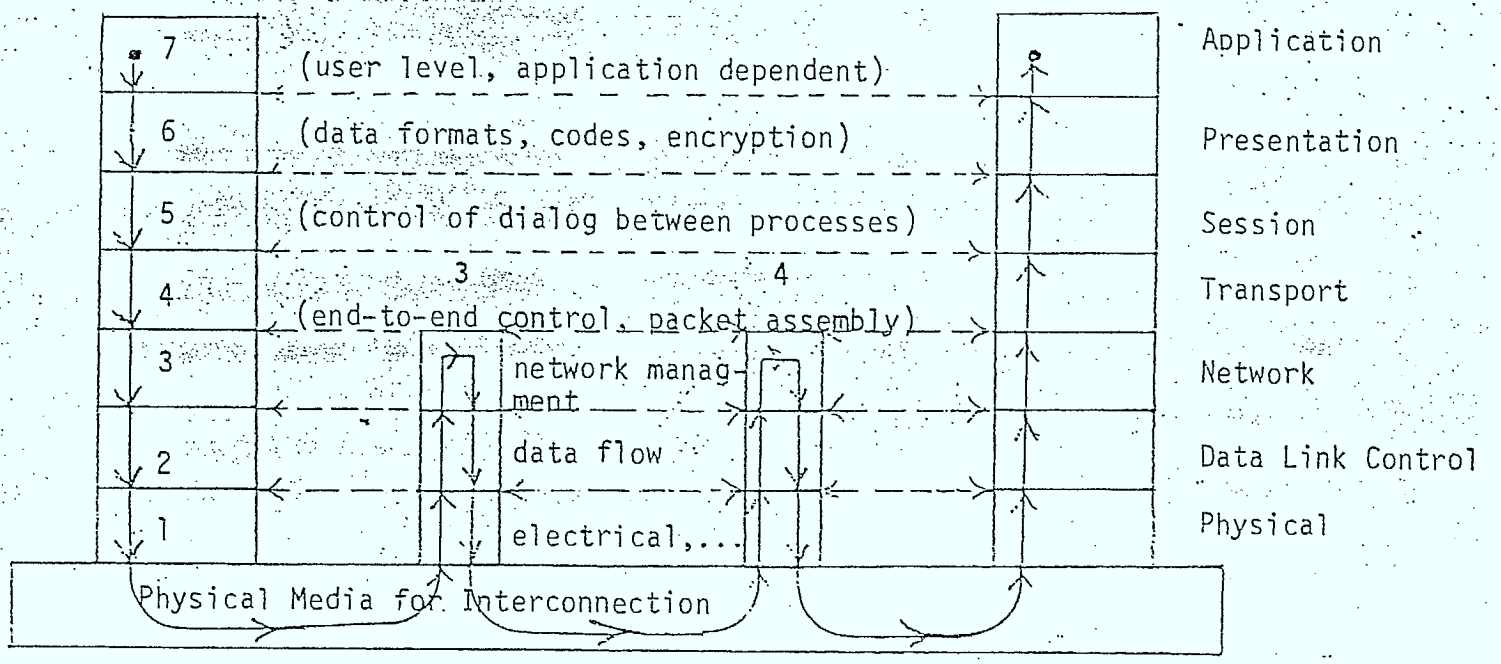


Figure 3.1. - The OSI Layered Model (See Figure 3.2)

Figure 3.2. - The OSI Layered Model (Vertical Arrangement)

This diagram shows the International Standards Organization's (ISO's) Open System Interconnection Reference Model (OSI/RM). Communication between applications at two different stations 1 and 2 is illustrated with intermediate stops at 3 and 4 (as, for example, in a packet-switched network). The user at level 7 of station 1 only "sees" his homologue at station 2 at the same level, the lower layers being essentially transparent to him. As a message of level 7 is transmitted, it descends through the lower levels at each of which appropriate information is added. This information is "stripped off" as the message ascends to level 7 of its destination.



this value-added service to the entities in the next higher level.

Communication between the entities of a given layer makes exclusive use of the services offered by the next lower layer. In particular, communication between entities of the same layer and of the same system, e.g., for sharing resource, is not covered by the OSI Architecture. Entities in the lowest layer communicate through the physical media. Cooperation between the entities in the same layer is ruled by the protocols of that layer which define how the entities work together using the service of the next lower level in order to offer service to the next higher level (Figure 3.2).

The services performed by a given layer are offered to the higher layer at the service access points which represent the logical interface between entities of the two contiguous layers. Communication between these entities are ruled not by protocols but by the interface.

The OSI Reference Architecture consists of the following seven specific layers (see Figure 3.2).

1. Physical Layer: Provides for transmission of transparent bit streams between data link entities

across physical connections;

2. Data Link Layer: Provides specific techniques for different communications media (e.g., telephone lines) to be used in order to transmit data between systems despite a relatively high error rate;

3. Network Layer: Provides a connection path (network connection) between a pair of transport entities;

4. Transport Layer: Control of data transportation from source-end system to destination-end system (which need not be performed in the intermediate nodes);

5. Session Layer: Bind and unbind distributed activities into a logical relationship that controls the data exchange with respect to synchronization and structure;

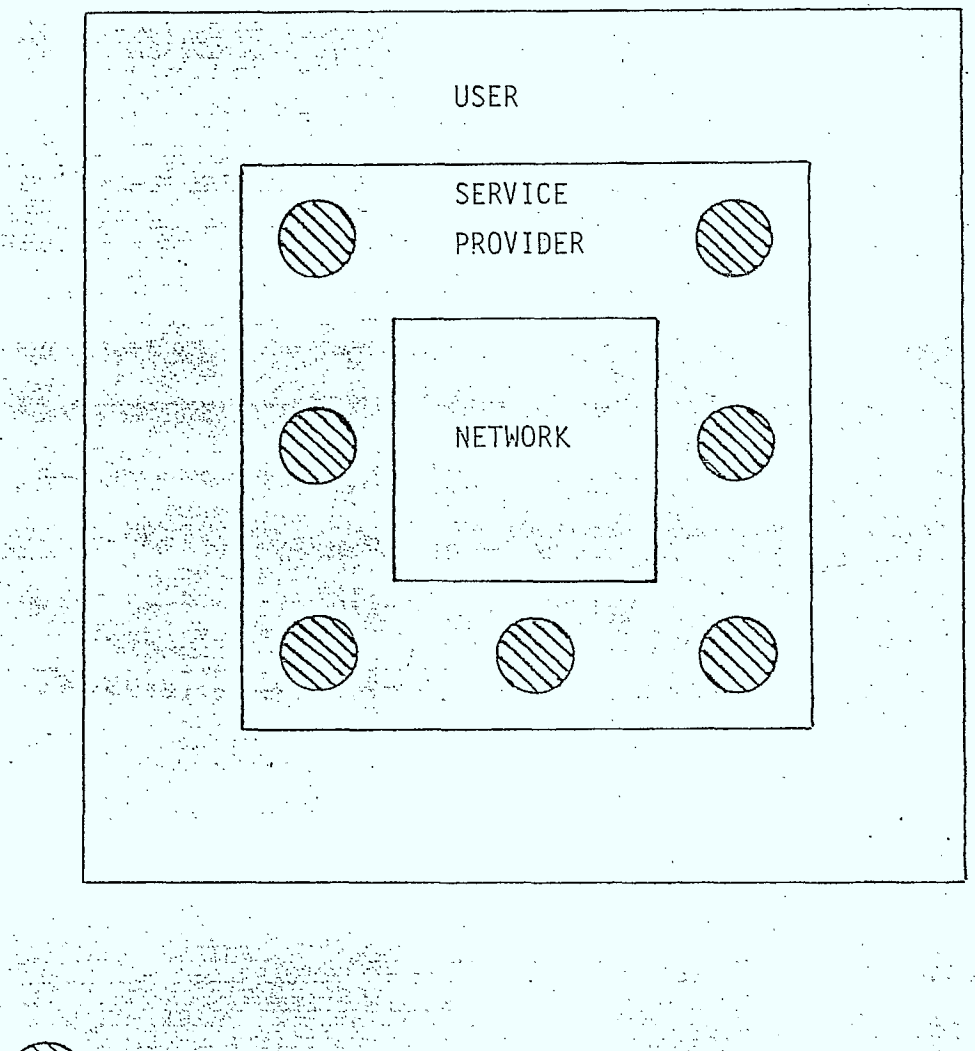
6. Presentation Layer: Perform functions related to representation and manipulation of structured data for the benefit of application programs;

7. Application Layer: Provide the distributed information service appropriate to an application, to its management and to system management.

To perform the services mentioned above, entities in a layer must cooperate via protocols. Detailed functioning of each layer is defined by the protocols specific to the layer in the framework of the Architecture model. OSI has made considerable effort in the definition of standards protocols for each layer. Standards already existing within the CCITT for the physical layer include X.21, V.26, V.35, etc. [BERT'80]. The most popular data link layer protocol is HDLC [CARL'80]. An important basis for the protocols in the network layer is the X.25 interface [RYBC'80, WECK'80] for the network layer. X.25 will be important for both users and providers of new telecommunication services because it allows the use of public data networks as carriers of new services.

Most new services will have to implement protocols for the three highest layers, namely, session Layer, presentation Layer and application Layer. The services these three layers offer are those carried out by what we have called the service provider. The entities in the service provider made use of the communication services offered by

Figure 3.3. - Relationship between Telecommunication Services and Networks



a network (Figure 3.3). Many new telecommunication services will make use of public data networks to establish communication between entities or "service processes" in the service provider. In this case, the most popular interface between the service provider and the public networks will probably be the X.25 interface. Some service providers however, will have private telecommunication networks to maintain communication between their service processes.

For most future telecommunication services, the interface between users and the service provider may not be direct. The users may have to make use of private or public networks to access the service provider. In this case, the network services such as virtual circuit or datagram would be the most popular.

The layered architecture will thus provide a framework for developping new services.

3.2 Architecture for Videotex Services

In this section, we consider planned and possible network structures for videotex services. This subject was the main theme of a workshop held at the Département d'informatique et de recherche opérationnelle, Université de Montréal,

in June 1980, whose results are summarized in an article by Ball et al [BALL'80]. We refer the reader to this article for a more complete discussion.

The users of the videotex service are represented by user terminals (UT). A typical videotex U.T. is connected to a communication medium, uses a conventional TV set for display and has a keypad for selecting the desired information or service. The terminal provides a basic character-oriented display mode (alphamosaic coding), and some systems more advanced capabilities as graphics (alphageometric) and photographic modes. The videotex service provider is a collection of videotex processing centers also called concentrators, videotex interface machines, videotex switches, or local access. The videotex centers are what we have previously called service processes and constitute what we will call a videotex network. The videotex centers provide effective interface between the videotex dervices and user terminals.

The simplest possible videotex network consists of a single computer with a centralized data base. All users and information provider terminals are connected directly to the centralcomputer through dedicated or dial-up telephone connections. Example of such simple videotex networks are the initial Prestel International System in England and various

Telidon field-trial configurations in Canada.

A more general version of netwroks with central data bases as shown in Figure 3.4, designed to serve several thousand users, is currently used in British Telecom's Prestel and its derivatives. The update center maintains a master data base and controls the interaction with information providers. A number of information retrieval centers are connected to the Update Center through leased synchronous lines; all the retrieval centers maintain identical copies of the master data base and provide dial-up ports user terminals.

As the quantity and diversity of information grow, the need arises for many independent data bases, which users select through a master service directory or by gateway pages within a given data-base. This design approach leads to the network of Figure 3.5 which is an extension of Prestel-like systems in two directions: This network (1) integrates independent third-party data bases and (2) includes a packet--switched network. In terms of layered architecture, the transport service is provided by a packet-switched network. At least two such videotex systems are currently planned, Bildschirmtext in Germany and a field trial at Vélizy, France. Independent third-party data bases and possibly other services

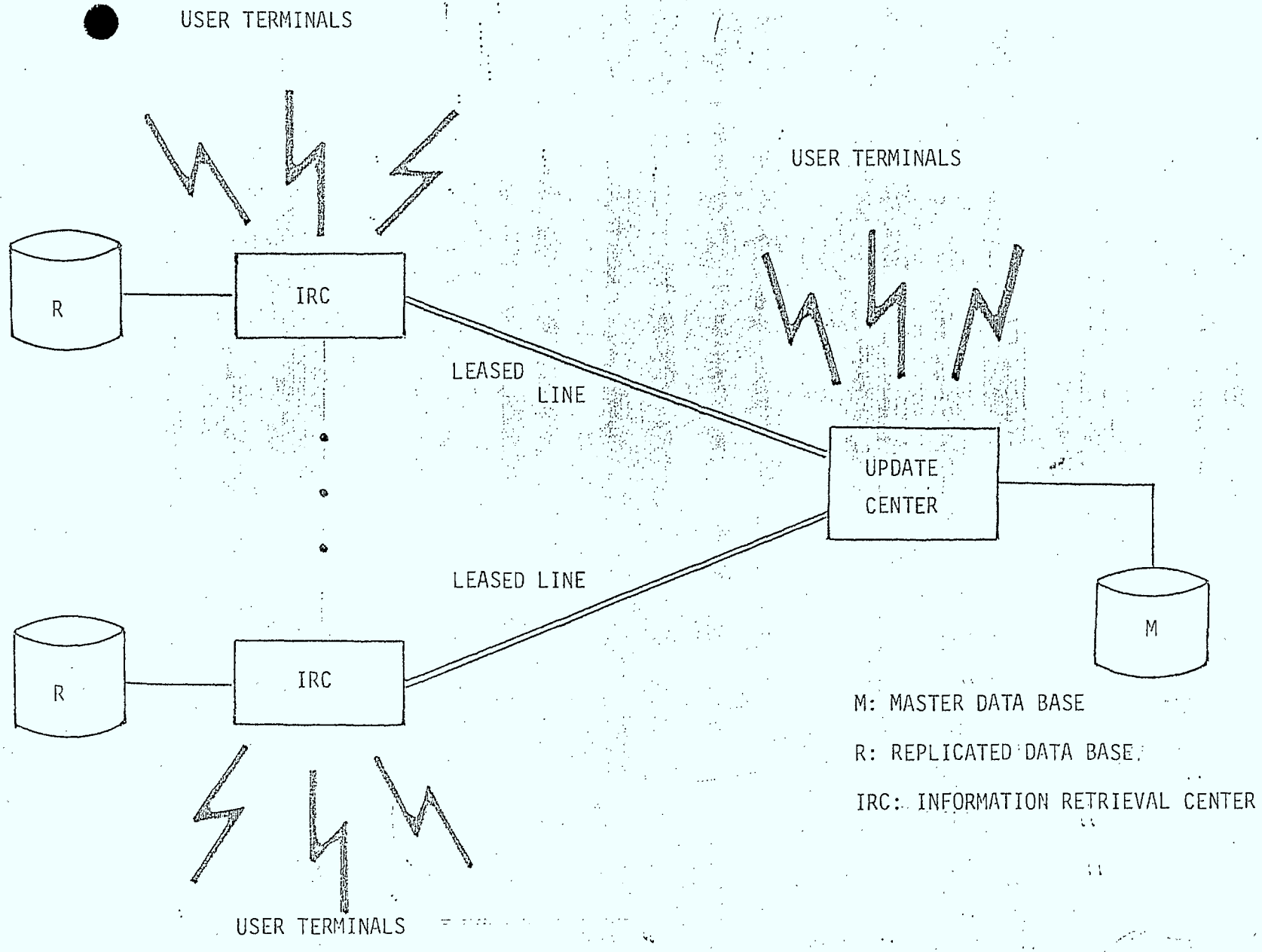


Figure 3.4. - Centralized Videotex Network

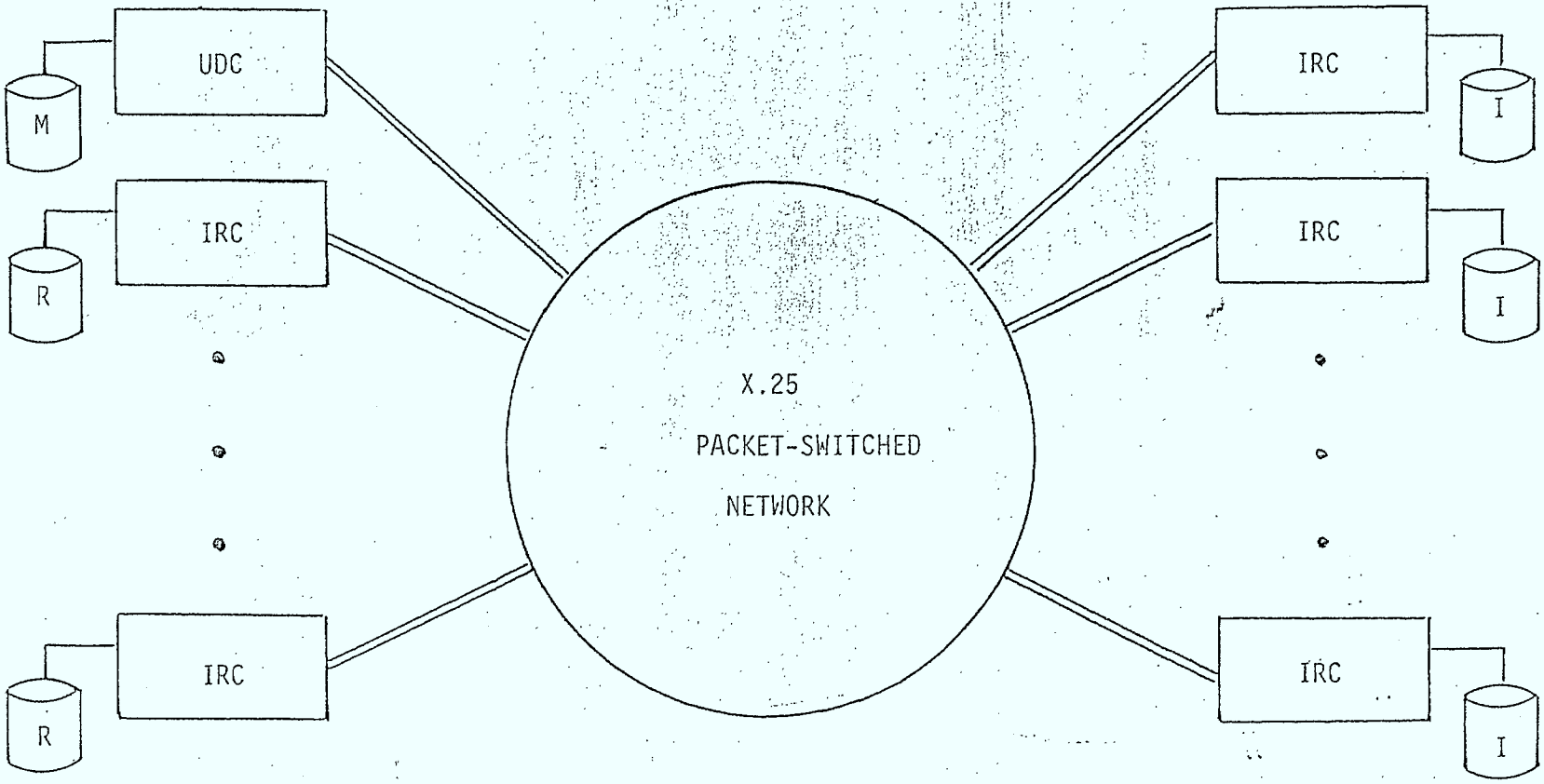


Figure 3.5. - Videotex in a Packet-Switched Network with Third-Party Databases.

may be accessed by the user. Such services are implemented on external computers connected to the public data network. An example of such a service is the catalog-lookup and order processing offered by the Quelle department in Germany.

In the networks discussed above, all network functions are performed in a single computer. In the French Antiope System, network functions are distributed. Users are connected through switched telephone lines to videotex centers, which handle interface functions for the user. The videotex centers, videotex data bases, the update computer with the videotex master data base, and independent third-party data bases are all connected through the Transpac public data network.

Some design issues for videotex services will now be discussed:

. Data Distribution: The merits of the two extreme solutions - complete duplication of all data in each data base versus partial data bases at different sites - are frequently discussed in the literature [LIEB'78]. For videotex applications, we have several reasons for maintaining data bases with partial information. Firstly, independent third-party data bases will be possible in the videotex network. Secondly, some information providers might prefer to

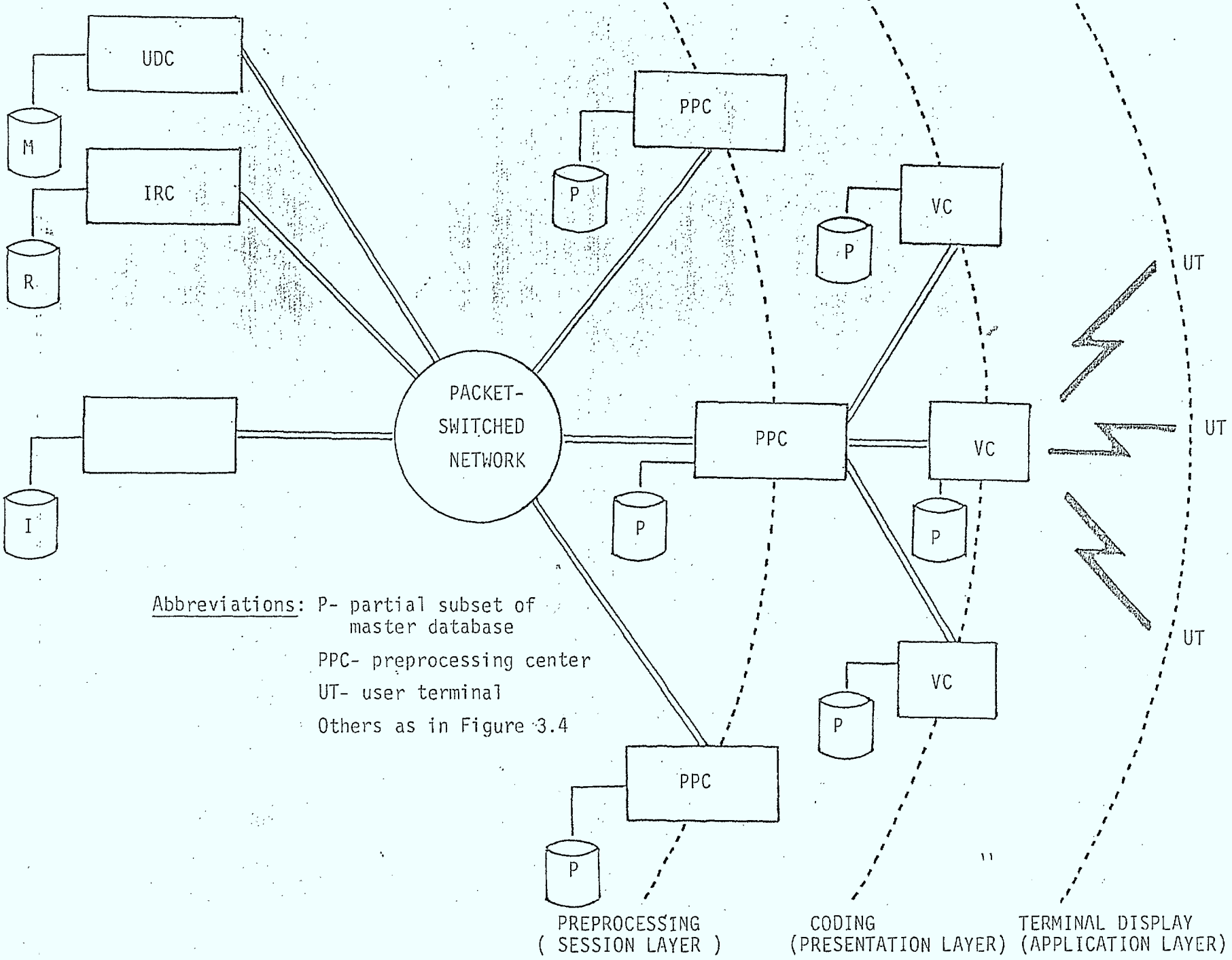
manage their own data. Thirdly, data bases are frequently geographically distributed to reduce communication costs. Thus, data distribution will be necessary for videotex service and the issue is how to realize this goal in a cost-effective way and free of performance problems such as data access bottlenecks.

. Directory and transparency: Data distribution requires some kind of metaservice directory to guide the user to the desired service or application. Another related issue is whether or not the distribution of information should be visible to the user. In videotex networks, both approches should be made available to the users with clear pricing policies.

. The rôle of Vidoetex Centers: Basically, the videotex centers act as concentrators to save communication costs. However, as the distribution of videotex service grows, more intelligence should be put into the videotex centers. Ideally, they should perform the functions of layer 5 (session layer) and layer 6 (presentation layer) of the OSI Architecture. These include: metaservices, virtual terminal standardization, access and control, storage management, software processing and personnal computing.

The preceeding discussion leads us to conclude that videotex of the future will have the distributed architecture as the one shown in Figure 3.6. Note that this architecture conforms with the layered architecture. The videotex center would perform function of the presentation layer, i.e., coding, verification and local videotex services. Larger videotex processing centers can perform session layer functions for user queries which require distributed processing. They will route user queries through a high speed packet switched network to the appropriate service computer, receive the reply, and return the requested information or service to the user.

Figure 3.6. -Future Videotex Network



Abbreviations: P- partial subset of master database

PPC- preprocessing center

UT- user terminal

Others as in Figure 3.4

4. IMPACT OF VIDEOTEX SERVICE ON THE TELEPHONE NETWORK

We have seen in the last section that the videotex centers are the concentrators of videotex user terminals. The latter access the former essentially through the telephone network. Thus, the videotex service will generate additional traffic in the telephone network. This section aims at evaluating the impact of videotex traffic on the telephone network.

4.1 Modeling the Telephone network for Performance Analysis

The methodogy for evaluating the performance of a communication network has been discussed at length in our previous report [FINL'80a]. For telephone networks, the principal performance measure of interest is the network grade-of-service given in terms of the blocking probability for an arbitrary call arriving to the network.

To conduct a network performance evaluation project, the following steps have to be carried out: (i) network characterization, (ii) service characterization and (iii) network modeling. We have successfully developped theoretical as well as software tools for carrying out the essentiel steps in a circuit-switched netwrok performance evaluation endeavor. Our results are presented in detail in Annexes I,

II and III. The reader is referred to these annexes for more complete discussion. There we will only summarize the major results.

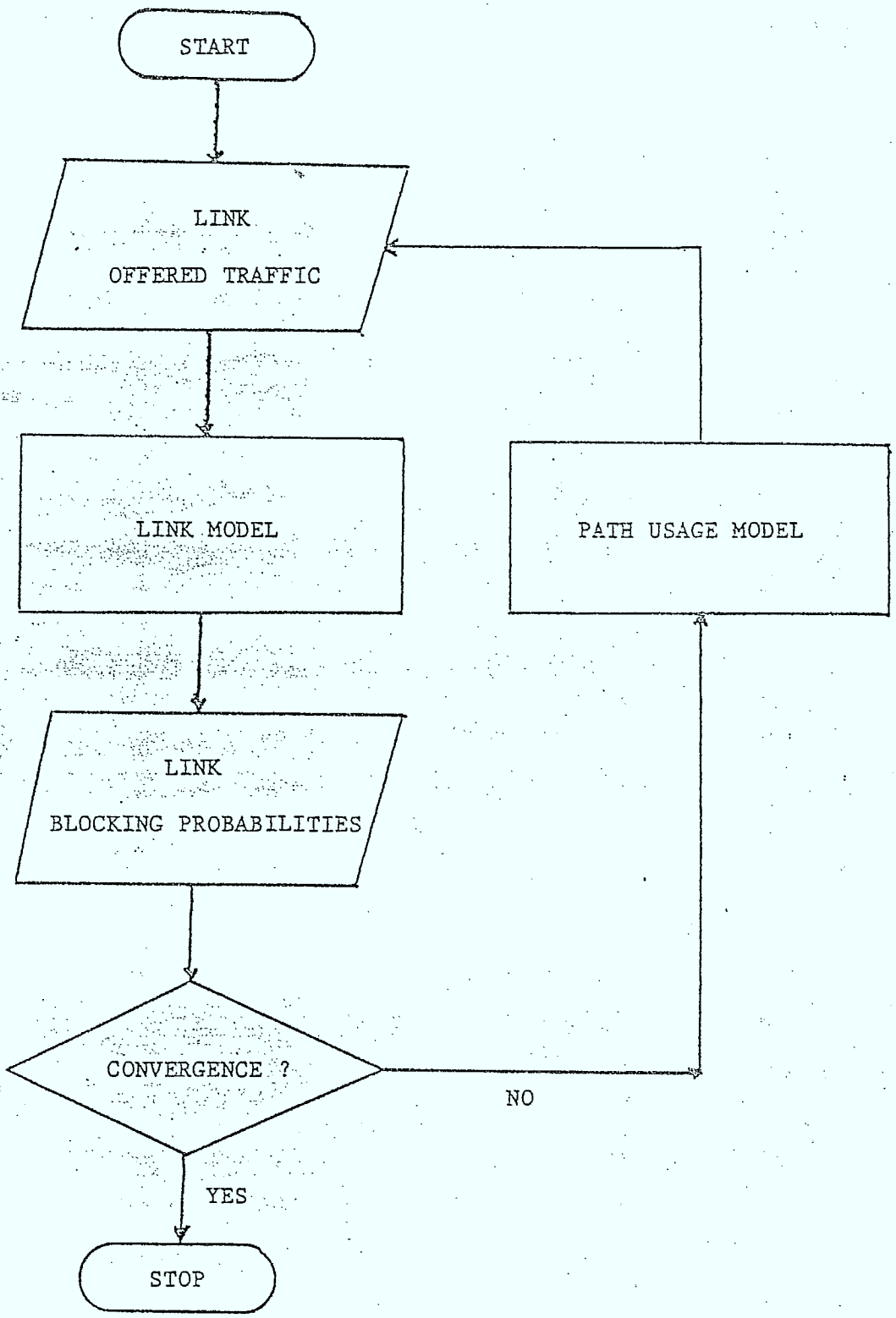
. Modeling Method for Performance Analysis of Circuit Switched Networks (Annexe I).

The theory of traffic engineering in a circuit-switched network is developed in Annexe I. The network is characterized by its configuration and by its routing scheme. Telecommunication services are characterized by their mean holding times. Network modeling is usually carried out by an iterative method and the iterative technique which has often been adopted breaks the network modeling process into two distinct steps:

1. Link Modeling: In this step, assuming that the link offered traffic is given, a model, which will be called the link model, is constructed to calculate all link blocking probabilities;

2. Path Usage Model: In this step, supposing that link blocking probabilities are known, a model, which will be called path usage model, is sought to estimate the link offered traffic.

Figure 4.1. - Iterative Algorithm for Network Modeling



Once these two models become available, the state vector of the network, i.e., the vector of link blocking probabilities, is obtained by iteration as depicted in Figure 4.1. At each iteration, the link blocking probabilities are calculated by the link model using the link offered traffic obtained by the path usage model in the previous iteration.

The link model usually adopted is Erlang's formula. However, the question as to whether Erlang's formula still holds for the case in which the link is offered a number of independent streams of Poisson traffic with distinct holding time distributions has not been raised.

Furthermore, the individual impact of each traffic stream on the link cannot be studied solely by using Erlang's formula. Since circuit-switched networks are essentially carriers of an increasing number of new and old telecommunication services each of which generates traffic with distinct holding time distributions, a link model which can take into account the difference between holding time distributions of different telecommunication services must be sought. In Annexe I, we propose such a link model using a closed queueing network.

In [LIN'78] it is shown that the routing scheme for an origin-destination (O-D) node pair is completely described by an augmented route tree from which a path-loss sequence can be generated. The latter specifies the alternative paths, the order in which they are successively attempted and the conditions under which a call is blocked. The augmented route tree concept constitutes an efficient and unified method for circuit-switched network characterization. It also assists considerably the path usage modeling. Nevertheless, the calculation envolved in the path usage modeling remains untractable and must relie on algorithms similar to those which have been developped for the terminal-pair reliability problem. These algorithms are unfortunately not very simple to implement. A breakthrough has been provided in [CHAN'80] where it is suggested that recursive programming technique could be used. Inspired from this suggestion, we propose in Annexe I a very simple recursive algorithm to calculate path usage probabilities and node-to-node blocking probabilities.

Among various routing schemes which have been proposed and studied, the successive office routing control (SOC) strategy is perhaps the most adopted, especially in North American telephone networks. Given the complete sequential routing policies in the SOC, we derive in Annexe I

a very simple analytic formula for calculating path usage probabilities and end-to-end grades of service in networks with SOC. To our knowledge, it is the first time that exact formulas are available for this very important class of networks.

. A PL/1 Program for Traffic Analysis in Circuit-Switched networks (Annexe II).

The theory developed in Annexe I is implemented in a PL/1 computer program documented in Annexe II. The computer program is applicable to circuit-switched networks with SOC, OOC (Originating Office Control) or OOC with spill-forward (see our previous report [FIN'80$_a$]for the definition of OOC).

The computer program requires as input:

- Traffic Intensity Matrix: This is the table which specifies, in ccs or erlangs, the offered traffic for all origin-destination pairs.

- Trunk Group Capacity: For each trunk group, the number of trunks must be given. This information is given in a table.

- Routing Table: This information is also given in a table.

The computer output includes:

- End-to-end grade-of-service (blocking probability) for all origin-destination pairs.

- Link blocking probabilities for all trunk groups in the networks.

- Link offered traffic for all trunk groups.

- Link carried traffic for all trunk groups.

- Network grade-of-service: this is the probability that an arbitrary call arriving to the network is blocked.

Our computer program is useful for traffic management in any circuit switched network. It can identify bad grades-of service for certain origin-destination node pairs. It can also identify bottlenecks in trunk groups.

The computer program has been tested with simple

and complex telephone networks. The test results and computer run time for various tests are given in Annexe II. Comparisons between the results obtained by the computer program and the simulator (to be discussed shortly) are also given in Annexe II.

. A Simulator for Telephone Networks (Annexe III)

The theory of traffic engineering that we have developed is based on two important assumptions:

1. Each trunk group is offered a traffic having Poisson arrival.

2. The occupancies of links are statistically independent.

Assumption 1 is not realistic becuase overflow traffic is known not to be Poisson. However, when the network is complex each trunk group is offered many streams of overflow traffic, the superposition of which may become Poisson. Thus, for small netwroks, Assumption 1 may be violated.

Assumption 2 is also not realistic because of the

routing strategy used. Again, when the network is sufficiently large, Assumption 2 may be acceptable.

We have built a digital simulator for telephone networks with a twofold objective. Firstly, we wish to validate our theory developed in Annexe I by simulation. In particular for small and medium-sized networks, we wish to assess the accuracy of our theory. Secondly, the simulator will be useful for network management in which accuracy is important and the transient phase is of interest. It must be pointed out that simulation is very costly and time consuming so that analytical theory should be used when possible.

The input and ouptut of the simulator are the same as for the computer program discussed in the last section. The simulator is of the event-driven type and is written in the SIMSCRIPT language. Documentation and the user guide for the simulator are presented in Annexe III.

4.2 Validation of the Analytical Method by Simulation

We have made a number of runs of our computer program and simulator with two networks. The first network is an abstraction of a telephone network segment provided

to us by Bell Canada, which we will refer to as the BELL network. The original network provided by Bell Canada has seventeen nodes of which sixteen nodes are class-5 offices and one node is a class-4 office. All nodes are connected by high usage trunk groups so that the network is fully connected. The sixteen class-5 offices, we retain only ten and omit six which have very small originating traffic. Thus, our telephone network BELL is a eleven node network. The second network is the European AUTOVON which we have discussed at length in last year's report [FINL'80a] to which the reader referred for a detailed description. The AUTOVON adopts the OCC with spill-forward routing strategy but we have changed it to SOC. Thus, the routing in AUTOVON is much more complexe than in BELL. BELL is therefore a samll network while AUTOVON is a medium-size network.

For each network, about ten computer runs of the simulator (SIMRET) and the computer program (ETARET) were made. The first run was made with the original traffic of the network. In the subsequent runs, traffic in the network is increased or decreased uniformly by a factor F, i.e., each element in the traffic intensity matrix is multiplied by the same factor F (F = 1 for the first run; F can be inferior to 1).

The results of these runs are shown in Figure 4.2 It can be observed that the network grades-of-service given by the computer program (analytical method) and those given by the simulator are very close. The agreement between our theory and the simulation is indeed very good. Thus, for traffic engineering purposes, our computer program (ETARET) can be used.

4.3 Prototype of Local Telephone Network

We have argued in our previous report [FINL'80] that videotex traffic would affect the telephone network significantly primarily at the local areal level. This can be seen from our discussion of the future videotex network (see Figure 3.6) where we indicate that user terminals are concentrated at the videotex centers (VC) The connection between user terminals and VC will be made through the telephone network. Since a VC, in a distributed videotex network, serves mainly a local area, the impact of the videotex service on the telephone network will be mainly at the local level. Long distance calls to a VC would be negligible in quantity with respect to local calls.

To evaluate the impact of the videotex service upon the entire telephone network, we should have an idea

TABLE I

Class 5 Office Details

Office	1979 Working Lines	Installed Lines SXS	XBAR	SP-1
01	54370	18291	15542	21950
02	32892	0	18748	17307
03	1450	1811	0	0
04	8948	9641	0	0
05	4467	0	0	5038
06	12756	0	0	13687
07	31940	16655	16656	0
08	4452	5474	0	0
09	43517	27796	17270	0
10	1465	1696	0	0
11	24876	0	26474	0
12	2115	2613	0	0
13	2996	3380	0	0
14	5426	6348	0	0
15	2018	2438	0	0
16	3010	0	0	3600

Note: The above data and that of Tables II and III were graciously provided to the authors by Bell Canada. They represent data for a "typical" local network (see text).

Table II

Geographic Distribution of Offices

Distance Matrix in kilofeet

Office

Office	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
01	0.0	15.8	87.6	26.9	51.2	68.6	50.7	45.4	13.7	45.9	39.6	59.1	74.4	105.0	154.1	42.2
02	15.8	0.0	103.4	42.7	67.0	63.3	45.4	61.2	29.5	61.7	23.8	63.3	58.6	89.2	138.3	45.4
03	87.6	103.4	0.0	60.7	85.0	156.2	138.3	133.0	101.3	143.3	127.2	166.7	162.0	192.6	241.7	129.8
04	26.9	42.7	60.7	0.0	24.3	95.5	77.6	72.3	40.6	72.8	66.5	86.0	101.3	131.9	181.0	69.1
05	51.2	67.0	85.0	24.3	0.0	119.8	101.9	96.6	64.9	97.1	90.8	110.3	125.6	156.2	205.3	93.4
06	68.6	63.3	156.2	95.5	119.8	0.0	17.9	54.3	82.3	114.5	87.1	125.6	121.9	152.5	201.6	108.7
07	50.7	45.4	138.3	77.6	101.9	17.9	0.0	36.4	64.4	96.6	69.2	107.7	104.0	134.6	183.7	90.8
08	45.4	61.2	133.0	72.3	96.6	54.3	36.4	0.0	59.1	91.3	85.0	104.5	119.8	150.4	199.5	87.6
09	13.7	29.5	101.3	40.6	64.9	82.3	64.4	59.1	0.0	32.2	50.1	45.4	80.2	110.8	159.9	28.5
10	45.9	61.7	143.5	72.8	97.1	114.5	96.6	91.3	32.2	0.0	82.3	77.6	112.4	143.0	192.1	60.7
11	39.6	23.8	127.2	66.5	90.8	87.1	69.2	85.0	50.1	82.3	0.0	38.2	34.8	65.4	114.5	21.6
12	59.1	62.3	166.7	86.0	110.3	125.6	107.7	104.5	45.4	77.6	38.2	0.0	73.3	103.9	153.0	16.9
13	74.4	58.6	162.0	101.3	125.6	121.9	104.0	119.8	80.2	112.4	34.8	73.3	0.0	30.6	79.7	56.4
14	105.0	89.2	192.6	131.9	156.2	152.5	134.6	150.4	110.8	143.0	65.4	103.9	30.6	0.0	49.1	87.0
15	154.1	138.3	241.7	181.0	205.3	201.6	183.7	199.5	159.9	192.1	114.5	153.0	79.7	49.1	0.0	136.1
16	42.2	45.4	129.8	69.1	93.4	108.7	90.8	87.6	28.5	60.7	21.6	16.9	56.4	87.0	136.1	0.0

Note: All offices are class-5, except number 17, which is a class-4 co-located with office number 1.

Table II

Geographic Distribution of Offices

Distance Matrix in kilofeet

Office	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
01	0.0	15.8	87.6	26.9	51.2	68.6	50.7	45.4	13.7	45.9	39.6	59.1	74.4	105.0	154.1	42.2
02	15.8	0.0	103.4	42.7	67.0	63.3	45.4	61.2	29.5	61.7	23.8	63.3	58.6	89.2	138.3	45.4
03	87.6	103.4	0.0	60.7	85.0	156.2	138.3	133.0	101.3	143.3	127.2	166.7	162.0	192.6	241.7	129.8
04	26.9	42.7	60.7	0.0	24.3	95.5	77.6	72.3	40.6	72.8	66.5	86.0	101.3	131.9	181.0	69.1
05	51.2	67.0	85.0	24.3	0.0	119.8	101.9	96.6	64.9	97.1	90.8	110.3	125.6	156.2	205.3	93.4
06	68.6	63.3	156.2	95.5	119.8	0.0	17.9	54.3	82.3	114.5	87.1	125.6	121.9	152.5	201.6	108.7
07	50.7	45.4	138.3	77.6	101.9	17.9	0.0	36.4	64.4	96.6	69.2	107.7	104.0	134.6	183.7	90.8
08	45.4	61.2	133.0	72.3	96.6	54.3	36.4	0.0	59.1	91.3	85.0	104.5	119.8	150.4	199.5	87.6
09	13.7	29.5	101.3	40.6	64.9	82.3	64.4	59.1	0.0	32.2	50.1	45.4	80.2	110.8	159.9	28.5
10	45.9	61.7	143.5	72.8	97.1	114.5	96.6	91.3	32.2	0.0	82.3	77.6	112.4	143.0	192.1	60.7
11	39.6	23.8	127.2	66.5	90.8	87.1	69.2	85.0	50.1	82.3	0.0	38.2	34.8	65.4	114.5	21.6
12	59.1	62.3	166.7	86.0	110.3	125.6	107.7	104.5	45.4	77.6	38.2	0.0	73.3	103.9	153.0	16.9
13	74.4	58.6	162.0	101.3	125.6	121.9	104.0	119.8	80.2	112.4	34.8	73.3	0.0	30.6	79.7	56.4
14	105.0	89.2	192.6	131.9	156.2	152.5	134.6	150.4	110.8	143.0	65.4	103.9	30.6	0.0	49.1	87.0
15	154.1	138.3	241.7	181.0	205.3	201.6	183.7	199.5	159.9	192.1	114.5	153.0	79.7	49.1	0.0	136.1
16	42.2	45.4	129.8	69.1	93.4	108.7	90.8	87.6	28.5	60.7	21.6	16.9	56.4	87.0	136.1	0.0

Note: All offices are class-5, except number 17, which is a class-4 co-located with office number 1.

Table IV

Traffic Intensity Matrix for BELL (see text)

To / From	1	2	3	4	5	6	7	8	9	10
1	-	11956	2848	1328	2238	5844	879	12578	3322	11726
2	10833	-	717	420	1007	2534	331	6851	5614	3803
3	2623	661	-	375	-	-	355	1302	231	1588
4	1354	359	368	-	0	0	0	821	137	819
5	1800	860	0	0	-	3743	238	449	364	2759
6	5441	2407	0	0	3962	-	948	2430	910	7081
7	992	332	349	0	376	848	-	430	0	879
8	12063	6401	1058	915	576	2269	373	-	3699	3879
9	5325	5595	272	162	426	918	88	3885	-	3727
10	11837	3545	1182	1062	2526	6007	637	3478	3482	-

Note: This matrix was obtained from that of Table III by assuming 2% link blocking probability.

Table IV

Traffic Intensity Matrix for BELL (see text)

From \ To	1	2	3	4	5	6	7	8	9	10
1	-	11956	2848	1328	2238	5844	879	12578	3322	11726
2	10833	-	717	420	1007	2534	331	6851	5614	3803
3	2623	661	-	375	-	-	355	1302	231	1588
4	1354	359	368	-	0	0	0	821	137	819
5	1800	860	0	0	-	3743	238	449	364	2759
6	5441	2407	0	0	3962	-	948	2430	910	7081
7	992	332	349	0	376	848	-	430	0	879
8	12063	6401	1058	915	576	2269	373	-	3699	3879
9	5325	5595	272	162	426	918	88	3885	-	3727
10	11837	3545	1182	1062	2526	6007	637	3478	3482	-

Note: This matrix was obtained from that of Table III by assuming 2% link blocking probability.

The data on trunk group capacity were not given by Bell Canada. The number of trunks in a trunk group must be derived from Erlang's formula assuming a typical trunk blocking probability of 2% in the busy hour. This was suggested by Bell Canada. Table V shows the number of trunks obtained in this manner.

Our model of a local telephone network corresponds to that of a typical urban area which might be served by a videotex center. In the next section, we shall study the impact of videotex service on this model of a local telephone network.

Using our computer program ETARET (See Annexe II) and the data in Tables IV and V, we obtain a value of 0.02 for network grade-of-service (blocking probability) for our prorotype of a local telephone network. This grade-of-service agrees with current grade-of-service offered by all North-American telephone networks.

To study the sensitivity of our network model with respect to increase in traffic, we must calculate, with the aid of our computer program, the increase in network blocking probability as function of the increase in network traffic. Figures 4.2 and 4.3 show the network blocking

Table V

Number of Trunks in BELL (see text)

To / From	1	2	3	4	5	6	7	8	9	10
1	-	546	135	56	91	265	39	602	205	616
2	546	-	35	17	45	131	14	313	292	208
3	135	35	-	22	0	0	22	63	13	90
4	56	17	22	-	0	0	0	47	7	64
5	91	45	0	0	-	187	15	21	14	154
6	265	131	0	0	187	-	46	98	45	352
7	39	14	22	0	15	46	-	16	2	50
8	602	313	63	47	25	98	16	-	186	208
9	205	292	13	7	14	45	2	186	-	204
10	616	208	90	64	154	352	50	208	204	-

Note: Calculated using Erlang's formula assuming 2% blocking probability on the links.

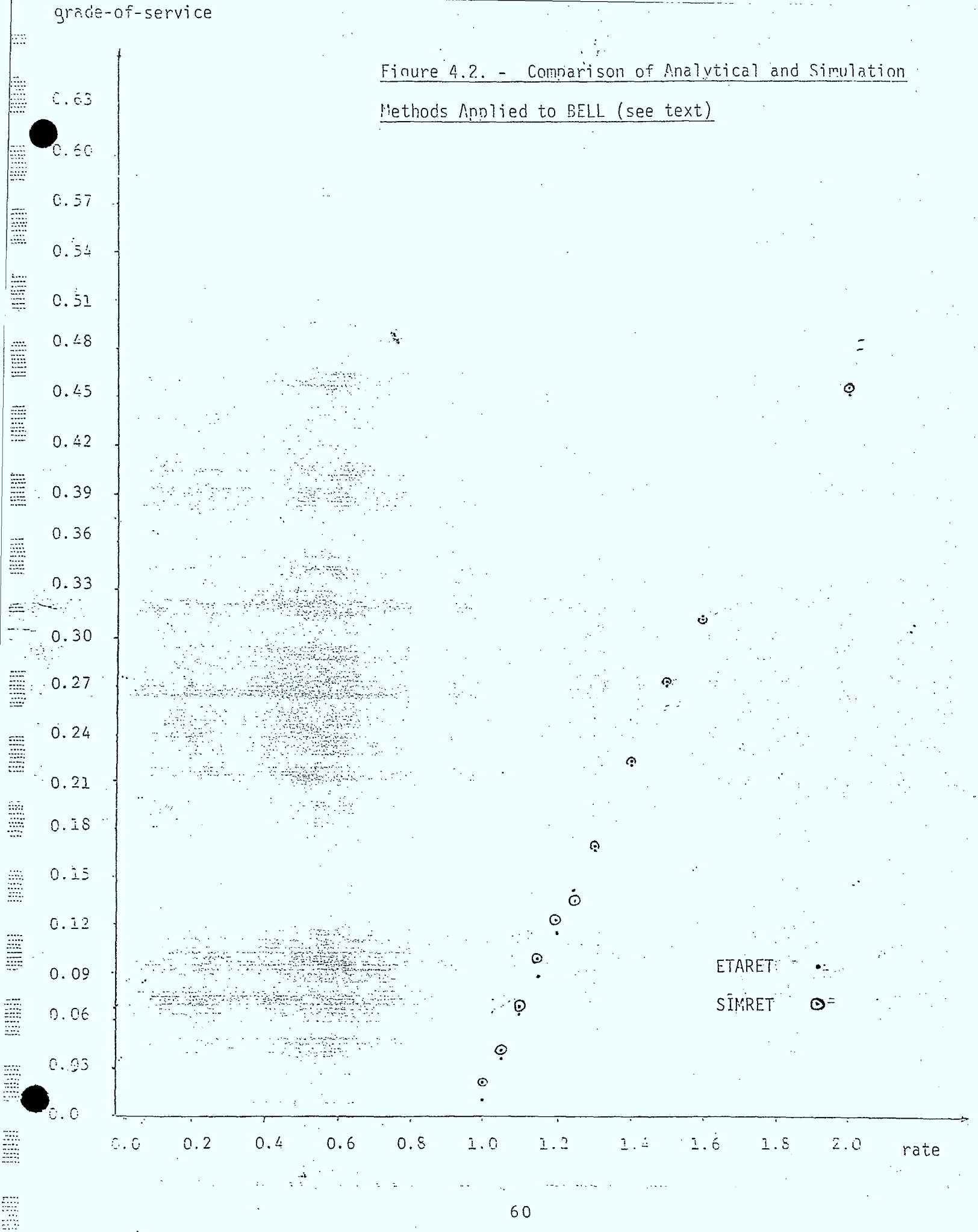


Figure 4.2. - Comparison of Analytical and Simulation Methods Applied to BELL (see text)

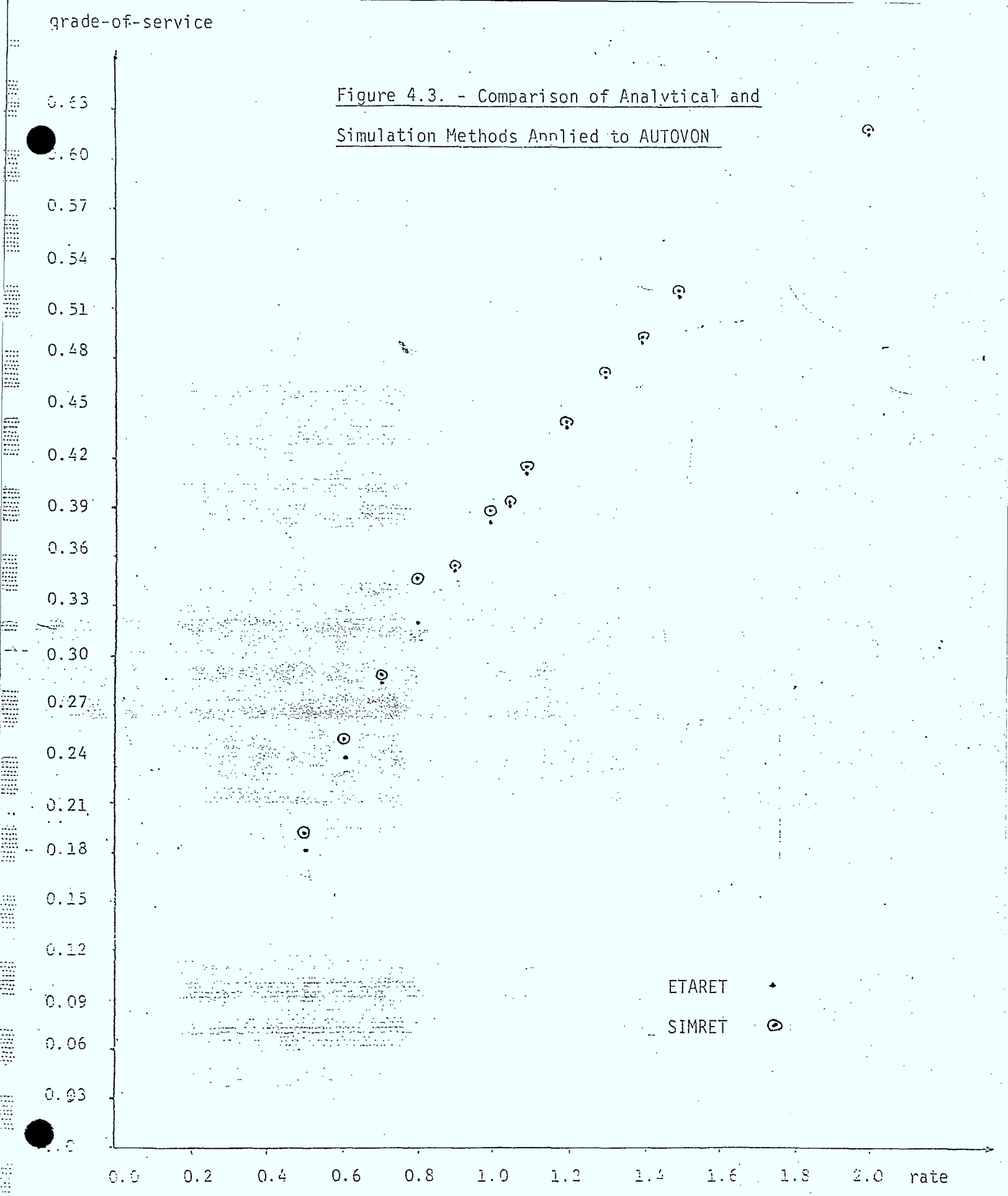


Figure 4.3. - Comparison of Analytical and Simulation Methods Applied to AUTOVON

probability as a function of increase in network traffic. It can be seen that although the current network grade-of-service is excellent, the network is very sensitive to traffic increase. For example, an increase of 25% in the traffic entails an increase in network blocking probability to 0.1 and the network blocking probability is about .25 when traffic is increased by 50%. These are, of course, draconian overall increased and merely serve as illustrations. Finer turning showing the precise effect of videotex traffic will be given.

4.4 Results of Analysis

Having obtained a prototype of local telephone network, we now try to assess the impact of the videotex service on this particular model for a local telephone network. This task is rendered difficult on two accounts: First, the traffic pattern generated by videotex service in not yet determined.

The first difficulty is partly reduced by one of our theoretical findings, namely, that the exact form of the distribution of videotex user holding times is not relevant, only the traffic intensity as measured in erlangs or CCS is really needed (see our link model in Annexe I).

Thus, videotex traffic should have the same qualitative impact as telephone traffic on the network. Quantitatively, little is known about videotex traffic. During field trials of the Prestel system in England, it has been observed that the average videotex user holding time is about 5 minutes. If we suppose, as it has been observed in the telephone service, a videotex user makes on the average 2 inquiries per busy hour then videotex traffic per user is 2 X (5/60) = .166 erlang or 6 CCS. Suppose also that a videotex center be designed to serve about 2000 users, then the videotex traffic per videotex center is about 12,000 CCS. Theses numbers might serve as a guideline for the design of videotex network. In the following we will study the impact of videotex traffic on the local telephone network in a large range, rather than at a fixed level, of videotex traffic. The aim of the study is to estimate the capacity of telephone network in order to assess the feasibility of implementing within it a videotex service.

The second difficulty is also partly reduced by our discussion of videotex network in 4.2 where we have argued that user terminals are concentrated at the videotex centers which are the outermost nodes in the videotex network. Connections between the videotex centers and user terminals are made vis the telephone network. Hence, a Videotex center must be located at a telephone network node

where the local subscriber loops are concentrated. The obvious location for a videotex center is to make it appear as an isolated class-5 office. This is the approach we adopt here.

Thus, we suppose that the videotex center is a class-5 office co-located with the class-4 office in our local area network model. Every videotex call has to be routed to this videotex center via the class-4 office through the local telephone network. Between the videotex center and the class-4 office is a fictitious trunk group with an infinite number of trunks, i.e. , we assume no blocking at the videotex center.

Figures 4.4 and 4.5 show the capacity of our local netwrok vis-à-vis the videotex service. Figure 4.4 shows the network blocking probability as seen by an arbitrary call while Figure 4.5 shows the network blocking probability as seen by videotex calls. The significance of these two figures is presented in the following section (4.5).

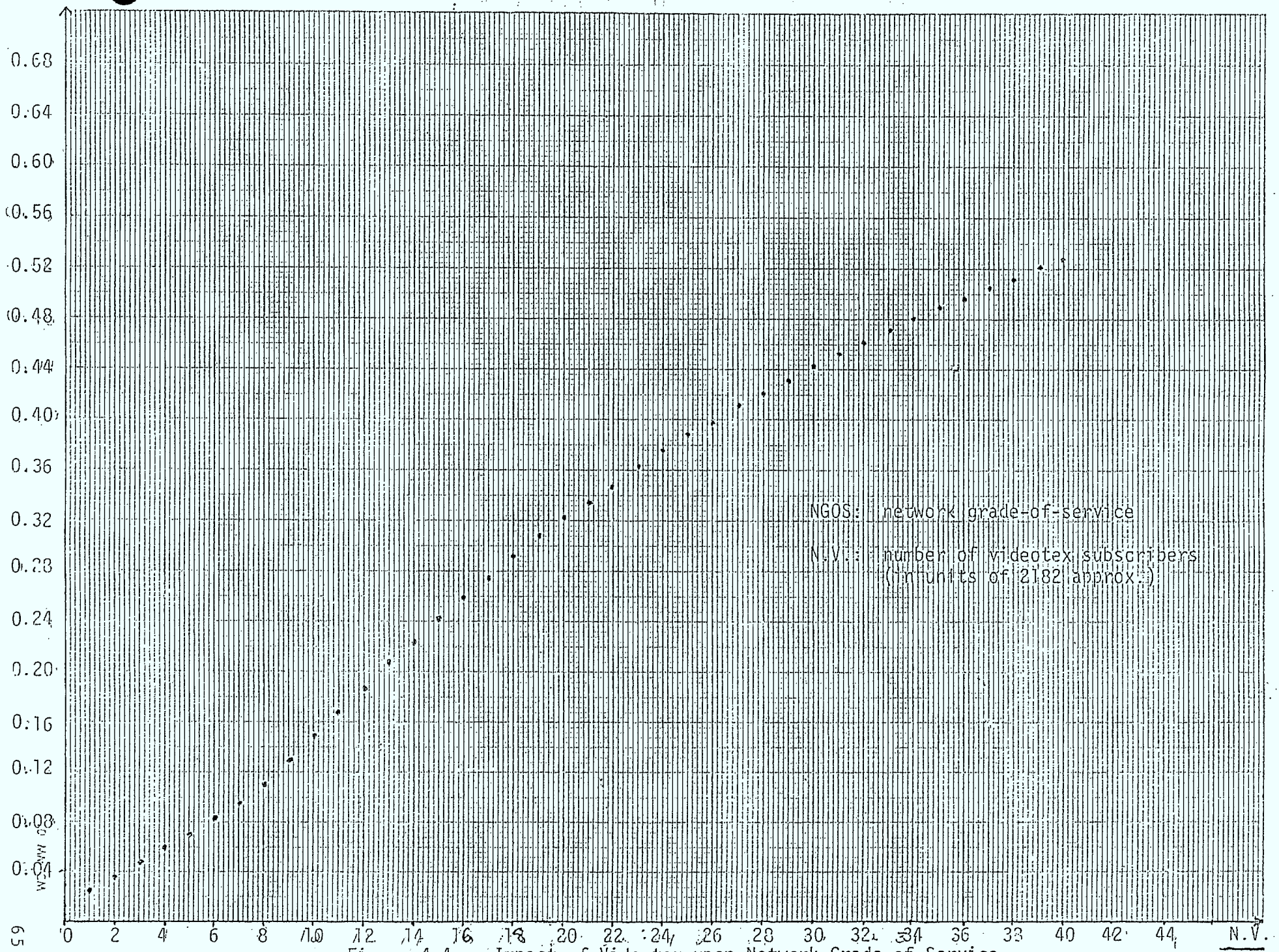


Figure 4.4. - Impact of Videotex upon Network Grade-of-Service

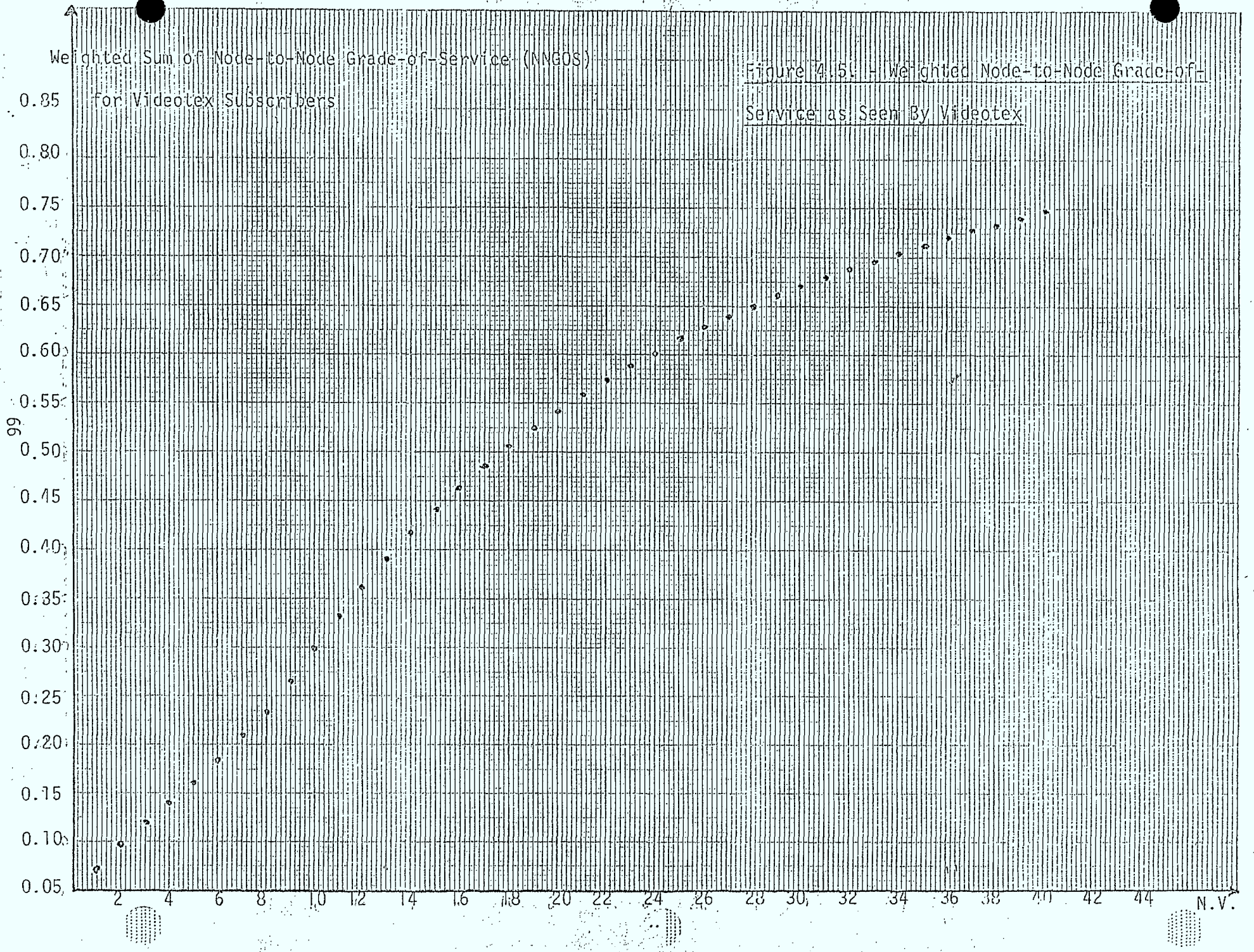


Figure 4.5. - Weighted Node-to-Node Grade-of-Service as Seen By Videotex

4.5 The Impact of Videotex upon an Idealized Local Telephone Network.

Recall that our hypothetical local telephone network, BELL, was a simplification (and an interpretation) of data kindly furnished to us by Bell Canada and that our hypothesis about videotex was based upon information obtained from the Prestel experience. Thus, while we have evidence that our models (analytical and simulation) are correct, the results obtained using these models are no better than the data upon which they must operate. With this caveat in mind, we now proceed to discuss the rather impressive implications of Figures 4.4, and 4.5, in particular 4.4

Figure 4.4 indicates the increase in the average network grade-of-service (NGOS) as a function of increase in videotex subscribers, or approximately K percent of the total subscriber population ($2,182 \div 218,218 \tilde{=} 1\%$). For our network BELL one notices a dramatic increase in the NGOS: for K=1, the NGOS is about 0.02, for K=2 it is about 0.0^4, and for K= 8, it is about 0.10, etc. More bluntly, an increase in videotex traffic to include 10% of the subscriber population brings the NGOS to 0.15, 15% this becomes 0.24!

In the case of BELL, the congestion points are clear: the trunks relating nodes (class-5) 1 to 9 to node 10 (class-4), thence to the videotex node, 11, become saturated. Presumably, increasing their capacity would bring the NGOS down to acceptable values for greater videotex traffec.

Figure 4.5 indicates the sum of the weighted node-to-node grades-of-service (NNGOS) from the class-5 nodes to node 11, hence the increase in the average NNGOS due to videotex. Again, one remarks the dramatic increase in this quantity with relatively small percentage increase in videotex usage. Indeed, at 2% the effective NGOS seen by videotex is almost 0.1, at 5% it to superior to 0.15!

Thus, our results show that the local subscriber loop network might be extremely fragile as far as tolerating videotex when more than 5% of the subscribers usethis service.

5. TRENDS IN THE LOCAL SUBSCRIBER NETWORKS

5.1 Introduction

In the preceeding chapter, we have seen that the local subscriber loop network, as represented by our highly simplified model BELL, may be extremely sensitive to the additional traffic generated by new services such as videotex. Assuming no blocking at the switches, the culpret is easily identified, namely, the trunks connecting the class-5 offices to the class-4 "hub" to which the videotex center is in turn itself connected as a special kind of class-5 office. Thus, raising the capacity of these trucks and, correspondingly, that of the class-4 hub switch, presumably would alleviate the problem, i.e., make the NGOS less sensitive to increases in videotex traffic. There remains, however, the question of the capacity of the class-5 switches and subscriber loops themselves. As the demand for more band-width consuming new services grows, the local subscriber loop network will need to be revamped.

Thus nowhere are the effects of the current telecommunications revolution more promissing and more far-reaching than in the architecture and design of that part of any telecommunications network that deals with the distribution to and from the subscriber's premisses of information: i.e., the local subscriber loop

plant. Technological innovations such as microwave, satellites, LSI and VLSI, and all digital optical fiber trucks have had and will continue to have an increasing impact upon long-haul communications wherein many information channels have been multiplexed by any one of a growing multitude of techniques into larger channel groups which are then transmitted over a single physical channel. Telephony, television and cable television have been direct beneficiaries of these innovations and, in the North American context at least, there has been a noticeable increase in the quality and the range of services offered.

The local distribution plant, however, remains a different story. The investment in the current copper wire plant, both in telephony as well as in cable television, is so great that serious economic arguments arise when one attempts to introduce the latest technological innovations at this level. Moreover, even if this investment were to be instantaneously amortized by some means or another, there remains the non-trivial question of exactly how subscribers would use the awesome capacity for broad and narrow-band information handling that would be theirs. Finally, even if we answer this question by the creation of various cost-effective (read "profitable") new services, there remain many practical technological developmental questions concerning such things as the design and production of home and office terminals, network interfaces, both hardware

and software, and many others.

In this section, a brief summary is given of recent developments in the local distribution plant. While the main subject of this discussion is the local telephone network, in line with the major thrust of this work, as we have seen in the brief review of videotex systems given in Chapter 2, one can no longer exclude cable television distribution networks, local computer networks, and even local satellite ("roof-top") applications.

In network planning, in particular in the determination of a coherent holistic national program for the development of telecommunications services, trends such as the ones mentioned in the following pages must be carefully watched. Policies concerning, among others, the one versus two-wire access to subscribers premisses, access to one network from another, "gateways" depend upon these trends as well as those concerning the guarantee of the delivery of conventional services to which the public has become accustomed. At a more elementary level, network planning tools, such as those developped by the authors, must be modified in order to conduct the appropriate impact studies. Finally, in a field which is undergoing such truly fundamental changes - one suspects that the telecommunications revolution has but begun - a synthesis of current trends is

essential as a planning aid. As a result of our current impact studies, the authors foresee the possibility of suggesting modifications to the current local plant in order that various proposed new services be better accomodated.

5.2 Towards the Integrated Services Digital Telephone Network (ISDN)

In a recent article, Habara and Aratani [HABA'80] suggest a three phase development of a local integrated services digital network (ISDN) in the Japanese context. In Japan, unlike Canada, the Nippon Telegraph and Telephone Public Corporation (NTT) is the only organization solely charged with the responsibility to furnish domestic telecommunication services. However, as far as the current Canadian telephone networks are concerned, these two authors have presented a developmental strategy that seems to capture current North American trends as well (see [CHAN'80b], [BANK'80], [MILL'80]). Indeed, at least one senior executive at Canada's largest telephone carrier forsees a similar trend in Eastern Canada [TERR'80].

The three stages proposed by NTT are the following (see Table VI):

1. Digitalization at the local switch (class 5 office)

TABLE VI

Three Possible Stages for the Development of a Local ISDN

(adapted from Habara and Aratani, HABA'80)

	Local Network Configuration	Information Sources	Subscriber Loop Transmission Medium	Local Network Construction
First Stage	Essentially use existing copper-wire plant, but introduce some digital links; switch becomes all-digital	Telephone (still analog) and new services (up to 64 kbits/sec)	Existing subscriber copper-wire cable	Digitalize local switch and line concentrators. Analog telephone or 64 kbits/sec new services
Second Stage	As above, but introduce optical fiber feeder cables where appropriate (new links, to replace old or bulky wire, etc.). Digitalize telephone.	Digital telephone (64 kbits/sec) and new services (up to 64 kbits/sec)	As above with optical fiber cable for feeders	ISDN with up to 2x64 kbits/sec services. Distributive routing function before trunk network
Third Stage	As above, but introduce large bandwidth optical fiber cable to subscriber's terminating equipment (a "second wire") Introduce space-division switcher at central office to handle broadband (video) switching, time-division switch remains as in previous stages.	As above, but with new services up to about 100 Mbits/sec	As above, with more emphasis placed on replacing wire cable with optical fiber cable	ISDN with 2x64 kbits/sec servic and a true broad band (100 Mbits/ sec) service. Us wave-length mult plexing and dis- tributive routin functions at cen tral office ter- minating equipme

mixed analog and digital subscriber cables. This will be realized by the mid-1980's.

2. Digitalization of all subscriber cables; use of pair gain equipment or optical fibers. This will be done by the late 1980's and will serve to establish a basis for economical introduction of non-telephone services.

3. Introduction of optical fiber cable and appropriate switching facilities for broadband communications. The trunks will have to be upgraded correspondingly. This phase is foreseen for the mid-1980's.

Various network architectures are considered: star (the underlying hierarchical architecture underlying most telephone systems), ring, and the web (see Table VII). A star structure may be preferable in phase I and II.

Finally, hierarchical levels in an ISDN are illustrated in Figure 5.1.

5.3 The Cable Television Subscriber Plant

At this writing, Télécâble-Vidéotron of Montreal

Table VII

Comparison of Three Local Network Configurations

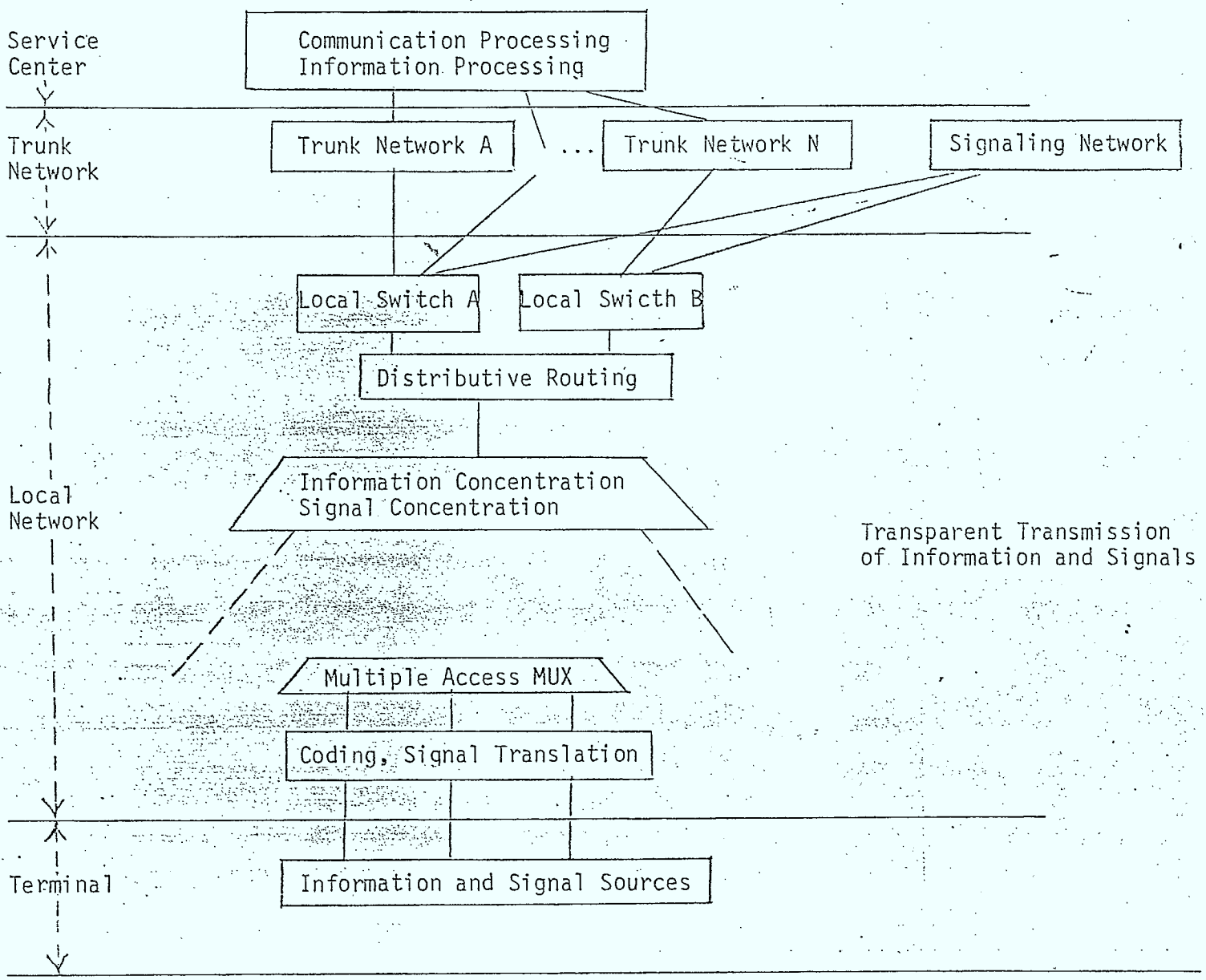
(adapted from Habara and Aratani [HABA'80])

Network Configuration*	Advantages	Disadvantages
Star network: Local switch at hub, feeders to concentrators and to subscriber terminals	-Network design is easy -Errors and faults easily localized -Easily expanded	-Uses links inefficiently
Ring network: Local switch and concentrators arranged in a ring; terminals in star configuration off of concentrators	-High efficiency due to large link capacity and its utilization -Short average circuit length due to intra-ring calls	-Network design not so easy -Expansion is difficult
Mesh: Combination of above with local switch and concentrators both in star and ring configurations		-Network design is difficult -Route handling is complicated

*Please see previous year's report (FINL'80) for definition of these types of networks.

Figure 5.1.- Hierarchical Structure of an ISDN

(adapted from Habara and Aratani HABA'80)



[DUFR'81] is planning the introduction of a new service, namely SID, or Système d'information à domicile (Home Information System). A field trial will be conducted in early 1982. The company foresees three phases, the first introducing teletext, the second so-called "narrow-casting", the third interactive videotext with tele-alarm and telemeasuring capabilities. Interfacing with the telephone companies has been considered.

Such a trend is typical at this time in the cable television industry and is made possible by the development of superior amplifiers permitting extension of cable bandwidth up to 400 MHz and eliminating the ingress noise problems that plagued early two-way cable television. Within, say, a space of fifteen years, cable television networks could offer a form of videotelephony. Thus the one-versus two-wire problem must be dealt with.

5.4 Local Area Networks

The subject of local area networks is currently the object of intensive research and development both in industry as well as in the academic environment. Motivating this intense interest are considerations, backed, to be sure, by economic arguments, such as the following:

- interconnecting a set of homogeneous computers (micro, mini or maxi) and terminal devices at relatively modest costs to permit sharing of distributed resources;

- distributing databases and processing according to local geographic needs;

- offloading overburdened centralized computers;

- automating the office;

- integrating data, video and voice on one local network using one transmission medium.

In all cases, the qualifier "local area" implies that the network extends over a geographically restricted area, often to one or several proximate buildings and not atypically belonging to a single organization.

One definition of local area networks that has been proposed by the IEEE Computer Society reads as follows:

"A Local Area Network is a <u>data communications system</u> [the underlining is the author's] which

allows a number of independent data devices to communicate with each other. A Local Area Area Network is distinguished from other types of data networks in that the communication is usually confined to a moderate sized geographic area such as a single office building, a warehouse, or a campus, and can depend on a physical communication channel of moderate to high data rate which has a consistently low error rate. This is in contrast to long-distance networks which interconnect facilities which may be in different parts of the country, and networks of data devices within a single piece of equipment. ... The applications environment of the Local Area Network is intended to be commercial and light industrial..." [IEEE'80]

Notice that this definition limits itself to data communications. The author wishes to include in the concept of local area network other forms of communication as well, specifically, video and voice. Sometimes local networks with the capacity for these latter are called local (area) broadband networks or integrated local networks.

The IEEE Computer Society definition cited above

attempts to preserve consistency with the International Standards Organization's (ISO) Open Systems Interconnection model [ISO'79, ZIMM'80]. In the latter, a layered approach is given to the characterization of a (computer) network such that processes at a given level communicate with their "peers" at, the same level, passing through the lower levels, including eventually the media or transmission layer, as needed in a way that to them is transparent. This model is summarized in Figure 3.2. The IEEE provisionally proposed model is shown in Figure 5.2 and the correspondance with the ISO model is indicated in Figure 5.3.

The IEEE Project 802 on Local Area Network Standards is currently attempting to standardize protocols, access methods and hardware specifications for the class of networks covered by the definition quoted above [IEEE'81].

Networks covered by that restrictive definition include Ethernet and its relatives [XERO'80], among others. The reader to asked to note that the architecture of a network is specified more by the layered-protocols and access methods rather than by network topology.

One notices that the geographic scope of a local network is generally much more restricted than that of the local

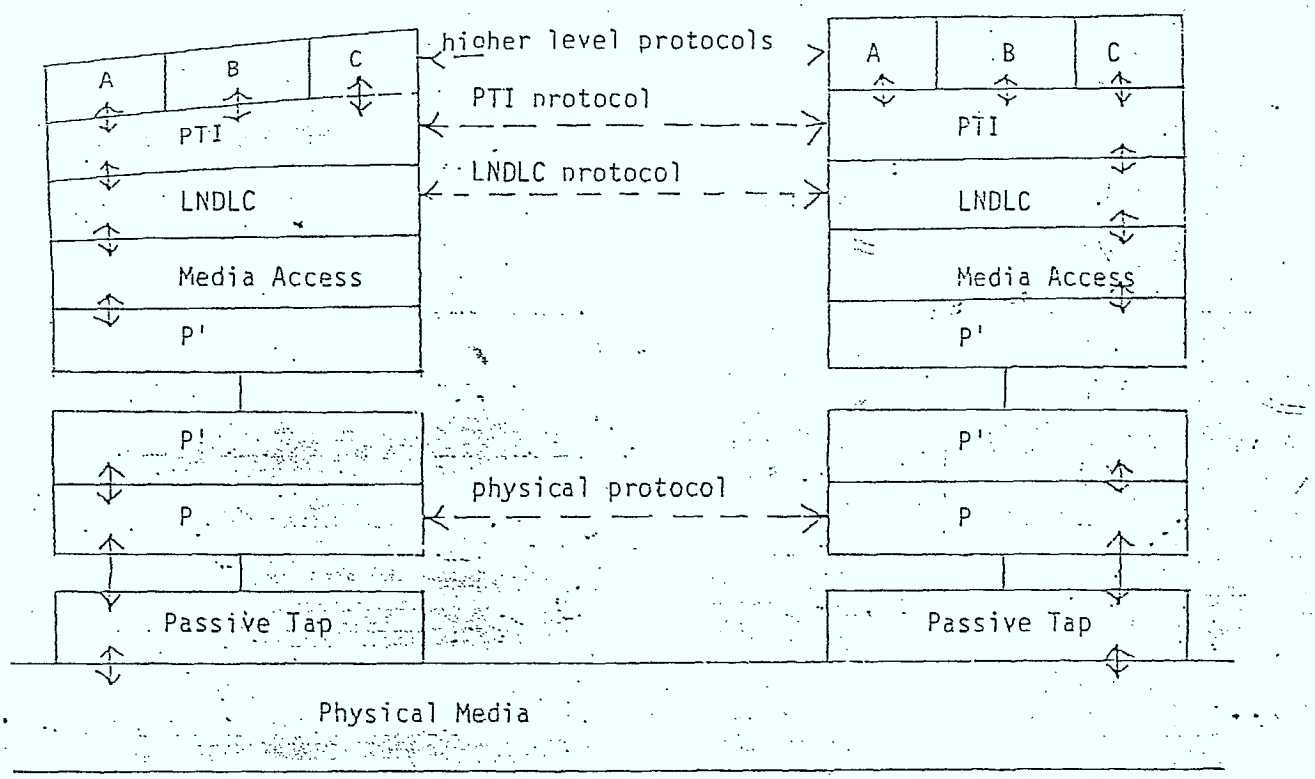


Figure 5-2- IEEE Local Area Network Reference Model

The LNDLC layer is the Local Network Data Link Level, the PTI is the Protocol Type Identification level. The latter serves to identify the protocols for the various higher level services indicated here simply as A, B, and C. The media interface is split into two parts: one is located in the data terminating equipment (DTE), the other may be remote from the DTE. Finally, the tap is by definition passive (the active part being located in P).

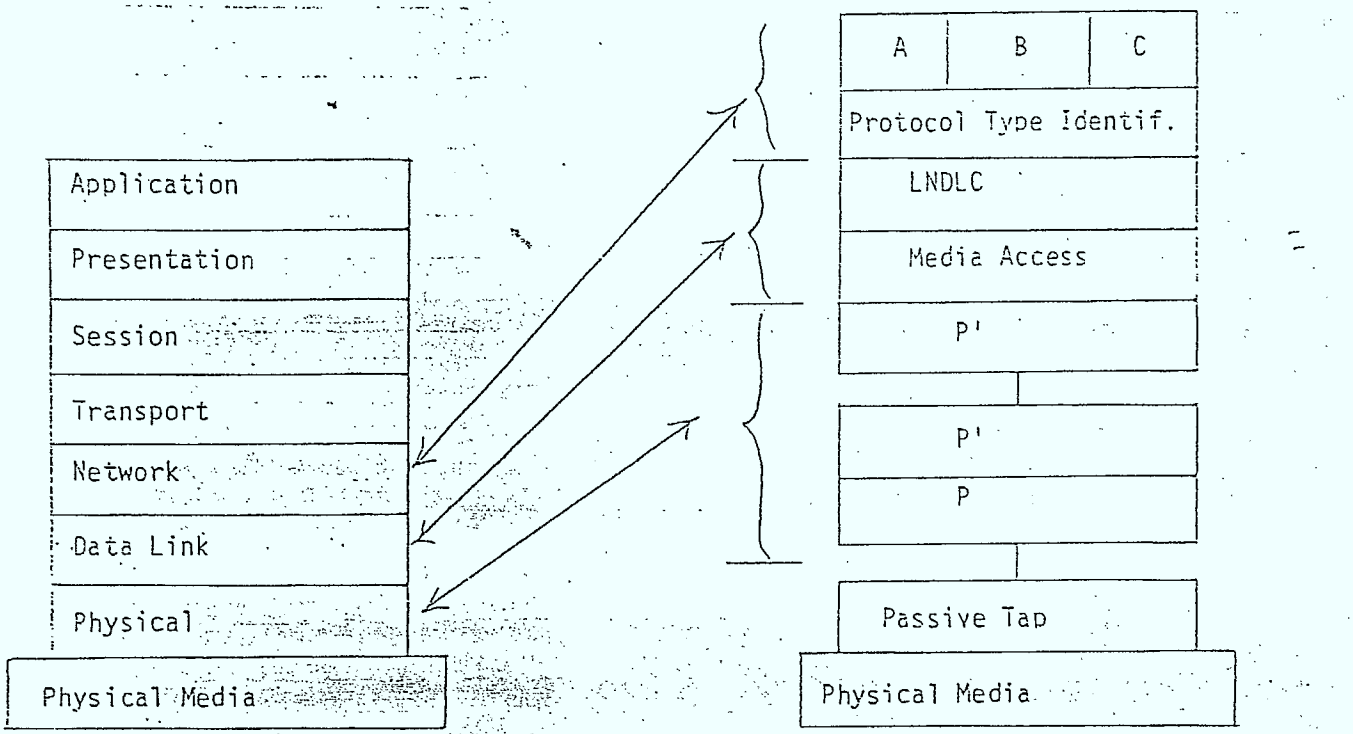


Figure 5.3 Relationship of the ISO's OSI to the IEEE Local Area Network Reference Model

subscriber telephone network. It may, however, have a "gateway" or entry point to, and with time and technological improvements, replace part or all of, the latter. For, integrating telephony into a local area network seems highly feasible in the near future [SHOC'80]. Figure 5.4 illustrates this concept in the case of Ethernet and Figure 5.5 indicates a possible digital Ethernet telephone set.

It is clear, therefore, that local area networks can furnish telephone network-compatible videotex traffic to the telephone network. Moreover, a specialized local area network could become a "videotex machine", i.e. a videotex center with distributed databases, entry points to other networks, etc.

At Laval University, the authors and one other colleague have recently acquired a network functioning on principles similar to Ethernet namely NET/ONE manufactured by Ungermann-Bass [FINL'81a]. They are currently examining just such possibilities as those suggested above. One of the authors is also studying the use of optical fibers in such networks [FINL'81b].

Local area networks represent a major technological development of the 1980's. They will certainly have an effect upon the realization of new services and upon local telephone and cable television network archtectures.

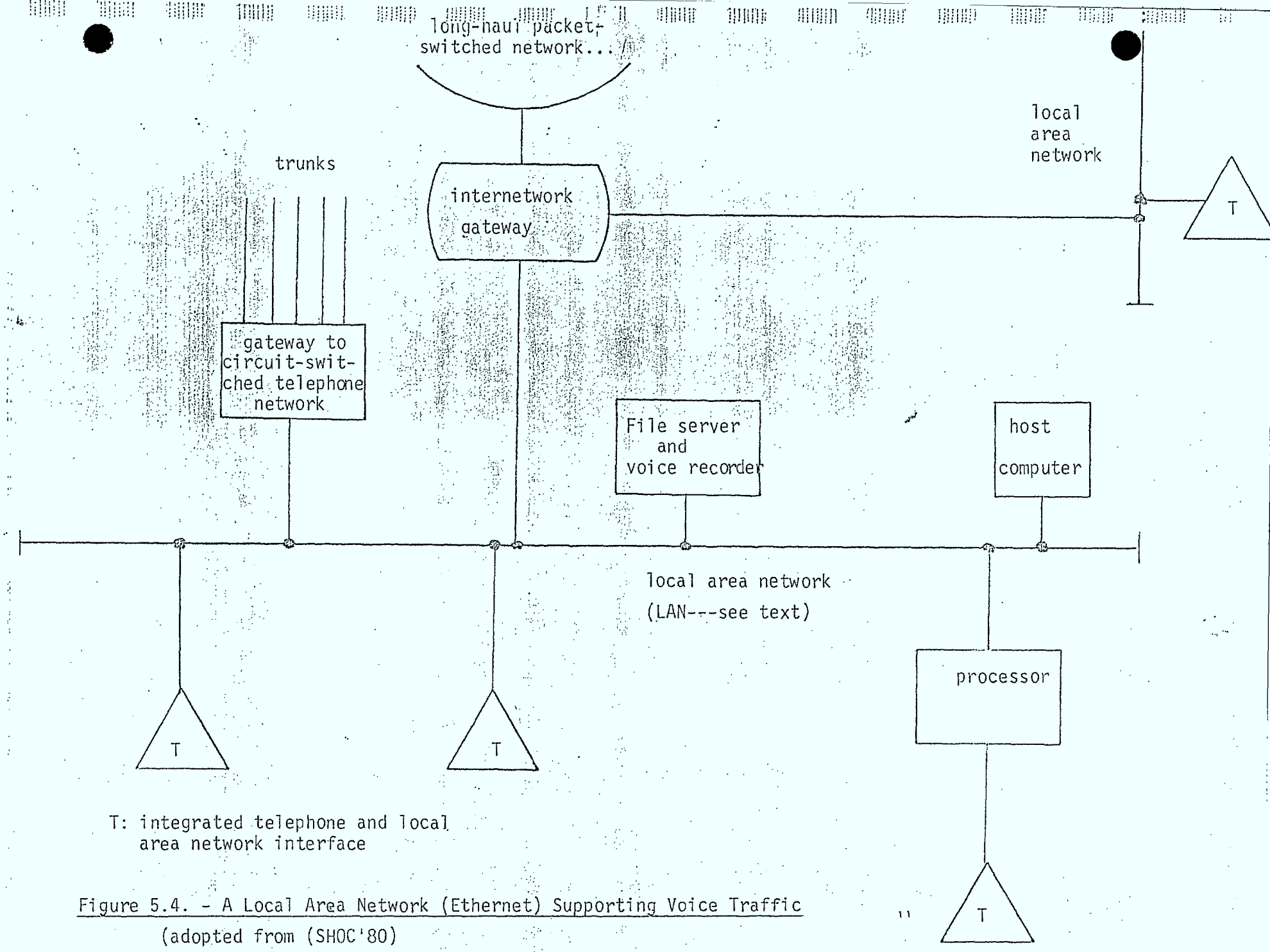


Figure 5.4. - A Local Area Network (Ethernet) Supporting Voice Traffic

(adopted from (SHOC'80)

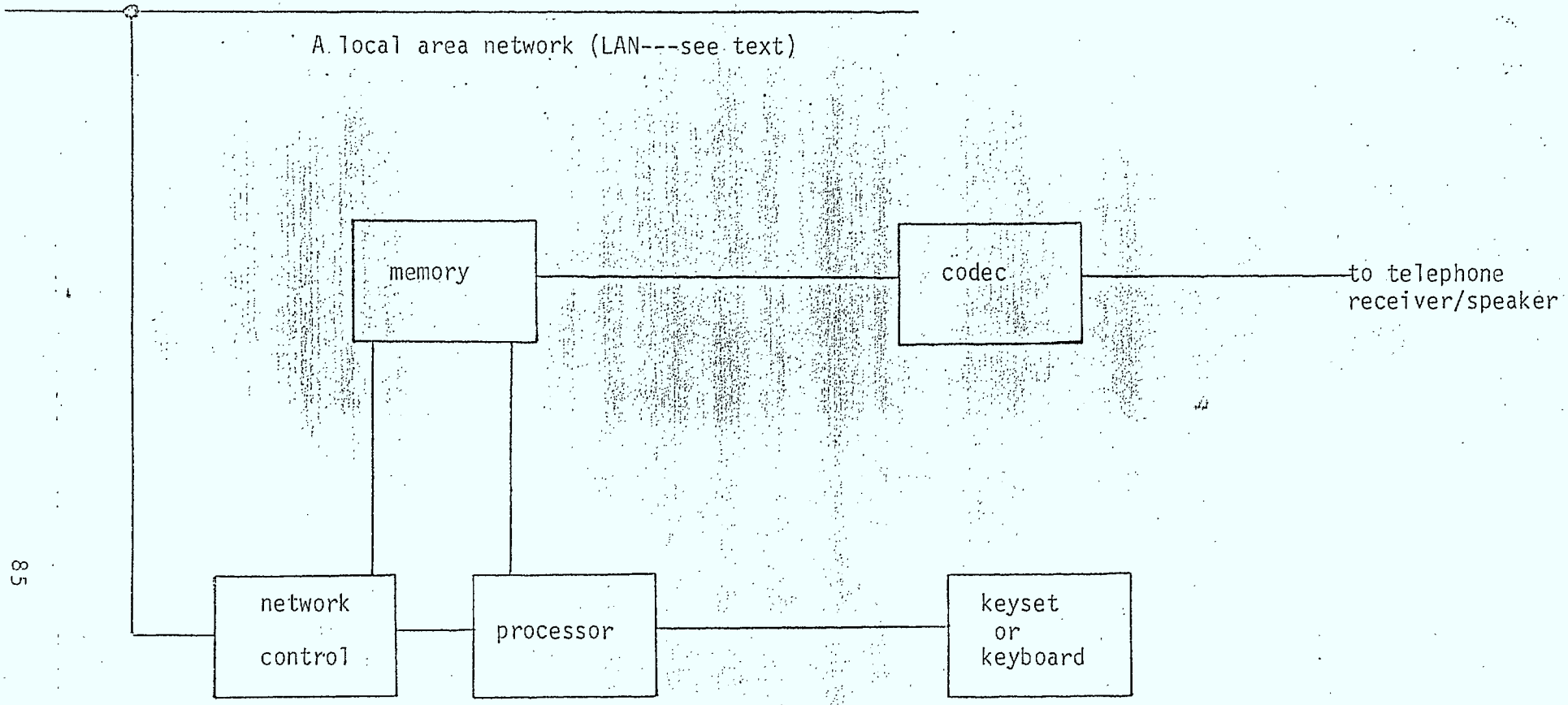


Figure 5.5. - An Integrated Telephone-Local Area Network Interface
(adopted from (SHOC'80)

Bibliography

[AT & T'81a]

American Telephone and Telegraph Company: Videotex Standard: Presentation Level Protocol, American Telephone and Telegraph Company, 5 Wood Hollow Road, Parsippany, New Jersey, May 1981.

----------: Bell System Announces New Videotex Standard, News Release May 20, 1981, AT & T, 195 Broadway, New York, New York.

[BALL'80]

Ball, A.J.J., Bochmann, G.V., and J. Gecsei: Videotex Networks, Computer, Vol. 13, No. 12, December 1980, pp 6-14.

[BANK'80]

Banks, F.M. and R.J. Mills: Evolving the Subscriber Loop Network in the Digital World, Telecom'79, Geneva, Switzerland, 19-21, September 1979.

[BERK'81]

Berkman, S.: from a speach made at Videotex'81, Toronto, 20-22 May 1981, on AT & T's proposed videotex presentation-level protocol.

[BERT'80]

Bertina, H.V.: Physical Level Protocols, IEEE Transactions Communications, Vol. COM-28, No. 4, pp. 433-444, April 1980.

[CARL'80]

Carlson, D.E.: Bit-Oriented Data Link Control Procedures, IEEE Transactions Communications, Vol. COM-28, No. 4, pp. 455-467, April 1980.

[CCIT'80]

CCITT (Comité Consultatif International Télégraphique et Téléphonique): International Information Exchange for Interactive Videotex (Study Group VIII - Contribution No 164), 4 June 1980.

[CHAN'80b]

Chang, K.Y.: Fiberguide Systems in the Subscriber Loop, Proceedings IEEE, special issue on optical communication, Vol. 68, No. 10, pp. 1291-1298, October 1980.

[COST'80]

COSTPRO (Canadian Organization for the Simplification of Trade Procedures): Canadian Trade Information System, Management Overview, Volume I, 151 Sponks Street, Suite 302, Ottawa, June 1980.

[DUFR'81]

Dufrese, M.: SID, Un premier pas vers l'avenir (Système d'information à domicile), Official Technical Records, 24-th Annual Convention, Trade Show and Vidéothèque, 11-14 May 1981, Québec City Canadian Cable Television Association, pp. II-41-47.

[FEDI'79]

Fedida, S. and R. Malik: The Viewdata Revolution, Associated Business Press, Fleet Street, London, 1979.

[FINL'77]

Finley, M.R.: Micro-Processor-Based Information Systems for Small Enterprises - A Pilot Study, Proceeding of the Int'l. Symposium on Mini and Micro-Computers, MIMI'77, Montréal, 11-18 November 1977, pp. 164-168.

[FINL'80a]

Finley, M.R. and T. Vo-Dai: A Technical and Economic Study on the Impact of the Introduction of New Services on Existing Telecommunications Networks, Final Report prepared for the Department of Communications of Canada, 1980, Université Laval, Département d'informatique.

[FINL'80b]

Finley, M.R.: Systèmes d'information utilisant les fibres optiques, CJIS/RCSI, Vol. 5, pp 31-60, mai 1980.

[FINL'81a]

Finley, M.R. (with A. Dubuque and T. Vo-Dai): An experimental Optical Fiber-Based Visual Information System at Laval University, Proceedings Canadian Information Society Session'81, CIPS'81, Waterloo, 8-10 June 1980, pp 1.2.1-1.2.1.2.

[FINL'81b]

Finley, M.R.: Optical Fibers in Local Area Networks, article in preparation for Fiber and Integrated Optics, 1981.

[GREE'80]

Green P.E., An introduction to Network Architectures and Protocols, IEEE Trans. Com., Vol. COM-28, No. 4, pp. 413-424, April 1980.

[HABA'80]

Habara, K. and T. Aratani: Toward Local Network Digitalization - The View from Japan, IEEE Trans. on Communications, Vol. COM-28, No. 7, July 1980, p. 956-966.

[IEEE'81]

IEEE Computer Society, Project 802, Local Area Network Reference Model (LAN/RM), draft document, 5 February 1981.

[ISO'79]

International Organization for Standardization, Open Systems Interconnection, Reference Model of Open Systems Interconnection ISO/TC97/SC16, Version 4, June 1979.

[LIEB'78]

Liebowitz, B.H. and J.H. Carson: Distributed Processing, Long Beach, California, IEEE, 1978.

[LIN'78]

Lin, P.M., J. Leon and C.R. Steward: Analysis of Circuit-Switched Networks Employing Originating Office Control with Spill-Forward", IEEE Trans. on Com., Vol. COM-26, No. 6, 1978, p. 754-765.

[LINK'81]

LINK: Viewdata/Videotex Report, Vol. 2, No. 3, LINK Resources Corporation, 215 Park Avenue South, New York, New York, March 1981.

[MILL'80]

Mills, R.J.: The Future of the Subscriber Loop Network, WESCON'80, 16-18 September 1980, Anaheim, California.

[PRES'80]

Prestel Group (British Telecom): publicity brochure on Picture Prestel entitled Prestel - the Future, 1980 (?).

[RYBC'80]

Rybczynski, A.: X.25 Interface and End-to-End Virtual Circuit Service Characteristics, IEEE Trans. Com., Vol. COM-28, No. 4, pp. 500-509, April 1980.

[SCHU'81]

Schubin, M.: Videodisco as an Information Storage Medium, Proceedings Videotex'81, 20-22 May 1981, Toronto, pp. 303-307.

[SHOC'80]

Shoch, J.F.: Carrying Voice Traffic through an Ethernet Local Network... A General Overview, IFIP WG G.4 International Workshop on Local Area Computer Networks, Zurich, August 1980.

[TERR'80]

Terreault, C.: personal communication, August 1980.

[VIDE'81]

Videotex'81: Proceedings of Videotex'81, 20-22 May 1981, Toronto, available from Online Conferences Ltd and Infomart, 1981. Rem: This Proceedings contains many papers covering Prestel, Télétel, Telidon, CAPTAINS and videotex and videotex-related activities.

[WECK'80]

Wecker, S.: DNA - The Digital Network Architecture, IEEE Trans. Comm., Vol. COM-28, No. 4, April 1980, pp. 510-526.

[XERO'80]

Xerox: The Ethernet Local Network: Three Reports, CSL-80-2, February 1980, Xerox Palo Alto, California (countains three well-known reports on Ethernet).

[ZIMM'80]

Zimmerman, H.: OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection, IEEE Transactions on Communications, COM-28, 4, April 1980, pp. 425-432.

ANNEX I

Modeling Methods for Performance Analysis of Circuit-Switched Networks

(Méthodes de Modélisation pour l'Analyse de la Performance des Réseaux Téléphoniques)

ABSTRACT

This annex presents a modeling method for performance evaluation of circuit-switched networks. First, a closed queueing network with multiple service classes is developed to model network links. Secondly, a recursive algorithm is proposed for determining the node-to-node grades of service and path usage probabilities in networks of arbitrary switching scheme . Thirdly, an exact analytical formula is derived for determining path usage probabilities and node-to-node grades of service in networks using a successive office control routing strategy.

CONTENTS

1. INTRODUCTION

Circuit-switched networks such as telephone networks continue to be essential carriers for an increasing number of telecommunication services [1]. Existing and future telecommunication services will generate significant additional traffic whose impact on circuit-switched networks will be important and will need to be evaluated in order to manage effectively network capacity planning. Thus, performance evaluation in circuit-switched networks is a critical problem for network managers.

Network performance evaluation requires that the telecommunications services and the networks be properly characterized and that quantitative models be built to characterize adequately the interaction between telecommunication services and networks [2]. For circuit-switched networks, the telecommunication services can be characterized by traffic intensities and the networks themselves by parameters such as network configuration, routing plan, and route control strategy [3,4]. Circuit-switched network performance is measured in terms of grades of service and link utilization [4]. The former assesses the quality of service the network offers to subscribers while the latter indicates network efficiency. Given network and service characteristics, mathematical models can be built to estimate network performance [5,6].

This paper attacks the problem of modeling a circuit-switched network for performance evaluation purpose. The same problem has recently been treated in [5] and [6] and this paper follows closely the methodology exposed in these works. The contribution of this paper is manifold. First, a model is sought to represent network links (trunk groups) in such a way that account may be

taken of the differences in holding times of different types of telecommunication services. To this end, a closed queueing network with multiple service classes is developed. Our results show that Erlang's formula still holds for the case of multiple service classes and provides an analytical tool for obtaining insight into each service class. Our model is also an interesting example of application of queueing network theory to telephony. Secondly, inspired from the work in [7,8], we propose a very simple recursive algorithm to calculate path usage probabilities and node-to-node grade of service. Our algorithm is also a simple solution to the terminal-pair reliability problem [9]. The algorithm has been implemented in PL/1 and the computer program is included in this paper. Finally, a very simple analytical formula is presented to calculate path usage probabilities and node-to-node blocking probabilities for successive-office routing control strategy. To our knowledge, this is the first time an exact formula has been proposed for the very important successive-office control circuit-switched networks.

The organization of this paper is as follows: in Section 2, we present some definitions, assumptions and fundamental facts about circuit-switched networks. In Section 3, a closed queueing network with multiple classes of services is developed to represent network links. In Section 4, the problem of calculating path usage probabilities is addressed. Detailed solution of the queueing network is given in Appendix A and an PL/1 implementation of the recursive algorithm for the terminal-pair reliability problem is presented in Appendix B.

2. DEFINITIONS AND ASSUMPTIONS

A circuit-switched network consists of nodes (offices) interconnected by links (trunk groups) whose main object is to set up a connection between some originating and destination node pair via one of a number of alternative paths. The alternative paths between such an originating-destination (O - D) node pair are attempted in a predefined order according to a routing scheme. The routing scheme is defined by two factors: the routing plan giving the set of alternative paths and the route control strategy which decides the choice of path.

In [5], it is shown that the routing scheme for an O - D node pair is completely described by an augmented route tree from which a path-loss sequence can be generated. The latter specifies the alternative paths, the order in which they are successively attempted and the conditions under which a called is blocked, i.e., lost to the network. In an augmented route tree, a tip is labeled either with a destination node or with L which stands for a fictitious "loss node" [5]. There is a unique path from the root of the tree to every tip. If the tip is a destination node, the corresponding path is designated by a P_j and is called a completion path. If the tip is a loss node, the corresponding path is designated by a L_k and is referred to as a loss-path.

In [5], augmented route trees and path-loss sequences for the three following principal route control strategies are discussed: Successive-Office Control (SOC), Originating Office Control (OOC) and Originating Office Control with Spill-Forward. These route control strategies possess special

properties which can be recognized from the augmented route trees: For SOC every node except the destination node is connected to a node L, while for OOC only the originating node is connected to a node L and for OOC with spill-forward only certain nodes, called spill-forward nodes, are connected to a node L.

Under a number of standard assumptions usually made in telephony [6], the set of parameters used to describe the state of a circuit-switched network is the set of link blocking probabilities. In [6], these assumptions and their implications are rigorously discussed. An important implication is that each link is offered a traffic which is a superposition of independent streams of Poisson traffic. In this paper we will not assume the same holding time for all streams in order to take into account of the differences between various telecommunication services.

Analysis of complex circuit-switched networks is usually carried out by an iterative method. An iterative technique which has often been adopted breaks the network modeling process into two distinct steps:

1. Link Modeling: In this step, assuming that all link offered traffics are given, a model, which will be called link model, is constructed to calculate all link blocking probabilities.

2. Path Usage Modeling: In this step, supposing that all link blocking probabilities were known, a model, which will be called path usage model, is sought to estimate all link offered traffics.

Once these two models become available, the state vector of the network, i.e. the vector of link blocking probabilities, is obtained by iteration.

as depicted in Figure 1. At each iteration, the link blocking probabilities are calculated by the link model using the link offered traffics obtained by the path usage model in the previous iteration. In [5], this iterative algorithm is identified as the fixed-point algorithm. Thus convergence to the "true" network state vector is guaranteed if the link model and the path usage model are consistant. However, these models being based on a considerable number of assumptions [4,5] which are sometimes difficult to justify, convergence is not guaranteed. Nevertheless, experience has shown that convergence seems to always occur and results given by the iterative algorithm were satisfactory [5,10]. The rest of this paper will be concerned mainly with the building of the link model and the path usage model.

Some terms must now be defined. A link is a collection of trunks and a link is free if at least one of its trunk is unoccupied, otherwise it is congested. A link set is an ordered set of links and a link set is free if all its links are free, otherwise it is congested. A path is a link set whose elements constitute an actual path for a call. Only link sets which are subsets of paths will be considered in this article, except for the sets V_i defined in Section 4.2.

3. LINK MODEL

Consider now a link on some path between an O - D node pair. Let the link be offered M independant streams of Poisson traffic with parameters $\lambda_1, \lambda_2, \ldots, \lambda_M$ respectively. Let the mean holding times for each stream of Poisson traffic be $1/\mu_1, 1/\mu_2, \ldots, 1/\mu_M$. These traffic streams can be thought of as belonging to M telecommunication services for which the mean holding times may be different. The total link offered traffic is thus a Poisson process with parameter given by:

$$\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_M \quad , \tag{3.1}$$

and any arriving call has a fixed probability p_i of belonging to service class i given by:

$$p_i = \lambda_i/\lambda, \quad i = 1, 2, \ldots, M \quad . \tag{3.2}$$

Let N be the total number of servers (trunks) in the link. Since calls which are offered to the link and find all the servers busy overflow the link, the total number of calls simultaneously in progress never exceeds N, the link can thus be modeled as a closed queueing network as shown in Figure 2. Our queueing model resembles the Central Server Model widely used to model multiprogramming Computer system [11]. Our model comprises a central server with service rate λ and M "channel" servers with state-dependent service rates being of the infinite capacity type. After being served at the central server, a call has a probability p_i of going to the non-central server i after which it always returns to the central server. Let the total number of calls in our closed queueing network be N. It can be seen that the main

function of the central server is to generate call arrivals at a rate of λ as long as the total number of calls in the link is less than N, i.e., as long as the total number of calls at the central server is greater than zero.

The state of our closed queueing network is described by a vector:

$$\underset{\sim}{n} = (n_0, n_1, \ldots, n_M)$$

with n_0 being the number of calls at the central server, n_1 number of calls at the server 1, etc... The number of distinct states the network can have is defined by the constraint:

$$n_0 + n_1 + \ldots + n_M = N \quad . \tag{3.3}$$

The i-th non-central server has a state-dependant service rate of the form:

$$\mu_i(\underset{\sim}{n}) = n_i \mu_i, \quad i = 1, \ldots M \quad . \tag{3.4}$$

Our closed queueing network belongs to the class of separable queueing networks which have been intensively studied by computer scientists [12,13]. Let $P(\underset{\sim}{n})$ be the network stationary-state probability. It is shown in the Appendix A (Eq. A.6) that $P(\underset{\sim}{n})$ admits a separable solution of the form:

$$P(\underset{\sim}{n}) = \frac{(1/\lambda)^{n_0}}{G(N)} \prod_{i=1}^{M} \frac{(p_i/\mu_i)^{n_i}}{n_i!} \quad , \tag{3.5}$$

Where $G(N)$ is given by:

$$G(N) = (1/\lambda)^N \sum_{j=0}^{N} (\lambda/\mu)^j \quad , \tag{3.6}$$

λ is given by Eq. (3.1) and μ is defined by:

$$1/\mu \triangleq \sum_{i=1}^{M} (p_i/\mu_i) \qquad (3.7)$$

From Eq. (3.5), the probability π_k of finding k calls simultaneously in progress is obtained as (Eq. A.13):

$$\pi_k = \frac{a^k/k!}{\sum_{j=0}^{M} (a^j/j!)} \qquad (3.8)$$

where we define $a = \lambda/\mu$ as the traffic intensity. Now, Eq. (3.8) is just Erlang's well-known formula. As it is known that Erlang's formula holds for any M/G/c/c queueing [14] system, Eq. (3.8) is not really surprising. In fact, Eq. (3.8) can be obtained from the queue-size distribution in a M/G/c/c system if it can be shown that μ defined by Eq. (3.7) is the mean service rate for the case of multiple service classes. Our derivation of Erlang's formula for this case has the advantage of being direct and elementary. Furthermore, Eq. (3.6) allows us to derive the marginal distribution of service-class i calls as:

$$\Pr\{n_i = k\} = S(N,k)\ (a_i^k/k!) \qquad (3.9)$$

where

$$S(N,k) = \frac{(1/\lambda)^N}{G(K)} \sum_{m=0}^{N-k} \Big[\sum_{\substack{j=1 \\ j\neq i}}^{M} (p_j/\mu_j)\Big]^m/m! \quad , \qquad (3.10)$$

and

$$a_i = p_i(\lambda/\mu_i) = \lambda_i/\mu \quad . \qquad (3.11)$$

Eq. (3.9) can be used to obtain statistical properties related to service class i.

Eq. (3.8) can be used to obtain the link blocking probability y for the link being considered:

$$y = \frac{a^N/N!}{\sum_{j=0}^{N} a^j/j!} \qquad (3.12)$$

This equation is used by the link model to compute the link blocking probabilities. The same equation has been used in previous works but its validation for the case of multiple service classes with different holding times is justified here.

4. PATH USAGE MODEL

4.1 Arbitrary route control strategy

In this section, we examine the problem of calculating link offered traffic given link blocking probabilities. Consider a loss-path sequence $U_1, U_2, \ldots, U_k$ and a link j on a path U_i between some O - D node pair. We will assume that the occupancies of the links are statiscally independant.

Let $x_j = 1 - y_j$, where y_j is the blocking probability of link j, be the availability of link i. In [6] it is shown that the traffic offered to link j is given by:

$$a_j = \frac{c_j}{1 - y_j} \qquad (4.1)$$

where c_j is defined as:

$$c_j = A \Pr \{U_1, \ldots, U_{i-1} \text{ congested} \mid U_i \text{ free}\} \cdot \Pr \{U_i \text{ free}\} \qquad (4.2)$$

where A is the originating traffic. In [6], a_j is called reduced offered traffic and in [5], c_j is called carried traffic. Here we call a_j link offered traffic and emphasize that it is this traffic that must be used in Erlang's formula to calculate the link blocking probability.

It can be seen from Eq. (4.2) that the heart of the problem is to calculate the probability that the path U_i is used by a call, which is given by:

$$\Pr \{U_i \text{ used}\} = \Pr \{U_1, \ldots, U_{i-1} \text{ congested} \mid U_i \text{ free}\} \cdot \Pr \{U_i \text{ free}\} \ . \qquad (4.3)$$

We will show now that there is a one-to-one correspondance between the problem of determining path usage probabilities and the terminal-pair reliability problem [9]. From an elementary theorem in Probability Theory, we have:

$$\begin{aligned}\Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_i \text{ free}\} &= \Pr\{U_i \text{ free}\} \\ &+ \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free}\} \\ &- \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free}, U_i \text{ free}\} \end{aligned} \tag{4.4}$$

Conditioning the last term of this equation on path U_i being free, we rewrite it as:

$$\begin{aligned}&\Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_i \text{ free}\} = \\ &\Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free}\} + \Pr[U_i \text{ free}] \\ &\cdot[1 - \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free} \mid U_i \text{ free}\}]\end{aligned} \tag{4.5}$$

Comparing Eq. (4.5) with Eq. (4.3) and noting that:

$$\begin{aligned}\Pr\{U_1, \ldots U_{i-1} \text{ congested} \mid U_i \text{ free}\} &= 1 \\ &- \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free} \mid U_i \text{ free}\}\end{aligned}$$

we can write Eq. (4.5) as:

$$\begin{aligned}\Pr\{U_i \text{ used}\} &= \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_i \text{ free}\} \\ &- \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free}\}.\end{aligned} \tag{4.6}$$

This equation means that the problem of determining all the path usage probabilities, $\Pr\{U_i \text{ used}\}$, for $i = 1, 2, \ldots$, is equivalent to solving all the terminal-pair reliability problems, i.e. calculating all the $\Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_k \text{ free}\}$. For this reason, we will concentrate our attention on the terminal-pair reliability problem.

For any two link sets L_i and L_j, define:

$$L_{j(i)} \triangleq L_j - L_i$$

$$\triangleq \{\ell | \ell \text{ is in } L_j \text{ and not in } L_i\}.$$

Then, by definition we have:

$$\Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_{i-1} \text{ free} \mid U_i \text{ free}\}$$

$$= \Pr\{U_{1(i)} \text{ or } U_{2(i)} \ldots \text{ or } U_{i-1(i)} \text{ free}\} \tag{4.7}$$

Let R be a function defined on the set of all possible link sets by:

$$R(U_1, U_2, \ldots, U_i) = \Pr\{U_1 \text{ or } U_2 \ldots \text{ or } U_i \text{ free}\}. \tag{4.8}$$

Substituing Eq. (4.7) into (4.5) and using Eq. (4.8) we obtain:

$$R(U_1) = \prod_{\ell \in U_1} x_\ell, \tag{4.9}$$

$$R(U_1, U_2, \ldots U_i) = R(U_1, U_2, \ldots U_{i-1})$$

$$+ [1 - R(U_{1(i)}, \ldots, U_{i-1(i)})] \prod_{\ell \in U_i} x_\ell, \quad \text{for } i > 1 \tag{4.10}$$

Eqs. (4.9) and (4.10) provides us with a simple recursive relation for solving the terminal-pair reliability problem. A computer implementation of this relation is given in Algorithm 1 which is in the form of a recursive procedure written in Pidgin ALGOL [15].

To measure the amount of computation envolved in evaluating $R(U_1, U_2, \ldots U_i)$ we let n_i be the number of recursive calls in Algorithm 1. For $i = 1$ it is obvious that:

$$n_1 = 1 \tag{4.11}$$

PROCEDURE $R(U_1, U_2, \ldots, U_i)$:

IF $i = 1$ THEN RETURN $\prod_{\ell \in U_i} x_\ell$

ELSE

BEGIN

FOR $k \leftarrow 1$ STEP 1 UNTIL $i - 1$ DO

$V_k \leftarrow U_k - U_i$

$R \leftarrow R(U_1, U_2, \ldots, U_{i-1})$

$+ \left[(1 - R(V_1, V_2, \ldots, V_{i-1})\right] \prod_{\ell \in U_i} x_\ell$

RETURN R

END

Algorithm 1: recursive procedure for solving the terminal pair reliability problem.

For $i > 1$, to evaluate $R(U_1, \ldots U_i)$, we first make 2 recursive calls, namely one for $R(U_1, U_2, \ldots U_{i-1})$ and one for $R(V_1, V_2, \ldots, V_{i-1})$, and each of them in turn will make n_{i-1} recursive calls. Hence:

$$n_i = 2 + 2n_{i-1} \tag{4.12}$$

Solving the difference equation (4.12) with boundary condition (4.11), we obtain:

$$n_i = 2^{i-1} - 2, \quad i > 1 \tag{4.13}$$

Since from Eq. (4.3) we have:

$$\Pr\{U_i \text{ used}\} = \left(\prod_{\ell \epsilon U_i} x_\ell\right) \cdot [1 - R[U_{1(i)}, U_{2(i)}, \cdots U_{i-1(i)}]\,, \tag{4.14}$$

the evaluation of $\Pr\{U_i \text{ used}\}$ using Algorithm 1 will necessitate $n_{i-1} = 2^{i-2} - 2$ recursive calls. In [7], a recursive algorithm is proposed which requires $2^{i-1} - 1$ recursive calls in order to calculate $\Pr\{U_i \text{ used}\}$ [7, Algorithm 1]. In [7] a recursive algorithm [7, Algorithm 2] for the terminal-pair reliability problem is also proposed which require $2^{i-1} - 1$ recursive calls. No serious attempt will be made here to compare the efficency of our algorithm with those in [7] except to observe that our Algorithm 1 appears to be simpler in form. A PL/1 implementation of Algorithm 1 is presented in Appendix B together with the computer output.

Under some circumstances, Eq. (4.9) may be simplified:

1. When $U_{k(i)} = U_k$ for $k = 1, 2, \ldots i - 1$, then:

$$R(U_1, U_2, \ldots, U_i) = \prod_{\ell \epsilon U_i} x_\ell + \left(1 - \prod_{\ell \epsilon U_i} x_\ell\right) R(U_1, U_2, \ldots, U_{i-1}) \tag{4.15}$$

2. When the U_k's are disjoint, then:

$$R(U_1, U_2, \ldots, U_i) = R(U_1, \ldots, U_{i-1}) + [\prod_{k=1}^{i-1} (1 - \prod_{\ell \in U_k} x_\ell)] \prod_{j \in U_i} x_j \quad (4.16)$$

The second term in Eq. (4.16) is actually equal to Pr {U_i used} and we do not need the function R to evaluate the latter.

3. If the paths $U_1, U_2, \ldots, U_i$ may be devided into two disjoint sets, namely $\{U_{j_1}, \ldots, U_{j_k}\}$ and $\{U_{j_{k+1}}, \ldots, U_{j_i}\}$, then:

$$R(U_1, U_2, \ldots U_i) = R(U_{j_1}, \ldots, U_{j_k})$$
$$+ R(U_{j_1}, \ldots, U_{j_k}) - R(U_{j_1}, \ldots, U_{j_k}) R(U_{j_{k+1}}, \ldots, U_{j_i}) \quad (4.17)$$

Although Eqs. (4.15 - 4.17) simplify the calculation, they are probably not advantageous for computer implementation due to the substantial computation required to verify the conditions under which they hold. They should be used only in analytical analysis.

4.2 Successive Office Control Strategy

In SOC, alternative choice of outgoing links for a call are permitted at tandem nodes and the choice at a tandem node is independant of those made at other tandem nodes, both in the past and in the future. Furthermore, with SOC, a call is routed to the first tandem node of a path if its first link is free. These properties will help us to derive a simple analytical formula for path usage probabilities for networks with SOC.

Let V_i be the set consisting of those first links taken from $U_{1(i)}, \ldots, U_{i-1(i)}$, where $U_1, U_2, \ldots, U_i, \ldots$ are successive paths in a path-loss sequence. Let us retain in V_i only distinct links, i.e. $|V_i| \leq i$.

It can be seen that V_i can be constructed solely from the completion path P_j's. In fact, let L_k be a loss-path found between P_1 and P_i. By definition, all links in L_k except the last one which is a fictitions "empty link", are overlapping links with some completion path, say P_h, immediately before L_k. Since P_i has no "empty link", we have:

$$L_h - P_i \subset P_h - P_i$$

This means that the first link of $L_{k(i)}$ is also the first link of $P_{k-1(i)}$. Hence in constructing V_i, we can entirely ignore the loss-paths.

From Eq. (4.3) we have:

$$\Pr\{P_i \text{ used}\} = \Pr\{U_{1(i)}, U_{2(i)} \cdots U_{i-1(i)} \text{ congested}\} \cdot \Pr\{P_i \text{ free}\} \quad (4.18)$$

Since in SOC a link set $U_{k(i)}$ is congested if its first link is congested, from the definition of the link set V_i, it can be seen easily that:

$$\begin{aligned}&\Pr\{U_{1(i)}, U_{2(i)}, \cdots U_{i-1(i)} \text{ congested}\}\\ &= \Pr\{\text{all links in } V_i \text{ are congested}\}\\ &= \prod_{x_\ell \in V_i} (1 - x_\ell) \quad (4.19)\end{aligned}$$

Substituting Eq. (4.19) in Eq. (4.18), we obtain:

$$\Pr\{P_i \text{ used}\} = \prod_{\ell \in P_i} x_\ell \left[\prod_{m \in V_i} (1 - x_m)\right] \quad (4.20)$$

Eq. (4.20) yields the desired path usage probabilities for networks with SOC. The node-to-node grade of service (NNGOS) for a network under

SOC is thus given by:

$$NNGOS = 1 - \sum_i \left(\prod_{\ell \in P_i} x_\ell \right) \left[\prod_{m \in V_i} (1 - x_m) \right] \tag{4.21}$$

where the sum extends to all completion paths in the path-loss sequence.

As an example, let us consider the path-loss sequence generated by the augmented route tree shown in Fig. 3. The event "P_4 is used" occurs if the following events occur: link OA is congested (otherwise P_1 or L_1 is used), link OC is free (otherwise L_8 is used), link CA is congested (otherwise P_3 or L_4 is used) and links CB, BA and AD are free. Hence:

$$\begin{aligned} \Pr\{P_4 \text{ used}\} &= (1 - x_{OA})(1 - x_{OB})x_{OC} \\ &\quad \cdot (1 - x_{CA})x_{CB}\, x_{BA}\, x_{AD} \\ &= (1 - x_{OA})(1 - x_{OB})(1 - x_{CA})x_{OC}\, x_{CB}\, x_{BA}\, x_{AD} \end{aligned} \tag{4.22}$$

On the other hand, we have:

$$P_1 - P_4 = \{OA, AD\},$$

$$P_2 - P_4 = \{OB, BA\},$$

$$P_3 - P_4 = \{CA, AD\}$$

so that:

$$V_4 = \{OA, OB, CA\}.$$

Using V_4 in Eq. (4.20), we obtain (4.22). This verifies Eq. (4.20) for the augmented route tree of Figure 3.

5. CONCLUSION

We have provided a framework for modeling circuit-switched networks which takes advantage of results in Queueing Network Theory and of most recently advanced ideas for circuit-switched analysis, namely, the use of augmented route tree and recursive programming technique. By modeling a network link as a closed queueing network with multiple service classes, we have offered a simple and direct proof of Erlang's formula for the situation in which the link is offered multiple streams of Poisson traffic with different mean holding times. This result is of special importance because modern circuit-switched networks are carriers of an increasing number of distinct telecommunications services. Furthermore, our queueing network model can be employed to answer performance - related questions concerning each service class. The calculation of path usage probabilities and end-to-end blocking probabilities has always been a difficult problem. We have demonstrated that the use of recursive programming technique offers a very simple and easily implemented solution to this problem under the form of a recursive algorithm. The availability of many programming languages with recursivity facility and computer systems with virtual memory makes our algorithm practical for circuit-switched networks analysis. Finally, for the very important class of circuit-switched networks operating under successive office routing control, we have derived an exact formula for calculating path usage probabilities and end-to-end blocking probabilities. This formula is valuable for analysis of networks such as the telephone networks.

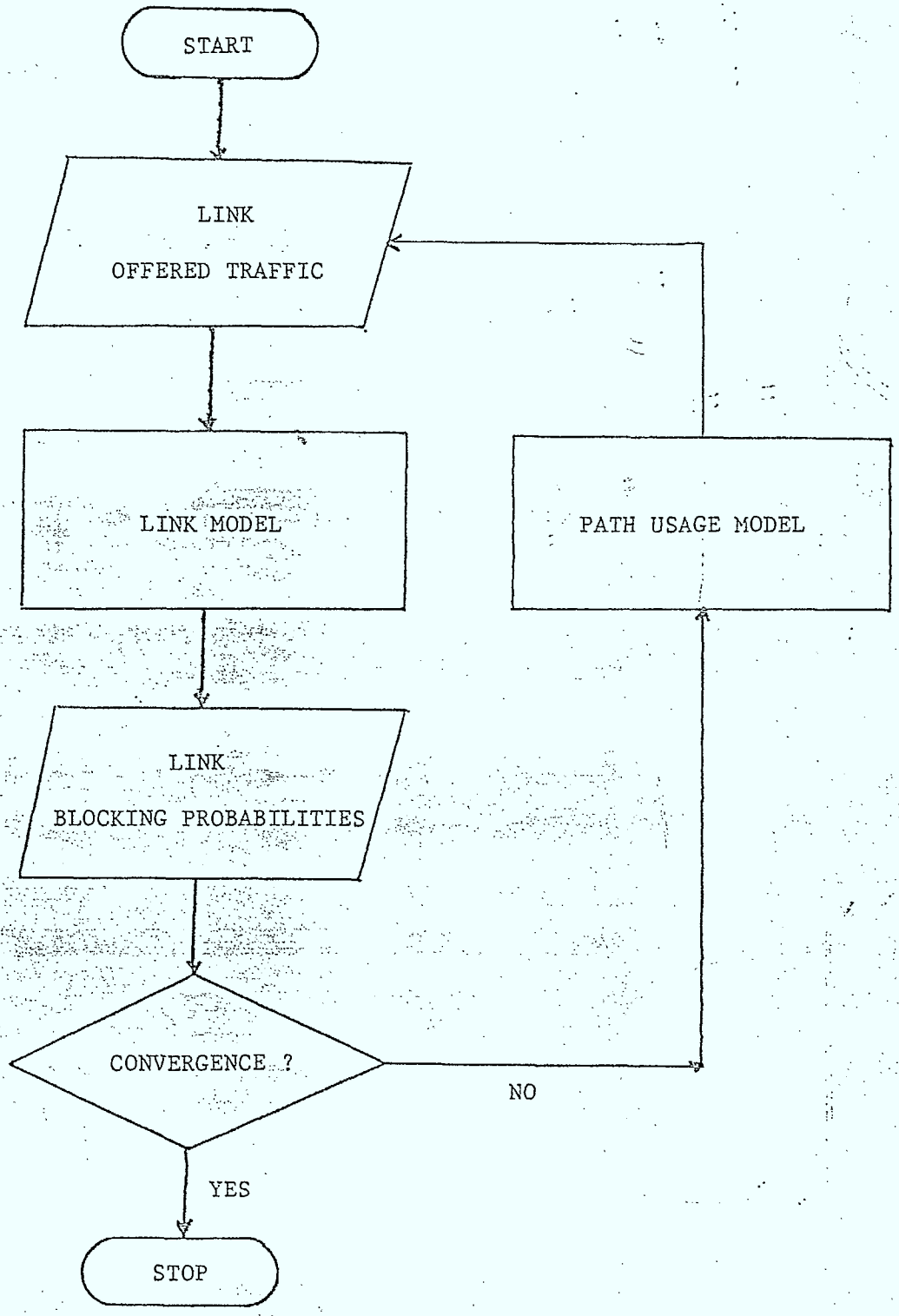


Figure 1: Iterative algorithm for network modeling

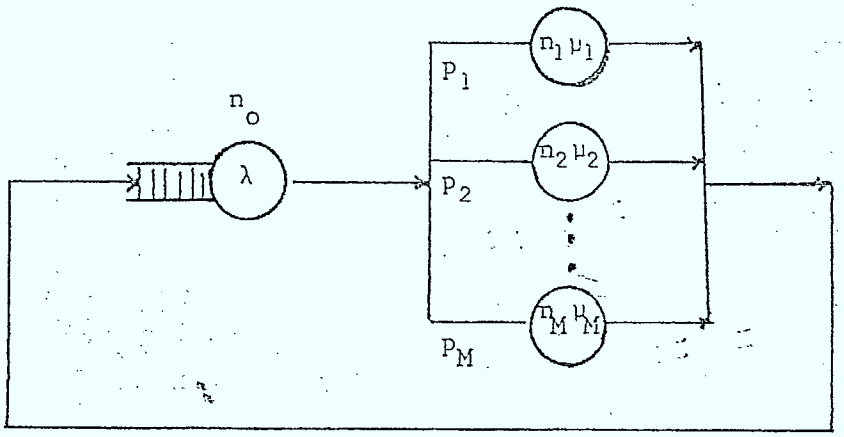


Figure 2: Closed queueing network model for telephone trunk groups.

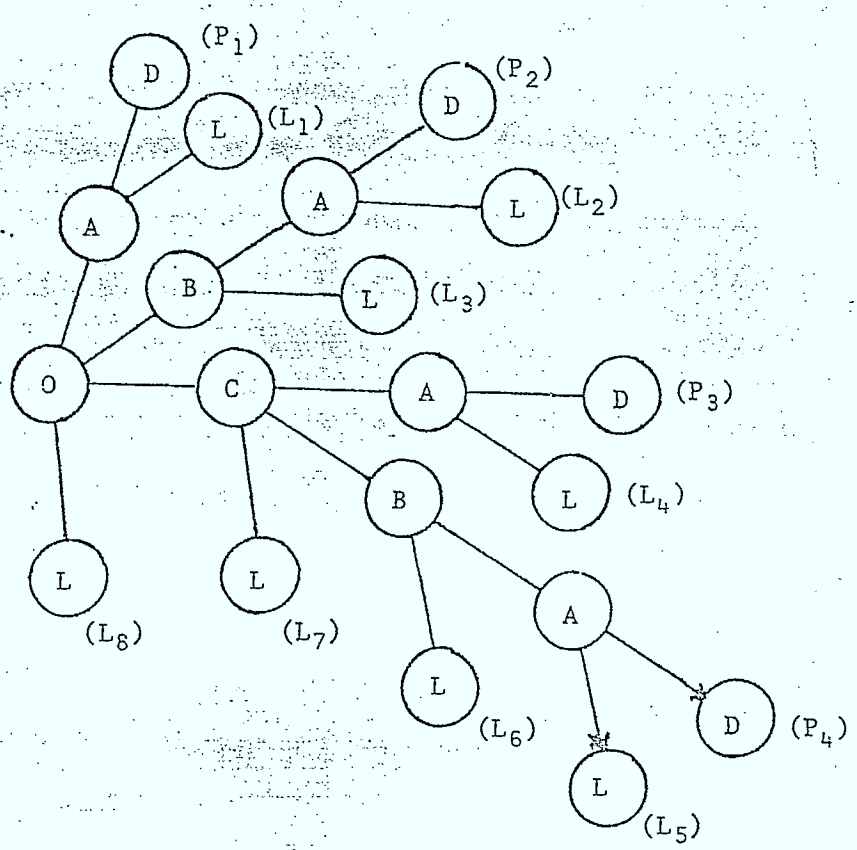


Figure 3: Example of augmented route tree for an O - D node pair with S.O.C.

REFERENCES

[1] M. Finley, Jr., and T. Vo-Dai: *"A Technical and Economic Study on the Impact of the Introduction of New Services on Existing Telecommunication Networks"*, Final Report submitted to the Canadian Department of Communications under Contract No. 20ST.36100-9-9525, MSS Serial No. OST79-00036, March 1980.

[2] D. Ferrari: *"Computer Systems Performance Evaluation"*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1978.

[3] S. Katz: *"Statistical Performance Analysis of a Schwitched Communication Network"*, Proc. 5th Int'l Teletraffic Congress, pp. 566-579, June 1967.

[4] R.I. Wilkinson: *"Theory for Toll Traffic Engineering in the U.S.A."*, BSTJ, Vol. 35, pp. 421-454, 1956.

[5] P.M. Lin, J. Leon and Chris R. Steward: *"Analysis of Circuit-Switched Networks Employing Originating - Office Control with Spill-Forward"*, IEEE Trans. on Comm., Vol. COM-26, No. 6, pp. 754-765, 1978.

[6] R. Manfield and T. Downs: *"On the One-Moment Analysis of Telephone Traffic Networks"*, IEEE Trans. Comm., Vol. COM-27, No. 8, pp. 1169-1174, 1979.

[7] W.S. Chan: *"Recursive Algorithm for Computing End-to-End Blocking in a Network with Arbitrary Routing Plan"*, IEEE Trans. Comm., Vol. COM-28, No. 2, pp. 153-164, 1980.

[8] M.D. Gaudreau: *"Recursive Formulas for the Calculation of Point-to-Point Congestion"*, IEEE Trans. Comm., Vol. COM-28, No. 3, pp. 313-316, 1980.

[9] P.M. Lin, R.J. Leon and T.C. Huang: *"A New Algorithm for Symbolic System Reliability Analysis"*, IEEE Trans. on Reliability, Vol. R-25, pp. 2-15, 1976.

[10] D.A. Calabrese, M.J. Fischer, B.E. Hoiem and E.P. Kaiser: *"Modeling a Voice Network with Preemption"*, IEEE Trans. on Comm., Vol. COM-28, No. 1, 1980.

[11] J.P. Buzen: *"Queueing Network Models of Multiprogramming"*, Ph.D. Thesis, Division of Engineering and Applied Sciences, Harward University, Cambridge, Mass., 1971.

[12] F. Basket, K.M. Chandy, R.R. Muntz and F.G. Palacios: *"Open, Closed and Mixed Networks of Queues with Different Classes of Customers"*, Jour. ACM, Vol. 22, No. 2, pp. 248-260, 1975.

[13] H. Kobayashi: *"Modeling and Analysis: An Introduction to System Performance Evaluation Methodogy"*, Chapter 3, Addison-Wesley Publishing Company, 1978.

[14] D. Gross and C.M. Harris: *"Fundamentals of Queueing Theory"*, John Wiley & Sons, Inc., pp. 266-272, 1974.

[15] A.V. Aho, J.E. Hopcroft and J.D. Ullman: *"The Design and Analysis of Computer Algorithms"*, Addison-Wesley Publishing Compagny, pp. 33-39, 1974.

[16] J.P. Buzen: *"Computational Algorithms for Closed Queueing Networks with Exponential Servers"*, Comm. of the ACM, Vol. 16, No. 9, pp. 527-531, 1973.

APPENDIX A

SOLUTION OF THE QUEUEING NETWORK FOR TELEPHONE LINKS

The closed queueing network shown in Figure 2 belongs to the class of separable networks which have been studied intensively by computer scientists [12,13]. The state of the network is described by a vector:

$$\underset{\sim}{n} = (n_0, n_1, \ldots n_M) \tag{A-1}$$

such that

$$\sum_{i=0}^{M} n_i = N \tag{A-2}$$

where n_i is the number of calls at station i. The solution for equilibrium state probability $P(\underset{\sim}{n})$ can be shown to have the form [12]:

$$P(\underset{\sim}{n}) = [1/S(N)] \prod_{i=0}^{M} [e_i^{n_i}/M_i(n_i)] \tag{A-3}$$

where $S(N)$ is the normalization constant, and e_i is given by:

$$e_i = p_i e_0 \ , \ i = 1, 2, \ldots M \ , \tag{A-4}$$

$M_i(n_i)$ is given by:

$$M_i(n_i) = \prod_{j=1}^{n_i} j\mu_i = \mu_i n_i!, \quad i = 1, 2, \ldots M$$

and

$$\mu_0(n_0) = \lambda^{n_0} \ , \quad i = 1 \ . \tag{A-5}$$

Substituting Eqs. (A-4) and (A-5) in Eq. (A-3) we obtain:

$$P(\underline{n}) = \frac{(1/\lambda)^{n_0}}{G(N)} \prod_{i=1}^{M} \frac{(p_i/\mu_i)^{n_i}}{n_i!} \tag{A-6}$$

where $G(N) = e_o^N/S(N)$.

To obtain $G(N)$, let us apply the convolution method [16]. Define π_k as the probability of finding k calls simultaneously in progress. From (A-6) we have:

$$\pi_k = \frac{1}{G(N)} \left(\frac{1}{\lambda}\right)^{N-k} \sum^{k} \prod_{i=1}^{M} \frac{(p_i/\mu_i)^{n_i}}{n_i!} \tag{A-7}$$

where the sum $\sum^{k}$ is over all possible $n_1, n_2, \ldots n_M$ such that $n_1 + n_2 + \ldots n_M = k$. Using the algebric relation:

$$\sum^{k} \prod_{i=1}^{M} \frac{x_i}{n_i!}^{n_i} = \frac{1}{k!} \left(\sum_{i=1}^{M} x_i\right)^k , \tag{A-8}$$

we obtain:

$$\pi_k = \frac{(1/\lambda)^{N-k}}{G(N)k!} \left[\sum_{i=1}^{M} (p_i/\mu_i)\right]^k . \tag{A-9}$$

The normalization constant is obtained by the constraint:

$$\sum_{k=0}^{N} \pi_k = 1 \tag{A-10}$$

with the result:

$$G(N) = (1/\lambda)^N \sum_{k=0}^{N} \left[\lambda \sum_{i=1}^{M} (p_i/\mu_i)\right]^k/k!$$

$$= (1/\lambda)^N \sum_{k=0}^{N} (\lambda/\mu)^k \tag{A-11}$$

where μ is defined as:

$$\mu = \sum_{i=1}^{M} p_i/\mu_i \quad . \tag{A-12}$$

Substituting Eq. (A-12) into (A-7) we have:

$$\pi_k = \frac{\lambda^k/k!}{\sum_{i=1}^{M} (\lambda/\mu)^k} \cdot (1/\mu)^k$$

$$= \frac{a^k/k!}{\sum_{k=0}^{N} (a^j/j!)} \tag{A-13}$$

where we define $a = \lambda/\mu$.

The marginal distribution for n_i is given by:

$$P_i(k) = \Pr\{n_i = k\}$$

$$= (p_i/\mu_i)^k/(k!G(N)) \cdot \sum\nolimits' (1/\lambda)^{n_0} \prod_{\substack{j=1 \\ j\neq i}}^{M} (p_i/\mu_i)^{n_j}/n_j! \tag{A-14}$$

where the summation $\sum'$ is over all vectors $(n_o, n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_M)$ such that $n_o + n_1 + \ldots + n_{i-1} + n_{i+1} + \ldots + n_M = N - k$. The summation can be evaluated by the convolution technique and by using Eq. (A-8). We have:

$$P_i(k) = (P_i/\mu_i)^k/(k!G(N))$$

$$\cdot \sum_{m=0}^{N-k} \{(1/\lambda)^{N-k-m} [\sum_{\substack{j=1 \\ j\neq i}}^{M} (p_i/\mu_i)]^m/m!\}$$

From this, we obtain:

$$P_i(k) = S(N,k)\ (a_i^k/k!)$$

where:

$$S(N,k) = \frac{(1/\lambda)^N}{G(N)} \sum_{m=0}^{N-k} \{[\lambda \sum_{\substack{j=1 \\ j\neq i}}^{M} (p_i/\mu_j)]^m/m!\} \qquad \text{(A-15)}$$

and

$$a_i = p_i(\lambda_i/\mu_i) = \lambda_i/\mu_i \qquad \text{(A-16)}$$

APPENDIX B

PL/1 IMPLEMENTATION OF ALGORITHM 1

In this Appendix, a PL/1 implementation of Algorithm 1 is given under the form of a PL/1 recursive procedure as:

```
R : PROCEDURE (U, PATH_U, I,X,P,T) RECURSIVE
```

The user stores the augmented route tree in the array U, specifies completion and loss paths by the array PATH_U, the number of paths by the integer I, link availabilities by the array X. Path usage probabilities will be stored in the array P if the user sets T=0.

Links availabilities x_1, x_2, ..., are stored in addresses X(1), X(2), ..., of the array X. Path U_k is stored in U in a sequence of contiguous addresses beginning at address PATH_U(k). Thus, the elements PATH_U(1), PATH_U(2), ..., point to the addresses of U which contain the first link number of path U_1, U_2, ...

Listing of the PL/1 recursive procedure for Algorithm 1 together with listing of the main program to evaluate path usage probabilities of an augmented route tree are given in the following. Also included in this Appendix is numerical results for the augmented route tree depicted in Figure 3 for:

$$X_{OA} = 0.95 \quad , \quad X_{OB} = 0.92 \quad , \quad X_{OC} = 0.12$$

$$X_{CA} = 0.98 \quad , \quad X_{CB} = 0.82 \quad , \quad X_{BA} = 0.37$$

$$X_{AD} = 0.87 \quad .$$

Numerical results agree with Eq. (4.20) as should be.

```
PL/I OPTIMIZING COMPILER          R:PROCEDURE(U,PATH_U,I,X,P,T) RECURSIVE;

                    SOURCE LISTING

 NUMBER LEV NT

     10       0   R:PROCEDURE(U,PATH_U,I,X,P,T) RECURSIVE;

     30   1   0   DECLARE (U(*),PATH_U(*),V(50),PATH_V(50))        FIXED(2),
                      (I,J,K,L,N)                                  FIXED(2),
                      T                                            FIXED(1),
                      (X(*),P(*),PR,XL,PI)                         FLOAT(16);

     80   1   0   XL=1;
     90   1   0   DO N=PATH_U(I) TO PATH_U(I+1)-1;
    100   1   1      XL = XL * X(U(N));
    110   1   1   END;
    120   1   0   IF I=1 THEN
                     DO;
    140   1   1        PR = XL;
    150   1   1        IF T=0 THEN P(I) = PR;
    160   1   1     END;
    170   1   0   ELSE
                     DO;
    190   1   1        L = 1; PATH_V(1) = 1;
    200   1   1        DO N=1 TO I-1;
    210   1   2           DO J=PATH_U(N) TO PATH_U(N+1)-1;
    220   1   3              DO K=PATH_U(I) TO PATH_U(I+1)-1
                              WHILE (U(K) ¬= U(J));
    240   1   4              END;
    250   1   3              IF K=PATH_U(I+1) THEN
                                DO;
    270   1   4                    V(L) = U(J); L = L + 1;
    280   1   4                END;
    290   1   3           END;
    300   1   2           PATH_V(N+1) = L;
    310   1   2        END;
    320   1   1        PI = (1 - R(V,PATH_V,I-1,X,P,1)) * XL;
    330   1   1        IF T=0 THEN
                        DO;
    350   1   2           P(I) = PI; PR = R(U,PATH_U,I-1,X,P,0);
    360   1   2        END;
    370   1   1     ELSE
                        DO;
    390   1   2           PR = R(U,PATH_U,I-1,X,P,1);
    400   1   2        END;
    410   1   1     PR = PR + PI;
    420   1   1  END;
    430   1   0  RETURN(PR);
    440   1   0  END R;
```

```
PL/I OPTIMIZING COMPILER          PRINC: PROCEDURE OPTIONS(MAIN);

                    SOURCE LISTING

 NUMBER LEV NT

     10      0  PRINC: PROCEDURE OPTIONS(MAIN);
     20    1 0  DECLARE (U(50),PATH_U(50) ,I,L,N,J,UI)      FIXED(2),
                       (X(15),P(15))                        FLOAT(16),
                       NAME(15)                             CHAR(2),
                       LIGNE_I(52)                          CHAR(1);

     70    1 0  DO J=1 TO 52;
     80    1 1     LIGNE_I(J)='I';
     90    1 1  END;

    110    1 0  GET EDIT ((X(J) DO J=1 TO 11)) (11F(4,2));
    120    1 0  GET EDIT (I) (SKIP,F(2));
    130    1 0  GET EDIT ((NAME(J) DO J=1 TO I)) (SKIP,12(A(2)));

    150    1 0  L=1;
    160    1 0  PATH_U(1)=1;
    170    1 0  DO N=1 TO I;
    180    1 1     GET EDIT (UI) (SKIP,F(2));
    190    1 1     DO WHILE (UI ¬= 0);
    200    1 2        U(L)=UI;
    210    1 2        L=L+1;
    220    1 2        GET EDIT (UI) (F(2));
    230    1 2     END;
    240    1 1     PATH_U(N+1)=L;
    250    1 1  END;

    270    1 0  PROB=R(U,PATH_U,I,X,P,0);

    290    1 0  PUT PAGE EDIT ('RELIABILITY FROM NODE A TO NODE D')
                       (X(15),A);
    310    1 0  PUT EDIT ((LIGNE_I(J) DO J=1 TO 52))
                       (SKIP(3),X(05),52A);
    330    1 0  PUT EDIT ('I','I','I','I')
                       (SKIP,X(05),A,X(13),A,X(11),A,X(24),A);
    350    1 0  PUT EDIT ('I','PATH NUMBER','I','PATH NAME','I',
                        'PATH USAGE PROBABILITY','I')
                       (SKIP(1),X(05),7(A,X(1)));
    380    1 0  PUT EDIT ('I','I','I','I')
                       (SKIP,X(05),A,X(13),A,X(11),A,X(24),A);
    400    1 0  PUT EDIT ((LIGNE_I(J) DO J=1 TO 52))
                       (SKIP,X(05),52A);
    420    1 0  DO N=1 TO I;
    430    1 1     PUT EDIT ('I','I','I','I')
                          (SKIP,X(05),A,X(13),A,X(11),A,X(24),A);
    450    1 1     PUT EDIT ('I',N,'I',NAME(N),'I',P(N),'I')
                          (SKIP,X(05),A,X(5),F(2),X(6),A,X(4),
                           A(2),X(5),A,X(8),F(8,6),X(6),A);
    480    1 1  END;
    490    1 0  PUT EDIT ('I','I','I','I')
                       (SKIP,X(05),A,X(13),A,X(11),A,X(24),A);
    510    1 0  PUT EDIT ((LIGNE_I(J) DO J=1 TO 52))
                       (SKIP,X(05),52A);
    530    1 0  PUT EDIT ('I','I','I')
                       (SKIP,X(05),A,X(25),A,X(24),A);
    550    1 0  PUT EDIT ('I','TERMINAL-PAIR','I','I')
                       (SKIP,X(05),A,X(6),A,X(6),A,X(24),A);
    570    1 0  PUT EDIT ('I','RELIABILITY','I',PROB,'I') (SKIP,
                        X(05),A,X(7),A,X(7),A,X(8),F(8,6),X(8),A);
    590    1 0  PUT EDIT ('I','I','I')
                       (SKIP,X(05),A,X(25),A,X(24),A);
    610    1 0  PUT EDIT ((LIGNE_I(J) DO J=1 TO 52))
                       (SKIP,X(05),52A);
    630    1 0  END PRINC;
```

RELIABILITY FROM NODE A TO NODE D

PATH NUMBER	PATH NAME	PATH USAGE PROBABILITY
1	P1	0.826500
2	L1	0.123500
3	P2	0.014807
4	L2	0.002213
5	L3	0.028980
6	P3	0.000409
7	L4	0.000061
8	P4	0.000003
9	L5	0.000000
10	L6	0.000005
11	L7	0.000002
12	L8	0.003520
TERMINAL-PAIR RELIABILITY		1.000000

ANNEX II

A PL/1 Program for Traffic Analysis in Circuit-Switched Networks

(Un programme en PL/1 pour l'analyse de la performance de réseaux commutés Téléphoniques)

ABSTRACT

In this report, the ETARET package is presented. In the first section, a user's guide is given in which will be found a description of the various options available, of the error messages and of input card formats. In the second section, a programmer's guide is given in which for each program involved the following are specified: a definition of the function reealized by the program, its logic diagram, a dictionary of variables used and its listing. Finally, an example of the use of this package is given.

RESUME

On présente dans ce rapport le package ETARET. La première section du rapport représente un guide à l'usager dans lequel on présente la description des options disponibles à l'usager, la description des messages d'erreur, ainsi que la description des cartes de données. Dans la deuxième section du rapport, on présente un guide du programmeur dans lequel on donne pour chaque programme les informations suivantes: une définition de la tâche qu'accomplit le programme, son diagramme logique, son dictionnaire des noms des variables employées ainsi que l'imprimé du programme. Pour terminer, on montre, par l'entremise d'un exemple d'utilisation, comment employer ce logiciel.

TABLE DES MATIERES

1. INTRODUCTION

Depuis qu'Alexandre Graham Bell a inventé le téléphone en 1876, cet appareil a connu un essor prodigieux. Aujourd'hui, il occupe une place de premier plan dans la vie de tous les jours des être humains. Que ce soit pour annoncer une bonne ou une mauvaise nouvelle, parler à un ami, règler des affaires, ..., le téléphone est le moyen de communication qui est le plus employé. D'ici quelques années, de nouveaux services tel VIDEOTEX seront offerts aux usagers via les réseaux téléphoniques. Il est donc indispensable que ces derniers maintiennent ou améliorent le niveau de la qualité de leur service. Par conséquent, il est très important de bien concevoir les nouveaux réseaux ou de bien analyser l'impact des modifications sur les réseaux existants.

Pour ce faire, il existe plusieurs techniques. Parmi les plus connues, on retrouve la simulation et la méthode analytique. La première nous permet, après avoir conçu un modèle du réseau téléphonique, de l'expérimenter sur un ordinateur digital pour en étudier le comportement. Pour ce qui est de la deuxième, cette dernière nous donne la possibilité de paramétriser le modèle du réseau téléphonique sous forme d'équations mathématiques. La résolution de ce modèle mathématique nous permet d'en dériver la solution analytique qui est utilisée pour prédire le comportement dudit réseau.

L'objectif de ce travail consistera à développer un outil informatique qui fera l'analyse des performances d'un réseau téléphonique quelconque. La technique d'analyse utilisée sera la méthode analytique.

2. GUIDE DE L'USAGER

A) Introduction

ETARET est un package qui effectue l'étude analytique d'un réseau téléphonique. Sa conception est basée sur le package STARTUP. Les buts de ce dernier sont l'analyse des performances et la mise à jour de la table d'acheminement des réseaux téléphoniques qui emploient la stratégie OOC avec SPILL FORWARD tout en considérant les temps d'établissement des appels comme étant négligeables.

Dans ETARET, nous n'avons conservé que la partie analyse des performances du package STARTUP. En fait, elle ne constitue que le squelette d'ETARET car de nombreuses modifications ont été effectuées. Parmi ces dernières, les plus importantes sont le remplacement des algorithmes d'acheminement (génération des arbres) et du calcul de la probabilité qu'un chemin soit utilisé.

Dans STARTUP, l'algorithme d'acheminement, en ne permettant qu'un noeud SPILL entre chaque Origine-Destination, n'observe qu'une partie de la définition de la stratégie OOC avec SPILL FORWARD et par conséquent ne représente qu'un cas très particulier d'acheminement dans les réseaux téléphoniques. Quant à l'algorithme du calcul de probabilité, ce dernier a été développé pour satisfaire les besoins du cas décrit ci-haut. De plus, il limite encore plus les types de réseaux traités puisqu'il n'autorise que cinq (5) chemins entre chaque paire de noeuds Origine-Destination.

Donc, l'aspect trop restrictifs des algorithmes nous ont forcés à en développer d'autres.

Etant donné que l'un des buts de notre package est de permettre l'analyse des performances de n'importe quel réseau téléphonique, nous avons implanté les trois (3) stratégies d'acheminement les plus connues soient: SOC, OOC et OOC avec SPILL FORWARD. Dans la version du package qui est présentée, cette dernière ne contient pas la stratégie OOC avec SPILL FORWARD puisqu'elle est présentement en développement.

Pour ce qui est du calcul de la probabilité qu'un chemin soit utilisé, nous avons implanté deux (2) algorithmes qui ont été conçus par M. VO-DAI. Ceux-ci sont expliqués dans la partie théorique de l'annexe I. Disons tout simplement que l'un de ces algorithmes (PROBOOC) est récursif et qu'il peut réaliser le calcul de la probabilité pour les trois (3) techniques d'acheminement que nous avons mentionnées précédemment. Quant à l'autre (PROBOOC) il correspond à une formule analytique qui peut être employée que pour la stratégie SOC. Cependant, elle est très avantageuse car elle nous permet de sauver du temps CPU.

Finalement, il est à noter que tous les autres programmes (MAIN, ACTGOS, BLKPRB, TGOL4 et SORT 1), pour accommoder les nouveaux algorithmes, ont subi des modifications. La plus importante est survenue dans le programme BLKPRB. Ce dernier calcule le GOS d'un lien en utilisant la formule d'ERLANG. De la manière dont cette dernière a été programmée, elle impose une limite quant au nombre de TRUNKS que l'on retrouve dans chaque lien. C'est pourquoi nous l'avons reprogrammée en utilisant un algorithmes de récurrence pour ladite formule.

B) Caractéristiques d'ETARET (options)

ETARET, dans la version qui est présentée dans ce rapport, est conçu pour traiter quinze (15) noeuds, cent cinq (105) liens et vingt-cinq (25) chemins entre chaque paire de noeuds Origine-Destination. Même si ces chiffres semblent très restrictifs quant aux réseaux qui peuvent être analysés par l'entremise du package, il est à noter que ces limites peuvent être levées en modifiant tout simplement les dimensions des vecteurs et des matrices employés dans les programmes du package.

Pour donner plus de flexibilité à l'usager, ETARET offre les options suivantes:

1. ENUMERATION DES CHEMINS

A partir de la table d'acheminement, les algorithmes PATHOOC et PATHSOC obtiennent, pour un réseau donné, tous les chemins entre chaque paire de noeuds Origine-Destination (Arbre d'acheminement). Les chemins sont générés comme une séquence de noeuds à partir de l'origine jusqu'à la destination. L'ordre de la génération des Origine-Destination est: 1→2, 1→3, ..., 1→N, 2→1, 2→3, ..., N→1, N→2, ..., N→(N-1)

NPPO est la variable qui contrôle l'option d'impression des arbres d'acheminement pour chaque paire de noeuds Origine-Destination. Lorsque,

NPPO ← 1, tous les chemins pour chaque paire de noeuds sont imprimés.

2. STRATEGIE D'ACHEMINEMENT

Partout où l'on a plus d'un chemin pour compléter un appel, il est nécessaire d'établir une stratégie d'acheminement pour déterminer les routes possibles ainsi que leur ordre de préférence. Il existe plusieurs règles pour le contrôle de l'appel. Dans cette version d'ETARET seulement deux (2) d'entre elles ont été retenues: SOC et OOC.

ROUSTR est la variable qui contient la stratégie d'acheminement choisie par l'utilisateur.

Lorsque,

ROUSTR ← 'SOC' , la stratégie d'acheminement suit les règles de SOC.

ROUSTR ← 'OOC' , la stratégie d'acheminement suit les règles d'OOC.

3. LA FIABILITE D'UN CHEMIN (PATH RELIABILITY) ET LE TRAFIC AMENE PAR CHAQUE CHEMIN (PATH CARRIED LOAD).

A partir de la table d'acheminement et du GOS (GRADE OF SERVICE) de chaque lien (TRUNK GROUP), on calcule le terme [PR(Pi used)] et le trafic amené par chaque chemin de l'arbre d'acheminement d'une paire de noeuds Origine-Destination.

NDPO (Detail Printing Option) est la variable qui contrôle l'option d'impression des valeurs qui ont été caluclées i.e la probabilité d'utilisation d'un chemin et le trafic amené par ledit chemin.

Lorsque,

NDPO ← 1, en se basant sur la table d'acheminement, la PR [Pi used] du i ième chemin et son trafic associé sont imprimés.

NDPO ← 0, l'option d'impression est supprimée.

4. LES UNITES DU TRAFIC

Les unités des valeurs que l'on retrouve dans la matrice du trafic sont soit des ERLANG soit des CCS (HUNDRED CALL SECONDS). Cependant, les sorties du programme sont toujours indiquées en termes de CCS.

LUNIT (LOAD UNIT) est la variable qui gère cette option

Lorsque,

LUNIT ← 'CCS', les valeurs que l'on retrouve dans la matrice du trafic sont considérées comme étant en unités de CCS (HUNDRED CALL SECOND).

LUNIT ← 'ERG', les valeurs que l'on retrouve dans la matrice du trafic sont considérées comme étant en unité d'ERLANG.

5. LA SURCHARGE DU TRAFIC OU LES HEURES DE POINTE (TRAFIC OVERLOAD)

La performance d'un réseau est généralement analysée en utilisant, entre les paires de noeuds, la valeur nominale du trafic. Cependant, durant une journée il y a toujours des heures de pointe et par conséquent, la surcharge du réseau est très fréquente.

Pour pouvoir analyser les effets des heures de pointe, nous avons inclus dans ETARET une option qui nous permet de modifier les valeurs contenues dans la matrice du trafic.

RATE est la variable qui contrôle cette option. Elle nous indique si l'on se retrouve dans une condition normale, de surcharge (OVERLOAD) ou de sous-charge (UNDERLOAD) du réseau.

Chaque élément de la matrice est multiplié par cette variable pour former une nouvelle matrice. Par exemple, s'il y a une augmentation de 10% du trafic à une heure quelconque de la journée, la variable RATE sera initialisée à la valeur 1.1.

La spécification de la condition dans laquelle on se retrouve est faite comme suit:

RATE ← q

ou,

q > 1 indique que le réseau se retrouve dans une condition de surcharge.

q = 1 indique que le réseau se retrouve dans une condition normale.

5. LA SURCHARGE DU TRAFIC OU LES HEURES DE POINTE (TRAFIC OVERLOAD)

La performance d'un réseau est généralement analysée en utilisant, entre les paires de noeuds, la valeur nominale du trafic. Cependant, durant une journée il y a toujours des heures de pointe et par conséquent, la surcharge du réseau est très fréquente.

Pour pouvoir analyser les effets des heures de pointe, nous avons inclus dans ETARET une option qui nous permet de modifier les valeurs contenues dans la matrice du trafic.

RATE est la variable qui contrôle cette option. Elle nous indique si l'on se retrouve dans une condition normale, de surcharge (OVERLOAD) ou de sous-charge (UNDERLOAD) du réseau.

Chaque élément de la matrice est multiplié par cette variable pour former une nouvelle matrice. Par exemple, s'il y a une augmentation de 10% du trafic à une heure quelconque de la journée, la variable RATE sera initialisée à la valeur 1.1.

La spécification de la condition dans laquelle on se retrouve est faite comme suit:

RATE ← q

ou, $q > 1$ indique que le réseau se retrouve dans une condition de surcharge.

$q = 1$ indique que le réseau se retrouve dans une condition normale.

$q < 1$ indique que le réseau se retrouve dans une condition où la quantité de trafic dans le réseau est sous la condition normale.

Finalement, il peut être intéressant de vérifier l'effet, sur le GOS du réseau, de plusieurs conditions; telles que la surcharge de 10%, de 20% et de 30% et la sous-charge de 5%, de 15% et de 20%. Nous avons donc inclus dans ETARET une option qui permet, pour différentes valeurs de la variable RATE, la reprise de tous les calculs.

NBTAUX est la variable qui contrôle cette option. Elle contient le nombre de répétitions des calculs i.e le nombre de conditions que l'on veut vérifier.

6. LA VARIABLE BPMAX

BPMAX est une variable qui nous indique la probabilité de blocage maximale pour toutes les paires de noeuds Origine-Destination.

C) Messages d'erreur

Dans ETARET, on retrouve plusieurs messages d'erreurs. Ces derniers peuvent être facilement interprétés par un usager.

Un message est imprimé quand un de ces évènements survient:

A- Les cartes contenues dans les cartes de données sont hors séquence;

B- Une carte décrivant la configuration du réseau ou de la table d'acheminement a été oubliée;

C- La table d'acheminement contient des chemins illégaux.

Lorsqu'une erreur est détectée dans les cartes de données ou dans la table d'acheminement (ROUTING TABLE), on termine l'exécution du programme. Tout dépendant de l'erreur diagnostiquée, un des messages suivants apparaît:

A- "TRAFFIC LOAD UNITS INCORRECTLY SPECIFIED
EXECUTION TERMINATED"

B- "ROUTING STRATEGY INCORRECTLY SPECIFIED
EXECUTION TERMINATED"

C- "THE ORIGINAL ROUTING TABLE HAS AT LEAST ONE LOOP IN IT.
LOOP WAS ENCOUNTRED ON THE PATHS FROM __ TO __.
EXECUTION OF THE PROGRAM HAS BEEN TERMINATED"

Un autre message d'erreur peut être rencontré quand le vecteur contenant le GOS de chaque lien (TRUNK GROUP) ne converge pas vers une valeur. Cette dernière correspond à une limite et est initialisée dans le programme. La forme du message est:

AFTER 20 ITERATIONS ALL ACTUEL TRUNK GROUP GOS COULD NOT BE DETERMINED WITHIN AN ERROR BOUND OF __.
THE TRUNK GROUP GOS ARE ASSUMED TO HAVE BEEN DETERMINED WITHIN AN ERROR BOUND OF__.

Lorsque cette erreur est rencontrée, on ne termine pas l'exécution du programme.

D. Description des cartes de données

Dans cette partie du rapport, les cartes de données requises pour obtenir l'analyse des performances d'un réseau sont présentées.

Avant de préparer les cartes de données, il faut s'assurer que les étapes suivantes ont été réalisées.

ETAPE 1:

Numéroter tous les noeuds du réseau de 1 à NS et tous les liens de 1 à NL. NS et NL sont respectivement le nombre de noeuds et de liens que l'on retrouve dans le graphe du réseau.

ETAPE 2

Attribuer à chaque lien un nombre de tronçons (TRUNKS).

ETAPE 3

Si vous ne disposez pas d'une table d'acheminement pour le réseau alors en préparer une tout en vous assurant qu'elle ne contient pas de chemin illégal.

ETAPE 4

Obtenir la matrice du trafic pour le réseau.

ETAPE 5

Choisir la valeur de BPMAX et les options que vous voulez utiliser.

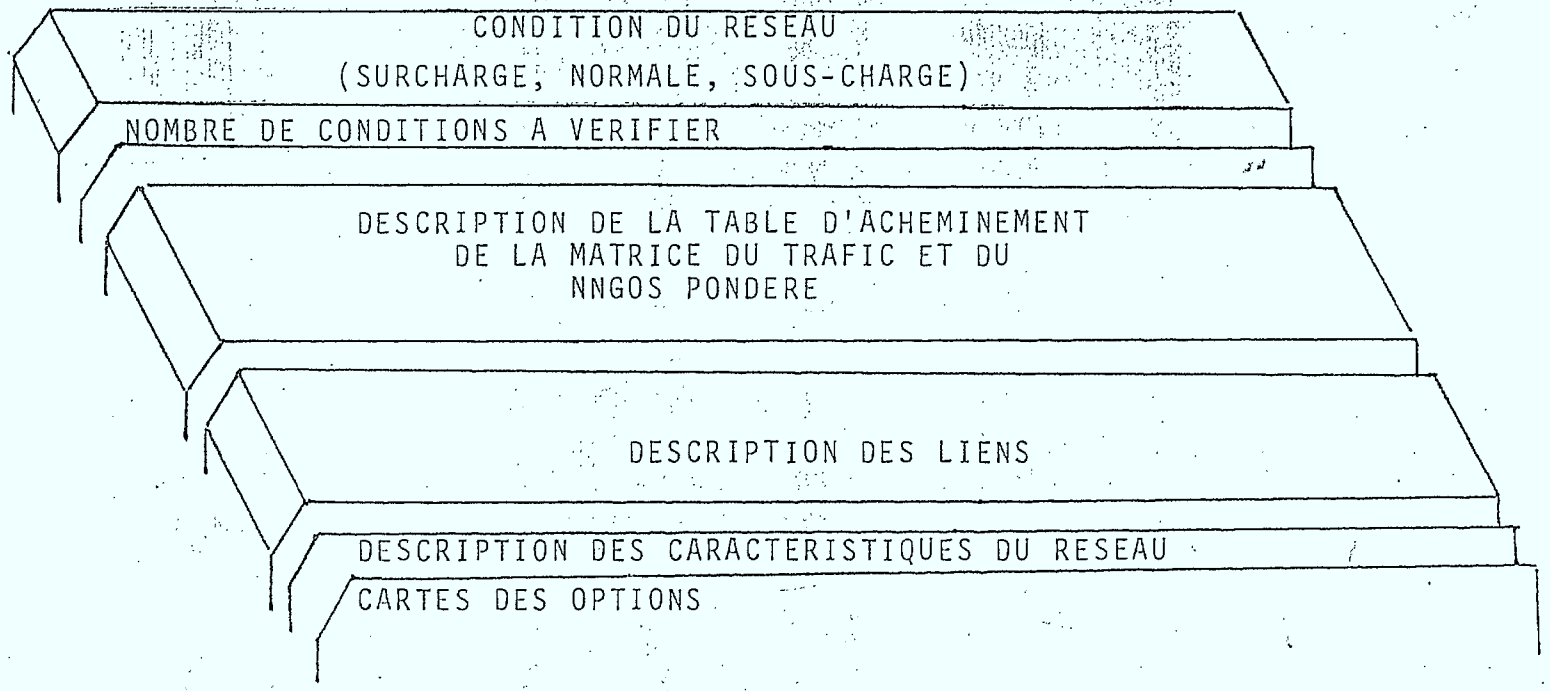


FIGURE 1 - SEQUENCE DES CARTES DE DONNEES DU PACKAGE ETARET

La procédure pour traduire ces informations en cartes de données est simple. La figure 1 nous montre la séquence des cartes de données du package ETARET. Dans les pages qui vont suivre, nous allons vous donner la description de chaque carte de données.

NUMERO DE LA CARTE	DESCRIPTION	NUMERO DES COLONNES
1	CARTES DES OPTIONS	
	A- IMPRESSION DES CHEMINS	
	Si cette option est désirée, on place dans ce champ la valeur 1 (NPPO ← 1) autrement on place la valeur 0 (NPPO ← 0).	1
	B- PROBABILITE D'UTILISATION DES CHEMINS AINSI QUE LE TRAFIC AMENE PAR CES DERNIERS	
	Si cette option est désirée, on place dans ce champ la valeur 1 (NDPO ← 1) autrement on place la valeur 0 (NDPO ← 0).	2
	C- CHOIX DE L'UNITE DU TRAFIC	
	1- On place dans ce champ la chaîne de caractères 'CCS' si le trafic est en unités de CCS (HUNDRED CALL SECONDS)	3 à 5
	2- On place dans ce champ la chaîne de caractères 'ERG' si le trafic est en unités d'ERLANG.	
	D- CHOIX DE LA STRATEGIE D'ACHEMINEMENT	
	1- On place dans ce champ la chaîne de caractères 'SOC', si l'acheminement du réseau suit les règles de la stratégie SOC.	6 à 8

NUMERO DE LA CARTE	DESCRIPTION	NUMERO DES COLONNES
	2- On place dans ce champ la chaîne de caractères 'OOC' si l'acheminement du réseau suit les règles de la stratégie OOC.	
2	DESCRIPTION DES CARACTERISTIQUES DU RESEAU	
	A- On place dans ce champ le nombre de noeuds (NS) que l'on retrouve dans le graphe du réseau.	1 à 2
	B- On place dans ce champ le nombre de liens (NL) que l'on retrouve dans le graphe du réseau.	3 à 5
	C- On place dans ce champ la valeur maximale permise pour la probabilité de blocage (BPMAX).	6 à 11
3 à (3+(NL-1))	DESCRIPTION DES LIENS	
	A- Dans ce champ , on indique les deux (2) noeuds auxquels se rattache un lien	1 à 4
	1- Premier Noeud	1 à 2
	2- Deuxième Noeud	3 à 4
	B- On place dans ce champ le numéro du lien	5 à 7

NUMERO DE LA CARTE	DESCRIPTION	NUMERO DES COLONNES
	C- On place dans ce champ , le nombre de tronçons (TRUNK) contenu dans le lien.	9 à 15
(3 + NL) à [(3 +NL) + (NS (NS-1)-1)]	DESCRIPTION DE LA TABLE D'ACHEMINEMENT DE LA MATRICE DU TRAFIC ET DU NNGOS PONDERE (NNGOS WEIGHT)	
	A- Dans ce champ on indique le numéro du noeud d'origine (IS).	1 à 2
	B- Dans ce champ on indique le numéro du noeud destination (ID).	3 à 4
	C- Dans ce champ , on indique le numéro du noeud qui correspond au premier noeud (Noeud adjacent ou noeud d'origine) de la route primaire entre une paire de noeuds ORIGINE-DESTINATION.	5 à 7
	D- Dans ce champ , on indique le numéro du noeud qui correspond au premier noeud (Noeud adjacent au noeud d'origine) de la première route secondaire entre une paire de noeuds ORIGINE-DESTINATION. S'il n'existe pas de première route secondaire, alors on place dans ce champ la valeur 0.	8 à 10
	E- Dans ce champ , on indique le numéro du noeud qui correspond au premier noeud (noeud adjacent au noeud d'origine) de la deuxième route secondaire entre une paire de noeuds ORIGINE-DESTINATION.	11 à 13

NUMERO DE LA CARTE.	DESCRIPTION	NUMERO DES COLONNES
	E- SUITE. S'il n'existe pas de deuxième route secondaire alors on place dans ce champ la valeur 0.	
	F- Dans ce champ on indique le trafic amené de l'origine (IS) à la destination (ID).	14 à 19
	G- Dans ce champ on indique le facteur de pondération du NNGOS pour la paire de noeuds ORIGINE (IS)-DESTINATION (ID).	20 à 25
	N.B. Il est à noter que dans cette version d'ETARET, un appel qui est acheminé d'une origine à une destination peut, à chaque noeud de son trajet, avoir, pour pouvoir continuer son trajet, au plus trois (3) choix (i.e qu'il peut aller à au plus trois (3) noeuds qui lui sont adjacents).	
[3 + NL + (NS (NS-1))]	Sur cette carte, on indique le nombre de conditions (surcharge et/ou sous-charge du réseau) que l'on veut vérifier (NBTAUX)	1 à 2
[4 + NL + (NS (NS 1)) à (4 + NL + (NS (NS-1)) + (NBTAUX-1)]	CONDITION DU RESEAU Sur cette carte, on indique la condition du réseau	1 à 6

NUMERO DE LA CARTE	DESCRIPTION	NUMERO DES COLONNES

RATE.

1. Si RATE > 1 alors on se retrouve dans une condition de surcharge.

2. Si RATE = 1 alors on se retrouve dans une condition normale.

3. Si RATE < 1 alors on se retrouve dans une condition de sous-charge.

3. GUIDE DU PROGRAMMEUR

A. Introduction

A partir du package STARTUP et de la théorie présentée dans l'annexe I du rapport, un programme utilisant un modèle analytique, a été conçu. Ce dernier est appelé ETARET (Etude Analytique d'un Réseau Téléphonique).

Le langage PL/1 a été employé pour la programmation. Celle-ci s'est effectuée sur un ordinateur IBM 370/158.

ETARET est un package qui effectue l'analyse des performances d'un réseau téléphonique. Ce dernier peut employer comme stratégie d'acheminement soit OOC soit SOC.

B) Structure des programmes

ETARET est composé d'un programme principal et de huit sous-programmes. La figure 2 nous donne un aperçu de la disposition des programmes. Les lignes dans cette figure illustrent la structure des appels entre les sous-programmes.

Les cases pointillées renfermant chacune deux sous-programmes indiquent, selon la stratégie d'acheminement choisie, quel est le sous-programme qui est appelé.

Si la stratégie d'acheminement sélectionnée est:

1) SOC

a) Pour générer les arbres d'acheminement, on utilise le sous-programme PATHSOC

b) Pour calculer la probabilité qu'un chemin soit utilisé on emploie le sous-programme PROBSOC.

2) OOC

a) Pour générer les arbres d'acheminement, on utilise le sous-programme PATHOOC

b) Pour calculer la probabilité qu'un chemin soit utilisé on emploie le sous-programme PROBOOC.

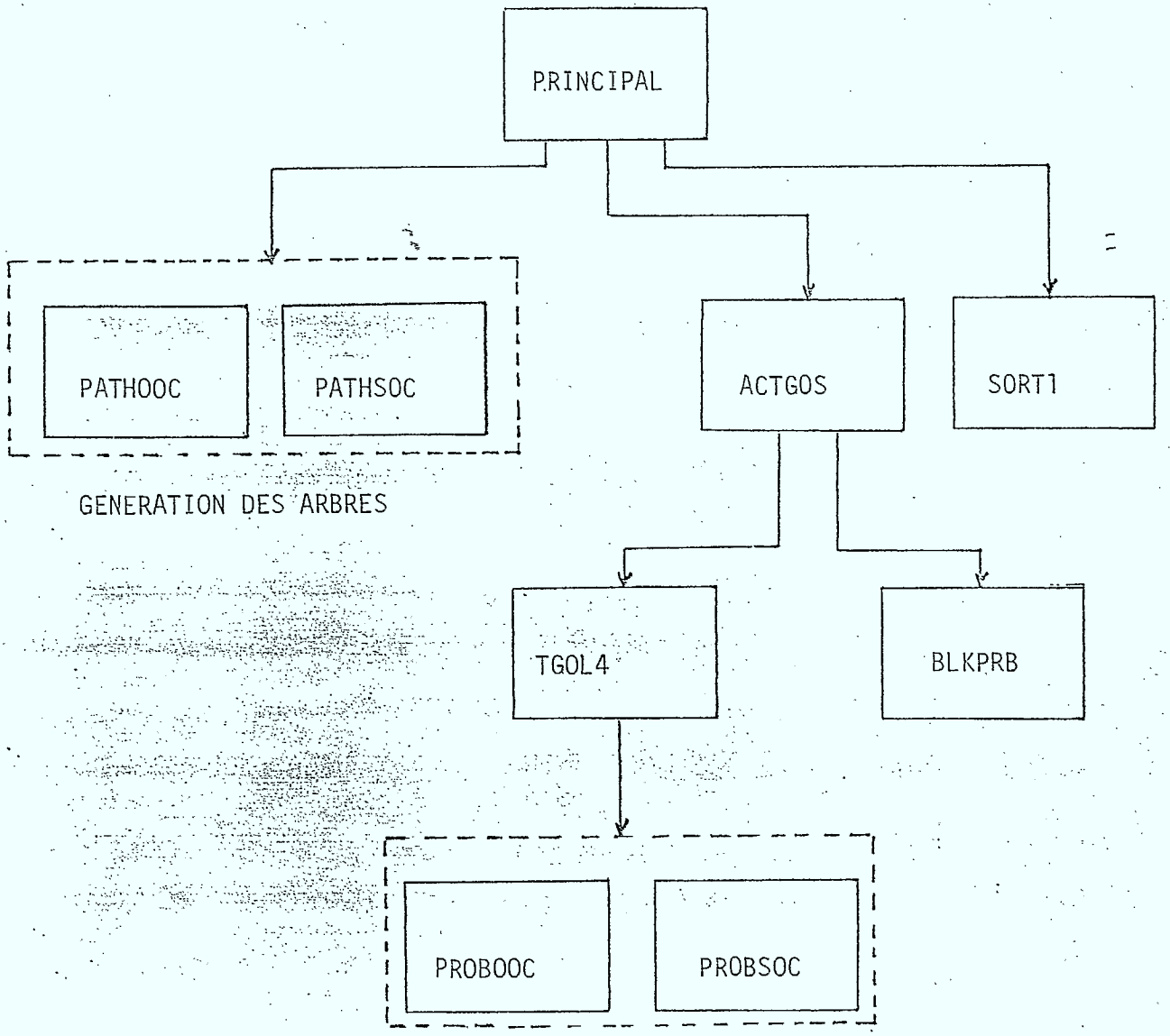


FIGURE 2 STRUCTURE DES PROGRAMMES DU PACKAGE ETARET

C) EXPLICATION DES PROGRAMMES

Dans cette section nous donnerons, pour chaque programme du package ETARET, une définition de la tâche qu'il accomplit, son diagramme logique, son dictionnaire des variables ainsi que son "listing".

La figure 3 donne la signification des symboles logiques que nous avons utilisés pour construire les diagrammes logiques.

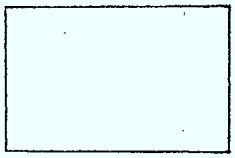
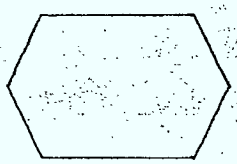
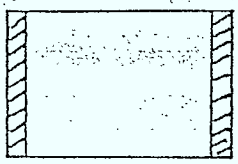
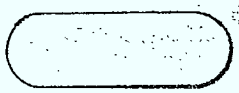
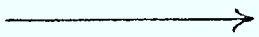
SYMBOLES	SIGNIFICATIONS
	Le rectangle est employé pour définir une opération réalisant une action définie
	Ce symbole caractérise le diagramme logique. Il sert à indiquer une décision, un test ou une comparaison. Il possède une entrée et deux sorties. (oui et non). Si la condition est satisfaite il branche au traitement indiqué par la porte <u>oui</u> autrement il branche au traitement indiqué par la porte <u>non</u>.
	Appel du sous-programme qui porte le nom 'nom'.
	Le cercle sert à indiquer un branchement à une autre partie du diagramme logique.
	Le symbole terminal permet de spécifier des opérations de départ et d'arrêt d'opérations.

FIGURE 3



Ces lignes se terminant par une flèche joignent des symboles des classes précédentes. Elles servent à indiquer le sens du cheminement de l'information. Elles peuvent être horizontales ou verticales.

FIGURE 3 (FIN)

1) Programme principal

définition de la tâche

Le programme PRINCIPAL du package ETARET est celui qui dirige l'analyse des performances. Sa première fonction est de lire les données du réseau. Par la suite, selon la stratégie d'acheminement choisie, il appelle le sous-programme qui génère les arbres d'acheminement pour chaque paire de noeuds ORIGINE-DESTINATION. Ensuite si les chemins définis par la table d'acheminement sont sans erreur, il effectue, par l'entremise de quelques sous-programmes, l'analyse des performances du réseau. Pour terminer, il imprime certaines statistiques du réseau telles que le GOS du réseau et la paire de noeuds ORIGINE-DESTINATION possèdant le plus grand NNGOS.

La figure 4 présente le diagramme logique de ce programme.

DIAGRAMME LOGIQUE DU PROGRAMME PRINCIPAL

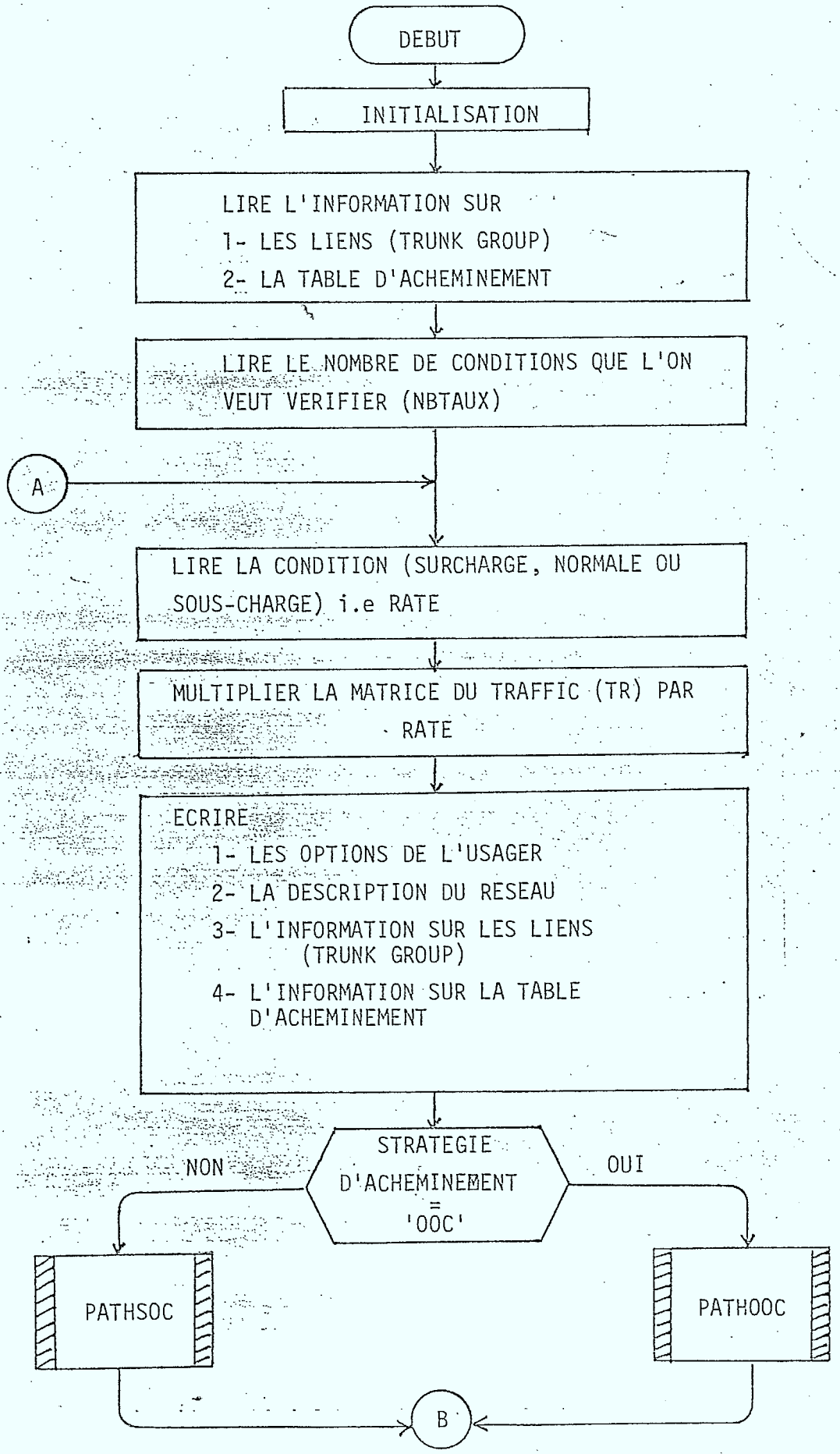


FIGURE 4

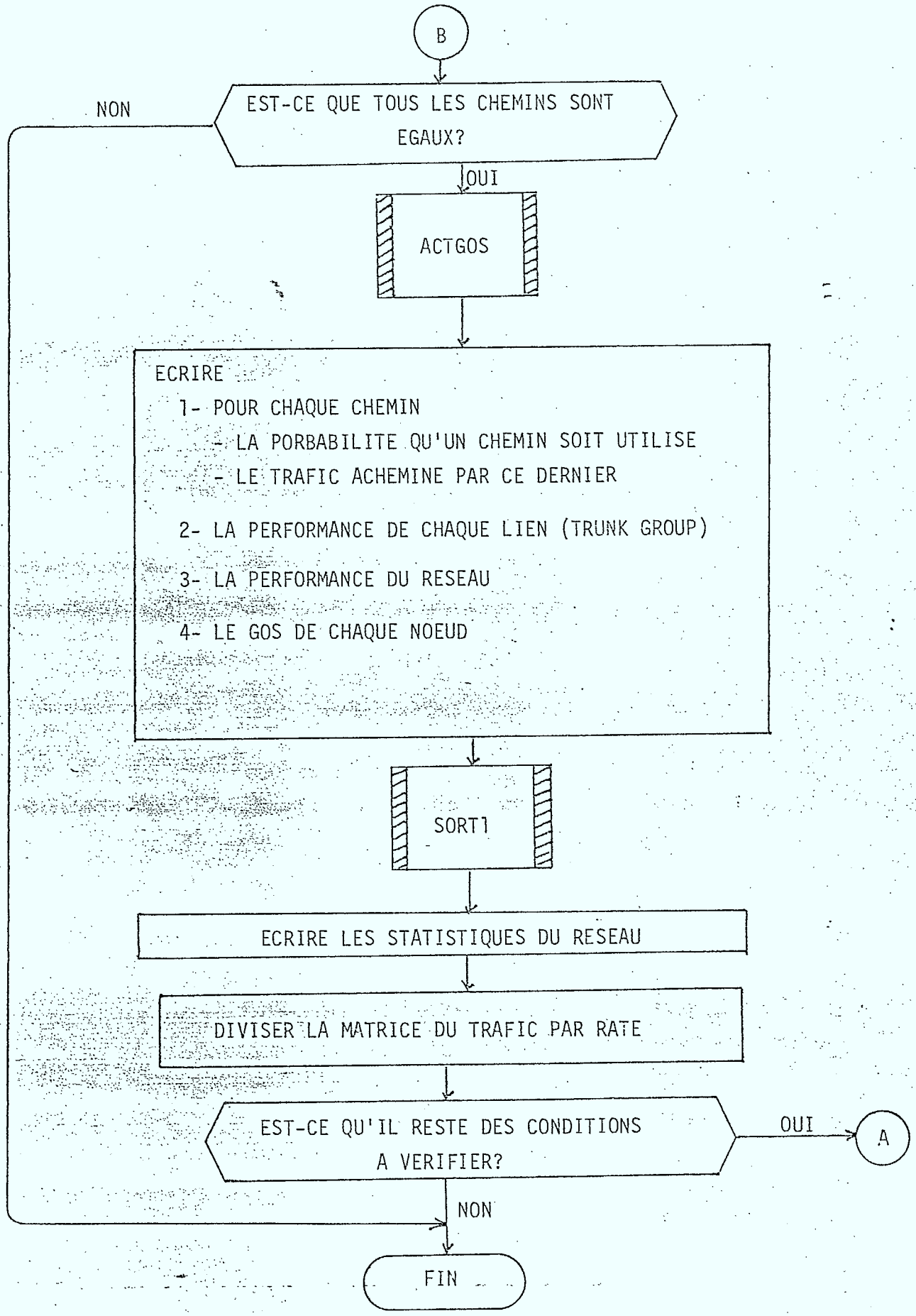

B
EST-CE QUE TOUS LES CHEMINS SONT EGAUX?
NON
OUI
ACTGOS
ECRIRE
1- POUR CHAQUE CHEMIN
- LA PORBABILITE QU'UN CHEMIN SOIT UTILISE
- LE TRAFIC ACHEMINE PAR CE DERNIER
2- LA PERFORMANCE DE CHAQUE LIEN (TRUNK GROUP)
3- LA PERFORMANCE DU RESEAU
4- LE GOS DE CHAQUE NOEUD
SORT1
ECRIRE LES STATISTIQUES DU RESEAU
DIVISER LA MATRICE DU TRAFIC PAR RATE
EST-CE QU'IL RESTE DES CONDITIONS A VERIFIER?
OUI
A
NON
FIN

FIGURE 4 (FIN)

DICTIONNAIRE DES VARIABLES DU PROGRAMME PRINCIPAL

BIGBP = variable qui contient la plus grande probabilité de blocage rencontrée pour une paire de noeuds ORIGINE-DESTINATION.

BP = la non-fiabilité d'un lien

BPMAX = variable contenant, pour toutes les paires de noeuds ORIGINE-DESTINATION, la probabilité maximale de blocage autorisée

CLS(I) = le trafic total qui est perdu au noeud I

CLDS(IS,ID) le trafic (en CCS) qui est perdu du noeud IS au noeud ID

CSSUM = le trafic (en CCS) qui est perdu dans le réseau

CTSUM = le trafic (en CCS) provenant (originating) du réseau

FLTG (Ln) = le trafic (en CCS) offert au lien Ln

GOS (Ln) la fiabilité du lien Ln. Il est à noter que le "GRADE OF SERVICE" du lien Ln est égal à 1-GOS (Ln)

GOSNTK = variable qui contient le "GRADE OF SERVICE" du réseau

ICON = variable qui nous indique si un chemin défini par la table d'acheminement contient une boucle (I Con= 1).
Si ICON = 0 alors tous les chemins sont légaux.

IRT (Is,Id,I) =structure contenant la table d'acheminement du réseau

LNUM(Is,Id) = contient le numéro du lien que se situe entre les noeuds IS et ID

LOCATI(Is,Id,N) = contient la position dans le vecteur NPATH où commence le $N^{ième}$ chemin entre la paire de noeuds ORIGINE-DESTINATION IS et ID

LOCATF (Is,Id,N) =contient la position dans le vecteur NPATH où se termine le $N^{ième}$ chemin entre la paire de noeuds ORIGINE-DESTINATION IS et ID

NPATH (I) = structure qui contient les numéros des noeuds constituant les chemins du réseau. Ces derniers ont été trouvés à partir de la table d'acheminement

LUNIT = variable nous indiquant l'unité de trafic qui a été choisie

NBTAUX = variable qui contient le nombre de conditions que l'on veut vérifier

NPPO = variable nous indiquant si les chemins seront imprimés (NPPO =1)

NL = nombre de liens que l'on retrouve dans le réseau

NDPO = variable nous indiquant si pour chaque chemin leur probabilité d'utilisation ainsi que le trafic qu'il amène seront imprimés (NDPO= 1).

NS = nombre de noeuds que contient le réseau

NTRK (Ln)=contient le nombre de tronçons qui se retrouvent dans le lien LN

P (I) = contient, pour une paire de noeuds ORIGINE-DESTINATION, la probabilité d'utilisation du $I^{ième}$ chemin

PL (IS, ID,K)= le trafic (en CCS) amené par le $K^{ième}$ chemin de la paire de noeuds ORIGINE-DESTINATION IS et ID

PR(IS,ID,K) = probabilité que le $K^{ième}$ chemin de la paire de noeuds ORIGINE-DESTINATION IS et ID soit utilisé

RATE = variable qui contient la condition du réseau

ROUSTR = variable nous indiquant la stratégie d'acheminement qui a été choisie ('SOC' et 'OOC')

SBP = le "grade of service" d'un noeud

SDBP = la probabilité de blocage pour une paire de noeuds ORIGINE-DESTINATION (NNGOS)

STR (I) = le trafic total (ORIGINATING) qui provient du noeud I

TR (IS, ID) = structure qui contient la matrice du trafic du réseau

WCSSUM = le trafic pondéré (en CCS) qui est perdu dans le réseau

WGOSNK = variable qui contient le "grade of service" pondéré du réseau

WSDBP = la probabilité de blocage pondérée pour une paire de noeuds ORIGINE-DESTINATION (Weighted NNGOS)

WT (IS, ID) = facteur par lequel la probabilité de blocage du trafic qui est envoyé du noeud IS au noeud ID sera multipliée pour calculer la probabilité de blocage pondérée.

LISTING DU PROGRAMME PRINCIPAL

```
I OPTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

                    SOURCE LISTING

NUMBER LEV NT

  10        0   PRINC:PROCEDURE OPTIONS(MAIN);

                /*****************************************************************/
                /*                                                               */
                /* OBJECTIFS:                                                    */
                /*       1- DIRIGER L'ANALYSE DES PERFORMANCES D'UN RESEAU       */
                /*          QUELCONQUE                                           */
                /*       2- IMPRIMER DES STATISQUES SUR LE RESEAU                */
                /*                                                               */
                /*****************************************************************/

  120   1   0   DECLARE (P(25),GOS(105),FLTG(105),
                         TR(15,15),WT(15,15),CLSD(15,15),
                         PL(15,15,25),PR(15,15,25))          FLOAT(16)      EXTERNAL,
                        (NS,NPATH(5000),
                         IRT(15,15,4))                       FIXED(2)       EXTERNAL,
                        (NL,LNUM(15,15))                     FIXED(3)       EXTERNAL,
                        (LOCATI(15,15,25),LOCATF(15,15,25)) FIXED(4)       EXTERNAL,
                         NTRK(105)                           FIXED(7)       EXTERNAL,
                        (LUNIT,ROUSTR)                       CHAR(3)        EXTERNAL,
                        (W,BP,ERR,SBP,DUM,SDBP,DUM1,DUM2,
                              CSSUM,WSDBP,BPMAX,BIGBP,
                         CTSUM,WCSSUM,GOSNTK,WGOSNK,
                         CLS(15),STR(15),RATE)               FLOAT(16),
                        (ICON,NDPO,NPPO)                     FIXED(1),
                        (IS,ID,IDUM1,IDUM2,IDUM3,NBTAUX)     FIXED(2),
                        (I,J,K,LN,NEXD,II,II1,II2)           FIXED(3),
                         SWTG(210)                           FIXED(2),
                         IDUM                                FIXED(7),
                         KOUNTK                              FIXED(4),
                         NPRINT                              CHAR(3);

                /* LIRE ET ECRIRE LES OPTIONS                                    */

  370   1   0   GET EDIT (NPPO,NDPO,LUNIT,ROUSTR) (SKIP,F(1),F(1),A(3),A(3));
  380   1   0   PUT PAGE EDIT(NPPO,NDPO,LUNIT,ROUSTR) (2(X(1),F(1)),2(X(1),A(3)));

                /* VERIFIER SI LES UNITES DU TRAFIC SONT SPECIFIES EN CCS        */
                /* OU EN ERLANG                                                  */

  450   1   0   IF ((LUNIT='CCS') | (LUNIT='ERG'))
                THEN DO;

                /* LIRE ET ECRIRE LES CARACTERISTIQUES DU RESEAU                 */
```

```
( OPTIMIZING COMPILER          PPINC:PROCEDURE OPTIONS(MAIN);

NUMBER LEV NT

   520   1  1          GET EDIT (NS,NL,BPMAX)
                             (SKIP,F(2),F(3),F(6,4));
   540   1  1          PUT EDIT(NS,NL,BPMAX)
                             (SKIP,F(2),F(3),F(7,4));


                 /* INITIALISATION DES MATRICES CREUSES                    */

   610   1  1          DO I=1 TO NS;
   620   1  2             CLS(I)=0;
   630   1  2             STR(I)=0;
   640   1  2             DO J=1 TO NS;
   650   1  3                LNUM(I,J)=0;
   660   1  3                WT(I,J)=1;
   670   1  3             END;
   680   1  2             LNUM(I,NS+1)=NL+I;
   690   1  2             GDS(I+NL)=1;
   700   1  2          END;


                 /* LIRE ET ECRIRE LA DESCRIPTION DES LIENS                */

   760   1  1          DO I=1 TO NL;
   770   1  2             GET EDIT(IS,ID,LN,IDUM)
                                (SKIP(1),2(F(2)),F(3),X(1),F(7));
   790   1  2             PUT EDIT(IS,ID,LN,IDUM)
                                (SKIP,X(1),2(F(2)),F(3),X(6),F(7));
   810   1  2             LNUM(IS,ID)=LN;
   820   1  2             LNUM(ID,IS)=LN;
   830   1  2             NTRK(LN)=IDUM;
   840   1  2             SWTG(LN)=IS;
   850   1  2             K=I05+LN;
   860   1  2             SWTG(K)=ID;
   870   1  2             GDS(I)=.9;
   880   1  2          END;


                 /* LIRE ET ECRIRE LA DESCRIPTION DE LA TABLE D'ACHEMINE-  */
                 /* MENT, DE LA MATRICE DU TRAFIC ET DU FACTEUR DE PON-    */
                 /* DERATION DU NNGOS                                      */


   960   1  1          IDUM=NS*(NS-1);
   970   1  1          DO I=1 TO IDUM;
   980   1  2             GET EDIT (IS,ID,IDUM1,IDUM2,IDUM3,DUM,W)
                                (SKIP,2(F(2)),3(F(3)),2(F(6,2)));
  1000   1  2             PUT EDIT (IS,ID,IDUM1,IDUM2,IDUM3,DUM,W)
                                (SKIP,X(1),2(F(2)),3(F(3)),2(F(6,2)));
  1020   1  2             IRT(IS,ID,1)=IDUM1;
  1030   1  2             IRT(IS,ID,2)=IDUM2;
  1040   1  2             IRT(IS,ID,3)=IDUM3;
  1050   1  2             IRT(IS,ID,4)=0;
```

```
OPTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

ABF LEV NT

1060    1   2                IF LUNIT = 'ERG' THEN
                                 DUM=DUM*36;
1080    1   2                ELSE;
1090    1   2                TR(IS,ID)=DUM;
1100    1   2                IF W ¬= 0 THEN
                                 WT(IS,ID)=W;
1120    1   2                ELSE;
1130    1   2            END;

                /*************************************************************/
                /*                                                           */

                /* LIRE LE NOMBRE DE CONDITIONS A VERIFIER                   */

1230    1   1   GET EDIT (NBTAUX) (SKIP,F(2));
1240    1   1   DO II=1 TO NBTAUX;

                /* LIRE UNE CONDITION                                        */

1300    1   2       GET EDIT (RATE) (SKIP,F(6,2));

                /* MULTIPLIER LA MATRICE DU TRAFIC PAR LA VALEUR DE LA       */
                /* CONDITION                                                 */

1370    1   2       DO II1=1 TO NS;
1380    1   3           DO II2=1 TO NS;
1390    1   4               TR(II1,II2)=TR(II1,II2)*RATE;
1400    1   4           END;
1410    1   3       END;

                /*                                                           */
                /*************************************************************/

                /* ECRIRE LES OPTIONS DE L'USAGER                            */

1510    1   2       PUT PAGE EDIT ('T E L E P H O N E   N E T W O R K   ',
                                   'A N A L Y T I C A L   S T U D Y')
                                  (COL(34),2(A));
1540    1   2       PUT EDIT ('=========================================',
                              '==========')
                             (SKIP(1),COL(34),2(A));
1570    1   2       PUT EDIT ('USER SPECIFIED OPTIONS')
                             (SKIP(10),COL(28),A);
1590    1   2       PUT EDIT ('----------------------')
```

```
OPTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

BER LEV NT

                                (SKIP(1),COL(28),A);
 610   1  2                PUT EDIT ('ROUTING STRATEGY',':',ROUSTR)
                                (SKIP(6),COL(35),A,COL(93),A,X(2),A(3));
 630   1  2                NPRINT='YES';
 640   1  2                IF NDPO = 0 THEN
                              NPRINT='NO ';
 660   1  2                ELSE;
 670   1  2                PUT EDIT ('LIST OF PATHS REQUIRED',':',NPRINT)
                                (SKIP(3),COL(35),A,COL(93),A,X(2),A(3));
 690   1  2                NPRINT='YES';
 700   1  2                IF NDPO=0 THEN
                              NPRINT='NO ';
 720   1  2                ELSE;
 730   1  2                PUT EDIT ('DETAILS OF PATH RELIABILITY AND PATH CARRIED LOAD',
                                ' REQUIRED:',NPRINT)
                                (SKIP(3),COL(35),2(A),X(2),A(3));
 760   1  2                PUT EDIT ('MAXIMUM ALLOWABLE BLOCKING PROBABILITY(BPMAX)',':',
                                BPMAX)
                                (SKIP(3),COL(35),A,COL(93),A,X(2),F(6,3));

          /* ECRIRE LA DESCRIPTION DES CARACTERISTIQUES DU RESEAU    */

 840   1  2                PUT PAGE EDIT ('NETWORK DESCRIPTION','----------')
                                (COL(28),A,SKIP(1),COL(28),A);
 860   1  2                PUT EDIT ('NUMBER OF SWITCHES IN NETWORK',':',NS)
                                (SKIP(6),COL(35),A,COL(68),A,X(2),F(2));
 880   1  2                PUT EDIT ('NUMBER OF TRUNK GROUPS IN NETWORK:',NL)
                                (SKIP(3),COL(35),A,X(2),F(3));
 900   1  2                PUT EDIT ('RATE',':',RATE)
                                (SKIP(3),COL(35),A,COL(68),A,X(2),F(6,2));
 920   1  2                PUT EDIT ('EXISTING NUMBER','LINK NUMBER','SWITCH PAIR',
                                'OF TRUNKS')
                                (SKIP(6),COL(83),A,SKIP(1),COL(35),A,X(13),A,X(16),
                                A,SKIP(3));

          /* ECRIRE LA DESCRIPTION DES LIENS                         */

2010   1  2                DO I=1 TO NL;
2020   1  3                   K=I+105;
2030   1  3                   PUT EDIT (I,SWTG(I),SWTG(K),NTRK(I))
                                (SKIP(1),COL(39),F(3),X(18),F(2),X(5),F(2),X(18),
                                F(7));
2060   1  3                END;
2070   1  2                PUT PAGE EDIT('SWITCH-TO-SWITCH TRAFFIC','ROUTING TABLE',
                                '----------','----------')
                                (COL(13),A,X(41),A,SKIP(1),COL(13),A,X(41),A);
2100   1  2                PUT EDIT ('FROM','TO','TRAFFIC IN CCS','CHOICE 1','CHOICE 2',
                                'CHOICE 3','WEIGHT')
                                (SKIP(2),COL(13),6(A,X(9)),A);
```

```
OPTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

ER LEV NT


                /* ECRIRE LA DESCRIPTION DE LA TABLE D'ACHEMINEMENT DE LA  */
                /* MATRICE DU TRAFIC ET DU FACTEUR DE PONDERATION DU NNGOS */

190     1   2           DO I=1 TO NS;
200     1   3             DO J=1 TO NS;
210     1   4               IF I ¬= J THEN
                              PUT EDIT(I,J,TR(I,J),IRT(I,J,1),IRT(I,J,2),
                                IRT(I,J,3),WT(I,J))
                             (SKIP(1),COL(14),F(2),X(10),F(2),X(12),
                              F(10,2),X(13),2(F(3),X(14)),F(3),X(12),
                              F(6,2));
270     1   4               ELSE;
280     1   4             END;
290     1   3           END;


                /* VERIFIER S'IL FAUT IMPRIMER LES CHEMINS                 */

350     1   2           IF NPPO ¬= 0 THEN
                          PUT PAGE EDIT('PATHS DEFINED BY ORIGINAL ROUTING TABLE',
                                        '-----------------------------------')
                             (COL(28),A,SKIP(1),COL(28),A);
390     1   2           ELSE;


                /* VERIFIER S'IL Y A, DANS LA TABLE D'ACHEMINEMENT, DES    */
                /* CHEMINS ILLEGAUX                                        */

460     1   2           ICON=0;
470     1   2           IF ROUSTR = 'DOC' THEN
                          CALL PATHDOC(NPPO,ICON,IS,ID);
490     1   2           ELSE
                          CALL PATHSOC(NPPO,ICON,IS,ID);
510     1   2           IF ICON = 1 THEN
                          PUT EDIT('THE ORIGINAL ROUTING TABLE HAS AT LEAST ONE LOOP',
                                   ' IN IT',' A LOOP WAS ENCOUNTERER ON THE PATHS ',
                                   'FROM',IS,'TO',ID,'EXECUTION OF THE PROGRAM HAS ',
                                   'TERMINATED')
                                  (SKIP,2(A),SKIP,X(12),2(A),X(1),F(2),X(1),A,X(1),
                                   F(2),SKIP,X(1),2(A));
530     1   2           ELSE DO;
590     1   3                  ERR=0.001;
600     1   3                  CALL ACTGOS(ERR,ICON);


                /* VERIFIER S'IL FAUT, POUR CHAQUE CHEMIN, IMPRIMER SA     */
                /* FIABILITE AINSI QUE LE TRAFIC QU'IL AMENE               */

670     1   3           IF NDPO ¬= 0
```

```
OPTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

NUMBER LEV NT

                        THEN DO;

            /* ECRIRE, POUR CHAQUE CHEMIN, SA FIABILITE AINSI QUE LE    */
            /* TRAFIC QU'IL AMENE                                       */

0000   1   4                    PUT PAGE EDIT ('PROBABILITY(PATH USED) AND PATH',
                                 ' CARRIED LOAD(CCS)',
                                 '-----------------------------',
                                 '-----------------')
                                (COL(28),2(A),SKIP(1),COL(28),
                                 2(A),SKIP(4));
0000   1   4                    DO I=1 TO NS;
0020   1   5                      DO J=1 TO NS;
0030   1   6                        IF I ¬= J
                                    THEN DO;
0050   1   7                          PUT EDIT('FROM',I,'TO',J)
                                          (SKIP(3),COL(35),A,X(3),
                                           F(2),X(4),A,X(3),F(2),
                                           SKIP(3));
0110   1   7                          DO K=1 TO 25;
0110   1   8                            KOUNTK=LOCATI(I,J,K);
0120   1   8                            IF KOUNTK = 0 THEN
                                          K=25;
0130   1   8                          ELSE
                                       IF NPATH(LOCATF(I,J,K)) ¬=
                                          (NS+1) THEN
                                          PUT EDIT('PR(PATH',K,')=',
                                                   PR(I,J,K),'LOAD',
                                                   ' CARRIED BY PAT',
                                                   'H',K,'=',
                                                   PL(I,J,K))
                                              (SKIP(1),COL(42),
                                               A,F(2),A,F(8,6),
                                               X(5),3(A),F(2),
                                               A,F(10,2));
0050   1   8                          ELSE;
0060   1   8                        END;
0070   1   7                      END;
0080   1   6                    ELSE;
0090   1   6                  END;
0100   1   5                END;
0110   1   4              END;
0120   1   3            ELSE;

            /* OBTENIR LA PERFORMANCE DES LIENS                         */

0000   1   3            PUT PAGE EDIT ('ACTUAL TRUNK GROUP PERFORMANCE',
                                        '-----------------------------')
                                       (COL(27),A,SKIP(1),COL(27),A);
0000   1   3            PUT EDIT ('LINK NUMBER','OFFERED LOAD(CCS)',
```

```
PTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

BER LEV NT

                                    'TRUNK GROUP ACTUAL GOS',
                                    'TRUNK GROUP RELIABILITY')
                                   (SKIP(6),COL(27),3(A,X(5)),A,SKIP(3));
0050    1  3                 DO I=1 TO NL;
0060    1  4                    BP=1-GOS(I);
0270    1  4                    PUT EDIT(I,FLTG(I),BP,GOS(I))
                                     (COL(27),X(4),F(3),X(13),F(10,2),X(14),
                                      F(9,6),X(17),F(9,6),SKIP(1));
0300    1  4                 END;
0310    1  3                 PUT PAGE EDIT ('NETWORK PERFORMANCE',
                                           '-------------------')
                                          (COL(26),A,SKIP(1),COL(26),A);
0340    1  3                 PUT EDIT ('SOURCE','DESTINATION','BLOCKING PROBABILITY',
                                    'WEIGHTED BLOCKING PROBABILITY')
                                   (SKIP(6),COL(26),3(A,X(5)),A,SKIP(3));

              /* CALCULER, POUR CHAQUE NOEUD, SON GOS ET LE GOS DU      */
              /* RESEAU                                                 */

0430    1  3                    CSSUM=0;
0440    1  3                    CTSUM=0;
0450    1  3                    WCSSUM=0;
0460    1  3                    DO I=1 TO NS;
0470    1  4                       DO J=1 TO NS;
0480    1  5                          IF I ¬= J
                                      THEN DO;
0500    1  6                             CSSUM=CSSUM+CLSD(I,J);
0510    1  6                             WCSSUM=WCSSUM+CLSD(I,J)*WT(I,J);
0520    1  6                             CTSUM=CTSUM+TR(I,J);
0530    1  6                             CLS(I)=CLS(I)+CLSD(I,J);
0540    1  6                             STR(I)=STR(I)+TR(I,J);
0550    1  6                             IF TR(I,J) = 0 THEN
                                            SDBP=0;
0570    1  6                             ELSE
                                            SDBP=CLSD(I,J)/TR(I,J);
0590    1  6                             WSDBP=SDBP*WT(I,J);

              /* ECRIRE POUR UNE PAIRE DE NOEUDS ORIGINE-DESTINATION,   */
              /* SA PROBABILITE DE BLOCAGE                              */

0600    1  6                             IF WSDBP > BPMAX THEN
                                            PUT EDIT(I,J,SDBP,WSDBP,'*')
                                                (COL(28),F(2),X(11),F(2),
                                                 X(15),E(10,3),X(15),E(10,3),
                                                 A,SKIP(1));
0610    1  6                             ELSE
                                            PUT EDIT(I,J,SDBP,WSDBP)
                                                (COL(28),F(2),X(11),F(2),
                                                 X(15),E(10,3),X(15),E(10,3),
                                                 SKIP(1));
```

```
PTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

MBER LEV NT

3770    1   6                                   END;
3780    1   5                               ELSE;
3790    1   5                                   END;
3800    1   4                           END;

                /* ECRIRE POUR CHAQUE NOEUD SON GOS                              */

3860    1   3                           PUT PAGE EDIT ('NODE GRADE OF SERVICE',
                                                       '----------------------')
                                                      (COL(25),A,SKIP(1),COL(25),A);
3890    1   3                           PUT EDIT ('NODE NUMBER','ORIGINATING LOAD(CCS)',
                                                  'NODE GRADE OF SERVICE','NODE RELIABILITY')
                                                 (SKIP(5),COL(25),3(A,X(5)),A,SKIP(3));
3920    1   3                       /*  DO I=1 TO NS;
3930    1   4                           IF STR(I) = 0.0E0 THEN
                                            DO;
3950    1   5                               PUT EDIT (I,'0.00','-','-')
                                                     (COL(29),F(2),X(18),A,X(22),A,X(21),
                                                      A);
3980    1   5                               END;
3990    1   4                           ELSE
                                            DO;
4010    1   5                               SBP=CLS(I)/STR(I);
4020    1   5                               DUM1=1.0E0-SBP;
4030    1   5                               DUM2=STR(I);
4040    1   5                               CLS(I)=0.0E0;
4050    1   5                               STR(I)=0.0E0;
4060    1   5                               PUT EDIT(I,DUM2,SBP,DUM1)
                                                    (COL(29),F(2),X(15),F(10,2),X(17),
                                                     F(9,6),X(13),F(9,6));
4090    1   5                               END;
4100    1   4                       END;
4110    1   3                   END;

                /* DETERMINER LA PAIRE DE NOEUDS ORIGINE-DESTINATION POS-        */
                /* SEDANT LE PLUS GRAND NNGOS PONDERE AINSI QUE LA VALEUR        */
                /* DE CE NNGOS                                                   */

4190    1   2   CALL SORT1(BPMAX,IS,ID,BIGBP,NEXD);

                /* ECRIRE LE GOS DU RESEAU, LE PLUS GRAND NNGOS PONDERE,         */
                /* ET LE NOMBRE DE NNGOS PONDERES EXEDANT LA VALEUR CON-         */
                /* TENUE DANS LA VARIABLE BPMAX                                  */

4270    1   2   PUT PAGE EDIT ('NETWORK STATISTICS','----------------')
                             (COL(27),A,SKIP(1),COL(27),A);
4290    1   2   PUT EDIT ('LARGEST WEIGHTED NNGOS(=',BIGBP,') IS FROM ',IS,' TO ',ID)
```

```
OPTIMIZING COMPILER          PRINC:PROCEDURE OPTIONS(MAIN);

BER LEV NT

                        (SKIP(10),COL(35),A,F(10,3),2(A,F(2)));
 310    1   2   PUT EDIT ('MAXIMUM ALLOWABLE BLOCKING PROSABILITY(BPMAX)',BPMAX)
                        (SKIP(3),COL(35),A,X(1),F(6,3));
 330    1   2   PUT EDIT ('NUMBER OF WEIGHTED NNGOS EXCEEDING BPMAX IN THE ORIGINAL ',
                         'ROUTING TABLE = ',NEXD)
                        (SKIP(3),COL(35),2(A),F(3));


 380    1   2   GOSNTK=CSSUM/CTSUM;
 390    1   2   WGOSNK=WCSSUM/CTSUM;

 420    1   2   PUT EDIT ('NETWORK GRADE OF SERVICE IS',GOSNTK,'WEIGHTED NETWORK ',
                         'GRADE OF SERVICE IS',WGOSNK)
                        (SKIP(3),COL(35),A,F(10,5),SKIP(3),COL(35),A,A,F(10,5));

                    /*****************************************************************/
                    /*                                                               */

                    /* REMETTRE LA MATRICE DU TRAFIC DANS SON ETAT ORIGINALE   */

 540    1   2       DO II1=1 TO NS;
 550    1   3         DO II2=1 TO NS;
 560    1   4           TR(II1,II2)=TR(II1,II2)/RATE;
 570    1   4         END;
 580    1   3       END;
 590    1   2   END;

                    /*                                                               */
                    /*****************************************************************/

 660    1   1     END;
 670    1   0   ELSE;

 690    1   0   END PRINC;
```

2) Le sous-programme ACTGOS

définition de la tâche

L'objectif du sous-programme ACTGOS est de trouver, par itération, la fiabilité de tous les liens (TRUNK GROUP RELIABILITIES). Comme dérivés de l'exécution de ce dernier, on obtient le trafic offert par chaque lien (TRUNK GROUPS OFFERED LOADS) et on calcule le trafic perdu entre chaque paire de noeuds. De plus, pour chaque paire de noeuds ORIGINE-DESTINATION, on calcule, pour tous les chemins qui relient ces deux noeuds, leur probabilité d'utilisation ainsi que le trafic acheminé par ces dits chemins.

La figure 5 présente le diagramme logique de ce sous-programme.

DIAGRAMME LOGIQUE DU SOUS-PROGRAMME ACTGOS

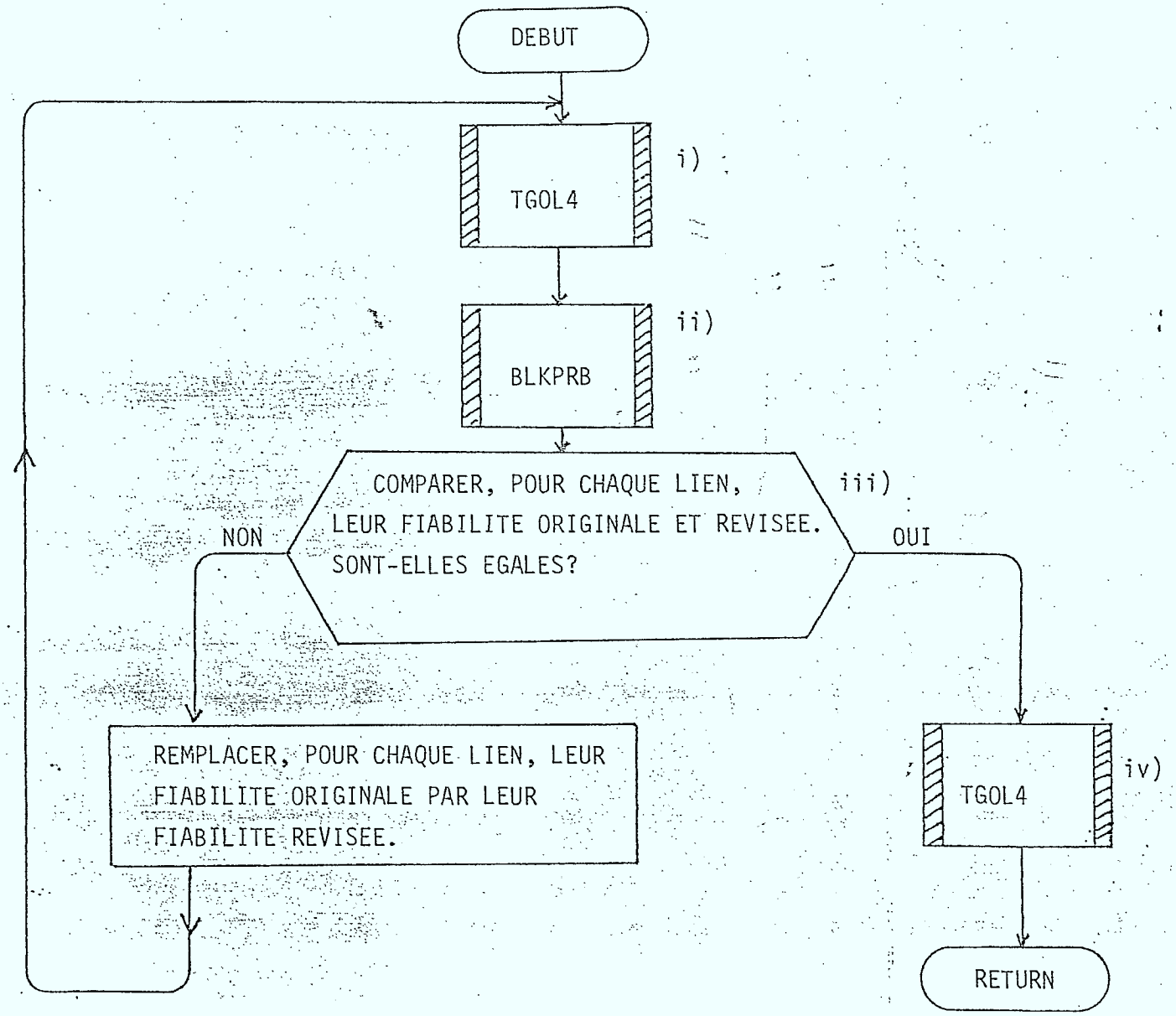


FIGURE 5

REMARQUES SUR LE DIAGRAMME LOGIQUE

i) On calcule le trafic offert par chaque lien (TRUNK GROUP OFFERED LOADS) en utilisant leur fiabilité originale (ORIGINAL TRUNK GROUP RELABILITIES).

ii) En utilisant la formule ERLANGB, on calcule, pour chaque lien, leur fiabilité révisée (REVISED TRUNK GROUP RELIABILITIES)

iii) On utilise une limite d'erreur. Elle a été spécifiée auparavant.

iv) On calcule le trafic offert par chaque lien (TRUNK GROUP OFFERED LOADS) en utilisant leur fiabilité révisée (REVISED TRUNK GROUP RELIABILITIES)

Il est à noter, qu'il y a au plus 20 itérations qui sont exécutées. En général, il a été démontré que la fiabilité d'un lien converge, avec une erreur $\leq 0{,}01$, vers une solution en prenant en moyenne, de 4 à 9 itérations. Lors de l'appel du sous-programme ACTGOS, la structure GOS contient un estimé de la fiabilité des liens.

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME ACTGOS

Les variables BP, CLSD, FLTG, GOS, NL, NTRK, PL et PR ont été définies dans le dictionnaire des variables du programme principal.

DUMMAX = après avoir calculé pour chaque lien la différence entre leurs fiabilités originale et révisée, on place dans DUMMAX la plus grande de ces différences.

ERR = variable dont la valeur est utilisée pour déterminer le point où les fiabilités d'un lien calculées lors de deux itérations successives sont considérées comme étant égales. ERR contient toujours la valeur 0.001

FL = variable qui contient le trafic offert par un lien (TRUNK GROUP OFFERED LOAD)

J = Après avoir calculé pour chaque lien la différence entre leurs fiabilités originale et révisée, la variable J nous indique si une ou plusieurs différences exédent la valeur contenue dans ERR.

N = Variable qui nous indique le nombre de tronçons contenus dans un lien

BP = variable qui nous indique le "GRADE OF SERVICE" du lien traité. Le trafic offert à ce lien ainsi que son nombre de TRUNK sont passés au sous-programme BLKPRB. Ce dernier retourne la valeur de la variable BP

DUM = variable qui contient la différence entre les fiabilités originale et révisée d'un lien

TRGOS = variable qui contient la fiabilité révisée d'un lien

LISTING DU SOUS-PROGRAMME ACTGOS

```
OPTIMIZING COMPILER          ACTGOS:PROCEDURE(ERR,ICON);

                SOURCE LISTING

NUMBER LEV NT

  10       0   ACTGOS:PROCEDURE(ERR,ICON);
               /*****************************************************************/
               /*                                                               */
               /* OBJECTIFS:                                                    */
               /*       1- OBTENIR, PAR ITERATION, LA FIABILITE DE TOUS         */
               /*          LES LIENS                                            */
               /*       2- OBTENIR LE TRAFIC OFFERT A CHAQUE LIEN               */
               /*       3- CALCULER LE TRAFIC PERDU ENTRE CHAQUE NOEUD          */
               /*       4- OBTENIR, POUR CHAQUE CHEMIN, LEUR PROBABILITE        */
               /*          D'UTILISATION AINSI QUE LE TRAFIC QU'IL AMENE        */
               /*                                                               */
               /*****************************************************************/

 150   1   0   DECLARE (GOS(105),FLTG(105),CLSD(15,15),
                  PL(15,15,25),PR(15,15,25))          FLOAT(16)      EXTERNAL,
                  IRT(15,15,4)                        FIXED(2)       EXTERNAL,
                NL                                    FIXED(3)       EXTERNAL,
                  NTRK(105)                           FIXED(7)       EXTERNAL,
                (BP,FL,DUM,ERR,TRGOS,DUMMAX)          FLOAT(16),
                (ICON,J)                              FIXED(1),
                (K,IS,ID)                             FIXED(2),
                  N                                   FIXED(7),
                  I                                   FIXED(3);

               /* DEBUT DES ITERATIONS                                          */

 300   1   0   DO K=1 TO 20;

               /* CALCULER LE TRAFIC OFFERT A CHAQUE LIEN                       */

 360   1   1     CALL TGOL4;
 370   1   1     J=0;
 380   1   1     DUMMAX=0;

               /* CALCULER POUR CHAQUE LIEN, EN UTILISANT LA FORMULE            */
               /* ERLANG B, SA PROBABILITE DE BLOCAGE                           */

 450   1   1     DO I=1 TO NL;
 460   1   2       FL=FLTG(I);
 470   1   2       N=NTRK(I);
 480   1   2       IF N = 0 THEN
                     GOS(I)=0;
 500   1   2       ELSE DO;
 510   1   3         CALL BLKPRB (BP,FL,N);
```

```
TIMIZING COMPILER          ACTGOS:PROCEDURE(ERR,ICON);

R LEV NT

20   1  3                    TRGOS=1-BP;
50   1  3                    DUM=ABS(TRGOS-GOS(I))/TRGOS;
0    1  3                    GOS(I)=TRGOS;

          /* VERIFIER SI LE CRITERE D'ERREUR EST SATISFAIT                */

00   1  3                    IF DUM > ERR
                             THEN DO;
50   1  4                       J=1;
0    1  4                       IF DUM > DUMMAX THEN
                                   DUMMAX=DUM;
50   1  4                       ELSE;
60   1  4                    END;
0    1  3                 ELSE;
0    1  3              END;
50   1  2        END;
00   1  1        IF J=0
                 THEN DO;
0    1  2              CALL TGOL4;
0    1  2              RETURN;
40   1  2           END;
50   1  1        ELSE;
10   1  1  END;
 0   1  0  PUT EDIT ('AFTER 20 ITERATIONS ALL ACTUAL TRUNK GROUP GOS COULD NOT B',
                    'E DETERMINED WITHIN AN ERROR BOUND OF',ERR,'THE TRUNK ',
                    'GROUP GOS ARE ASSUMED TO HAVE BEEN DETERMINED WITHIN AN',
                    ' ERROR BOUND OF',DUMMAX)
                   (SKIP,2(A),X(1),F(6,3),SKIP,X(1),3(A),X(1),F(6,3));
0    1  0  RETURN;
30   1  0  END ACTGOS;
```

3) Le programme BLKPRB

définition de la tâche

Etant donné le trafic offert par un lien et le nombre de tronçons (TRUNK) contenus dans ce dernier, le sous-programme BLKPRB calcule le GOS dudit lien (TRUNK GROUP)

Pour ce faire, il utilise la formule d'ERLANG B:

$$B(n,a) = \frac{a^n/n!}{\sum_{k=0}^{n} a^K/K!}$$

Cependant, si elle est programmée en utilisant cette relation, elle impose une limite quant au nombre de tronçons (TRUNK) que l'on retrouve dans chaque lien. C'est pourquoi, puisqu'il peut être démontré que B(n,a) admet une relation de récurrence, elle a été programmée en employant la formule suivante:

$$B(n,a) = \frac{a\ B(n-1,a)}{N+a\ B(n-1,a)} \qquad (1)$$

où $B(o,a) = 1$

Il est à noter que cette dernière formule peut être utilisée pour déterminer N i.e. le nombre de tronçons. Dans ce cas on calcule:

$$a = \frac{a_C}{1-B(n,a)}$$

où a_C: est le trafic amené par le lien (TRUNK GROUP CARRIER LOAD)

$B(n,a)$: probabilité de blocage maximale

Ensuite, on emploie la formule (1) pour trouver la valeur de N. Pour réaliser cette dernière étape, il faut tout simplement augmenter N jusqu'à ce que $B(n,a) \leq$ probabilité de blocage maximale.

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME BLKPRB

Les variables FL et N ont été définies dans le dictionnaire des variables du programme principal.

BNA = La non-fiabilité d'un lien i.e la probabilité de blocage du lien

LISTING DU SOUS-PROGRAMME BLKPRB

```
PL/I OPTIMIZING COMPILER          BLKPRB:PROCEDURE(BNA,FL,N);

                    SOURCE LISTING

NUMBER LEV NT

    10       0  BLKPRB:PROCEDURE(BNA,FL,N);
                /*************************************************************/
                /*                                                           */
                /* CE SOUS-PROGRAMME REALISE LA FONCTION SUIVANTE:           */
                /*                                                           */
                /*      CALCUL LA FORMULE D'ERLANG POUR OBTENIR LE ''GRA-    */
                /*      DE OF SERVICE'' D'UN LIEN                            */
                /*                                                           */
                /*************************************************************/
    120   1  0  DECLARE (BNA,FL,SI,A)                    FLOAT(16),
                (N,I)                                    FIXED(7);
    140   1  0  A=FL/36.0E0;
    150   1  0  BNA=1.0E0;

    180   1  0  DO I=1 TO N WHILE (BNA>1.0E-50);
    190   1  1       SI=A*BNA;
    200   1  1       BNA=SI/(FLOAT(I,16)+SI);
    210   1  1  END;
    220   1  0  END BLKPRB;
```

4) Le sous-programme TGOL4

définition de la tâche

L'objectif du sous-programme TGOL4 est de calculer le trafic offert par chaque lien (TRUNK GROUP) et le trafic perdu entre chaque paire de noeuds ORIGINE-DESTINATION du réseau. Il obtient aussi la probabilité d'utilisation ainsi que le trafic amené pour chacun des chemins des arbres d'acheminement du réseau.

La figure 6 présente le diagramme logique de ce sous-programme.

DIAGRAMME LOGIQUE DU SOUS-PROGRAMME TGOL4

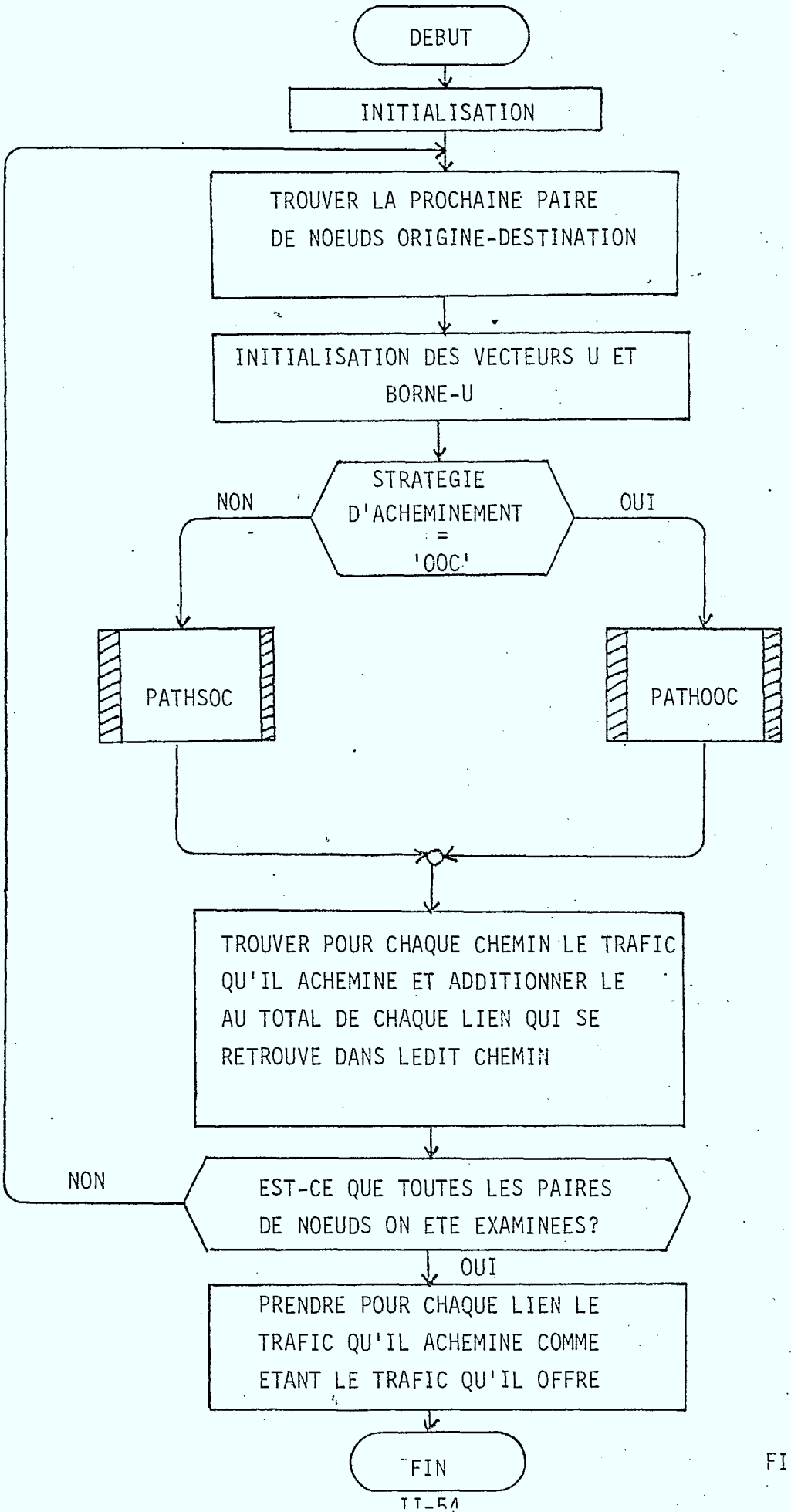


FIGURE 6

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME TGOL4

Les variables CLSD, FLTG, GOS, LOCATF, LOCATI, LNUM, NPATH, NS, NL, P, PL, PR, ROUSTR, TR et WT ont été définies dans le dictionnaire des variables du programme principal.

BORNE-U = vecteur nous indiquant la composante où commence un chemin dans le vecteur U. Le numéro d'une composante de ce vecteur correspond au numéro du chemin pour la paire de noeuds ORIGINE-DESTINATION. La composante (N+1) contient le numéro de la prochaine composante à remplir dans le vecteur U. Elle marque la limite supérieure de ce vecteur.

DUM = variable qui contient le trafic amené par le chemin qui est à l'étude

ID = variable qui contient le numéro du noeud destination

IS = variable qui contient le numéro du noeud source

ITS,ITD = variables qui contiennent les numéros des noeuds source et destination. Ces noeuds correspondent à des noeuds temporaires et sont utilisés pour trouver les numéros des liens du chemin qui est à l'étude.

KOUNTF = variable bidon qui contient, par l'entremise de LOCATF, la position dans le vecteur NPATH du dernier noeud du chemin qui est à l'étude

KOUNTI = variable bidon qui contient, par l'entremise de LOCATI, la position dans le vecteur NPATH du premier noeud du chemin qui est à l'étude

LN = variable qui contient un des numéros des liens du chemin qui est à l'étude

TOTAL = variable qui contient la somme de tous les trafics qui atteignent la destination

U = vecteur contenant tous les chemins pour une paire de noeuds ORIGINE-DESTINATION. Un chemin est représenté par tous les numéros des liens qui le constituent.

LISTING DU SOUS-PROGRAMME TGOL4

```
TIMIZING COMPILER          TGOL4:PROCEDURE;

                SOURCE LISTING

MBER LEV NT

 10      0   TGOL4:PROCEDURE;

             /*****************************************************************/
             /*                                                               */
             /* OBJECTIFS:                                                    */
             /*       1- CALCULER LE TRAFIC OFFERT A CHAQUE LIEN              */
             /*       2- CALCULER LE TRAFIC PERDU ENTRE CHAQUE PAIRE DE       */
             /*          NOEUDS ORIGINE-DESTINATION                           */
             /*       3- CALCULER LA PROBABILITE D'UTILISATION AINSI QUE      */
             /*          LE TRAFIC AMENE PAR CHACUN DES CHEMINS DES           */
             /*          ARBRES D'ACHEMINEMENT DU RESEAU                      */
             /*                                                               */
             /*****************************************************************/

 160   1   0   DECLARE (P(25),GOS(105),FLTG(105),
                 TR(15,15),WT(15,15),CLSD(15,15),
                 PR(15,15,25),PL(15,15,25))          FLOAT(16)       EXTERNAL,
                 (NS,NPATH(5000))                    FIXED(2)        EXTERNAL,
                 (NL,LNUM(15,15))                    FIXED(3)        EXTERNAL,
                 (LOCATI(15,15,25),LOCATF(15,15,25)) FIXED(4)        EXTERNAL,
                 (DUM,TOTAL)                         FLOAT(16),
                 (I,N,IS,ID,ITS,ITD,U(90),
                 BORNE_U(50))                        FIXED(2),
                 LN                                  FIXED(3),
                 (J,K,KOUNTI,KOUNTF)                 FIXED(4),
                 ROUSTR                              CHAR(3)         EXTERNAL;

             /* INITIALISATION DES MATRICES                                   */

 330   1   0   DO I=1 TO NS;
 340   1   1     DO J=1 TO NS;
 350   1   2       CLSD(I,J)=0;
 360   1   2     END;
 370   1   1   END;
 380   1   0   DO I=1 TO NL;
 390   1   1     FLTG(I)=0;
 400   1   1   END;

             /* EXAMINER CHAQUE PAIRE DE NOEUDS ET CALCULER LE TRAFIC         */
             /* OFFERT PAR CHAQUE LIEN                                        */

 470   1   0   DO IS=1 TO NS;
 480   1   1     DO ID=1 TO NS;
 490   1   2       IF IS ¬= ID
                   THEN DO;
```

```
L/I OPTIMIZING COMPILER          TGOL4:PROCEDURE;

NUMBER LEV NT

                    /* TROUVER POUR CHAQUE CHEMIN DES ARBRES D'ACHEMINEMENT  */
                    /* LEUR PROBABILITE D'UTILISATION                        */

                    /* ARRANGER LES VECTEURS U ET BORNE_U                    */

  600    1  3                  K=1;
  610    1  3                  BORNE_U(1)=1;
  620    1  3                  DO I=1 TO 25
                                  WHILE (LOCATI(IS,ID,I) ¬= 0);
  640    1  4                     DO J=LOCATI(IS,ID,I) TO LOCATF(IS,ID,I)-1;
  650    1  5                        U(K)= LNUM(NPATH(J),NPATH(J+1));
  660    1  5                        K=K+1;
  670    1  5                     END;
  680    1  4                     BORNE_U(I+1)=K;
  690    1  4                  END;
  700    1  3                  N=I-1;

  720    1  3                  IF ROUSTR = 'ODC' THEN
                                  AA=PROBODC(U,BORNE_U,N,0);
  740    1  3                  ELSE
                                  CALL PROBSDC(U,BORNE_U,N);

  770    1  3                  TOTAL = 0;

                    /* CALCULER, POUR CHAQUE CHEMIN, LE TRAFIC QU'IL AMENE   */

  830    1  3                  DO I=1 TO 25;
  840    1  4                     KOUNTI = LOCATI(IS,ID,I);
  850    1  4                     IF KOUNTI ¬= 0
                                  THEN DO ;
  870    1  5                              DUM=TR(IS,ID) * P(I);
  880    1  5                              TOTAL=TOTAL+DUM;
  890    1  5                              KOUNTF=LOCATF(IS,ID,I)-1;
  900    1  5                              PL(IS,ID,I)=DUM;
  910    1  5                              PR(IS,ID,I)=P(I) * WT(IS,ID);

                    /* ADDITIONNER A TOUS LES LIENS CONSTITUANT UN CHEMIN, LE */
                    /* TRAFIC AMENE PAR LE DIT CHEMIN                        */

  980    1  5                              DO J=KOUNTI TO KOUNTF;

                    /* TROUVER LE NUMERO DU LIEN                             */

 1040    1  6                                 ITS = NPATH(J);
 1050    1  6                                 K=J+1;
```

```
PL/I OPTIMIZING COMPILER          TGDL4:PROCEDURE;

NUMBER LEV NT

1060    1  6                                        ITD=NPATH(K);
1070    1  6                                        LN=LNUM(ITS,ITD);

                /* ADDITIONNER LE TRAFIC AMENE                          */

1130    1  6                                        FLTG(LN)=FLTG(LN)+DUM;
1140    1  6                                    END;
1150    1  5                                END;
1160    1  4                            ELSE;
1170    1  4                        END;

                /* CALCULER LE TRAFIC PERDU ENTRE CHAQUE PAIRE DE NOEUDS */
                /* ORIGINE-DESTINATION                                   */

1240    1  3                        CLSD(IS,ID) = TR(IS,ID)- TOTAL;
1250    1  3                    END;
1260    1  2                ELSE;
1270    1  2            END;
1280    1  1    END;

                /* PRENDRE, POUR CHAQUE LIEN, LE TRAFIC QU'IL AMENE COM- */
                /* ME ETANT LE TRAFIC QUI LUI EST OFFERT                 */

1350    1  0    DO I=1 TO NL;
1360    1  1        IF GOS(I) =0 THEN
                        FLTG(I)=0;
1380    1  1        ELSE
                        FLTG(I)=FLTG(I)/GOS(I);
1400    1  1    END;
1410    1  0    RETURN;
1420    1  0    END TGDL4;
```

5) Les sous-programmes PROBOOC et PROBSOC

définition de la tâche

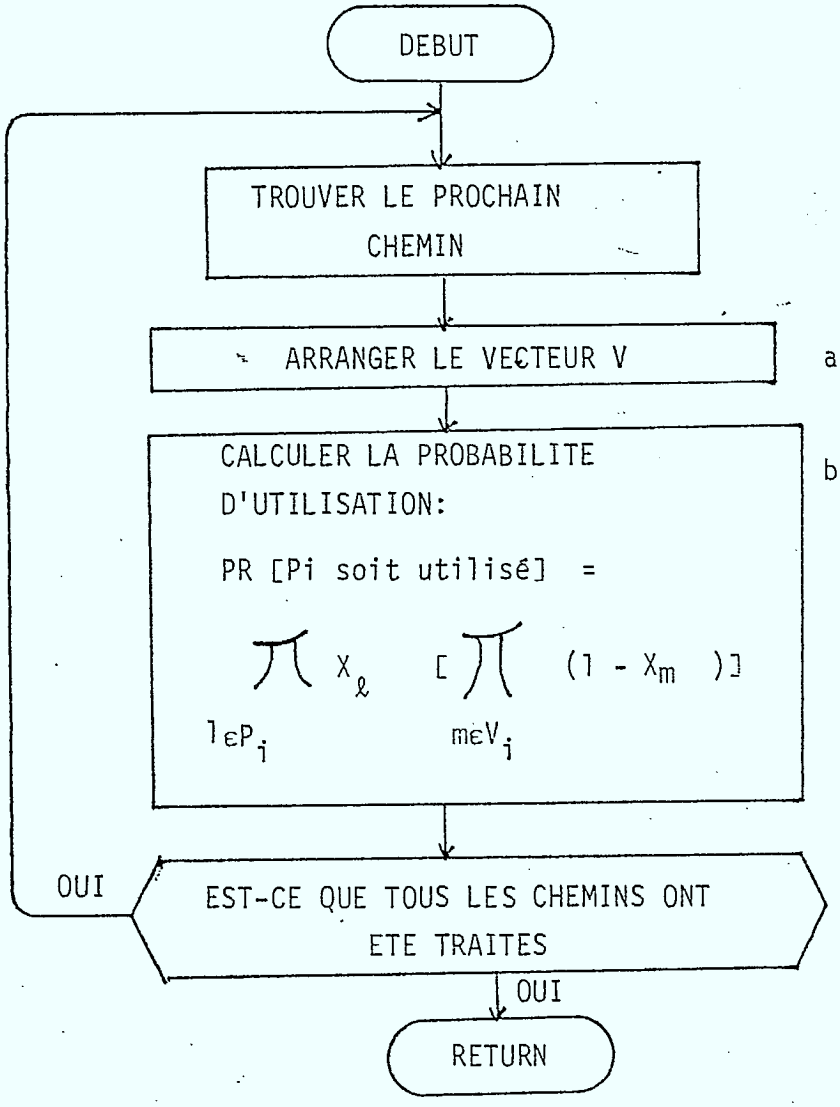
L'objectif de ces sous-programmes est de calculer, pour tous les chemins qui se retrouvent entre une paire de noeuds ORIGINE-DESTINATION, leur probabilité d'utilisation.

Selon la stratégie d'acheminement sélectionnée, on utilise l'un ou l'autre de ces sous-programmes. Ainsi, si la stratégie d'acheminement suit la règle OOC, on emploie le sous-programme PROBOOC autrement on emploie le sous-programme PROBSOC car à ce moment la stratégie choisie suit la règle SOC.

La figure 7 présente le diagramme logique du sous-programme PROBSOC. Ce dernier calcule, pour tous les chemins de l'arbre d'acheminement d'une paire de noeuds ORIGINE-DESTINATION, leur probabilité d'utilisation. La théorie concernant le diagramme logique est expliquée dans la section 4.2 de l'annexe I.

Pour ce qui est du sous-programme PROBOOC, la figure 8 illustre son algorithme. Ce dernier est écrit en PIDGIN ALGOL. Comme pour le sous-programme PROBSOC, PROBOOC calcule pour tous les chemins de l'arbre d'acheminement d'une paire de noeuds ORIGINE-DESTINATION, leur probabilité d'utilisation. La théorie concernant l'algorithme est décrite dans la section 4.1 de l'annexe I.

DIAGRAMME LOGIQUE DU SOUS-PROGRAMME PROBSOC



a) et b) Pour plus de renseignements voir la section 4.2 de l'annexe I.

FIGURE 7

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME PROBSOC

Les variables GOS et P ont été définies dans le dictionnaire des variables du programme principal. Pour ce qui est des varaibles BORNE-U et U, ces dernières ont été définies dans le dictionnaire des variables du sous-programme TGOL4.

M = variable qui contient le numéro de la prochaine composante à remplir dans le vecteur V. M-1 nous indique donc le nombre de composantes qui sont remplies dans le vecteur V.

N = variable qui nous indique le nombre de chemins qui existent entre la paire de noeuds ORIGINE-DESTINATION qui est à l'étude.

P1 = variable qui contient, pour la paire de noeuds ORIGINE-DESTINATION qui est à l'étude, la valeur suivante:

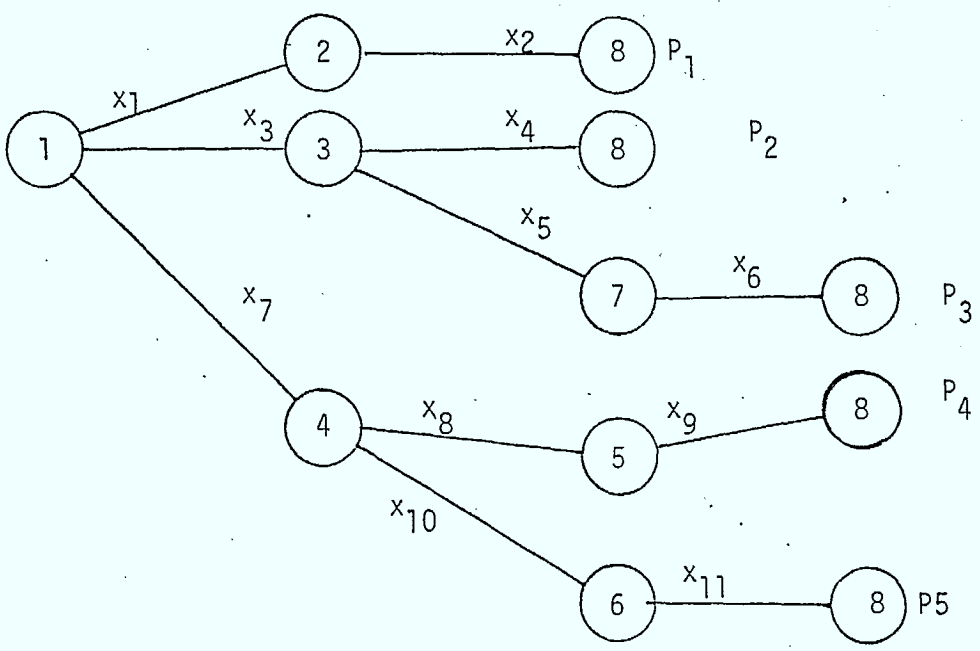
$\prod_{l \in P_i} x_l$ (voir section 4.2 de l'annexeI)

P2 = variable qui contient, pour la paire de noeuds ORIGINE-DESTINATION qui est à l'étude, la valeur suivante:

$\prod_{m \in V_j} (1-x_m)$ (voir section 4.2 de l'annexe I)

V = vecteur contenant, pour chaque chemin qui précède dans l'arbre d'acheminement le chemin qui est à l'étude, le numéro du premier lien qui est différent des numéros des liens constituant le chemin qui est à l'étude. Il est à noter que si deux ou plusieurs chemins ont le même numéro du premier lien, ce dernier n'apparaîtra qu'une fois dans le vecteur.

Par exemple, si on étudie le 5ième chemin (P5) de l'arbre d'acheminement suivant:



le vecteur V contiendra les numéros des liens x_1, x_3 et x_8.

LISTING DU SOUS-PROGRAMME PROBSOC

```
/* OPTIMIZING COMPILER          PROBSOC:PROCEDURE(U,BORNE_U,N);

                    SOURCE LISTING

NUMBER LEV NT

    10       0   PROBSOC:PROCEDURE(U,BORNE_U,N);

                 /*************************************************************/
                 /*                                                           */
                 /* OBJECTIFS:                                                */
                 /*   POUR UN RESEAU SUIVANT LES REGLES DE LA STRATEGIE       */
                 /*   D'ACHEMINEMENT SOC                                      */
                 /*      1- CALCULER, POUR TOUS LES CHEMINS QUI SE RETROU-    */
                 /*         VENT ENTRE UNE PAIRE DE NOEUDS ORIGINE-DESTINA-   */
                 /*         TION, LEUR PROBABILITE D'UTILISATION              */
                 /*                                                           */
                 /*************************************************************/

    140  1   0  DECLARE (P(25),GOS(105))                FLOAT(16)      EXTERNAL,
                        (P1,P2)                         FLOAT(16),
                        (I,J,K,L,N,U(*),BORNE_U(*))     FIXED(2),
                        (V(15),M,MM)                    FIXED(2);



                 /* PRENDRE LE PROCHAIN CHEMIN                                */

    250  1   0  DO I=1 TO N;

                 /* ARRANGER LE VECTEUR V                                     */

    310  1   1      M=1;
    320  1   1      DO J=1 TO I-1 WHILE (I>1);
    330  1   2          K=0;
    340  1   2          DO L=BORNE_U(J) TO BORNE_U(J+1)-1
                            WHILE (K ¬= BORNE_U(I+1));
    360  1   3              DO K=BORNE_U(I) TO BORNE_U(I+1)-1
                                WHILE (U(K) ¬= U(L));

    390  1   4              END;
    400  1   3          END;

    420  1   2          DO MM=1 TO M-1
                            WHILE ((M>1) & (V(MM) ¬= U(L-1)));
    440  1   3          END;

    460  1   2          IF MM=M THEN
                            DO;
    480  1   3              V(M)=U(L-1);
    490  1   3              M=M+1;
    500  1   3            END;
    510  1   2          ELSE;
```

```
PL/I OPTIMIZING COMPILER          PROBSOC:PROCEDURE(U,BORNE_U,N);

   NUMBER LEV NT

      530   1  2       END;

                       /* FAIRE LE PRODUIT DES PROBABILITES DE BLOCAGE DE TOUS   */
                       /* LES LIENS QUI SE RETROUVENT DANS LE VECTEUR V          */

      600   1  1       P2=1;
      610   1  1       DO MM=1 TO M-1
                           WHILE ((M>1) & (P2>1.0E-50));
      630   1  2           P2=P2*(1-GOS(V(MM)));
      640   1  2       END;

                       /* FAIRE LE PRODUIT DES FIABILITES DE TOUS LES LIENS QUI  */
                       /* CONSTITUENT LE CHEMIN                                  */

      710   1  1       P1=1;
      720   1  1       DO J=BORNE_U(I) TO BORNE_U(I+1)-1;
      730   1  2           P1=P1*GOS(U(J));
      740   1  2       END;

                       /* CALCULER LA PROBABILITE D'UTILISATION DU CHEMIN        */

      800   1  1       P(I)=P1*P2;
      810   1  1   END;
      820   1  0   END PROBSOC;
```

Algorithme du sous-programme PROBOCC

```
PROCEDURE R (U1, U2, ...., Ui, T)
IF i=1  Then Return  ∏ X1
                     1εVi

ELSE
    BEGIN
        FOR K ← 1 STEP UNTIL i-1  DO
            Vk ← Uk - U i

        PI=[1-R (V1, V2, ...., Vi-1, 1)] ∏ X1
                                         1εVi

        IF T= 0     Then
            BEGIN
                P (i) = PI
                R ← R(U1, U2, ...., Ui-1, 0)

            END
        ELSE  R ← R(U1, U2, ...., Ui-1 , 1)

        R ← R + PI

        RETURN R

END
```

FIGURE 8

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME PROBOCC

Les variables GOS et P ont été définies dans le dictionnaires des variables du programme principal. Pour ce qui est des variables BORNE-U et U, ces dernières ont été définies dans le dictionnaire des variables du sous-programmes TGOL4.

BORNE-V = vecteur nous indiquant la composante où commence un chemin dans le vecteur V. Le numéro d'une composante de ce vecteur correspond au numéro d'un chemin pour la paire de noeuds ORIGINE-DESTINATION. La composante N + 1 contient le numéro de la prochaine composante à remplir dans le vecteur V. Elle marque la limite supérieure de ce vecteur.

N = variable qui nous indique le nombre de chemins qui existent entre la paire de noeuds ORIGINE-DESTINATION qui est à l'étude.

PI = variable qui contient, pour le chemin qui est à l'étude, la probabilité d'utilisation de cedit chemin.

PROB = variable qui contient, pour une paire de noeuds ORIGINE-DESTINATION, la somme des probabilités d'utilisation de tous les chemins de son arbre d'acheminement.

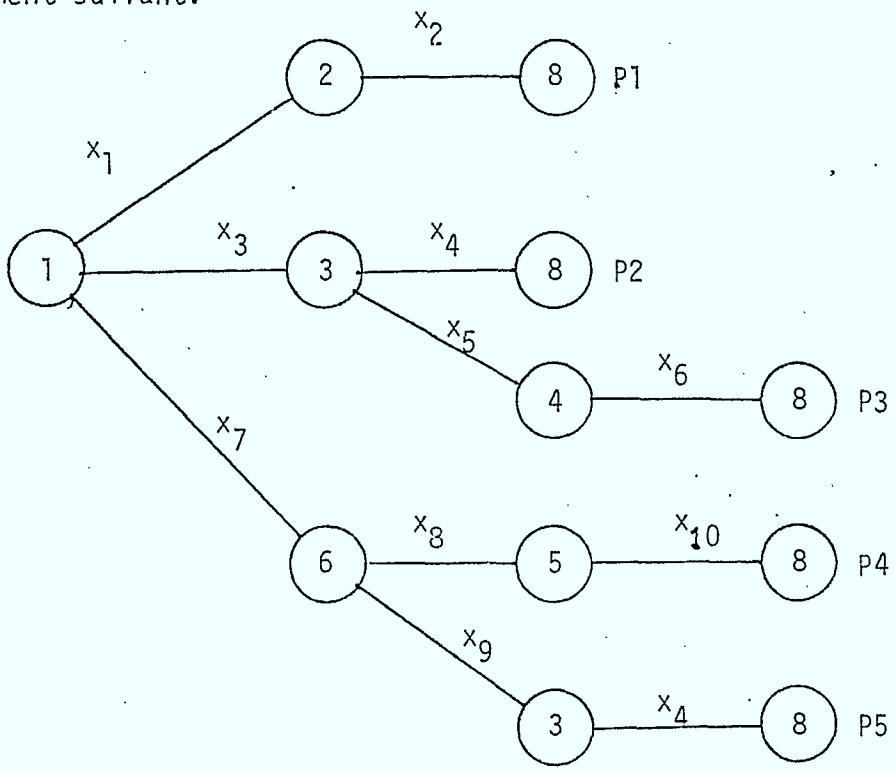
T = variable nous indiquant s'il faut emmagasiner la valeur de PI dans le vecteur V (T=0).

UI = variable contenant la valeur suivante:

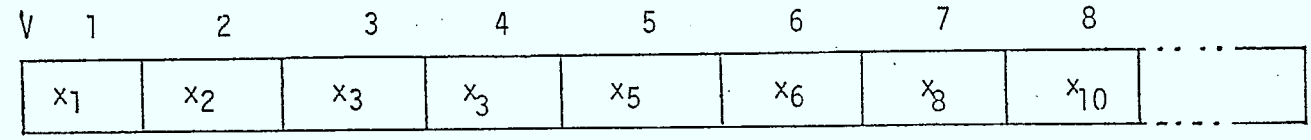
$$\prod_{\ell \in V_i} X_\ell$$

V = vecteur contenant, pour chaque chemin qui précède dans l'arbre d'acheminement le chemin qui est à l'étude, les numéros des liens qui sont différents des numéros des liens constituant le chemin qui est à l'étude.

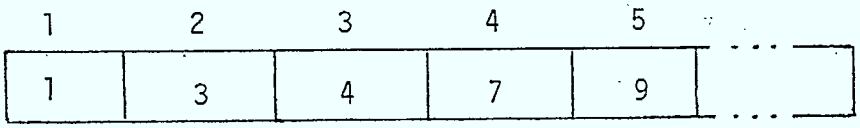
Par exemple, si on étudie le 5ième chemin de l'arbre d'acheminement suivant:



les vecteurs V et BORNE-V auront la forme suivante:



BORNE-V



LISTING DU SOUS-PROGRAMME PROBOOC

```
/* OPTIMIZING COMPILER          PROBOOC:PROCEDURE(U,BORNE_U,N,T) RECURSIVE;

                    SOURCE LISTING

NUMBER LEV NT

    10       0  PROBOOC:PROCEDURE(U,BORNE_U,N,T) RECURSIVE;

               /*************************************************************/
               /*                                                           */
               /* OBJECTIFS:                                                */
               /*    POUR UN RESEAU SUIVANT LES REGLES DE LA STRATEGIE      */
               /*    D'ACHEMINEMENT DOC                                     */
               /*       1- CALCULER, POUR TOUS LES CHEMINS QUI SE RETROU-   */
               /*          VENT ENTRE UNE PAIRE DE NOEUDS ORIGINE-DESTINA-  */
               /*          TION, LEUR PROBABILITE D'UTILISATION             */
               /*                                                           */
               /*************************************************************/

   140   1   0  DECLARE (P(25),GOS(10**5))                  FLOAT(16)      EXTERNAL,
                       (UI,PI,PROB)                        FLOAT(16),
                       T                                   FIXED(1),
                       (I,J,K,L,N,U(*),V(90),BORNE_U(*),
                        BORNE_V(50))                       FIXED(2);


               /* FAIRE LE PRODUIT DES FIABILITES DE TOUS LES LIENS QUI     */
               /* CONSTITUENT LE CHEMIN                                     */

   250   1   0  UI=1;
   260   1   0  DO I=BORNE_U(N) TO BORNE_U(N+1)-1;
   270   1   1     UI=UI*GOS(U(I));
   280   1   1  END;


               /* S'IL N'Y A QU'UN CHEMIN ALORS SA PROBABILITE D'UTILISA-    */
               /* TION EST EGALE AU PRODUIT CALCULE PRECEDEMMENT            */

   350   1   0  IF N=1
                THEN DO;
   370   1   1          PROB=UI;
   380   1   1          IF T=0 THEN
                            P(N)=PROB;
   400   1   1          ELSE;
   410   1   1       END;
   420   1   0  ELSE DO;


               /* SINON, ARRANGER LES VECTEURS V ET BORNE_V                 */


   480   1   1          L=1;
   490   1   1          BORNE_V(1)=1;
   500   1   1          DO I=1 TO N-1;
   510   1   2             DO J=BORNE_U(I) TO BORNE_U(I+1)-1;
```

```
PL/I OPTIMIZING COMPILER          PROBOOC:PROCEDURE(U,BORNE_U,N,T) RECURSIVE;

   NUMBER LEV NT

      520   1  3                    DO K=BORNE_U(N) TO BORNE_U(N+1)-1
                                        WHILE (U(K) ¬= U(J));
      540   1  4                    END;
      550   1  3                    IF K=BORNE_U(N+1)
                                    THEN DO;
      570   1  4                           V(L)=U(J);
      580   1  4                           L=L+1;
      590   1  4                       END;
      600   1  3                    ELSE;
      610   1  3                END;
      620   1  2                BORNE_V(I+1)=L;
      630   1  2            END;

                        /* CALCULER LA PROBABILITE D'UTILISATION DU CHEMIN      */

      690   1  1            PI=(1-PROBOOC(V,BORNE_V,N-1,1))*UI;
      700   1  1            IF T=0
                            THEN DO;
      720   1  2                   P(N)=PI;
      730   1  2                   PROB=PROBOOC(U,BORNE_U,N-1,0);
      740   1  2                END;
      750   1  1            ELSE
                                  PROB=PROBOOC(U,BORNE_U,N-1,1);
      770   1  1            PROB=PROB+PI;
      780   1  1        END;
      790   1  0   RETURN(PROB);
      800   1  0   END PROBOOC;
```

6) Les sous-programmes PATHOOC et PATHSOC

définition de la tâche

L'objectif de ces sous-programmes est de trouver, d'emmagasiner et d'écrire (si indiqué) les chemins entre toutes les paires de noeuds du réseau. Ils vérifient aussi si les chemins contiennent des boucles (circuits).

Les chemins qui sont trouvés par ces sous-programmes sont emmagasinés:

1) comme une séquence ordonnée de numéros de noeuds, et

2) selon leur ordre de préférence.

Finalement, tout dépendant de la stratégie d'acheminement choisie, on utilise l'un ou l'autre de ces sous-programmes.

Ainsi, si la stratégie d'acheminement suit la règle OOC, on emploie le sous-programme PATHOOC autrement on emploie le sous-programme PATHSOC car à ce moment la stratégie sélectionnée suit la règle SOC.

Les figures 9 et 10 présentent respectivement les diagrammes logiques des sous-programmes PATHSOC et PATHOOC.

DIAGRAMME LOGIQUE DU SOUS-PROGRAMME PATHSOC

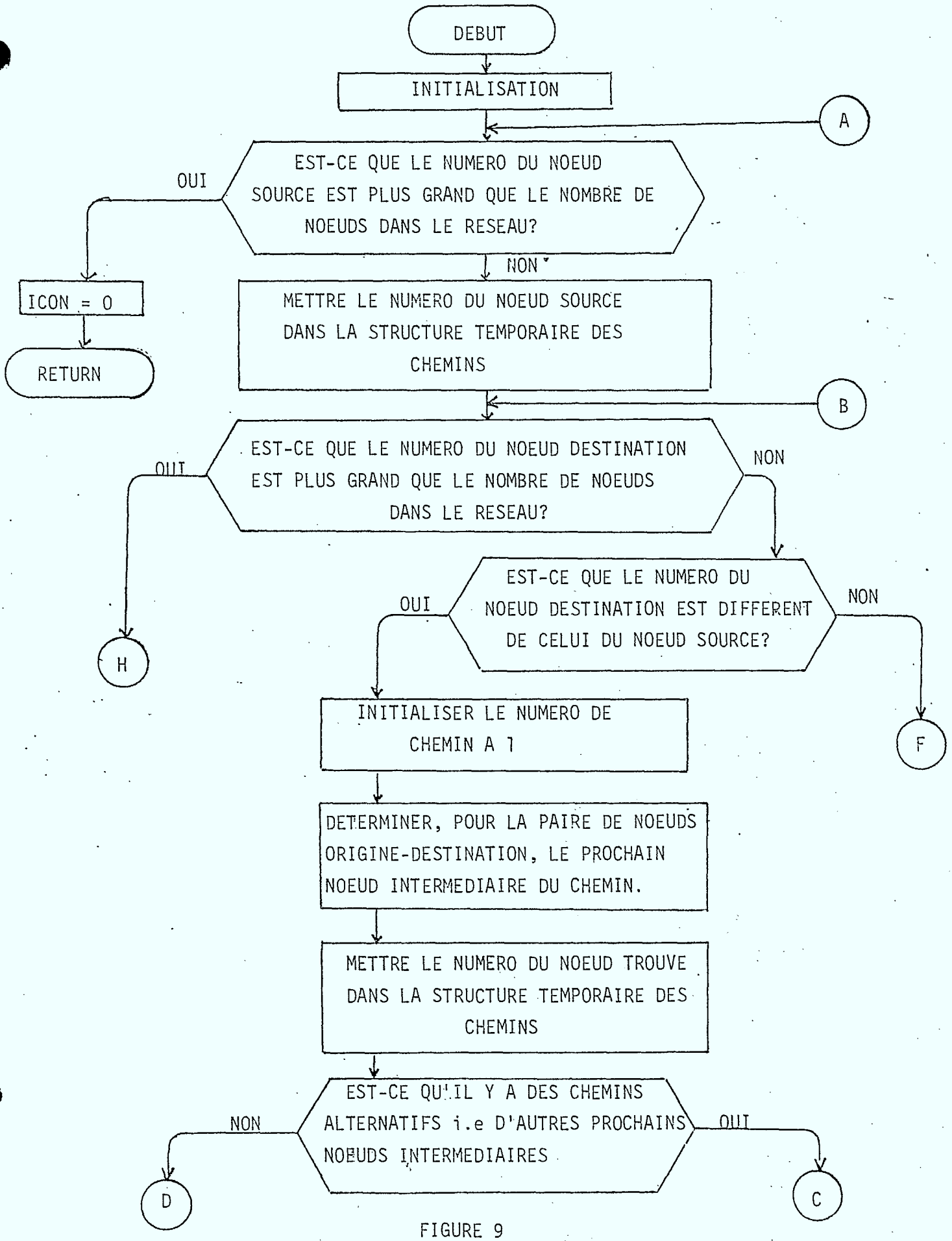


FIGURE 9

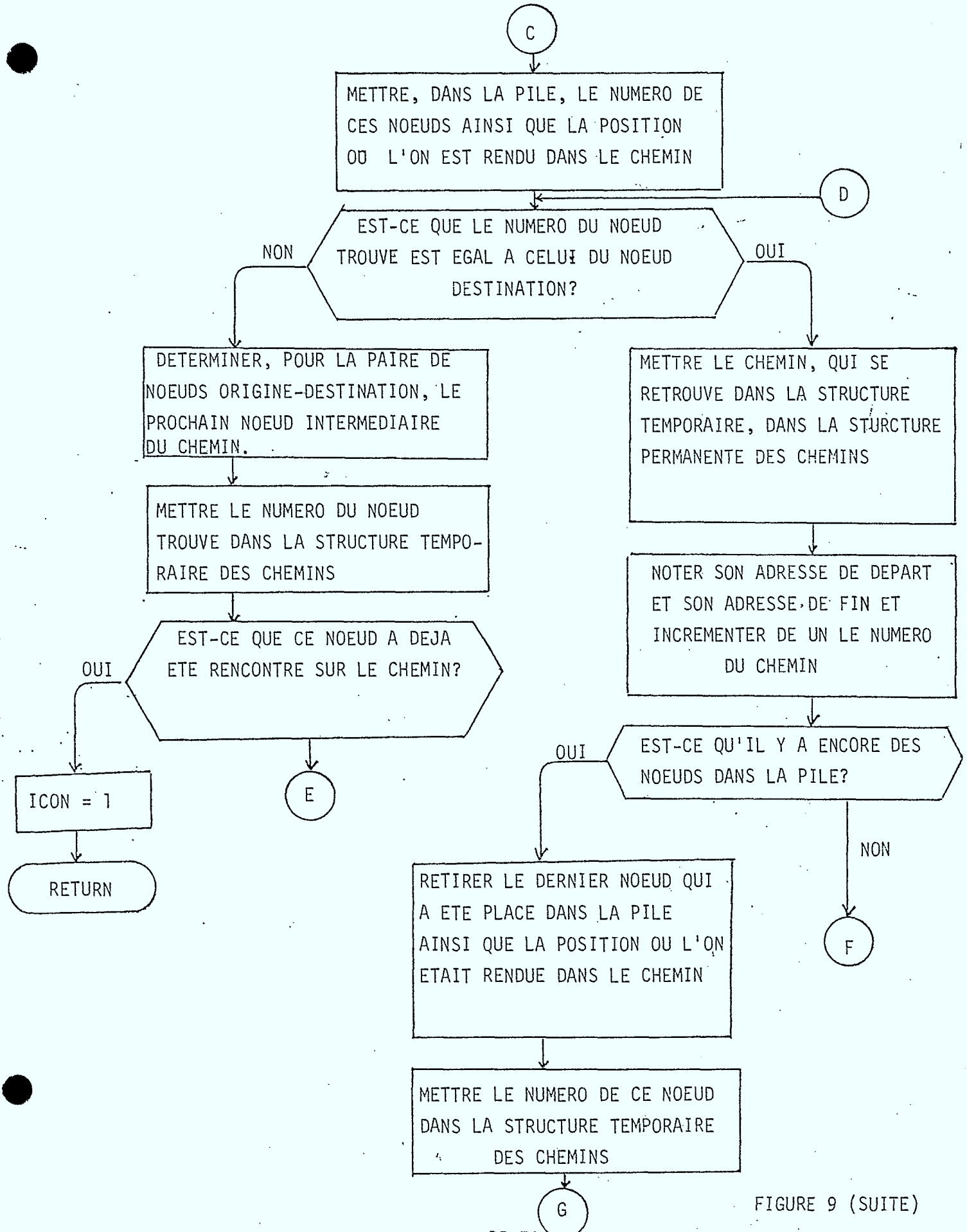


FIGURE 9 (SUITE)

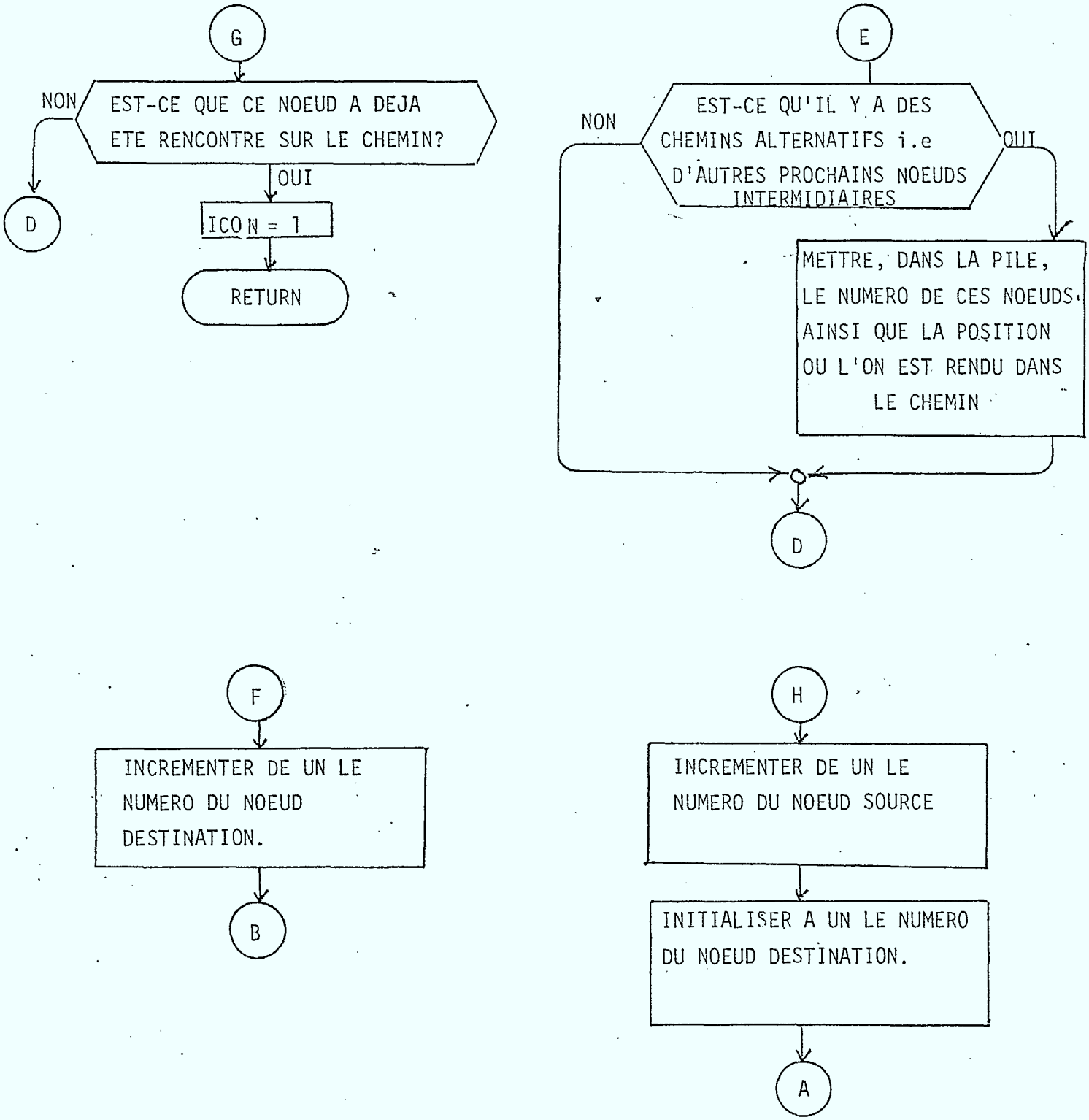

G
NON
EST-CE QUE CE NOEUD A DEJA ETE RENCONTRE SUR LE CHEMIN?
OUI
D
ICON = 1
RETURN
E
NON
EST-CE QU'IL Y A DES CHEMINS ALTERNATIFS i.e D'AUTRES PROCHAINS NOEUDS INTERMIDIAIRES
OUI
METTRE, DANS LA PILE, LE NUMERO DE CES NOEUDS AINSI QUE LA POSITION OU L'ON EST RENDU DANS LE CHEMIN
D
F
INCREMENTER DE UN LE NUMERO DU NOEUD DESTINATION.
B
H
INCREMENTER DE UN LE NUMERO DU NOEUD SOURCE
INITIALISER A UN LE NUMERO DU NOEUD DESTINATION.
A

FIGURE 9 (FIN)

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME PATHSOC

Les variables ICON, IRT, IWR, LOCATF, LOCATI, NPATH et NS ont été définies dans le dictionnaire des variables du programme principal.

IS = variable qui contient le numéro du noeud source qui est à l'étude. Si un chemin illégal est formé, elle contient, lorsque le contrôle est retourné au programme appelant, le numéro du noeud source dudit chemin.

ID = variable qui contient le numéro du noeud destination qui est à l'étude. Si un chemin illégal est formé, elle contient, lorsque le contrôle est retourné au programme appelant, le numéro du noeud destination dudit chemin.

INDPATH = variable qui contient, la position dans le vecteur NPATH de la dernière composante qui a été remplie.

IPATH = structure temporaire qui contient les numéros des noeuds constituant le chemin qui est en construction.

ITS = variable contenant le numéro du noeud source temporaire. Ce dernier correspond à un noeud qui se situe à la fin d'un lien sur le chemin en construction. Il peut être atteint en partant

du noeud source.

M = variable nous indiquant le nombre de noeuds + 1 qui, jusqu'ici ont été rencontrés sur le chemin en construction

NC = variable nous indiquant le numéro du chemin qui est en construction.

PILE (matrice de 2 colonnes) = pile où l'on emmagasine, selon l'ordre inverse de leur préférence, les destinations secondaires du noeud source et des noeuds sources temporaires (noeuds intermédiaires). De plus, pour chaque destination emmagasinée, on stocke la position où l'on est rendu dans la construction du chemin i.e la valeur de la variable M.

PTR = pointeur nous indiquant la prochaine composante à remplir dans la pile i.e le sommet de la pile.

OGRAMME PATHSOC

```
PATHSOC:PROCEDURE(IWR,ICON,IS,ID);

LISTING


PROCEDURE(IWR,ICON,IS,ID);

*****************************************************/
                                                     */
CTIFS:                                               */
UR UN RESEAU SUIVANT LES REGLES DE LA STRATEGIE      */
ACHEMINEMENT SOC                                     */
 1- TROUVER, STOCKER ET ECRIRE LES CHEMINS ENTRE     */
    TOUTES LES PAIRES DE NOEUDS DU RESEAU, ET        */
 2- VERIFIER SI CES CHEMINS CONTIENNENT DES BOU-     */
    CLES                                             */
                                                     */
*****************************************************/

E (NS,NPATH(5000),IRT(15,15,4))        FIXED(2)      EXTERNAL,
  (LOCATI(15,15,25),LOCATF(15,15,25)) FIXED(4)      EXTERNAL,
  (IWR,ICON)                           FIXED(1),
   INDPATH                             FIXED(4),
  (I,J,K,M,IS,ID,NC,ITS,PTR,WORK,
  IPATH(15),PILE(20,2))                FIXED(2);


ITIALISATION                                         */

=1 TO 15;
DO J=1 TO 15;
    DO K=1 TO 25;
       LOCATI(I,J,K)=0;
       LOCATF(I,J,K)=0;
    END;
END;
;


)PATH=0;
=1;
=1;
R=1;
2;


* TROUVER LE PROCHAIN NOEUD SOURCE                   */

)O WHILE (IS <= NS);

/* PLACER LE NUMERO DU NOEUD SOURCE DANS LA STRUCTURE */
```

```
5/I OPTIMIZING COMPILER          PATHSOC:PROCEDURE(IWR,ICON,IS,ID);

NUMBER LEV NT

                    /* TEMPORAIRE DES CHEMINS                                  */

  550    1  1          IPATH(1)=IS;

                    /* TROUVER LE PROCHAIN NOEUD DESTINATION                   */

  610    1  1          DO WHILE (ID <= NS);

                    /* INITIALISER LE NUMERO DU CHEMIN A 1                     */

  670    1  2              NC=1;

                    /* VERIFIER SI LE NOEUD DESTINATION EST EGAL AU NOEUD      */
                    /* SOURCE                                                  */

  740    1  2              IF ID ¬= IS THEN
                              DO;
  760    1  3                  IF IWR ¬= 0 THEN
                                  PUT EDIT ('FROM',IS,'TO',ID)
                                           (SKIP(3),COL(34),2(X(1),A,X(1),F(2)));
  790    1  3                  ELSE;

                    /* DETERMINER LE PROCHAIN NOEUD INTERMEDIAIRE DU CHEMIN    */
                    /* ET PLACER SON NUMERO DANS LA STRUCTURE TEMPORAIRE DES   */
                    /* CHEMINS                                                 */

  870    1  3                  ITS=IRT(IS,ID,1);
  880    1  3                  IPATH(M)=ITS;

                    /* S'IL Y A DES CHEMINS ALTERNATIFS ALORS METTRE DANS LA   */
                    /* PILE LE NUMERO DES AUTRES NOEUDS INTERMEDIAIRES AINSI   */
                    /* QUE LA POSITION OU L'ON EST RENDU DANS LE CHEMIN        */

  960    1  3                  DO I=1 TO 2;
  970    1  4                      WORK=IRT(IS,ID,4-I);
  980    1  4                      IF WORK ¬= 0 THEN
                                       DO;
 1000    1  5                        PILE(PTR,1)=WORK;
 1010    1  5                        PILE(PTR,2)=M;
 1020    1  5                        PTR=PTR+1;
 1030    1  5                    END;
 1040    1  4                  ELSE;
 1050    1  4              END;
```

```
P.L/I OPTIMIZING COMPILER          PATHSOC:PROCEDURE(IWR,ICON,IS,ID);

NUMBER LEV NT

  1060   1  3                    M=3;

  1090   1  3                    DO WHILE (M ¬= 2);
  1100   1  4                       DO WHILE (ITS ¬= ID);

                /* S'IL Y A DES CHEMINS ALTERNATIFS ALORS METTRE DANS LA    */
                /* PILE LE NUMERO DES AUTRES NOEUDS INTERMEDIAIRES AINSI    */
                /* QUE LA POSITION OU L'ON EST RENDU DANS LE CHEMIN         */

  1180   1  5                          DO I=1 TO 2;
  1190   1  6                             WORK=IRT(ITS,ID,4-I);
  1200   1  6                             IF WORK ¬= 0 THEN
  1220   1  7                                DO;
  1230   1  7                                   PILE(PTR,1)=WORK;
  1240   1  7                                   PILE(PTR,2)=M;
  1250   1  7                                   PTR=PTR+1;
  1260   1  6                                END;
  1270   1  6                             ELSE;
                                       END;

                /* DETERMINER LE PROCHAIN NOEUD INTERMEDIAIRE DU CHEMIN     */
                /* ET PLACER SON NUMERO DANS LA STRUCTURE TEMPORAIRE DES    */
                /* CHEMINS                                                  */

  1350   1  5                          ITS=IRT(ITS,ID,1);
  1360   1  5                          IPATH(M)=ITS;

                /* VERIFIER SI LE NOEUD A DEJA ETE RENCONTRE SUR LE         */
                /* CHEMIN                                                   */

  1430   1  5                          DO I=1 TO M
                                          WHILE (ITS ¬= IPATH(I));
  1450   1  6                          END;
  1460   1  5                          IF I < M THEN
                                          DO;
  1480   1  6                               ICON=1;
  1490   1  6                               PUT EDIT ('ERREUR *** ICON = ',ICON)
                                                  (SKIP(2),X(10),A,F(5));
  1510   1  6                               RETURN;
  1520   1  6                           END;
  1530   1  5                       ELSE;

  1550   1  5                       M=M+1;
  1560   1  5                    END;
```

```
PL/I OPTIMIZING COMPILER          PATHSDC:PROCEDURE(IWR,ICON,IS,ID);

 NUMBER LEV NT

                    /* UN CHEMIN A ETE TROUVE , METTRE LE CHEMIN QUI SE RE-     */
                    /* TROUVE DANS LA STRUCTURE TEMPORAIRE, DANS LA STRUCTURE   */
                    /* PERMANENTE DES CHEMINS, NOTER SON ADRESSE DE DEPART ET   */
                    /* SON ADRESSE DE FIN ET INCREMENTER LE NUMERO DU CHEMIN    */

   1660    1   4                    DO I=1 TO M-1;
   1670    1   5                      NPATH(INDPATH+I)=IPATH(I);
   1680    1   5                    END;

   1700    1   4                    IF IWR ¬= 0 THEN
                                      PUT EDIT ((IPATH(I) DO I=1 TO M-1))
                                               (COL(38),15(F(3)));
   1730    1   4                    ELSE;

   1750    1   4                    LOCATI(IS,ID,NC)=INDPATH+1;
   1760    1   4                    LOCATF(IS,ID,NC)=INDPATH+M-1;
   1770    1   4                    INDPATH=INDPATH+M-1;
   1780    1   4                    NC=NC+1;


   1810    1   4                    IF PTR ¬= 1 THEN
                                      DO;

                    /* RETIRER LE DERNIER NOEUD QUI A ETE PLACE DANS LA PILE    */
                    /* AINSI QUE LA POSITION DU L'ON ETAIT RENDU  DANS LE CHE-  */
                    /* MIN. METTRE LE NUMERO DE CE NOEUD DANS LA STRUCTURE      */
                    /* TEMPORAIRE DES CHEMINS                                   */

   1910    1   5                      PTR=PTR-1;
   1920    1   5                      ITS=PILE(PTR,1);
   1930    1   5                      M=PILE(PTR,2);
   1940    1   5                      IPATH(M)=ITS;

                    /* VERIFIER SI LE NOEUD A DEJA ETE RENCONTRE SUR LE         */
                    /* CHEMIN                                                   */

   2010    1   5                      DO I=1 TO M
                                        WHILE (ITS ¬= IPATH(I));
   2030    1   6                      END;
   2040    1   5                      IF I < M THEN
                                        DO;
   2060    1   6                      ICON=1;
   2070    1   6                      PUT EDIT ('ERREUR *** ICON = ',ICON)
                                               (SKIP(2),X(10),A,F(5));
   2090    1   6                      RETURN;
   2100    1   6                    END;
   2110    1   5                    ELSE;
   2120    1   5                    M=M+1;
   2130    1   5                  END;
```

```
PL/I OPTIMIZING COMPILER          PATHSOC:PROCEDURE(IWR,ICON,IS,ID);

   NUMBER LEV NT

     2140   1  4                          ELSE
                                             M=2;
     2160   1  4                       END;
     2170   1  3                    END;
     2180   1  2                 ELSE;
     2190   1  2                 ID=ID+1;
     2200   1  2           END;
     2210   1  1           IS=IS+1;
     2220   1  1           ID=1;
     2230   1  1     END;
     2240   1  0     ICON=0;
     2250   1  0     END PATHSOC;
```

DIAGRAMME LOGIQUE DU SOUS-PROGRAMME PATHOOC

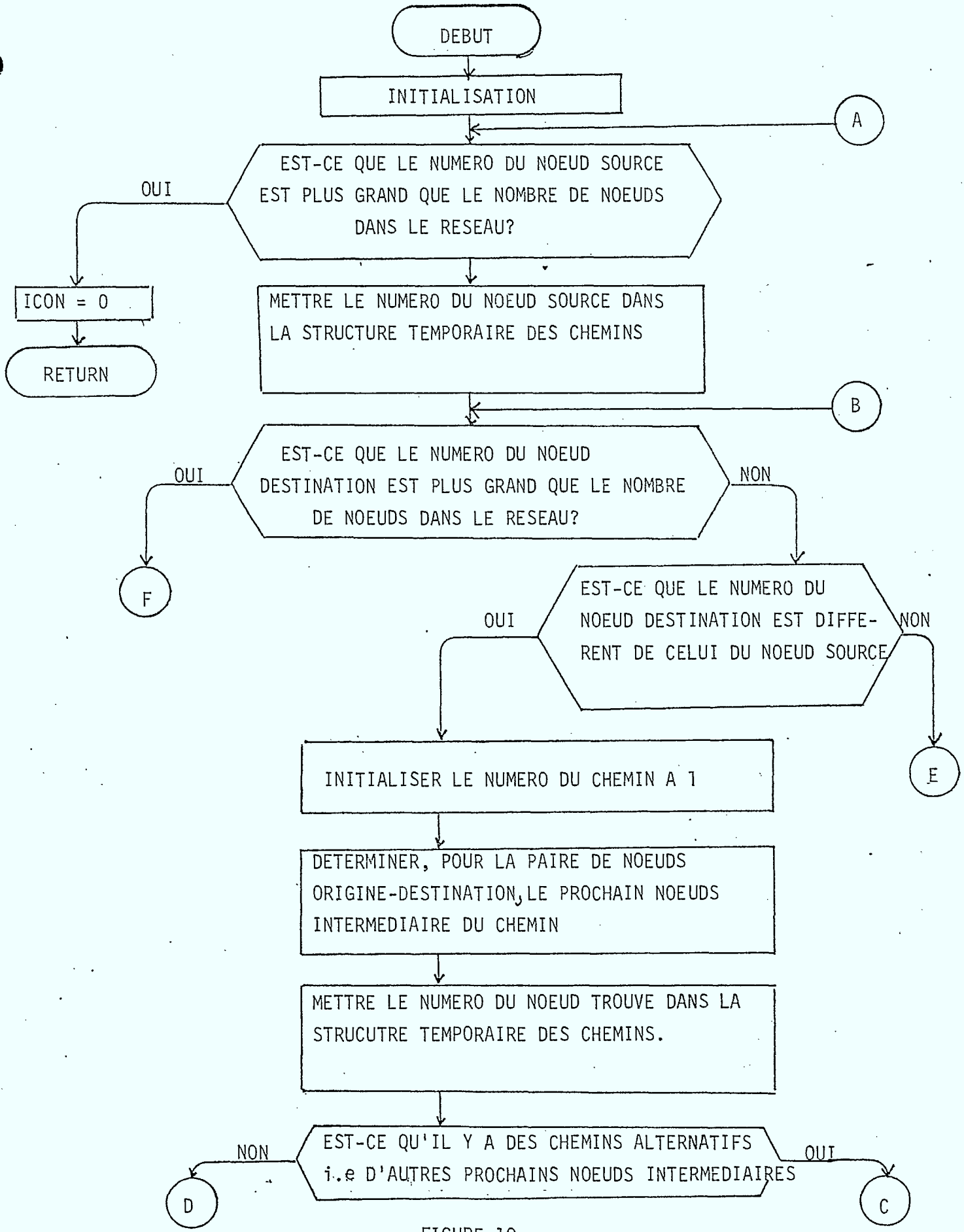


FIGURE 10

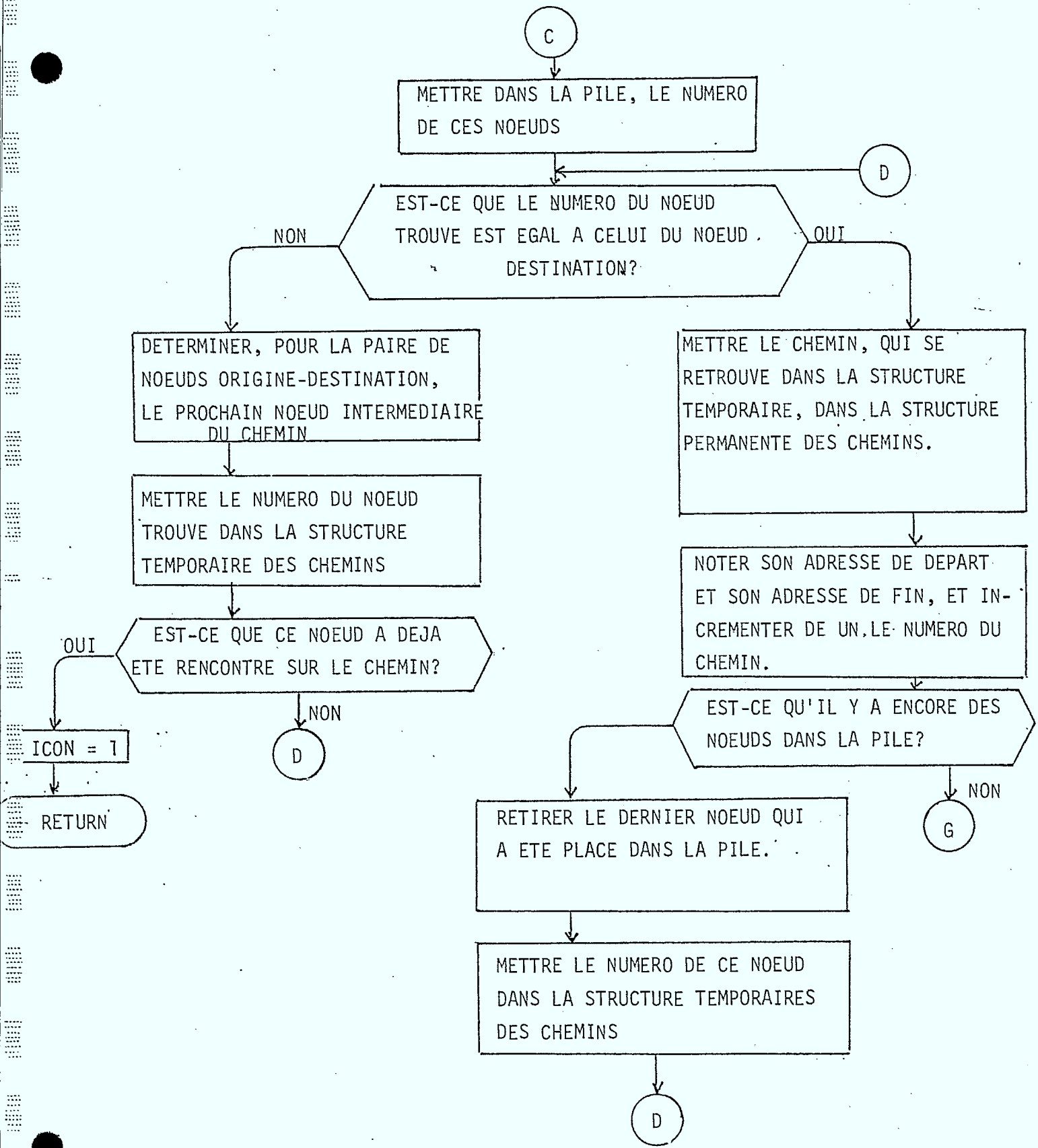


FIGURE 10 (SUITE)

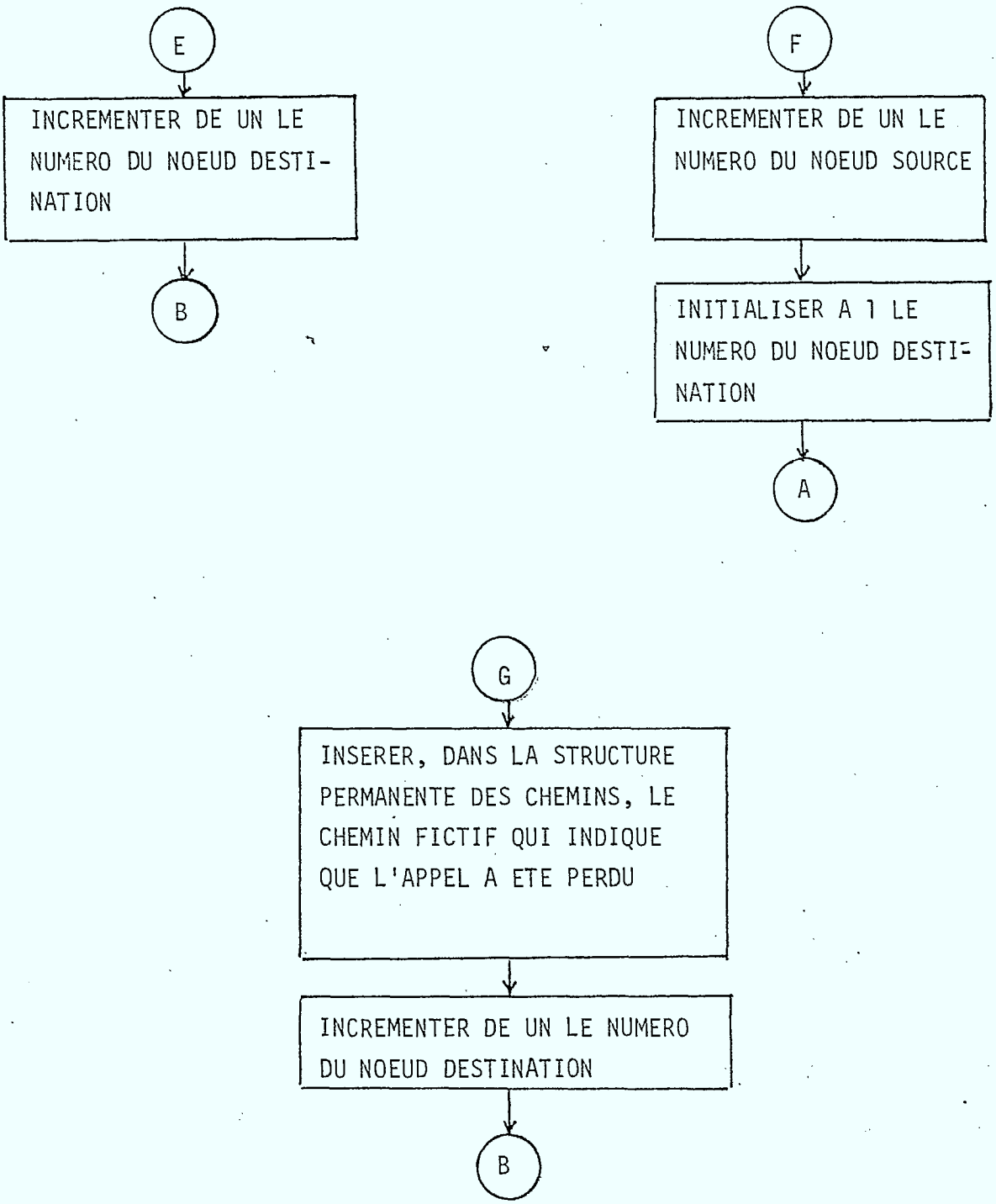


FIGURE 10 (FIN)

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME PATHOOC

Toutes les variables, à l'exception de la variable pile, ont la même signification que celle donnée dans le dictionnaire des variables du sous-programme PATHOOC.

PILE (vecteur) = pile où l'on emmagasine, selon leur ordre de préférence, les destinations secondaires du noeud source.

LOST = variable contenant la valeur NS + 1. Elle est employée pour construire le chemin fictif. Ce dernier est utilisée lorsqu'un appel est perdu.

LISTING DU SOUS-PROGRAMME PATHOOC

```
OPTIMIZING COMPILER          PATHOOC:PROCEDURE(IWR,ICON,IS,ID);

                SOURCE LISTING

NUMBER LEV NT

  10       0  PATHOOC:PROCEDURE(IWR,ICON,IS,ID);

             /*****************************************************************/
             /*                                                               */
             /* OBJECTIFS:                                                    */
             /*   POUR UN RESEAU SUIVANT LES REGLES DE LA STRATEGIE           */
             /*   D'ACHEMINEMENT OOC                                          */
             /*      1- TROUVER, STOCKER ET ECRIRE LES CHEMINS ENTRE          */
             /*         TOUTES LES PAIRES DE NOEUDS DU RESEAU, ET             */
             /*      2- VERIFIER SI CES CHEMINS CONTIENNENT DES BOU-          */
             /*         CLES                                                  */
             /*                                                               */
             /*****************************************************************/

 170   1   0 DECLARE (NS,NPATH(5000),IRT(15,15,4))      FIXED(2)      EXTERNAL,
                  (LOCATI(15,15,25),LOCATF(15,15,25)) FIXED(4)      EXTERNAL,
                  (IWR,ICON)                          FIXED(1),
                  (I,INDPATH)                         FIXED(4),
                  (J,K)                               FIXED(2),
                  (M,IS,ID,NC,ITS,PTR,LOST,WORK,
                   PILE(20),IPATH(15))                FIXED(2);

             /* INITIALISATION                                                */

 290   1   0 DO I=1 TO 15;
 300   1   1    DO J=1 TO 15;
 310   1   2       DO K=1 TO 25;
 320   1   3          LOCATI(I,J,K)=0;
 330   1   3          LOCATF(I,J,K)=0;
 340   1   3       END;
 350   1   2    END;
 360   1   1 END;

 380   1   0 INDPATH=0;
 390   1   0 IS=1;
 400   1   0 ID=1;
 410   1   0 PTR=1;
 420   1   0 LOST=NS+1;

             /* TROUVER LE PROCHAIN NOEUD SOURCE                              */

 490   1   0 DO WHILE (IS <= NS);
```

```
PL/I OPTIMIZING COMPILER          PATHDOC:PROCEDURE(IWR,ICON,IS,ID);

   NUMBER LEV NT

                    /* PLACER LE NUMERO DU NOEUD SOURCE DANS LA STRUCTURE      */
                    /* TEMPORAIRE DES CHEMINS                                  */

     560    1  1       IPATH(1)=IS;

                    /* TROUVER LE PROCHAIN NOEUD DESTINATION                   */

     620    1  1       DO WHILE (ID <= NS);

                    /* INITIALISER LE NUMERO DU CHEMIN A 1                     */

     680    1  2           NC=1;

                    /* VERIFIER SI LE NOEUD DESTINATION EST EGAL AU NOEUD      */
                    /* SOURCE                                                  */

     750    1  2           IF ID ¬= IS THEN
                              DO;
     770    1  3               IF IWR ¬= 0 THEN
                                  PUT EDIT ('FROM',IS,'TO',ID)
                                           (SKIP,2(X(1),A,X(1),F(2)));
     800    1  3               ELSE;

                    /* DETERMINER LE PROCHAIN NOEUD INTERMEDIAIRE DU CHEMIN    */
                    /* ET PLACER SON NUMERO DANS LA STRUCTURE TEMPORAIRE DES   */
                    /* CHEMINS                                                 */

     890    1  3                   ITS=IRT(IS,ID,1);
     900    1  3                   IPATH(2)=ITS;

                    /* S'IL Y A DES CHEMINS ALTERNATIFS ALORS METTRE DANS LA   */
                    /* PILE LE NUMERO DES AUTRES NOEUDS INTERMEDIAIRES         */

     970    1  3                   DO I=1 TO 2;
     980    1  4                       WORK=IRT(IS,ID,4-I);
     990    1  4                       IF WORK ¬= 0 THEN
                                            DO;
    1010    1  5                         PILE(PTR)=WORK;
    1020    1  5                         PTR=PTR+1;
    1030    1  5                       END;
    1040    1  4                   ELSE;
    1050    1  4               END;
```

```
pL/I OPTIMIZING COMPILER          PATHOOC:pROCEDURE(IWR,ICON,IS,ID);

  NUMBER LEV NT

    1060    1  3                    M=3;

    1080    1  3                    DO WHILE (M ¬=2);
    1090    1  4                       DO WHILE (ITS ¬= ID);

                  /* DETERMINER LE PROCHAIN NOEJD INTERMEDIAIRE DU CHEMIN ET */
                  /* METTRE SON NUMERO DANS LA STRUCTURE TEMPORAIRE DES      */
                  /* CHEMINS                                                 */

    1170    1  5                          ITS=IRT(ITS,ID,1);
    1180    1  5                          IPATH(M)=ITS;

                  /* VERIFIER SI LE NOEUD A DEJA ETE RENCONTRE SUR LE        */
                  /* CHEMIN                                                  */

    1250    1  5                          DO I=1 TO M
                                             WHILE (ITS ¬= IPATH(I));
    1270    1  6                          END;
    1280    1  5                          IF I < M THEN
                                             DO;
    1300    1  6                             ICON=1;
    1310    1  6                             PUT EDIT (;ERREUR *** ICON =-1,ICON)
                                                      (SKIP(2),X(10),A,F(5));
    1330    1  6                             RETURN;
    1340    1  6                          END;
    1350    1  5                          ELSE;
    1360    1  5                          M=M+1;
    1370    1  5                    END;

                  /* UN CHEMIN A ETE TROUVE . METTRE LE CHEMIN QUI SE RE-    */
                  /* TROUVE DANS LA STRUCTURE TEMPORAIRE, DANS LA STRUCTURE  */
                  /* PERMANENTE DES CHEMINS. NOTER SON ADRESSE DE DEPART ET  */
                  /* SON ADRESSE DE FIN ET INCREMENTER LE NUMERO DU CHEMIN   */

    1470    1  4                    DO I=1 TO M-1;
    1480    1  5                       NPATH(INDPATH+I)=IPATH(I);
    1490    1  5                    END;

    1510    1  4                    IF IWR ¬= 0 THEN
                                       PUT EDIT ((IPATH(I) DO I=1 TO M-1))
                                                (COL(2),15(F(3)));
    1540    1  4                    ELSE;

    1560    1  4                    LOCATI(IS,ID,NC)=INDPATH+1;
    1570    1  4                    LOCATF(IS,ID,NC)=INDPATH+M-1;
    1580    1  4                    INDPATH=INDPATH+M-1;
    1590    1  4                    NC=NC+1;
```

```
PL/I OPTIMIZING COMPILER          PATHOOC:PROCEDURE(IWR,ICON,IS,ID);

   NUMBER LEV NT

     1610    1  4                      IF PTR ¬= 1 THEN
                                          DO;

                        /* RETIRER LE DERNIER NOEUD QUI A ETE PLACE DANS LA PILE   */
                        /* ET METTRE LE NUMERO DE CE NOEUD DANS LA STRUCTURE TEM-  */
                        /* PORAIRE DES CHEMINS                                     */

     1700    1  5                            PTR=PTR-1;
     1710    1  5                            ITS=PILE(PTR);
     1720    1  5                            IPATH(2)=ITS;
     1730    1  5                            M=3;
     1740    1  5                         END;
     1750    1  4                   ELSE
                                       M=2;
     1770    1  4                   END;
     1780    1  3               END;
     1790    1  2           ELSE;

                        /* INSERER, DANS LA STRUCTURE PERMANENTE DES CHEMINS, LE   */
                        /* CHEMIN FICTIF QUI INDIQUE QUE L'APPEL A ETE PERDU       */

     1860    1  2           NPATH(INDPATH+1)=IS;
     1870    1  2           NPATH(INDPATH+2)=LOST;

     1890    1  2           IF IWR ¬= 0 THEN
                               PUT EDIT ((NPATH(I) DO I=INDPATH+1 TO INDPATH+2))
                                        (COL(2),2(F(3)));
     1920    1  2           ELSE;

     1940    1  2           LOCATI(IS,ID,NC)=INDPATH+1;
     1950    1  2           LOCATF(IS,ID,NC)=INDPATH+2;
     1960    1  2           INDPATH=INDPATH+2;
     1970    1  2           ID=ID+1;
     1980    1  2       END;

     2000    1  1       IS=IS+1;
     2010    1  1       ID=1;
     2020    1  1   END;
     2030    1  0   ICON=0;
     2040    1  0   END PATHOOC;
```

7) Le sous-programme SORT1

définition de la tâche

L'objectif du sous-programme SORT1 est de trouver la paire de noeuds ORIGINE-DESTINATION possédant la plus grande probabilité de blocage pondérée (WEITHTED NNGOS) ainsi que la valeur de cette dernière. De plus, il détermine le nombre de NNGOS pondérés qui dépassent la valeur de BPMAX i.e. La probabilité de blocage maximale autorisée.

La figure 11 présente le diagramme logique de ce sous-programme.

DIAGRAMME LOGIQUE DU SOUS-PROGRAMME SORT1

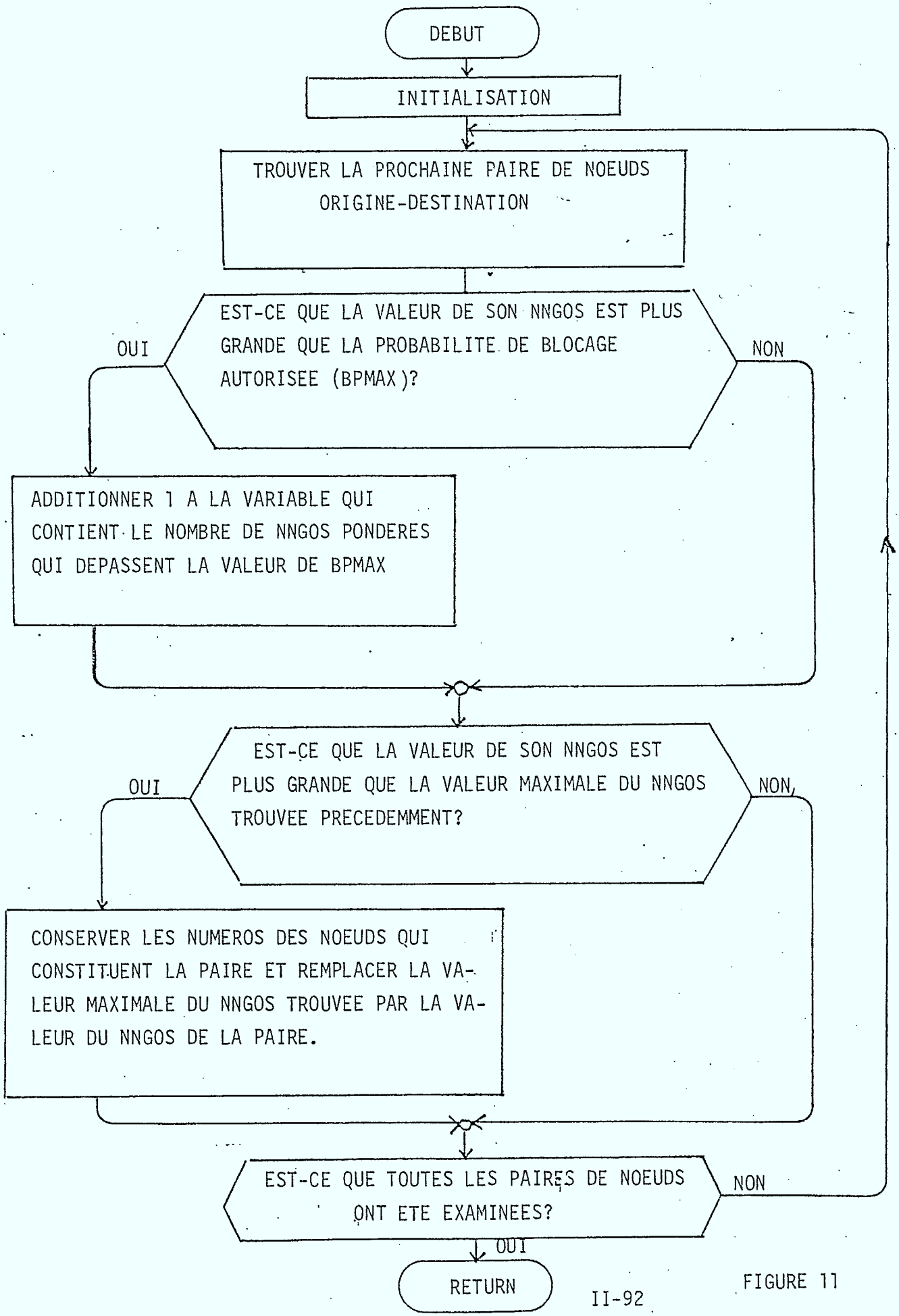


FIGURE 11

DICTIONNAIRE DES VARIABLES DU SOUS-PROGRAMME SORT1

Les variables BPMAX, CLSD, NS, TR et WT ont été définies dans le dictionnaire des variables du programme principal.

BIGBP = variable qui contient la plus grande probabilité de blocage pondérée, rencontrée pour une paire de noeuds ORIGINE-DESTINATION (LARGEST WEIGHTED NNGOS).

DUM = variable qui contient la probabilité de blocage pondérée de la paire de noeuds ORIGINE-DESTINATION qui est à l'étude (WEIGHTED NNGOS)

IS,ID = variables qui contiennent respectivement les noeuds source et destination de la paire de noeuds ORIGINE-DESTINATION qui possède la plus grande probabilité de blocage pondérée du réseau.

NEXD = variable qui contient le nombre de NNGOS pondérés qui exèdent la valeur contenue dans BPMAX.

LISTING DU SOUS-PROGRAMME SORT1

```
/I OPTIMIZING COMPILER          SORT1:PROCEDURE(BPMAX,IS,ID,BIGBP,NEXD);

                    SOURCE LISTING

UMBER LEV NT

   10       0   SORT1:PROCEDURE(BPMAX,IS,ID,BIGBP,NEXD);
                /*************************************************************/
                /*                                                           */
                /* OBJECTIFS:                                                */
                /*       1- TROUVER LA PAIRE DE NOEUDS POSSEDANT LE PLUS     */
                /*          GRAND NNGOS PONDERE AINSI QUE LA VALEUR DE CE    */
                /*          NNGOS                                            */
                /*       2- DETERMINER LE NOMBRE DE NNGOS PONDERES QUI       */
                /*          DEPASSENT LA VALEUR DE BPMAX                     */
                /*                                                           */
                /*************************************************************/
  140   1   0   DECLARE (TR(15,15),WT(15,15),CLSD(15,15))  FLOAT(16)      EXTERNAL,
                      NS                                   FIXED(2)       EXTERNAL,
                      (DUM,BPMAX,BIGBP)                    FLOAT(16),
                      (I,J,IS,ID)                          FIXED(2),
                      NEXD                                 FIXED(3);

                /* INITIALISATION                                            */

  240   1   0   NEXD=0;
  250   1   0   BIGBP=0;

                /* EXAMINER TOUS LES NNGOS ET TROUVER CELUI QUI EST LE       */
                /* PLUS ELEVE AVEC SA PAIRE O-D QUI LUI EST ASSOCIE          */

  320   1   0   DO I=1 TO NS;
  330   1   1     DO J=1 TO NS;
  340   1   2       IF I ¬= J THEN
                      DO;
  360   1   3         IF TR(I,J) ¬= 0.0E0 THEN
                        DO;
  380   1   4           DUM=CLSD(I,J)/TR(I,J)*WT(I,J);
  390   1   4           IF DUM >= BPMAX THEN
                          NEXD=NEXD+1;
  410   1   4           ELSE;
  430   1   4           IF DUM >= BIGBP THEN
                            DO;
  450   1   5             BIGBP=DUM;
  460   1   5             IS=I;
  470   1   5             ID=J;
  480   1   5           END;
  490   1   4         ELSE;
  500   1   4       END;
  510   1   3     ELSE;
```

```
PL/I OPTIMIZING COMPILER          SORT1:PROCEDURE(BPMAX,IS,ID,BIGBP,NEXD);

  NUMBER LEV NT

     520   1  3                    END;
     530   1  2                ELSE;
     540   1  2             END;
     550   1  1          END;
     560   1  0          END SORT1;
```

4. EXEMPLE D'UTILISATION

Dans cette section du rapport, nous vous présentons un exemple d'utilisation du package ETARET. Ce dernier vous montre comment employer le logiciel pour réaliser l'analyse des performances d'un réseau téléphonique.

Le réseau que nous voulons analyser, est un réseau fictif qui a été dérivé des données (partielles) que la compagnie BELL nous a fourni. Il est constitué de dix (10) noeuds et de 40 liens.

Après avoir réalisé les cinq étapes décrites dans la section II-D. du rapport, nous avons l'information suivante:

1) la structure du réseau (figure 12);

2) le nombre de tronçons pour chaque lien (Table 1);

3) la matrice du trafic (en CCS) (Table 2);

4) la table d'acheminement (TABLE 3), et

5) les options choisies pourle réseau:

a) l'impression de tous les chemins;

b) l'impression de la probabilité d'utilisation des chemins ainsi que le trafic amené par ces derniers;

c) la matrice du trafic sera entrée en unités d'ERLANG;

d) l'acheminement du réseau suit les règles de la stratégie SOC, et

e) la valeur de BPMAX: 0.02

Pour terminer, on détermine le nombre de conditions (sous-charge, normale et surcharge du réseau) que l'on veut vérifier ainsi que les valeurs de ces conditions.

La table 4 illustre la séquence des cartes de données pour ce réseau. La figure 13 illustre, pour le réseau décrit précédemment, les sorties du package ETARET.

Le réseau BELL

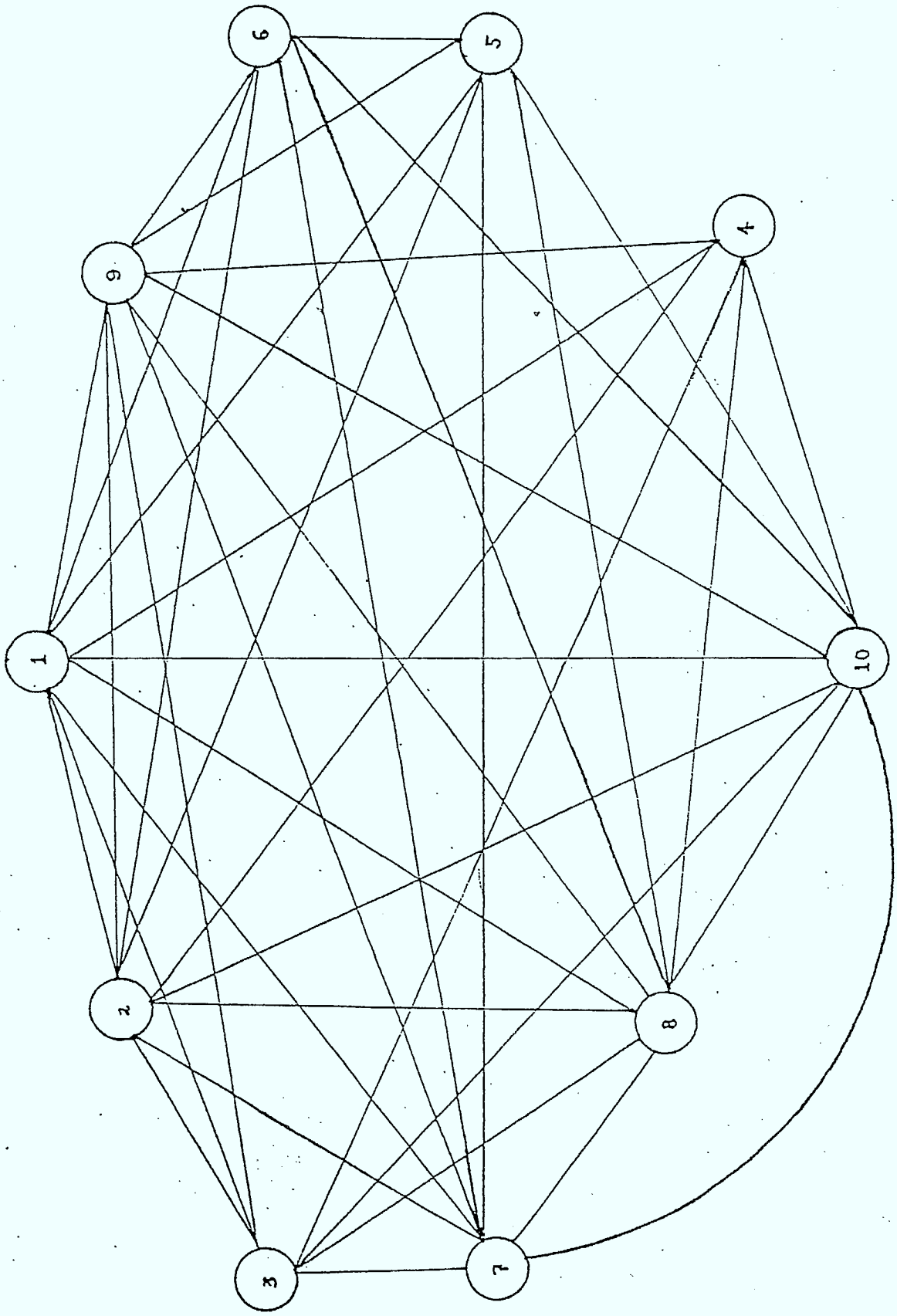


Figure 12

de \ vers	1	2	3	4	5	6	7	8	9	10
1	-	546	135	56	91	265	39	602	205	616
2		-	35	17	45	131	14	313	292	208
3			-	22	0	0	22	63	13	90
4				-	0	0	0	47	7	64
5					-	187	15	21	14	154
6						-	46	98	45	352
7							-	16	2	50
8								-	186	208
9									-	204
10										-

Table 1: Nombre de tronçons pour chaque lien.

vers de	1	2	3	4	5	6	7	8	9	10
1	-	11956	2848	1328	2238	5844	879	12578	3322	11726
2	10833	-	717	420	1007	2534	331	6851	5614	3803
3	2623	611	-	375	0	0	355	1302	231	1588
4	1354	359	368	-	0	0	0	821	137	819
5	1800	860	0	0	-	3743	238	449	364	2759
6	5441	2407	0	0	3962	-	948	2430	910	7081
7	922	332	349	0	376	848	-	430	0	779
8	12063	6401	1058	915	576	2269	373	-	3699	3879
9	5325	5595	272	162	426	918	88	3885	-	3727
10	11837	3545	1182	1062	2526	6007	637	3478	3482	-

Table 2: Matrice du trafic (en CCS)

vers de	1	2	3	4	5	6	7	8	9	10
1	-	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10
2	1,10	-	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10
3	1,10	2,10	-	4,10	10	10	7,10	8,10	9,10	10
4	1,10	2,10	3,10	-	10	10	10	8,10	9,10	10
5	1,10	2,10	10	10	-	6,10	7,10	8,10	9,10	10
6	1,10	2,10	3,10	10	5,10	-	7,10	8,10	9,10	10
7	1,10	2,10	3,10	10	5,10	6,10	-	8,10	9,10	10
8	1,10	2,10	3,10	4,10	5,10	6,10	7,10	-	9,10	10
9	1,10	2,10	3,10	4,10	5,10	6,10	10	8,10	-	10
10	1	2	3	4	5	6	7	8	9	-

Table 3: Table d'acheminement

DISPOSITION DES DONNEES SUR CARTES

	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
1	1 1 E R C S 0 C			C A R T E S D E S O P T I O N S				
2	1 0 4 0 0 . 2 0			D E S C R I P T I O N S D E S G A R A C T E R I S T I Q U E S D U R E S E A U				
3	1 2 1	5 4 6						
4	1 3 2	1 3 5						
5	1 4 3	5 6						
6	1 5 4	9 1						
7	1 6 5	2 6 5						
8	1 7 6	3 9						
9	1 8 7	6 0 2						
10	1 9 8	2 0 5						
11	1 1 0 9	6 1 6						
12	2 1 3 1 0	3 5						
13	2 1 4 1 1	1 7						
14	2 1 5 1 2	4 5						
15	2 1 6 1 3	1 3 1						
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
16	2 1 7 1 4	1 4						
17	2 1 8 1 5	3 1 3						
18	2 1 9 1 6	2 9 2						
19	2 1 1 0 1 7	2 0 8						
20	3 1 4 1 8	2 2		D I A N I S C E S Q U A R A N T I E C A R T E S , I D N I		D I O N N E U N E		
21	3 1 7 1 9	2 2		D E S C R I P T I O N D E S C H A M P S U N D E S L I E N S . L I G N A G E				
22	3 1 8 2 0	6 3		C A R T E C O M P L E M E N T D I A N S L ' O R D R E L I E N D E U D I T				
23	3 1 9 2 1	1 3		O R I G I N E L I E N D E U D I D E S T I N A T I O N , L E N U M E R O				
24	3 1 0 2 2	9 0		D U L I E N E T L I E N O M B R E D E T R O N C O N S				
25	4 1 8 2 3	4 7						
26	4 1 9 2 4	7						
27	4 1 0 2 5	6 9						
28	5 6 2 6	1 8 7						
29	5 7 2 7	1 5						
30	5 1 8 2 8	2 1						
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80

TABLE 4

DISPOSITION DES DONNEES SUR CARTES

	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
1	5 9 2 9	1 4						
2	5 1 0 3 0	1 5 4						
3	6 7 3 1	4 6						
4	6 8 3 2	9 8						
5	6 9 3 3	4 5						
6	6 1 0 3 4	3 5 2						
7	7 8 3 5	1 6						
8	7 9 3 6	2						
9	7 1 0 3 7	5 0						
10	8 9 3 8	1 8 6						
11	8 1 0 3 9	2 0 8						
12	9 1 0 4 0	2 0 4						
13	1 2 2 1 0	0 3 0 0 . 9 3	1 . 0					
14	1 3 3 1 0	0 7 2 . 8 7	1 . 0					
15	1 4 4 1 0	0 3 7 . 6 2	1 . 0					
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
16	1 5 1 5 1 0	0 1 5 0 . 0 0	1 . 0					
17	1 6 1 6 1 0	0 1 5 1 . 1 5	1 . 0					
18	1 7 1 7 1 0	0 2 5 . 6 2	1 . 0					
19	1 8 1 8 1 0	0 3 3 5 . 1 0	1 . 0					
20	1 9 1 9 1 0	0 1 4 7 . 9 2	1 . 0					
21	1 1 0 1 0 0	0 0 . 0 0	1 . 0					
22	2 1 1 1 0	0 3 3 2 . 1 2	1 . 0					
23	2 3 3 1 0	0 1 8 . 3 7	1 . 0					
24	2 4 4 1 0	0 9 . 7 8	1 . 0					
25	2 5 5 1 0	0 2 3 . 8 8	1 . 0					
26	2 6 6 1 0	0 6 6 . 8 5	1 . 0					
27	2 7 7 1 0	0 9 . 2 2	1 . 0					
28	2 8 8 1 0	0 1 7 7 . 8 4	1 . 0					
29	2 9 9 1 0	0 1 5 5 . 4 2	1 . 0					
30	2 1 0 1 0 0	0 0 . 0 0	1 . 0					
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80

TABLE 4 (SUITE)

DISPOSITION DES DONNEES SUR CARTES

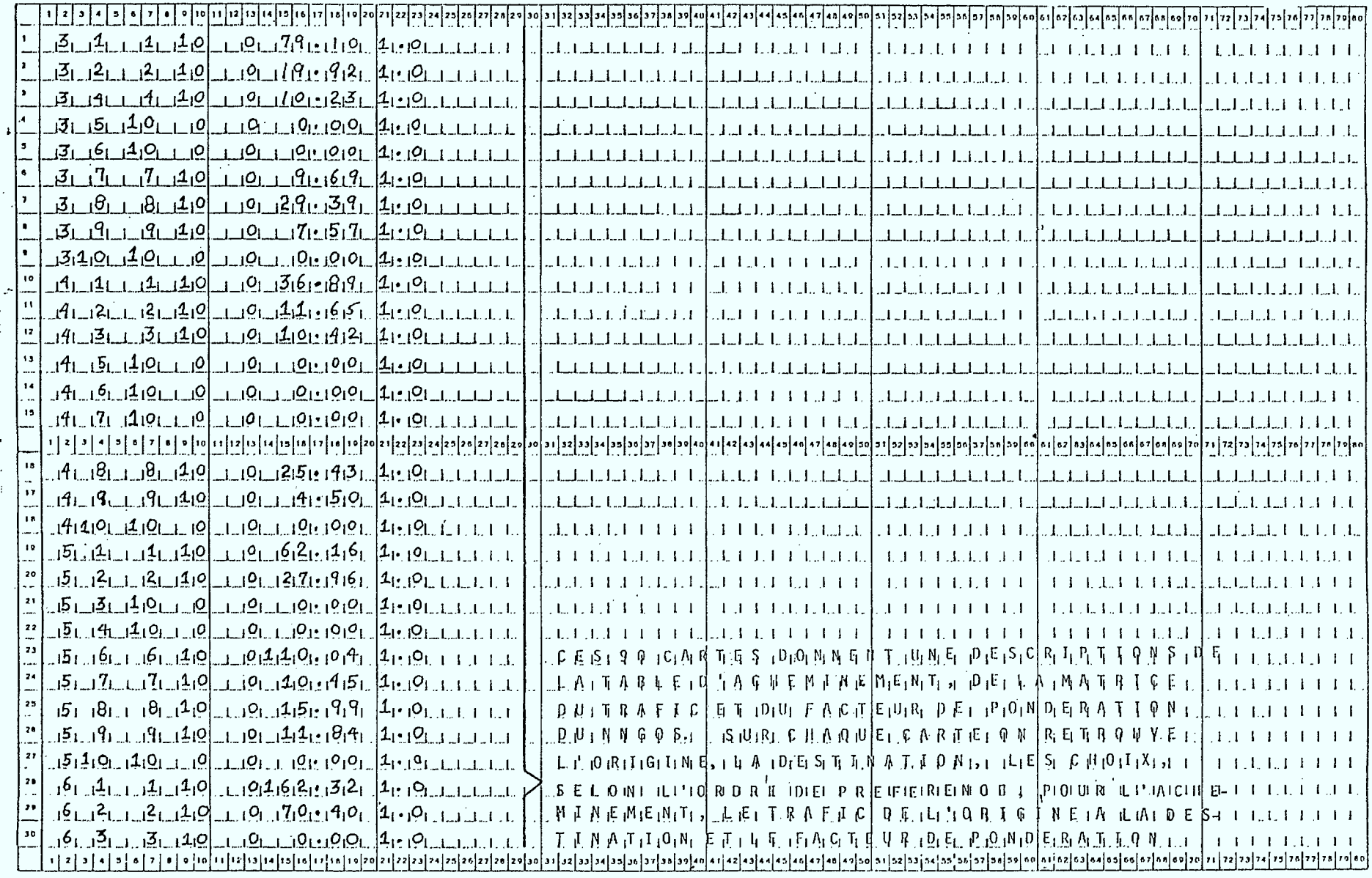


TABLE 4 (SUITE)

DISPOSITION DES DONNEES SUR CARTES

	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
1	6 4 1 0 0	0 0 0 0 0	1 0	DUNNGPS				
2	6 5 5 1 0	0 1 0 3 9 6	1 0					
3	6 7 7 1 0	0 2 3 5 6	1 0					
4	6 8 8 1 0	0 6 3 0 2	1 0					
5	6 9 9 1 0	0 2 5 4 9	1 0					
6	6 1 0 1 0 0	0 0 0 0 0	1 0					
7	7 1 1 1 0	0 2 4 4 2	1 0					
8	7 2 2 1 0	0 9 1 8	1 0					
9	7 3 3 1 0	0 9 8 5	1 0					
10	7 4 1 0 0	0 0 0 0 0	1 0					
11	7 5 5 1 0	0 6 6 2	1 0					
12	7 6 6 1 0	0 2 6 3 5	1 0					
13	7 8 8 1 0	0 1 0 3 6	1 0					
14	7 9 9 1 0	0 2 4 4	1 0					
15	7 1 0 1 0 0	0 0 0 0 0	1 0					
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
16	8 1 1 1 0	0 3 4 9 3 8	1 0					
17	8 2 2 1 0	0 1 9 0 3 2	1 0					
18	8 3 3 1 0	0 3 6 1 0	1 0					
19	8 4 4 1 0	0 2 2 8 0	1 0					
20	8 5 5 1 0	0 1 2 4 8	1 0					
21	8 6 6 1 0	0 6 7 5 1	1 0					
22	8 7 7 1 0	0 1 1 9 4	1 0					
23	8 9 9 1 0	0 1 0 7 9 2	1 0					
24	8 1 0 1 0 0	0 0 0 0 0	1 0					
25	9 1 1 1 0	0 9 2 2 8	1 0					
26	9 2 2 1 0	0 1 5 5 9 6	1 0					
27	9 3 3 1 0	0 6 4 3	1 0					
28	9 4 4 1 0	0 3 8 0	1 0					
29	9 5 5 1 0	0 1 0 1 0	1 0					
30	9 6 6 1 0	0 2 5 2 7	1 0					
	1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60	61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80

TABLE 4 (SUITE)

DISPOSITION DES DONNEES SUR CARTES

Ligne	1–10	11–20	21–30	31–80
1	9 7 10 0	0 0.00	1.0	
2	9 8 8 10	0102.76	1.0	
3	9 10 10 0	0 0.00	1.0	
4	10 1 1 0	0 0.00	1.0	
5	10 2 2 0	0 0.00	1.0	
6	10 3 3 0	0 0.00	1.0	
7	10 4 4 0	0 0.00	1.0	
8	10 5 5 0	0 0.00	1.0	
9	10 6 6 0	0 0.00	1.0	
10	10 7 7 0	0 0.00	1.0	
11	10 8 8 0	0 0.00	1.0	
12	10 9 9 0	0 0.00	1.0	
13	1			NOMBRE DE CONDITIONS A VERIFIER:
14	1.00			LA CONDITION QUI EST DANS CE CAS LA
15				CONDITION NORMALE

TABLE 4 (FIN)

FIGURE 13 (DEBUT)

```
 1  1 ERG SOC
10 40 0.0200
  1  2  1                546
  1  3  2                135
  1  4  3                 56
  1  5  4                 91
  1  6  5                265
  1  7  6                 39
  1  8  7                602
  1  9  8                205
  1 10  9                616
  2  3 10                 35
  2  4 11                 17
  2  5 12                 45
  2  6 13                131
  2  7 14                 14
  2  8 15                313
  2  9 16                292
  2 10 17                208
  3  4 18                 22
  3  7 19                 22
  3  8 20                 63
  3  9 21                 13
  3 10 22                 90
  4  8 23                 47
  4  9 24                  7
  4 10 25                 64
  5  6 26                107
  5  7 27                 15
  5  8 28                 21
  5  9 29                 14
  5 10 30                154
  6  7 31                 46
  6  8 32                 98
  6  9 33                 45
  6 10 34                352
  7  8 35                 16
  7  9 36                  2
  7 10 37                 50
  8  9 38                186
  8 10 39                208
  9 10 40                204
  1  2   2 10   0300.93   1.00
  1  3   3 10   0 72.87   1.00
  1  4   4 10   0 37.62   1.00
  1  5   5 10   0 50.00   1.00
  1  6   6 10   0151.15   1.00
  1  7   7 10   0 25.62   1.00
  1  8   8 10   0335.10   1.00
  1  9   9 10   0147.92   1.00
  1 10  10  0    0  0.00   1.00
  2  1   1 10   0332.12   1.00
  2  3   3 10   0 18.37   1.00
  2  4   4 10   0  9.98   1.00
  2  5   5 10   0 23.88   1.00
  2  6   6 10   0 66.85   1.00
  2  7   7 10   0  9.22   1.00
  2  8   8 10   0177.81   1.00
  2  9   9 10   0155.42   1.00
  2 10  10  0    0  0.00   1.00
```

9	7	10	0	0	0.00	1.00
9	8	8	10	0	102.76	1.00
9	10	10	0	0	0.00	1.00
10	1	1	0	0	0.00	1.00
10	2	2	0	0	0.00	1.00
10	3	3	0	0	0.00	1.00
10	4	4	0	0	0.00	1.00
10	5	5	0	0	0.00	1.00
10	6	6	0	0	0.00	1.00
10	7	7	0	0	0.00	1.00
10	8	8	0	0	0.00	1.00
10	9	9	0	0	0.00	1.00

3	2	2	10	0	19.22	1.00
3	4	4	10	0	10.23	1.00
3	5	10	0	0	0.00	1.00
3	6	10	0	0	0.00	1.00
3	7	7	10	0	9.69	1.00
3	8	8	10	0	29.38	1.00
3	9	9	10	0	7.57	1.00
3	10	10	0	0	0.00	1.00
4	1	1	10	0	36.89	1.00
4	2	2	10	0	11.65	1.00
4	3	3	10	0	10.42	1.00
4	5	10	0	0	0.00	1.00
4	6	10	0	0	0.00	1.00
4	7	10	0	0	0.00	1.00
4	8	8	10	0	25.43	1.00
4	9	9	10	0	4.50	1.00
4	10	10	0	0	0.00	1.00
5	1	1	10	0	62.16	1.00
5	2	2	10	0	27.96	1.00
5	3	10	0	0	0.00	1.00
5	4	10	0	0	0.00	1.00
5	6	6	10	0	110.04	1.00
5	7	7	10	0	10.45	1.00
5	8	8	10	0	15.99	1.00
5	9	9	10	0	11.84	1.00
5	10	10	0	0	0.00	1.00
6	1	1	10	0	162.32	1.00
6	2	2	10	0	70.40	1.00
6	3	3	10	0	0.00	1.00
6	4	10	0	0	0.00	1.00
6	5	5	10	0	103.96	1.00
5	7	7	10	0	23.56	1.00
6	8	8	10	0	63.02	1.00
6	9	9	10	0	25.49	1.00
6	10	10	0	0	0.00	1.00
7	1	1	10	0	24.42	1.00
7	2	2	10	0	9.18	1.00
7	3	3	10	0	9.85	1.00
7	4	10	0	0	0.00	1.00
7	5	5	10	0	6.62	1.00
7	6	6	10	0	26.35	1.00
7	8	8	10	0	10.36	1.00
7	9	9	10	0	2.44	1.00
7	10	10	0	0	0.00	1.00
8	1	1	10	0	349.38	1.00
8	2	2	10	0	190.32	1.00
8	3	3	10	0	36.16	1.00
8	4	4	10	0	22.80	1.00
8	5	5	10	0	12.48	1.00
8	6	6	10	0	67.51	1.00
8	7	7	10	0	11.94	1.00
8	9	9	10	0	107.92	1.00
8	10	10	0	0	0.00	1.00
9	1	1	10	0	92.28	1.00
9	2	2	10	0	155.96	1.00
9	3	3	10	0	6.43	1.00
9	4	4	10	0	3.80	1.00
9	5	5	10	0	10.10	1.00
9	6	6	10	0	25.27	1.00

TELEPHONE NETWORK ANALYTICAL STUDY

USER SPECIFIED OPTIONS

ROUTING STRATEGY	: SOC
LIST OF PATHS REQUIRED	: YES
DETAILS OF PATH RELIABILITY AND PATH CARRIED LOAD REQUIRED	: YES
MAXIMUM ALLOWABLE BLOCKING PROBABILITY(BPMAX)	: 0.020

NETWORK DESCRIPTION

NUMBER OF SWITCHES IN NETWORK : 10

NUMBER OF TRUNK GROUPS IN NETWORK: 40

RATE : 1.00

LINK NUMBER	SWITCH PAIR		EXISTING NUMBER OF TRUNKS
1	1	2	546
2	1	3	135
3	1	4	56
4	1	5	91
5	1	6	265
6	1	7	39
7	1	8	602
8	1	9	205
9	1	10	616
10	2	3	35
11	2	4	17
12	2	5	45
13	2	6	131
14	2	7	14
15	2	8	313
16	2	9	292
17	2	10	208
18	3	4	22
19	3	7	22
20	3	8	63
21	3	9	13
22	3	10	90
23	4	8	47
24	4	9	7
25	4	10	64
26	5	6	187
27	5	7	15
28	5	8	21
29	5	9	14
30	5	10	154
31	6	7	46
32	6	8	98
33	6	9	45
34	6	10	352
35	7	8	16
36	7	9	2
37	7	10	50
38	8	9	186
39	8	10	208

SWITCH-TO-SWITCH TRAFFIC

FROM	TO	TRAFFIC IN CCS
1	2	10833.48
1	3	2623.32
1	4	1354.32
1	5	1800.00
1	6	5441.40
1	7	922.32
1	8	12063.60
1	9	5325.12
1	10	0.00
2	1	11956.32
2	3	661.32
2	4	359.28
2	5	859.68
2	6	2406.60
2	7	331.92
2	8	6401.16
2	9	5595.12
2	10	0.00
3	1	2847.60
3	2	717.12
3	4	368.28
3	5	0.00
3	6	0.00
3	7	348.84
3	8	1058.04
3	9	272.52
3	10	0.00
4	1	1328.04
4	2	419.40
4	3	375.12
4	5	0.00
4	6	0.00
4	7	0.00
4	8	915.48
4	9	162.00
4	10	0.00
5	1	2237.76
5	2	1006.56
5	3	0.00
5	4	0.00
5	6	3961.44
5	7	376.20
5	8	575.64
5	9	426.24
5	10	0.00
6	1	5843.52
6	2	2534.40
6	3	0.00
6	4	0.00
6	5	3742.56
6	7	848.16
6	8	2268.72
6	9	917.64
6	10	0.00
7	1	879.12
7	2	330.48

ROUTING TABLE

CHOICE 1	CHOICE 2	CHOICE 3	WEIGHT
2	10	0	1.0
3	10	0	1.0
4	10	0	1.0
5	10	0	1.0
6	10	0	1.0
7	10	0	1.0
8	10	0	1.0
9	10	0	1.0
10	0	0	1.0
1	10	0	1.0
3	10	0	1.0
4	10	0	1.0
5	10	0	1.0
6	10	0	1.0
7	10	0	1.0
8	10	0	1.0
9	10	0	1.0
10	0	0	1.0
1	10	0	1.0
2	10	0	1.0
4	10	0	1.0
5	10	0	1.0
6	10	0	1.0
7	10	0	1.0
8	10	0	1.0
9	10	0	1.0
10	0	0	1.0
1	10	0	1.0
2	10	0	1.0
3	10	0	1.0
4	10	0	1.0
5	10	0	1.0
7	10	0	1.0
8	10	0	1.0
9	10	0	1.0
10	0	0	1.0
1	10	0	1.0
2	10	0	1.0

7	3	354.60	3	10	0	1.
7	4	0.00	10	0	0	1.
7	5	238.32	5	10	0	1.
7	6	948.60	6	10	0	1.
7	8	372.96	8	10	0	1.
7	9	87.84	9	10	0	1.
7	10	0.00	10	0	0	1.
8	1	12577.68	1	10	0	1.
8	2	6851.52	2	10	0	1.
8	3	1301.76	3	10	0	1.
8	4	820.80	4	10	0	1.
8	5	449.28	5	10	0	1.
8	6	2430.36	6	10	0	1.
8	7	429.84	7	10	0	1.
8	9	3885.12	9	10	0	1.
8	10	0.00	10	0	0	1.
9	1	3322.08	1	10	0	1.
9	2	5614.56	2	10	0	1.
9	3	231.48	3	10	0	1.
9	4	136.80	4	10	0	1.
9	5	363.60	5	10	0	1.
9	6	909.72	6	10	0	1.
9	7	0.00	10	0	0	1.
9	8	3699.36	8	10	0	1.
9	10	0.00	10	0	0	1.
10	1	0.00	1	0	0	1.
10	2	0.00	2	0	0	1.
10	3	0.00	3	0	0	1.
10	4	0.00	4	0	0	1.
10	5	0.00	5	0	0	1.
10	6	0.00	6	0	0	1.
10	7	0.00	7	0	0	1.
10	8	0.00	8	0	0	1.
10	9	0.00	9	0	0	1.

PATHS DEFINED BY ORIGINAL ROUTING TABLE

```
FROM  1 TO  2
      1  2
      1 10  2

FROM  1 TO  3
      1  3
      1 10  3

FROM  1 TO  4
      1  4
      1 10  4

FROM  1 TO  5
      1  5
      1 10  5

FROM  1 TO  6
      1  6
      1 10  6

FROM  1 TO  7
      1  7
      1 10  7

FROM  1 TO  8
      1  8
      1 10  8

FROM  1 TO  9
      1  9
      1 10  9

FROM  1 TO 10
      1 10

FROM  2 TO  1
      2  1
      2 10  1

FROM  2 TO  3
      2  3
      2 10  3

FROM  2 TO  4
      2  4
```

```
      2  10   4

FROM   2  TO   5
       2   5
       2  10   5

FROM   2  TO   6
       2   6
       2  10   6

FROM   2  TO   7
       2   7
       2  10   7

FROM   2  TO   8
       2   8
       2  10   8

FROM   2  TO   9
       2   9
       2  10   9

FROM   2  TO  10
       2  10

FROM   3  TO   1
       3   1
       3  10   1

FROM   3  TO   2
       3   2
       3  10   2

FROM   3  TO   4
       3   4
       3  10   4

FROM   3  TO   5
       3  10   5

FROM   3  TO   6
       3  10   6

FROM   3  TO   7
       3   7
       3  10   7
```

```
FROM   3 TO   8
       3   8
       3 10   8

FROM   3 TO   9
       3   9
       3 10   9

FROM   3 TO  10
       3 10

FROM   4 TO   1
       4   1
       4 10   1

FROM   4 TO   2
       4   2
       4 10   2

FROM   4 TO   3
       4   3
       4 10   3

FROM   4 TO   5
       4 10   5

FROM   4 TO   6
       4 10   6

FROM   4 TO   7
       4 10   7

FROM   4 TO   8
       4   8
       4 10   8

FROM   4 TO   9
       4   9
       4 10   9

FROM   4 TO  10
       4 10

FROM   5 TO   1
       5   1
       5 10   1
```

```
FROM   5 TO   2
       5   2
       5 10   2

FROM   5 TO   3
       5 10   3

FROM   5 TO   4
       5 10   4

FROM   5 TO   6
       5   6
       5 10   6

FROM   5 TO   7
       5   7
       5 10   7

FROM   5 TO   8
       5   8
       5 10   8

FROM   5 TO   9
       5   9
       5 10   9

FROM   5 TO  10
       5 10

FROM   6 TO   1
       6   1
       6 10   1

FROM   6 TO   2
       6   2
       6 10   2

FROM   6 TO   3
       6   3
       6 10   3

FROM   6 TO   4
       6 10   4

FROM   6 TO   5
       6   5
       6 10   5
```

```
FROM   6 TO   7
       6   7
       6 10  7

FROM   6 TO   8
       6   8
       6 10  8

FROM   6 TO   9
       6   9
       6 10  9

FROM   6 TO  10
       6 10

FROM   7 TO   1
       7   1
       7 10  1

FROM   7 TO   2
       7   2
       7 10  2

FROM   7 TO   3
       7   3
       7 10  3

FROM   7 TO   4
       7 10  4

FROM   7 TO   5
       7   5
       7 10  5

FROM   7 TO   6
       7   6
       7 10  6

FROM   7 TO   8
       7   8
       7 10  8

FROM   7 TO   9
       7   9
       7 10  9

FROM   7 TO  10
       7 10
```

```
FROM   8 TO   1
       8   1
       8  10   1

FROM   8 TO   2
       8   2
       8  10   2

FROM   8 TO   3
       8   3
       8  10   3

FROM   8 TO   4
       8   4
       8  10   4

FROM   8 TO   5
       8   5
       8  10   5

FROM   8 TO   6
       8   6
       8  10   6

FROM   8 TO   7
       8   7
       8  10   7

FROM   8 TO   9
       8   9
       8  10   9

FROM   8 TO  10
       8  10

FROM   9 TO   1
       9   1
       9  10   1

FROM   9 TO   2
       9   2
       9  10   2

FROM   9 TO   3
       9   3
       9  10   3

FROM   9 TO   4
```

```
      9   4
      9  10   4

FROM   9 TO   5
      9   5
      9  10   5

FROM   9 TO   6
      9   6
      9  10   6

FROM   9 TO   7
      9  10   7

FROM   9 TO   8
      9   8
      9  10   8

FROM   9 TO  10
      9  10

FROM  10 TO   1
     10   1

FROM  10 TO   2
     10   2

FROM  10 TO   3
     10   3

FROM  10 TO   4
     10   4

FROM  10 TO   5
     10   5

FROM  10 TO   6
     10   6

FROM  10 TO   7
     10   7

FROM  10 TO   8
     10   8

FROM  10 TO   9
```

PROBABILITY(PATH USED) AND PATH CARRIED LOAD(CCS)

FROM 1 TO 2		
pR(PATH 1)=0.853811	LOAD CARRIED BY PATH 1=	9249.74
pR(PATH 2)=0.136203	LOAD CARRIED BY PATH 2=	1475.56
FROM 1 TO 3		
pR(PATH 1)=0.855847	LOAD CARRIED BY PATH 1=	2245.16
pR(PATH 2)=0.144153	LOAD CARRIED BY PATH 2=	378.16
FROM 1 TO 4		
pR(PATH 1)=0.720864	LOAD CARRIED BY PATH 1=	976.28
pR(PATH 2)=0.279131	LOAD CARRIED BY PATH 2=	378.03
FROM 1 TO 5		
pR(PATH 1)=0.783060	LOAD CARRIED BY PATH 1=	1409.51
pR(PATH 2)=0.216940	LOAD CARRIED BY PATH 2=	390.49
FROM 1 TO 6		
pR(PATH 1)=0.830981	LOAD CARRIED BY PATH 1=	4521.70
pR(PATH 2)=0.169019	LOAD CARRIED BY PATH 2=	919.70
FROM 1 TO 7		
pR(PATH 1)=0.733135	LOAD CARRIED BY PATH 1=	676.18
pR(PATH 2)=0.261062	LOAD CARRIED BY PATH 2=	240.78
FROM 1 TO 8		
pR(PATH 1)=0.870353	LOAD CARRIED BY PATH 1=	10499.59
pR(PATH 2)=0.109779	LOAD CARRIED BY PATH 2=	1324.33
FROM 1 TO 9		
pR(PATH 1)=0.834620	LOAD CARRIED BY PATH 1=	4444.45
pR(PATH 2)=0.165380	LOAD CARRIED BY PATH 2=	880.67
FROM 1 TO 10		
pR(PATH 1)=1.000000	LOAD CARRIED BY PATH 1=	0.00
FROM 2 TO 1		
pR(PATH 1)=0.853811	LOAD CARRIED BY PATH 1=	10208.44
pR(PATH 2)=0.136203	LOAD CARRIED BY PATH 2=	1628.49
FROM 2 TO 3		
pR(PATH 1)=0.825819	LOAD CARRIED BY PATH 1=	546.13
pR(PATH 2)=0.162284	LOAD CARRIED BY PATH 2=	107.32
FROM 2 TO 4		
pR(PATH 1)=0.701763	LOAD CARRIED BY PATH 1=	252.13

```
                pR(PATH 2)=0.277861     LOAD CARRIED BY PATH 2=      99.83

FROM    2    TO    5
                pR(PATH 1)=0.805813     LOAD CARRIED BY PATH 1=     692.74
                pR(PATH 2)=0.180922     LOAD CARRIED BY PATH 2=     155.54

FROM    2    TO    6
                pR(PATH 1)=0.904488     LOAD CARRIED BY PATH 1=    2176.74
                pR(PATH 2)=0.088988     LOAD CARRIED BY PATH 2=     214.16

FROM    2    TO    7
                pR(PATH 1)=0.673812     LOAD CARRIED BY PATH 1=     223.65
                pR(PATH 2)=0.297298     LOAD CARRIED BY PATH 2=      98.68

FROM    2    TO    8
                pR(PATH 1)=0.837337     LOAD CARRIED BY PATH 1=    5359.93
                pR(PATH 2)=0.128327     LOAD CARRIED BY PATH 2=     821.44

FROM    2    TO    9
                pR(PATH 1)=0.911214     LOAD CARRIED BY PATH 1=    5098.35
                pR(PATH 2)=0.082722     LOAD CARRIED BY PATH 2=     462.84

FROM    2    TO   10
                pR(PATH 1)=0.931694     LOAD CARRIED BY PATH 1=       0.00

FROM    3    TO    1
                pR(PATH 1)=0.855847     LOAD CARRIED BY PATH 1=    2437.11
                pR(PATH 2)=0.144153     LOAD CARRIED BY PATH 2=     410.49

FROM    3    TO    2
                pR(PATH 1)=0.825819     LOAD CARRIED BY PATH 1=     592.21
                pR(PATH 2)=0.162284     LOAD CARRIED BY PATH 2=     116.38

FROM    3    TO    4
                pR(PATH 1)=0.878726     LOAD CARRIED BY PATH 1=     323.62
                pR(PATH 2)=0.121272     LOAD CARRIED BY PATH 2=      44.66

FROM    3    TO    5
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    3    TO    6
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    3    TO    7
                pR(PATH 1)=0.903297     LOAD CARRIED BY PATH 1=     315.11
                pR(PATH 2)=0.094600     LOAD CARRIED BY PATH 2=      33.00
```

FROM 3 TO 8			
pR(PATH 1)=0.882878	LOAD CARRIED BY PATH 1=	934.12
pR(PATH 2)=0.099174	LOAD CARRIED BY PATH 2=	104.93
FROM 3 TO 9			
pR(PATH 1)=0.771795	LOAD CARRIED BY PATH 1=	210.33
pR(PATH 2)=0.228205	LOAD CARRIED BY PATH 2=	62.19
FROM 3 TO 10			
pR(PATH 1)=1.000000	LOAD CARRIED BY PATH 1=	0.00
FROM 4 TO 1			
pR(PATH 1)=0.720864	LOAD CARRIED BY PATH 1=	957.34
pR(PATH 2)=0.279131	LOAD CARRIED BY PATH 2=	370.70
FROM 4 TO 2			
pR(PATH 1)=0.701763	LOAD CARRIED BY PATH 1=	294.32
pR(PATH 2)=0.277861	LOAD CARRIED BY PATH 2=	116.53
FROM 4 TO 3			
pR(PATH 1)=0.878726	LOAD CARRIED BY PATH 1=	329.63
pR(PATH 2)=0.121272	LOAD CARRIED BY PATH 2=	45.49
FROM 4 TO 5			
pR(PATH 1)=0.999984	LOAD CARRIED BY PATH 1=	0.00
FROM 4 TO 6			
pR(PATH 1)=0.999984	LOAD CARRIED BY PATH 1=	0.00
FROM 4 TO 7			
pR(PATH 1)=0.978238	LOAD CARRIED BY PATH 1=	0.00
FROM 4 TO 8			
pR(PATH 1)=0.877727	LOAD CARRIED BY PATH 1=	803.54
pR(PATH 2)=0.103533	LOAD CARRIED BY PATH 2=	94.78
FROM 4 TO 9			
pR(PATH 1)=0.675172	LOAD CARRIED BY PATH 1=	109.38
pR(PATH 2)=0.324823	LOAD CARRIED BY PATH 2=	52.62
FROM 4 TO 10			
pR(PATH 1)=0.999984	LOAD CARRIED BY PATH 1=	0.00
FROM 5 TO 1			
pR(PATH 1)=0.783060	LOAD CARRIED BY PATH 1=	1752.30
pR(PATH 2)=0.216940	LOAD CARRIED BY PATH 2=	485.46

```
FROM    5     TO     2
        pR(PATH 1)=0.805813      LOAD CARRIED BY PATH 1=     811.10
        pR(PATH 2)=0.180922      LOAD CARRIED BY PATH 2=     182.11

FROM    5     TO     3
        pR(PATH 1)=1.000000      LOAD CARRIED BY PATH 1=       0.00

FROM    5     TO     4
        pR(PATH 1)=0.999984      LOAD CARRIED BY PATH 1=       0.00

FROM    5     TO     6
        pR(PATH 1)=0.850656      LOAD CARRIED BY PATH 1=    3369.82
        pR(PATH 2)=0.149344      LOAD CARRIED BY PATH 2=     591.62

FROM    5     TO     7
        pR(PATH 1)=0.753883      LOAD CARRIED BY PATH 1=     283.61
        pR(PATH 2)=0.240764      LOAD CARRIED BY PATH 2=      90.58

FROM    5     TO     8
        pR(PATH 1)=0.677033      LOAD CARRIED BY PATH 1=     389.73
        pR(PATH 2)=0.273473      LOAD CARRIED BY PATH 2=     157.42

FROM    5     TO     9
        pR(PATH 1)=0.583927      LOAD CARRIED BY PATH 1=     248.89
        pR(PATH 2)=0.416073      LOAD CARRIED BY PATH 2=     177.35

FROM    5     TO    10
        pR(PATH 1)=1.000000      LOAD CARRIED BY PATH 1=       0.00

FROM    6     TO     1
        pR(PATH 1)=0.830981      LOAD CARRIED BY PATH 1=    4855.85
        pR(PATH 2)=0.169019      LOAD CARRIED BY PATH 2=     987.67

FROM    6     TO     2
        pR(PATH 1)=0.904488      LOAD CARRIED BY PATH 1=    2292.33
        pR(PATH 2)=0.088988      LOAD CARRIED BY PATH 2=     225.53

FROM    6     TO     3
        pR(PATH 1)=0.000000      LOAD CARRIED BY PATH 1=       0.00
        pR(PATH 2)=1.000000      LOAD CARRIED BY PATH 2=       0.00

FROM    6     TO     4
        pR(PATH 1)=0.999984      LOAD CARRIED BY PATH 1=       0.00

FROM    6     TO     5
        pR(PATH 1)=0.850656      LOAD CARRIED BY PATH 1=    3183.63
        pR(PATH 2)=0.149344      LOAD CARRIED BY PATH 2=     558.93
```

FROM 6 TO 7			
PR(PATH 1)=0.843801	LOAD CARRIED BY PATH 1=	715.68	
PR(PATH 2)=0.152802	LOAD CARRIED BY PATH 2=	129.60	
FROM 6 TO 8			
PR(PATH 1)=0.731567	LOAD CARRIED BY PATH 1=	1659.72	
PR(PATH 2)=0.227296	LOAD CARRIED BY PATH 2=	515.67	
FROM 6 TO 9			
PR(PATH 1)=0.818950	LOAD CARRIED BY PATH 1=	751.50	
PR(PATH 2)=0.181050	LOAD CARRIED BY PATH 2=	166.14	
FROM 6 TO 10			
PR(PATH 1)=1.000000	LOAD CARRIED BY PATH 1=	0.00	
FROM 7 TO 1			
PR(PATH 1)=0.733135	LOAD CARRIED BY PATH 1=	644.51	
PR(PATH 2)=0.261062	LOAD CARRIED BY PATH 2=	229.50	
FROM 7 TO 2			
PR(PATH 1)=0.673812	LOAD CARRIED BY PATH 1=	222.68	
PR(PATH 2)=0.297298	LOAD CARRIED BY PATH 2=	98.25	
FROM 7 TO 3			
PR(PATH 1)=0.903297	LOAD CARRIED BY PATH 1=	320.31	
PR(PATH 2)=0.094600	LOAD CARRIED BY PATH 2=	33.55	
FROM 7 TO 4			
PR(PATH 1)=0.978238	LOAD CARRIED BY PATH 1=	0.00	
FROM 7 TO 5			
PR(PATH 1)=0.753883	LOAD CARRIED BY PATH 1=	179.67	
PR(PATH 2)=0.240764	LOAD CARRIED BY PATH 2=	57.38	
FROM 7 TO 6			
PR(PATH 1)=0.843801	LOAD CARRIED BY PATH 1=	800.43	
PR(PATH 2)=0.152802	LOAD CARRIED BY PATH 2=	144.95	
FROM 7 TO 8			
PR(PATH 1)=0.649797	LOAD CARRIED BY PATH 1=	242.35	
PR(PATH 2)=0.290086	LOAD CARRIED BY PATH 2=	108.19	
FROM 7 TO 9			
PR(PATH 1)=0.536093	LOAD CARRIED BY PATH 1=	47.09	
PR(PATH 2)=0.453819	LOAD CARRIED BY PATH 2=	39.86	
FROM 7 TO 10			
PR(PATH 1)=0.978253	LOAD CARRIED BY PATH 1=	0.00	

```
FROM     8     TO     1
         PR(PATH 1)=0.870353      LOAD CARRIED BY PATH 1=   10947.02
         PR(PATH 2)=0.109779      LOAD CARRIED BY PATH 2=    1380.76

FROM     8     TO     2
         PR(PATH 1)=0.837337      LOAD CARRIED BY PATH 1=    5737.03
         PR(PATH 2)=0.128327      LOAD CARRIED BY PATH 2=     879.24

FROM     8     TO     3
         PR(PATH 1)=0.882878      LOAD CARRIED BY PATH 1=    1149.29
         PR(PATH 2)=0.099174      LOAD CARRIED BY PATH 2=     129.10

FROM     8     TO     4
         PR(PATH 1)=0.877727      LOAD CARRIED BY PATH 1=     720.44
         PR(PATH 2)=0.103533      LOAD CARRIED BY PATH 2=      84.98

FROM     8     TO     5
         PR(PATH 1)=0.677033      LOAD CARRIED BY PATH 1=     304.18
         PR(PATH 2)=0.273473      LOAD CARRIED BY PATH 2=     122.87

FROM     8     TO     6
         PR(PATH 1)=0.731567      LOAD CARRIED BY PATH 1=    1777.97
         PR(PATH 2)=0.227296      LOAD CARRIED BY PATH 2=     552.41

FROM     8     TO     7
         PR(PATH 1)=0.649797      LOAD CARRIED BY PATH 1=     279.31
         PR(PATH 2)=0.290086      LOAD CARRIED BY PATH 2=     124.69

FROM     8     TO     9
         PR(PATH 1)=0.858154      LOAD CARRIED BY PATH 1=    3334.03
         PR(PATH 2)=0.120108      LOAD CARRIED BY PATH 2=     466.63

FROM     8     TO    10
         PR(PATH 1)=0.846752      LOAD CARRIED BY PATH 1=       0.00

FROM     9     TO     1
         PR(PATH 1)=0.834620      LOAD CARRIED BY PATH 1=    2772.67
         PR(PATH 2)=0.165380      LOAD CARRIED BY PATH 2=     549.41

FROM     9     TO     2
         PR(PATH 1)=0.911214      LOAD CARRIED BY PATH 1=    5116.06
         PR(PATH 2)=0.082722      LOAD CARRIED BY PATH 2=     464.45

FROM     9     TO     3
         PR(PATH 1)=0.771795      LOAD CARRIED BY PATH 1=     178.66
         PR(PATH 2)=0.228205      LOAD CARRIED BY PATH 2=      52.82

FROM     9     TO     4
```

```
                pR(PATH 1)=0.675172     LOAD CARRIED BY PATH 1=      92.36
                pR(PATH 2)=0.324823     LOAD CARRIED BY PATH 2=      44.44

FROM     9     TO     5
                pR(PATH 1)=0.583927     LOAD CARRIED BY PATH 1=     212.32
                pR(PATH 2)=0.416073     LOAD CARRIED BY PATH 2=     151.28

FROM     9     TO     6
                pR(PATH 1)=0.818950     LOAD CARRIED BY PATH 1=     745.02
                pR(PATH 2)=0.181050     LOAD CARRIED BY PATH 2=     164.70

FROM     9     TO     7
                pR(PATH 1)=0.978253     LOAD CARRIED BY PATH 1=       0.00

FROM     9     TO     8
                pR(PATH 1)=0.858154     LOAD CARRIED BY PATH 1=    3174.62
                pR(PATH 2)=0.120108     LOAD CARRIED BY PATH 2=     444.32

FROM     9     TO    10
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     1
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     2
                pR(PATH 1)=0.931694     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     3
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     4
                pR(PATH 1)=0.999984     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     5
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     6
                pR(PATH 1)=1.000000     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     7
                pR(PATH 1)=0.978253     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     8
                pR(PATH 1)=0.846752     LOAD CARRIED BY PATH 1=       0.00

FROM    10     TO     9
```

pR(PATH 1)=1.000000 LOAD CARRIED BY PATH 1= 0.00

ACTUAL TRUNK GROUP PERFORMANCE

LINK NUMBER	OFFERED LOAD(CCS)	TRUNK GROUP ACTUAL GOS	TRUNK GROUP RELIABILITY
1	22789.80	0.146189	0.853811
2	5470.92	0.144153	0.855847
3	2682.36	0.279136	0.720864
4	4037.76	0.216940	0.783060
5	11284.92	0.169019	0.830981
6	1801.44	0.266865	0.733135
7	24641.28	0.129647	0.870353
8	8647.20	0.165380	0.834620
9	12030.19	0.000000	1.000000
10	1378.44	0.174181	0.825819
11	778.68	0.298237	0.701763
12	1866.24	0.194187	0.805813
13	4941.00	0.095512	0.904488
14	662.40	0.326188	0.673812
15	13252.68	0.162663	0.837337
16	11209.68	0.088786	0.911214
17	7670.27	0.068306	0.931694
18	743.40	0.121274	0.878726
19	703.44	0.096703	0.903297
20	2359.80	0.117122	0.882878
21	504.00	0.228205	0.771795
22	1518.09	0.000000	1.000000
23	1736.28	0.122273	0.877727
24	298.80	0.324828	0.675172
25	1332.09	0.000016	0.999984
26	7704.00	0.149344	0.850656
27	614.52	0.246117	0.753883
28	1024.92	0.322967	0.677033
29	789.84	0.416073	0.583927
30	3121.01	0.000000	1.000000
31	1796.76	0.156199	0.843801
32	4699.08	0.268433	0.731567
33	1827.36	0.181050	0.818950
34	5171.07	0.000000	1.000000
35	802.80	0.350203	0.649797
36	87.84	0.463907	0.536093
37	1460.78	0.021747	0.978253
38	7584.48	0.141846	0.858154
39	8635.08	0.153248	0.846752
40	4179.73	0.000000	1.000000

NETWORK PERFORMANCE

SOURCE	DESTINATION	BLOCKING PROBABILITY	WEIGHTED BLOCKING PROBABILITY
1	2	9.986E-03	9.986E-03
1	3	8.395E-12	8.395E-12
1	4	4.364E-06	4.364E-06
1	5	4.442E-12	4.442E-12
1	6	1.671E-16	1.671E-16
1	7	5.804E-03	5.804E-03
1	8	1.987E-02	1.987E-02
1	9	8.198E-15	8.198E-15
1	10	0.000E+00	0.000E+00
2	1	9.986E-03	9.986E-03
2	3	1.190E-02	1.190E-02
2	4	2.038E-02	2.038E-02*
2	5	1.326E-02	1.326E-02
2	6	6.524E-03	6.524E-03
2	7	2.889E-02	2.889E-02*
2	8	3.434E-02	3.434E-02*
2	9	6.065E-03	6.065E-03
2	10	0.000E+00	0.000E+00
3	1	8.395E-12	8.395E-12
3	2	1.190E-02	1.190E-02
3	4	1.896E-06	1.896E-06
3	5	0.000E+00	0.000E+00
3	6	0.000E+00	0.000E+00
3	7	2.103E-03	2.103E-03
3	8	1.795E-02	1.795E-02
3	9	1.330E-11	1.330E-11
3	10	0.000E+00	0.000E+00
4	1	4.364E-06	4.364E-06
4	2	2.038E-02	2.038E-02*
4	3	1.896E-06	1.896E-06
4	5	0.000E+00	0.000E+00
4	6	0.000E+00	0.000E+00
4	7	0.000E+00	0.000E+00
4	8	1.874E-02	1.874E-02
4	9	5.079E-06	5.079E-06
4	10	0.000E+00	0.000E+00
5	1	4.442E-12	4.442E-12
5	2	1.326E-02	1.326E-02
5	3	0.000E+00	0.000E+00
5	4	0.000E+00	0.000E+00
5	6	3.058E-12	3.058E-12
5	7	5.352E-03	5.352E-03
5	8	4.949E-02	4.949E-02*
5	9	8.539E-12	8.539E-12
5	10	0.000E+00	0.000E+00
6	1	1.556E-16	1.556E-16
6	2	6.524E-03	6.524E-03
6	3	0.000E+00	0.000E+00
6	4	0.000E+00	0.000E+00
6	5	3.058E-12	3.058E-12
6	7	3.397E-03	3.397E-03
6	8	4.114E-02	4.114E-02*

6	9	8.858E-15	8.858E-15
6	10	0.000E+00	0.000E+00
7	1	5.804E-03	5.804E-03
7	2	2.889E-02	2.889E-02*
7	3	2.103E-03	2.103E-03
7	4	0.000E+00	0.000E+00
7	5	5.352E-03	5.352E-03
7	6	3.397E-03	3.397E-03
7	8	6.012E-02	6.012E-02*
7	9	1.009E-02	1.009E-02
7	10	0.000E+00	0.000E+00
8	1	1.987E-02	1.987E-02
8	2	3.434E-02	3.434E-02*
8	3	1.795E-02	1.795E-02
8	4	1.874E-02	1.874E-02
8	5	4.949E-02	4.949E-02*
8	6	4.114E-02	4.114E-02*
8	7	6.012E-02	6.012E-02*
8	9	2.174E-02	2.174E-02*
8	10	0.000E+00	0.000E+00
9	1	8.042E-15	8.042E-15
9	2	6.065E-03	6.065E-03
9	3	1.330E-11	1.330E-11
9	4	5.079E-06	5.079E-06
9	5	8.539E-12	8.539E-12
9	6	8.873E-15	8.873E-15
9	7	0.000E+00	0.000E+00
9	8	2.174E-02	2.174E-02*
9	10	0.000E+00	0.000E+00
10	1	0.000E+00	0.000E+00
10	2	0.000E+00	0.000E+00
10	3	0.000E+00	0.000E+00
10	4	0.000E+00	0.000E+00
10	5	0.000E+00	0.000E+00
10	6	0.000E+00	0.000E+00
10	7	0.000E+00	0.000E+00
10	8	0.000E+00	0.000E+00
10	9	0.000E+00	0.000E+00

NODE GRADE OF SERVICE

NODE NUMBER	ORIGINATING LOAD(CCS)	NODE GRADE OF SERVICE	NODE RELIABILITY
1	40363.56	0.008751	0.991249
2	28571.40	0.014875	0.985125
3	5612.40	0.005035	0.994965
4	3200.04	0.008034	0.991966
5	8583.84	0.005109	0.994891
6	16155.00	0.006979	0.993021
7	3211.92	0.013450	0.986550
8	28746.36	0.026313	0.973687
9	14277.60	0.008017	0.991983
10	0.00	-	-

NETWORK STATISTICS

LARGEST WEIGHTED NNGOS(= 0.060) IS FROM 7 TO 8

MAXIMUM ALLOWABLE BLOCKING PROBABILITY(BPMAX) 0.020

NUMBER OF WEIGHTED NNGOS EXCEEDING BPMAX IN THE ORIGINAL ROUTING TABLE = 14

NETWORK GRADE OF SERVICE IS 0.01279

WEIGHTED NETWORK GRADE OF SERVICE IS 0.01279

FIGURE 13 (FIN)

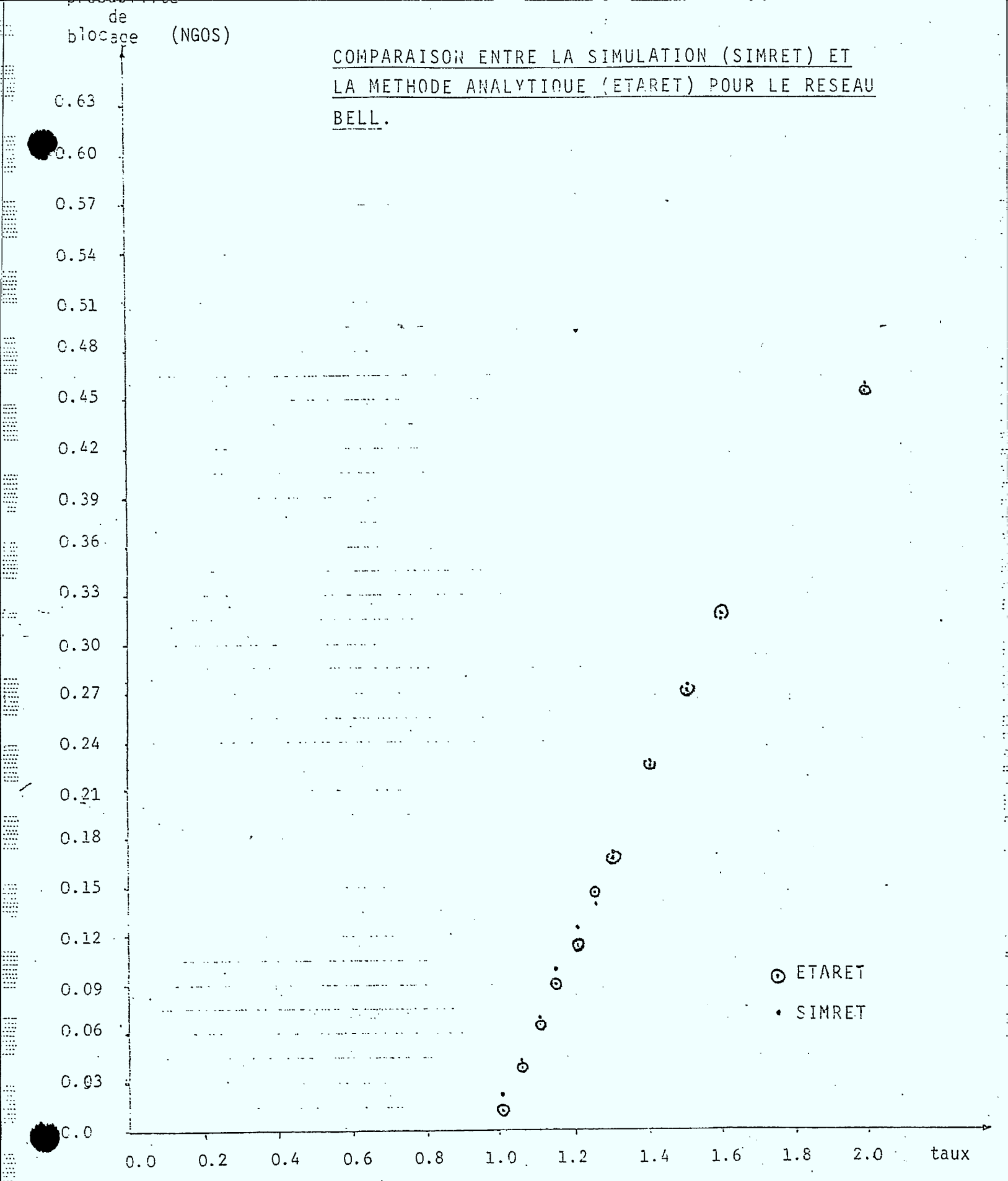
5. CONCLUSION

Parallèlement au développement du package ETARET, un autre package, SIMRET, a été conçu par M. Mario Lavoie. Son objectif, tout comme ETARET, est de réaliser, en utilisant la simulation, l'analyse des performances d'un réseau téléphonique quelconque.

Les personnes employant les techniques de simulation et de méthode analytique s'accordent pour dire que la simulation peut être employée pour valider le modèle mathématique. Nous avons donc comparé, pour deux réseaux, les résultats de ces deux packages. Le premier réseau est un réseau fictif qui a été dérivé des données fournies par Bell Canada. Les différentes matrices décrivant le réseau sont données dans la section de l'exemple d'utilisation. Le deuxième réseau correspond à une modification du réseau AUTOVON. Dans ce réseau, les arbres d'acheminement pour une paire de noeuds ORIGINE-DESTINATION sont plus complexes.

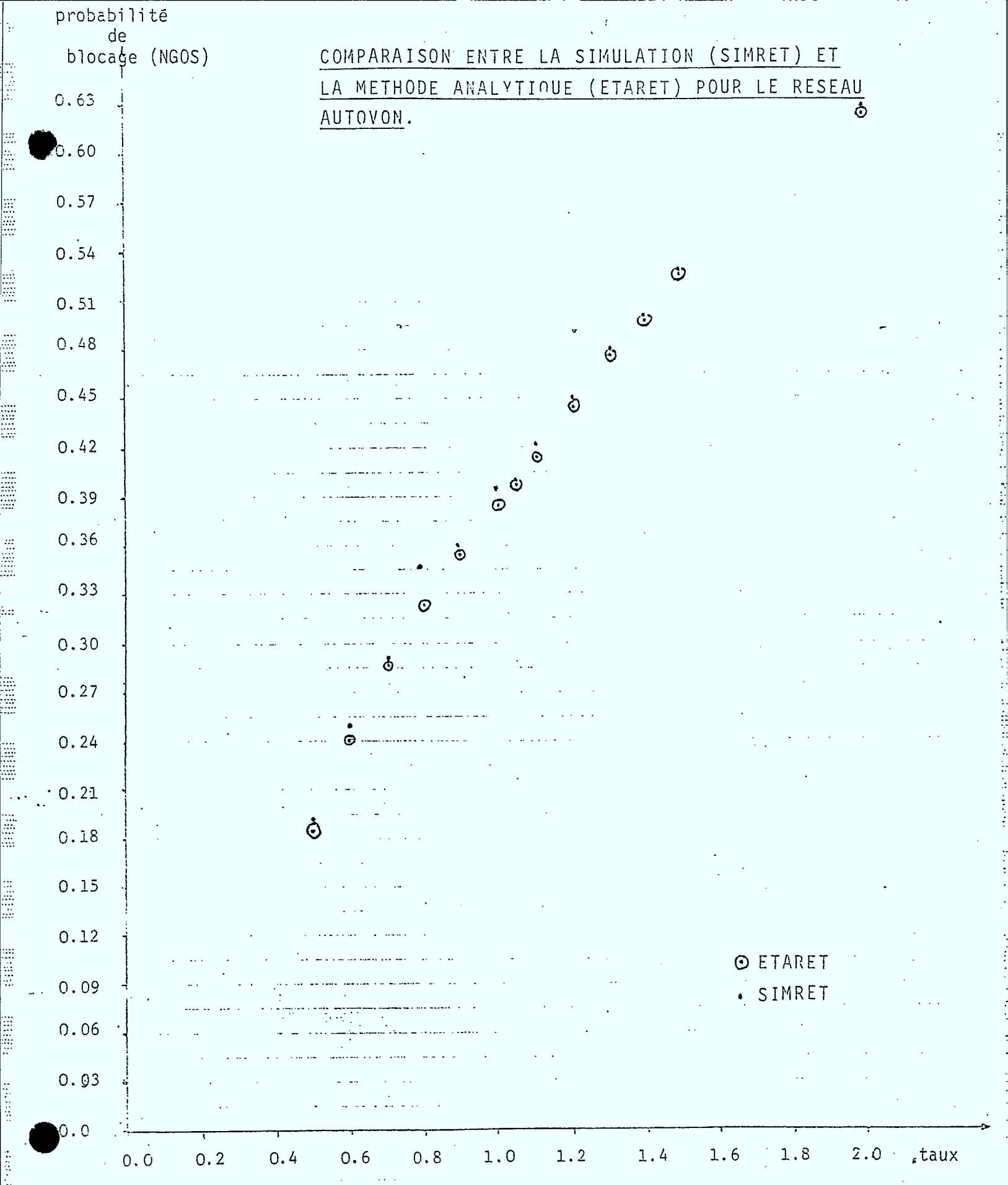
Nous avons basé notre comparaison sur le GRADE OF SERVICE du réseau. Pour chaque réseau, plusieurs conditions ont été vérifiées. De cette comparaison, comme nous l'indique les graphiques des figures 14 et 15, il en ressort que les deux packages nous donnent des résultats quasis identiques. Par conséquent, on peut dire que SIMRET valide jusqu'à un certain point ETARET.

Pour avoir une plus grande certitude quant à la validité d'ETARET, il faudrait le vérifier avec un réseau existant dont les performances sont déjà connues.



COMPARAISON ENTRE LA SIMULATION (SIMRET) ET LA METHODE ANALYTIQUE (ETARET) POUR LE RESEAU BELL.

FIGURE 14.



COMPARAISON ENTRE LA SIMULATION (SIMRET) ET LA METHODE ANALYTIQUE (ETARET) POUR LE RESEAU AUTOVON.

FIGURE 15.

6. BIBLIOGRAPHIE

LIVRE

Lin, P.M., B.J. Leon, H. Thapar, C. Stewart, STARTUP-a statistical analysis and routing table updating program for european AUTOVON, School of electrical engineering, Purdue University, Ind., Tech. Rept. TR-EE 76-22, August 1976, 315 pages.

ARTICLE

Lin, P.M., Benjamin J. Leon, Chris R. Stewart,

"Analysis of circuit-switched networks employing originating - Office control with spill-forward".

IEEE transactions on communications, Vol. COM-26, No 6, June 1978, pp. 754-765.

ANNEX III

A SIMULATOR FOR TELEPHONE NETWORKS

(UN SIMULATEUR POUR LES RESEAUX COMMUTES TELEPHONIQUES)

ABSTRACT

In this document, a simulator program called SIMRET is described which allows modeling of telephone networks with SOC (successive office control) routing strategies. Data structures, subprograms, input and output formats used are given in some detail. To illustrate use of this package, an example is given. Finally, a comparison is made of the results obtained with SIMRET and those obtained by an analytical method (ETARET).

RESUME

Ce document présente un programme de simulation appelé SIMRET qui permet de modéliser un réseau téléphonique employant la stratégie d'acheminement progressif (SOC). On y décrit les structures de données ainsi que chaque sous-programme, les entrées et les sorties. On montre aussi un exemple d'utilisation. On conclut par une comparaison entre le programme de simulation et un programme qui applique une méthode analytique au même réseau (ETARET).

TABLE DES MATIERES

TABLE DES MATIERES (suite)

Page

1. Introduction

Le téléphone est dans tous les foyers. Cependant, peu de personnes s'imaginent l'étendue des réseaux qui relient ces appareils entre eux. Saviez-vous par exemple que le réseau américain totalise un investissement supérieur à 75 milliards de dollars et qu'il relie environ 140 millions d'abonnés !

Cela démontre qu'on doit prévoir longtemps d'avance car chaque modification à une partie du réseau amène des dépenses se chiffrant à plusieurs milliers de dollars. L'impact d'un changement doit donc être calculé minutieusement.

Pour ce faire, deux méthodes sont offertes. Il y a les modèles analytiques. Ceux-ci sont des modèles où l'on essaie de paramétriser le réseau de façon à obtenir une série d'équations mathématiques qui résument le réseau. La simulation est la seconde méthode. C'est une technique plus coûteuse mais plus précise que la méthode analytique. Elle modélise un réseau et fait passer l'information dans le modèle de manière à représenter une situation réelle possible. On doit faire une simulation pour chaque situation différente.

Le but de ce travail était de concevoir un programme de simulation permettant de modéliser un réseau téléphonique quelconque. Le langage utilisé est SIMSCRIPT II.5, choisi pour sa puissance et ses possibilités.

2. L'environnement

Le langage SIMSCRIPT II.5, version 8H a été employé pour la programmation. Le tout s'est fait dans un environnement en lot (batch) avec un ordinateur IBM 370/158.

SIMSCRIPT est un langage très puissant qui permet de faire de la simulation par évènements. Il possède des facilités étonnantes pour le traitement des listes et la création de structures de données. Finalement, il offre de nombreuses fonctions pour cueillir les statistiques.

3. Les structures de données

Il est difficile de décrire les structures de données sans tout d'abord définir quelques termes utilisés par SIMSCRIPT. Ce sont les termes: entité, attribut, activité et événement.

Le terme entité est employé pour désigner un objet d'intérêt dans le système. Le terme attribut dénote une propriété de l'entité. Naturellement une entité peut posséder plusieurs attributs. Tout processus qui provoque un changement dans le système est appelé activité. Une activité est cernée par le début et la fin d'un évènement.

Avec SIMSCRIPT, on décrit un système avec des entités et des attributs. Pour plus d'efficacité, le langage fait une distinction entre les entités temporaires et les entités permanentes. Le premier type représente les entités pouvant être créées ou détruites durant l'exécution de la simulation tandis que le second décrit celles qui existent du début à la fin de la simulation. Par exemple dans notre programme (SIMRET) les appels sont considérés comme une entité temporaire; le nombre d'appels dans le système peut varier. Les liens par contre, sont des entités permanentes; leur nombre (NB.-LINKS) est lu au début du programme et demeure inchangé tout au long de la simulation.

Les structures de données sont d'une importance capitale car elles auront un effet concluant sur les performances du programme.

C'est pourquoi il faut leur porter une attention toute particulière.

Le programme (SIMRET) simule un réseau téléphonique. Un tel réseau est constitué de plusieurs centres de commutation (noeuds) reliés entre eux par des liens. Chaque lien est composé de plusieurs tronçons. Un tronçon permet de "porter" un appel entre deux noeuds adjacents (i.e reliés directement).

Etant donné que tous les noeuds ne sont pas connectés entre eux, il faut établir une stratégie d'acheminement qui va décider du chemin à prendre pour deux noeuds qui ne sont pas reliés directement. Il existe plusieurs stratégies; nous avons choisi la stratégie progressive connue sous le nom de S.O.C (Successive Office Control). Les informations requises pour cette stratégie sont données dans une matrice appelée table d'acheminement (routing table).

Dans certains réseaux il est avantageux de réserver des tronçons entre deux noeuds. Par exemple si 10 tronçons sont réservés entre deux noeuds A et B, ils ne peuvent servir pour des appels de X (un noeud quelconque) vers B en passant par A. Les 10 tronçons sont maintenus exclusivement pour les appels originant de A vers B. Naturellement, il doit exister un lien entre A et B.

Une autre possibilité c'est ce qu'on a appelé les appels retenus (STORE.AND.FORWARD). Si entre deux noeuds, à un moment

donné de la simulation, tous les chemins sont bloqués, on peut placer les appels dans une file d'attente (FIFO).

Il existe d'autres options que l'usager peut spécifier. Elles seront discutées plus en détails dans la section "Guide de l'usager". Pour l'instant, on s'intéresse aux options qui impliquent des structures de données plus complexes.

En termes de programmation il faut tout d'abord distinguer les entités du système. On crée deux entités permanentes:

- les liens (P. LINK)
- les liens réservés ou réservations (RES.LINK)

Les P. LINK possèdent neuf attributs. Ce sont le noeud d'origine, le noeud destination, le mode (direct ou retenu), le nombre de tronçons, le temps d'inter-arrivée des appels (correspond au trafic) et 4 autres attributs permettant de calculer le trafic acheminé et offert entre deux noeuds (peu importe le chemin), le trafic "porté" par les liens et le temps d'attente moyen dans la file d'attente (si on est en mode retenu). De plus, à chaque P. LINK est associé une file d'attente FIFO qui sert à retenir les appels.

Les RES. LINK n'ont que quatre attributs soient les noeuds origine et destination, le mode et le nombre de tronçons réservés.

Le langage SIMSCRIPT range les attributs comme des vecteurs indexables. Ainsi chaque attribut occupe un mot en mémoire.

En plus des entités permanentes le système comprend des objets dynamiques et éphémères. C'est ce que SIMSCRIPT appelle les entités temporaires.

La première qui nous vient à l'idée c'est l'appel. En effet, les appels sont de courte durée si on les compare au temps total de la simulation. Les entités temporaires ont aussi des attributs. L'appel en possède deux: le premier, POSITION, correspond à l'indice de l'entité permanent P.LINK ayant les caractéristiques de l'appel (origine, destination, mode); le second, ATIME, sert à calculer le temps d'attente d'un appel.

On utilise deux autres entités temporaires. PATH n'a pas d'attribut mais possède une queue, PATH.QU. l'autre entité, LINK, a un attribut VALUE qui donne l'indice d'un lien entre deux noeuds. Par lien on entend un P.LINK avec un nombre de tronçons positifs. La file PATH.QU est emplie de LINK.

Ces deux entités sont utilisées pour sauvegarder le chemin d'un appel. Ainsi, lorsqu'un appel (CALL) arrive dans le système, on lui associe un chemin (PATH) lequel est formé de liens (LINK). Chaque lien a une valeur (VALUE) qui correspond à l'indice de

l'entité permanente P. LINK dont les caractéristiques sont les mêmes que l'appel (origine, destination, mode).

La table d'acheminement est représentée par une matrice à trois dimensions nommée RT. Pour trouver le premier choix d'une origine-destination (O-D) quelconque, on indice la table comme suit: RT (O, D, 1), pour le deuxième choix: RT (O, D, 2) et ainsi de suite.

Pour de petits réseaux, cette technique peut être valable mais une technique qui utilise les matrices creuses (sparse matrix) serait avantageuse pour les réseaux de grande taille.

Ceci termine la section sur les structures de données. Les événements et routines sont décrits dans la section suivante.

4. La modélisation

Le programme étudié ici a été écrit de façon modulaire. A la base, on retrouve trois événements qui sont en fait la charpente de tout le programme. Ces trois événements sont:

- l'arrivée (ARRIVAL et RECALL)
- l'acheminement (ROUTING) et
- le départ (ENDING) d'un appel.

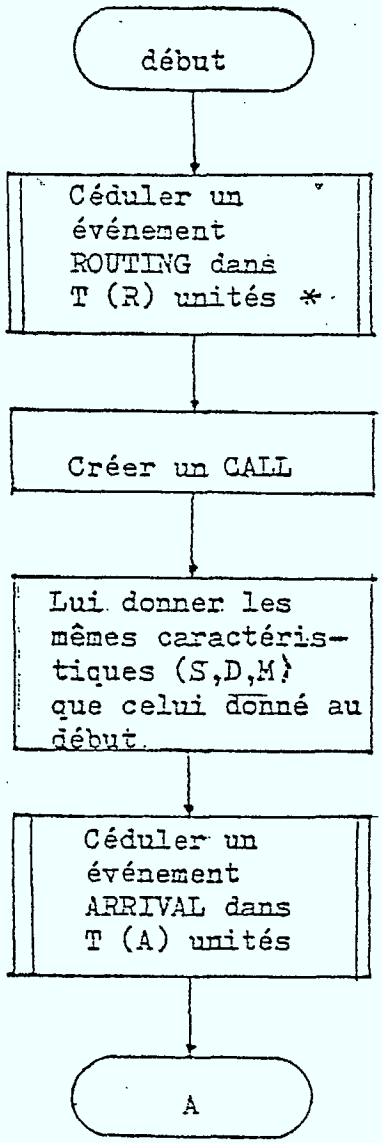
4.1 L'arrivée d'un appel

Comme son nom l'indique, l'arrivée d'un appel est cédulée chaque fois qu'un appel est introduit dans le réseau. Au départ, le programme cédule un appel pour chaque lien ayant un temps d'inter-arrivée positif. Tout au long de la simulation le nombre d'événements ARRIVAL sera le même et égal au nombre de liens ayant un temps d'inter-arrivée positif.

L'événement est très simple comme le montre l'organigramme de la figure 1. Il consiste à céduler un acheminement pour l'appel en cours, à créer un autre appel avec les mêmes caractéristiques (origine, destination, mode) et à céduler une autre arrivée selon le temps d'inter-arrivée du lien correspondant.

Une autre version de l'événement ARRIVAL est appelée RECALL (voir figure 2). Cet événement est utilisé lorsqu'un appel non acheminé est réessayé. RECALL est identique à ARRIVAL

Evénement ARRIVAL GIVEN CALL



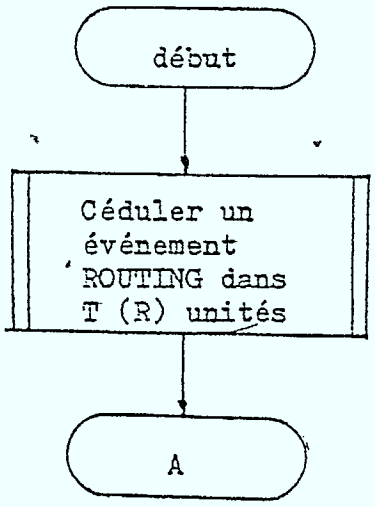
T (R) : Temps de commutation d'un appel
T (A) : Temps d'inter-arrivée d'un appel. Il suit une loi exponentielle

A : Signifie qu'on retourne le contrôle à la "timing routine", c'est-à-dire, la routine qui fait la gestion des listes d'événements

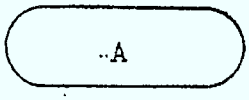
* Les unités sont définies par l'utilisateur

FIGURE 1

Evénement RECALL GIVEN CALL



T (R) : Temps de commutation d'un appel



: On redonne le contrôle à la "timing routine"

FIGURE 2

sauf qu'il ne cédule pas une prochaine arrivée pour l'appel en cours. Si c'était le cas le nombre d'arrivées pour ce lien irait en croissant de façon exponentielle.

4.2 L'acheminement

Le second événement constitue le coeur de la simulation. Il exécute la stratégie d'acheminement dans le réseau, en l'occurence la stratégie progressive (S.O.C). Les organigrammes des figures 3, 4 et 5 définissent la technique utilisée pour acheminer les appels selon cette stratégie.

Si aucun chemin n'est disponible pour acheminer l'appel, on vérifie s'il n'y aurait pas de réservations pour ce lien. Les réservations correspondent à des tronçons qui relient directement le noeud origine et destination de l'appel en cours. Il est à noter que le mode du tronçon doit être le même que le mode de l'appel.

Si après avoir vérifié les réservations, on ne peut atteindre la destination alors l'appel est "non acheminé". Dès lors, les actions à prendre dépendront du mode. L'appel en mode direct (MODE = 1) est perdu ou réessayé; celui en mode retenu (MODE = 2) est mis en attente dans une file. Il existe une file d'attente pour chaque lien. L'appel est donc placé dans une file correspondant à ses caractéristiques (origine, destination, mode).

Pour acheminer l'appel, l'événement ROUTING se sert de deux routines: VERIFY (figure 4) et RESERV (figure 5). La première routine vérifie s'il existe des tronçons disponibles pour l'origine, la destination et le mode qu'on lui fournit. Si tous les tronçons sont occupés, alors elle retourne zéro dans la variable FLAG. Autrement FLAG indique la position du lien physique où les tronçons sont disponibles.

On doit fournir trois arguments à la routine RESERV. Le premier PATH est une entité temporaire qui possède une file d'attente PATH.QU. Cette file contient la route qu'on doit libérer ou réserver. L'action à entreprendre est définie par le deuxième argument GUIDE. Le GUIDE est égal à -1, on libère les tronçons; l'appel est terminé. Le GUIDE est égal à 1 on réserve les tronçons; l'appel est acheminé. Le troisième argument sert pour calculer le trafic porté par chaque lien.

4.3 Le départ d'un appel

Le dernier événement (figure 6) est le départ de l'appel. Il se nomme ENDING et a trois arguments d'entrée qui sont: CALL, PATH et PERM. Les deux premiers sont l'appel et le chemin que ce dernier occupe en terme de tronçons. Ces deux entités ont été décrites plus tôt (voir "Les structures de données"). Le dernier, PERM est un fanion qui indique si l'appel utilise un lien réservé. Dans ce cas, PERM contient la position du lien réservé; autrement PERM vaut zéro.

Event ROUTING (CALL)

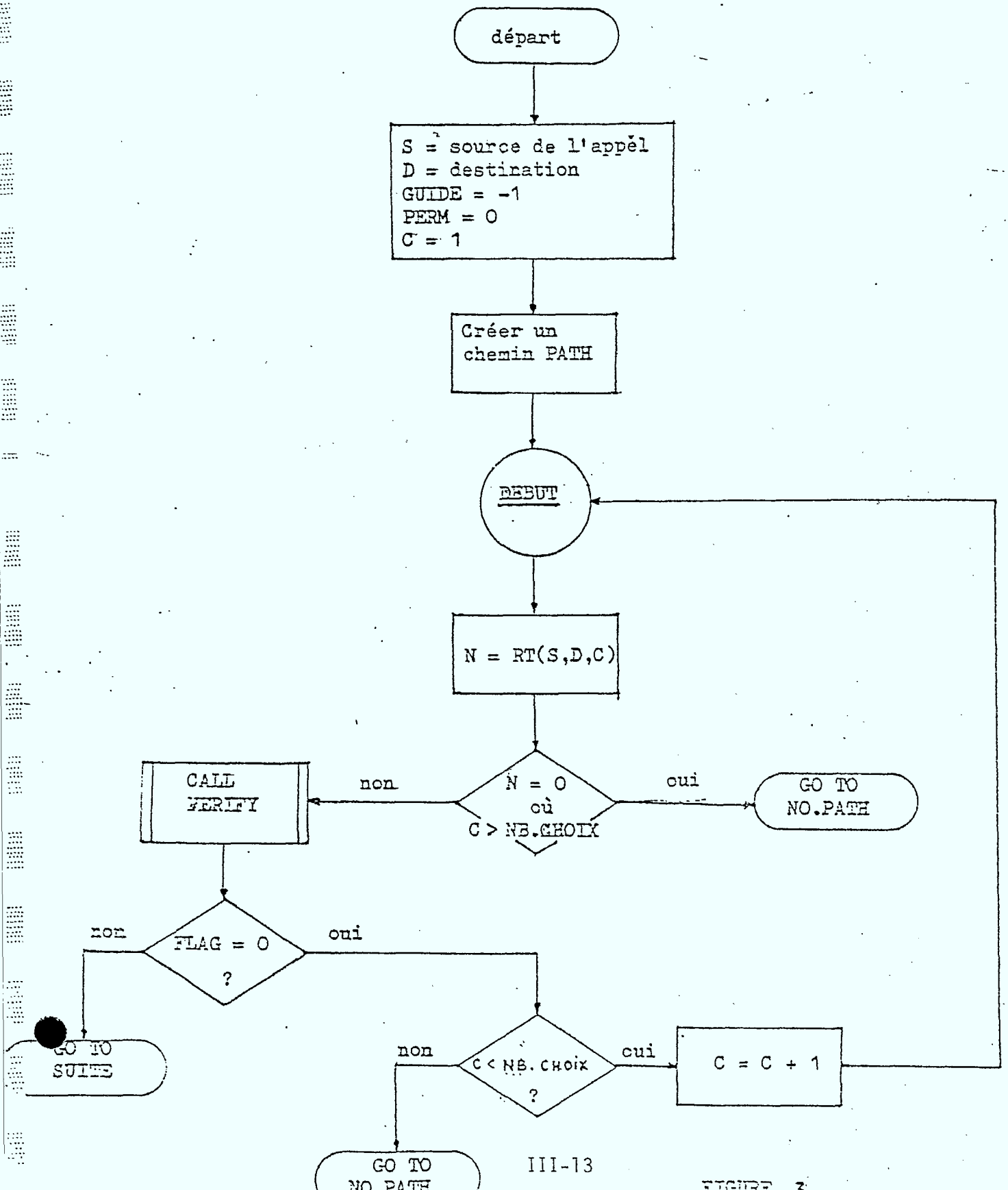


FIGURE 2

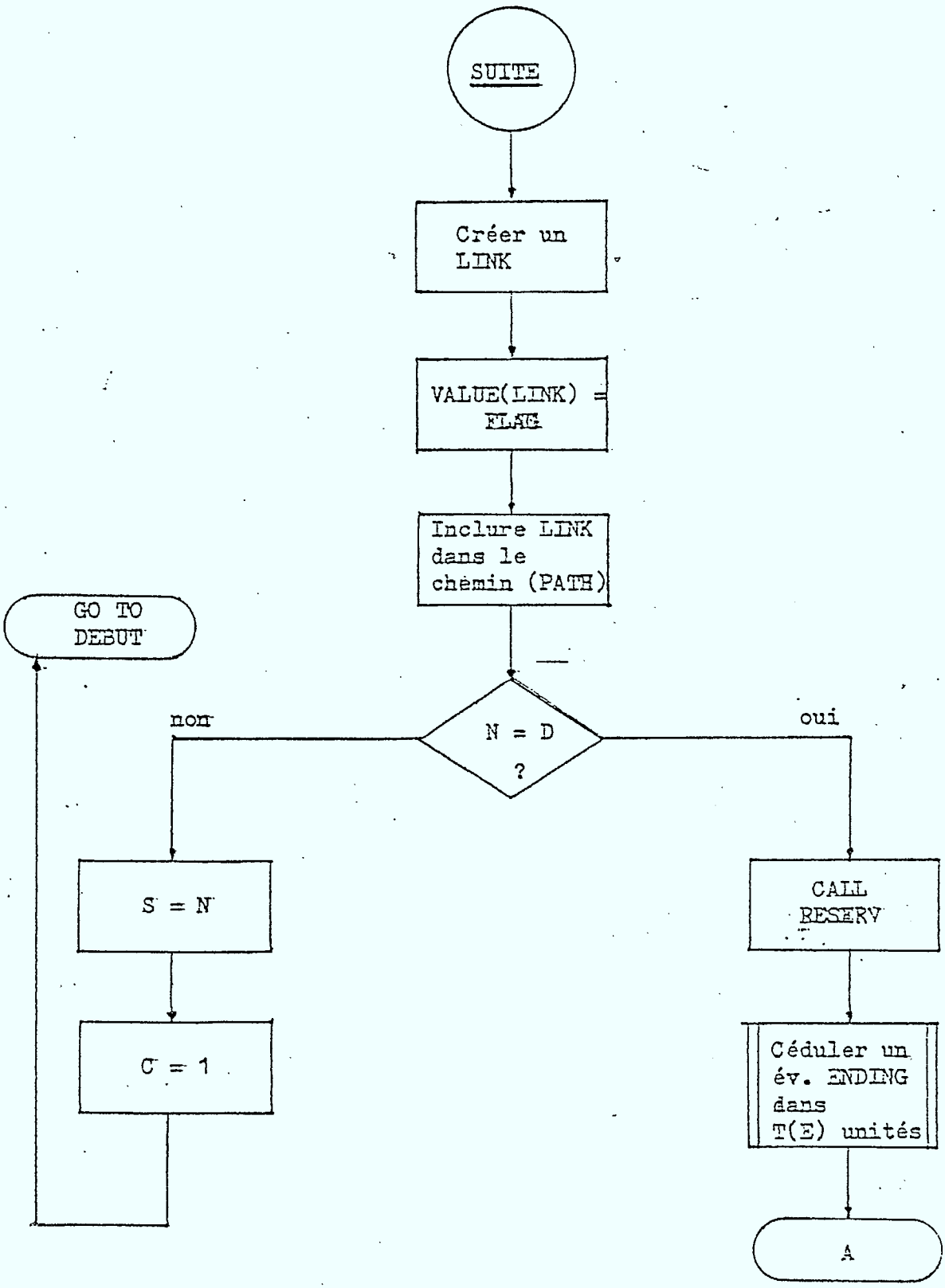


FIGURE 3 (SUITE.)

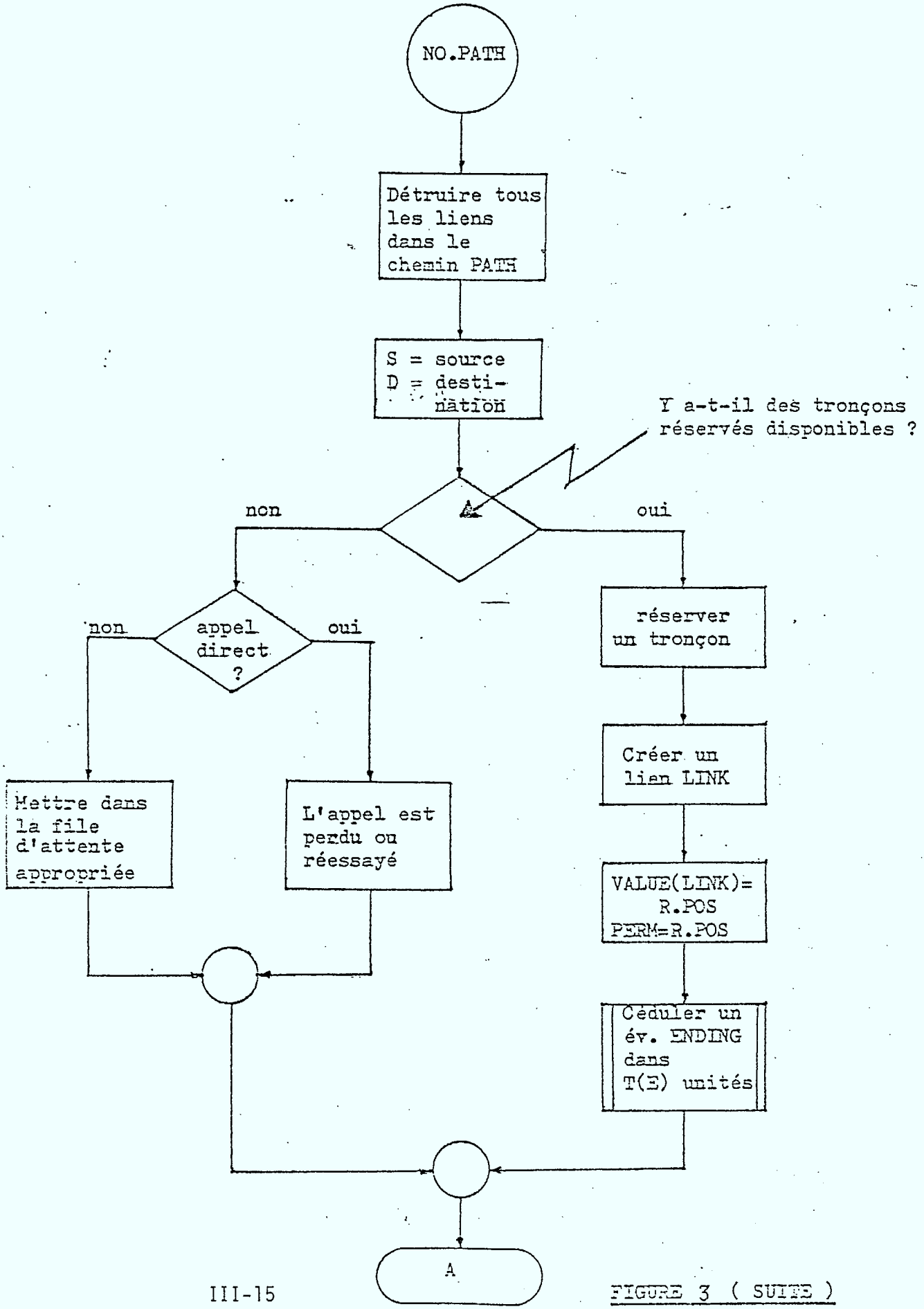


FIGURE 3 (SUITE)

LEGENDE : FIGURE 3

C : Détermine le choix

RT : Table d'acheminement. Elle est représentée par une matrice à trois dimensions.

NB.CHOIX : Nombre de choix possibles dans la table d'acheminement

FLAG : Cette variable est initialisée par la routine VERIFY. Elle indique l'indice du lien physique. Si FLAG = 0, c'est qu'il n'y a plus de tronçon disponible.

T(E) : Durée de l'appel

Routine VERIFY GIVEN S,D,M YIELDING POS

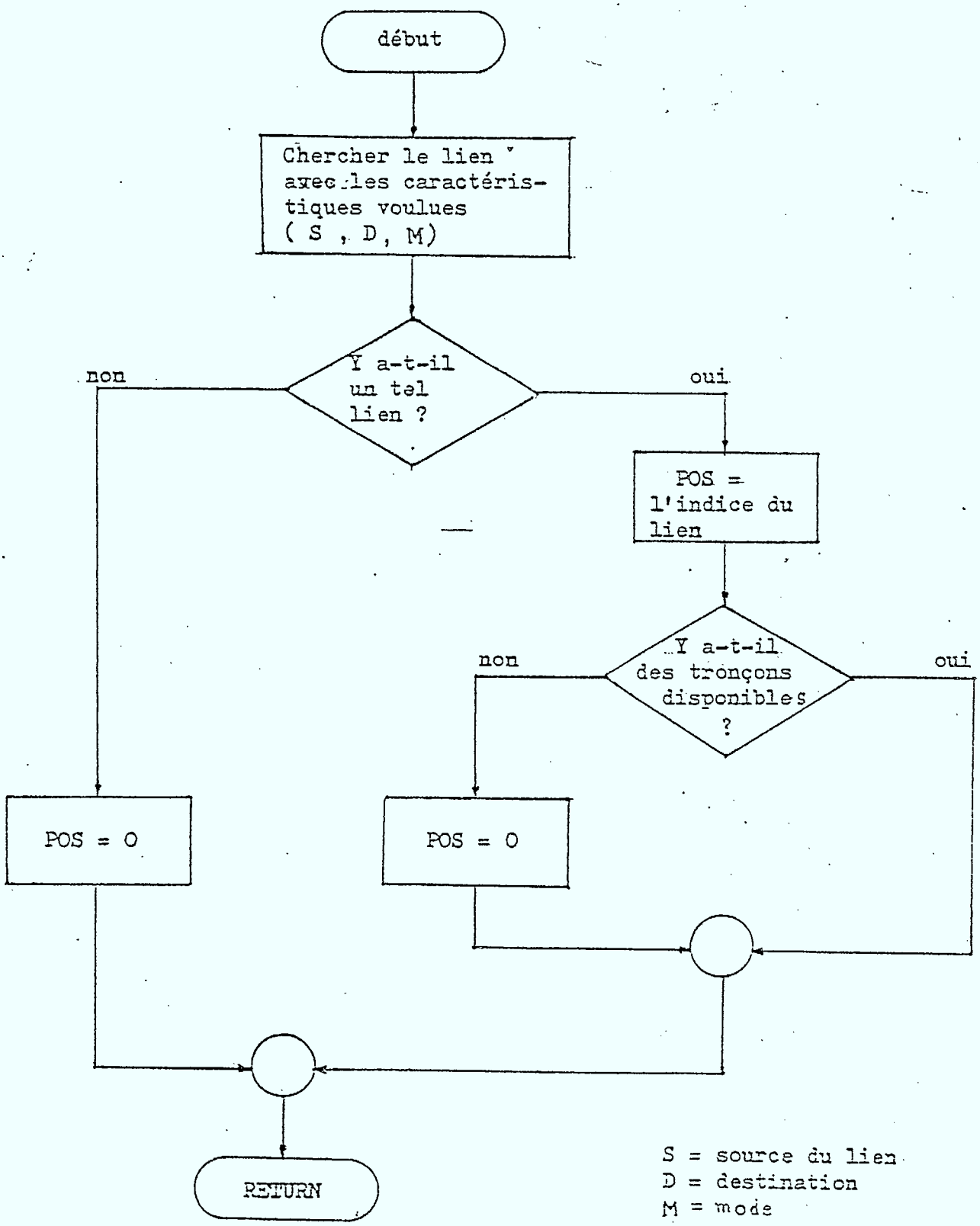
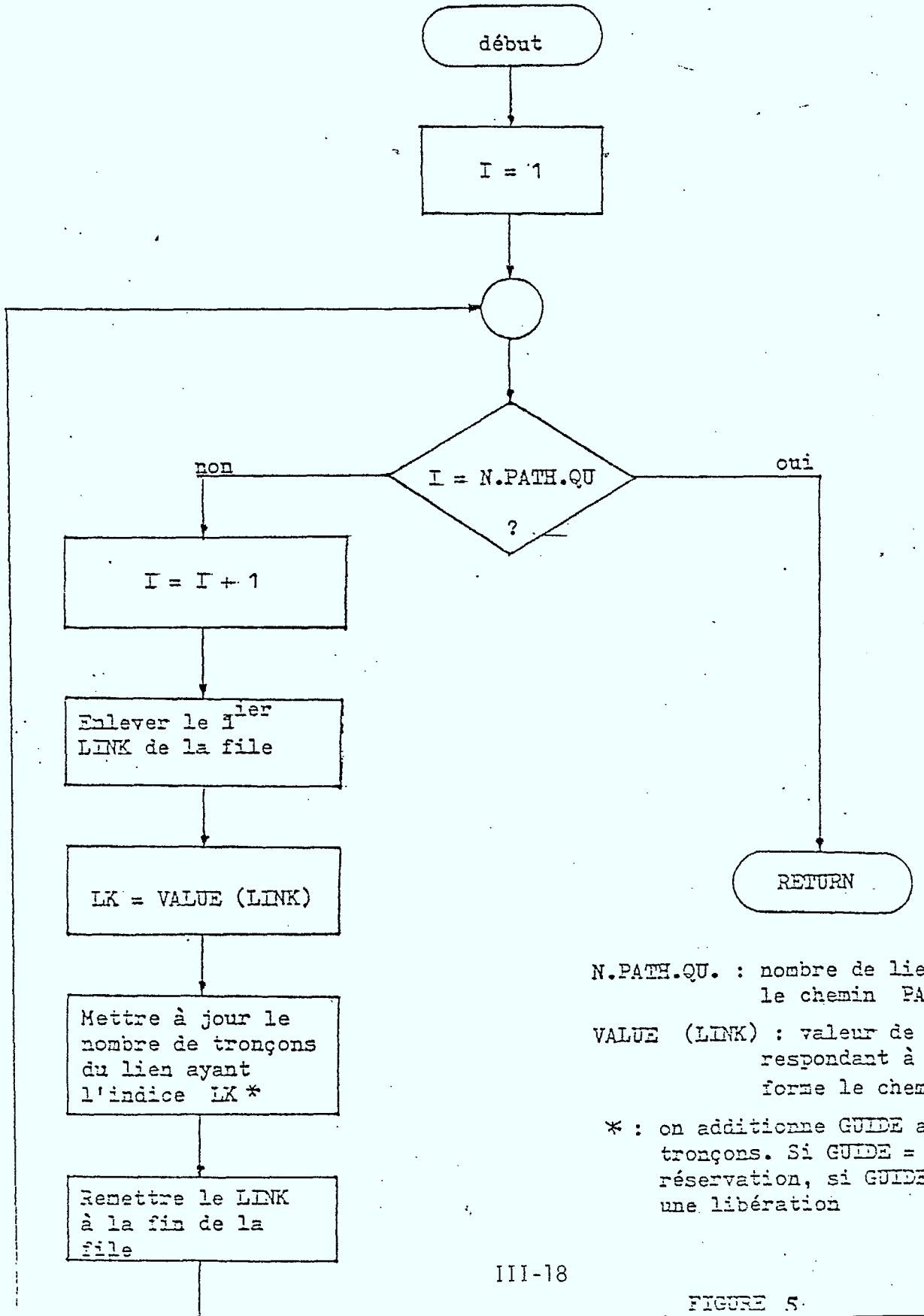


FIGURE 4

Routine RESERV GIVEN PATH , GUIDE , DUREE



N.PATH.QU. : nombre de liens qui forment le chemin PATH

VALUE (LINK) : valeur de l'indice correspondant à un lien qui forme le chemin

* : on additionne GUIDE au nombre de tronçons. Si GUIDE = -1 , c'est une réservation, si GUIDE = 1, c'est une libération

FIGURE 5

L'événement ENDING a trois fonctions. Tout d'abord il libère les liens occupés par l'appel. Il détruit les entités temporaires: les LINK contenus dans PATH.QU, l'appel (CALL) et le chemin associé (PATH). En dernier lieu il vérifie si des appels sont en attente dans la file correspondant aux caractéristiques (origine, destination, mode) de l'appel. Si tel est le cas alors il libère le premier arrivé dans la queue.

On vient de donner ici une description assez détaillée des événements essentiels du programme de simulation. Ceux-ci ont la tâche de manipuler les données en entrée de façon à générer de l'information à propos d'un réseau. A ces trois événements viennent s'ajouter quelques autres routines et événements qui complètent la simulation. Nous regroupons ces programmes sous le titre: "procédures de contrôle". Chacune de ces procédures est décrite brièvement dans ce qui suit.

4.4 Les procédures de contrôle

Par cette expression, on inclut tous les événements ou routines qui jouent un rôle secondaire dans la simulation. Il y en a cinq:

Evénement ENDING (CALL , PATH , PERM)

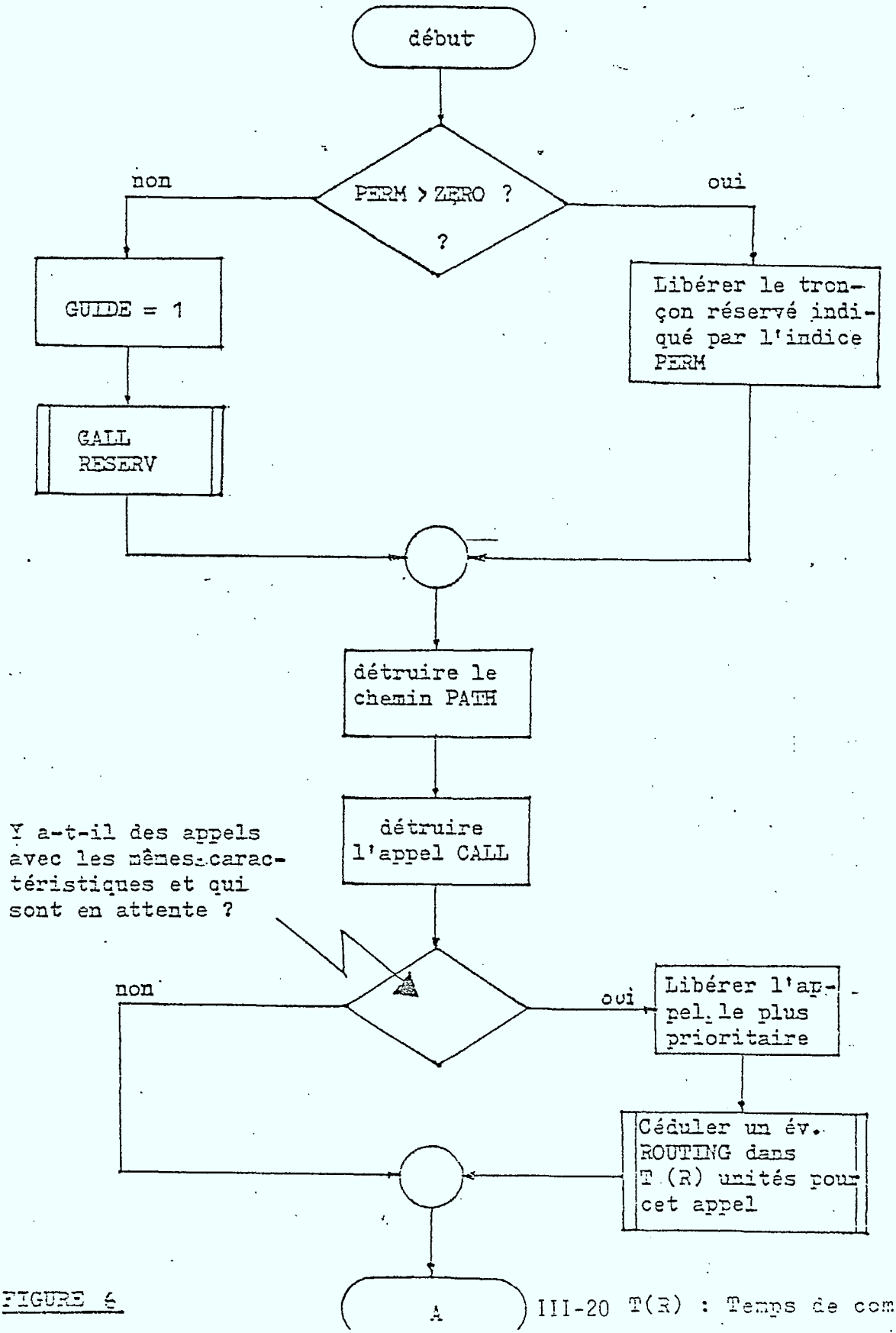


FIGURE 6

 T(R) : Temps de commutation.

Evénement ENDING (CALL , PATH , PERM)

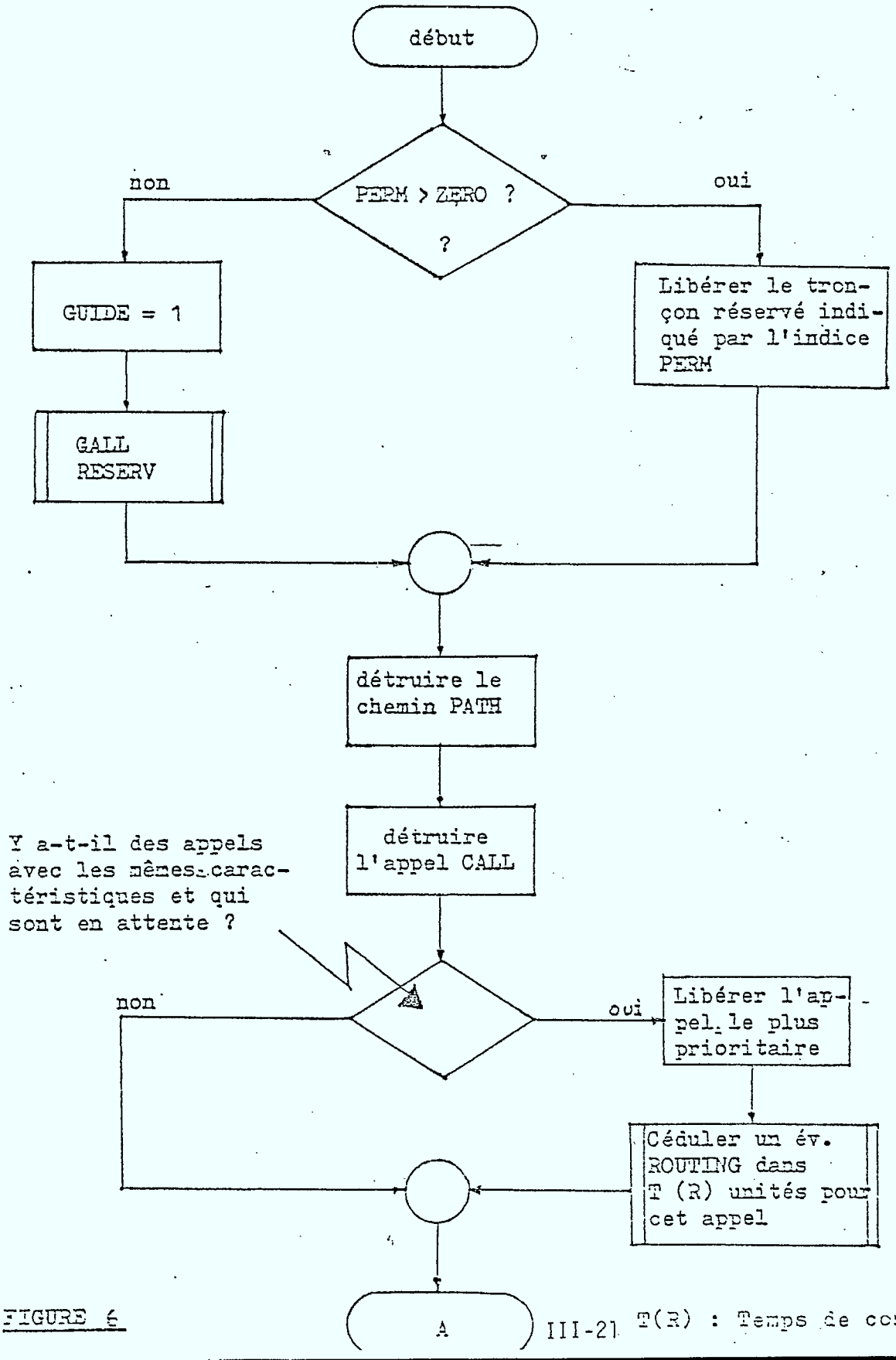


FIGURE 6

T(R) : Temps de commutation.

INIT

INITIAL.STATE

CHANGE

TRACES

END.OF.SIMULATION

La routine INIT est celle qui annonce la simulation en cédulant une arrivée (ARRIVAL) pour chaque lien ayant un temps d'inter-arrivée positif. Elle permet aussi d'initialiser le réseau pour atteindre plus rapidement l'état stable. Cette routine ne sert qu'au début du programme c'est pourquoi après l'avoir utilisée, on libère l'espace qu'elle occupe. Parmi les cinq procédures de contrôle INIT est la seule routine; les quatre autres sont des événements.

La seconde procédure, INITIAL.STATE imprime l'état initial du système et remet tous les compteurs à zéro. Cet événement est appelé chaque fois qu'on amorce une simulation. Elle décrit le réseau ainsi que les options de l'usager. Elle imprime aussi la table d'acheminement.

L'événement CHANGE permet de changer le taux des arrivées de façon linéaire. En d'autres mots il peut augmenter ou diminuer le trafic dans le réseau. Son utilisation est très simple (voir: guide de l'usager) et peut être exploité lors de tests pour optimiser un réseau.

L'événement suivant, TRACES, peut être cédulé autant de fois qu'on le désire pendant une simulation. Il donne toute l'information concernant le réseau à un moment donné de la simulation. Un événement TRACES est cédulé automatiquement à la fin de la simulation.

Les résultats fournis par cette procédure sont divisés en cinq parties comme suit:

. Contents of the event list

. Network progress report

. Network performance statistics

. Link performance statistics

. Summary of network performances.

Chaque partie sera explicitée dans une section prochaine intitulée "Les sorties". Il faut faire attention dans l'utilisation de cet événement car il peut générer beaucoup de lignes de sorties.

Il est à noter que l'impression de ces résultats au début de la simulation (horloge = 0.0) ne contient que trois parties. Les statistiques sur les performances des liens et du réseau sont éliminées.

Le dernier événement, END.OF.SIMULATION, vérifie tout d'abord s'il reste des données. Si la réponse est positive il amorce une

autre simulation avec de nouveaux paramètres. On ne peut pas modifier le réseau. S'il rencontre la fin des données, il termine le programme par l'impression du message suivant:

FIN NORMALE DE LA SIMULATION # # # #

Ceci termine la section sur la modélisation. L'usager devrait maintenant connaître le fonctionnement général du programme SIMRET. La section suivante est le guide de l'usager. Elle devrait lui permettre de décrire les données qui représentent le réseau.

5. Le guide de l'usager

Cette partie se veut le guide de l'usager du programme de simulation SIMRET. On y donne tous les paramètres, leur définition et l'ordre de leur lecture. Par la suite un exemple d'utilisation montre comment on peut modéliser un réseau à l'aide de SIMRET. Puis en dernier lieu on explique les sorties générées par le programme.

5. 1 Données d'entrée

On parle ici de cartes mais les formats en entrée sont tous des formats libres (sauf la CARTE-2). L'utilisation de cartes ne sert qu'à donner une certaine logique dans l'ordre des paramètres.

S'il le veut l'usager peut mettre toutes ses données les unes à la suite des autres en les séparant seulement par un blanc (espace). Voici la description de chaque "carte" rencontrée en entrée.

CARTE-1

Cette carte est obligatoire. Elle décrit le réseau à l'aide de deux nombres entiers. Le premier est le nombre de noeuds, le second le nombre de liens physiques du réseau. Dans un réseau de N noeuds, le nombre maximum de liens physiques est égal à (N *(N-1)/2).

CARTE-2

C'est la seule carte de type caractère. On y inscrit les unités avec lesquelles la simulation va se faire. Cette donnée est purement informative. Le programme prend les 16 premières colonnes de la carte. Aucune action n'est générée par cette carte lors de la simulation.

Il faut faire attention de bien mettre cette donnée sur une nouvelle carte. C'est la seule donnée qu'on ne peut écrire en format libre.

CARTE-3

On y place trois données de type REAL. La première indique le temps de commutation d'un appel. La seconde est le temps de réessai d'un appel. La dernière donne le pourcentage de chance qu'un appel soit perdu plutôt que réessayé lorsqu'il ne peut être acheminé.

CARTE-4

Celle-ci est une table qui indique le pourcentage qu'un appel soit d'une certaine durée. Sur cette carte, on doit inscrire des paires de nombres. Le premier de ces nombres est une probabilité cumulative, le second la durée moyenne de l'appel.

Un exemple expliquera mieux cela. Si on sait que la durée d'un appel est d'environ:

120 secondes	20% du temps
200 secondes	50% du temps
300 secondes	25% du temps
1000 secondes	5% du temps

alors on aura la carte suivante:

```
0.20  120.0  0.70  200.0  0.95  300.0  1.0  1000.0  *
```

Il est à noter que la dernière probabilité doit être égale à 1.0 et que le dernier caractère de la carte doit être un astérisque(*). On peut prendre le nombre de cartes que l'on veut.

CARTE-5

Cette carte donne un nombre entier appelé N.P.LINK. C'est le nombre de liens du réseau. Chaque lien dont on parle correspond à un noeud origine, un noeud destination et à un mode. Il doit y avoir au moins N * (N-1) liens, où N est le nombre de noeuds.

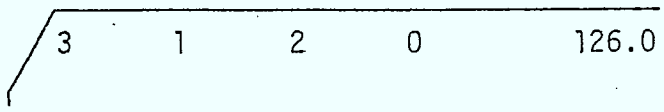
CARTE-6

En fait, cette carte contient cinq données dont quatre du type INTERGER et la dernière du type REAL. On doit avoir N.P. LINK CARTE-6 c'est-à-dire autant que le nombre mentionné par la CARTE-5.

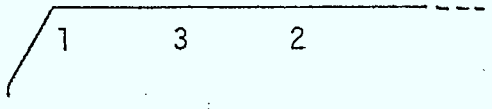
Cette carte contient dans l'ordre, le noeud origine, le noeud destination, le mode, le nombre de tronçons et le temps inter-arrivée des appels. Le mode doit être 1 pour les appels directs et 2 pour les appels retenus. Il faut obligatoirement avoir des modes identiques pour le lien origine-destination (O-D) et pour le lien destination-origine (D-O).

Un autre point à noter c'est que les cartes avec un noeud origine plus grand que le noeud destination ne doivent pas avoir de tronçons (NB.TRUNKS=0). Même si on inscrit un nombre positif, le programme n'en tient pas compte. Cela est dû au fait que les tronçons entre deux noeuds O et D sont les mêmes qu'entre les noeuds D et O.

Ainsi la carte:



sous-entend une carte commençant par



et avec un nombre de tronçons positifs.

CARTE-7

Ce nombre entier indique le nombre de réservations. Ce nombre, qui peut être égal à zéro est appelé N.RES.LINK. L'usage de cette option peut entraîner des sorties moins soignées.

CARTE-8

Cette carte définit une réservation. Il doit y avoir N.RES.LINK cartes de ce genre. Chaque carte est composée de quatre variables de type INTERGER. Dans l'ordre on a: l'origine, la destination, le mode et le nombre de tronçons.

Chacune de ces cartes doit être associée à une CARTE-6 qui contient les trois premiers attributs identiques et un nombre de tronçons supérieur ou égal. Le nombre de tronçons n'est pas nécessairement indiqué sur la CARTE-6 associée.

Ainsi, une réservation définie comme suit:

```
/3      1      2      5
```

implique deux CARTE-5. La première contient les trois premiers attributs identiques (3, 1, 2)

```
/3      1      2      0      212.0
```

mais étant donné que la source est plus grande que la destination (3 > 1), le nombre de tronçons pour le lien 3-1 sera indiqué sur la CARTE-5 suivante:

```
1     3     2     N     126.0
```

où N est plus grand ou égal à 5, le nombre de réservations.

CARTE-9

Cette carte comprend un nombre entier (NB.CHOIX) qui indique le nombre de choix (incluant la route primaire) qu'un appel peut avoir. On conseille un nombre plus petit que cinq pour que les sorties finales soient lisibles.

CARTE-10

Il doit y avoir exactement N * (N-1) cartes de ce genre où N est le nombre de noeuds du réseau. Chaque carte comprend (NB. CHOIX + 2) paramètres. Ce sont dans l'ordre, l'origine, la destination et les NB.CHOIX dans l'ordre de préférence. Le total de ces cartes constitue l'information nécessaire pour construire la table d'acheminement.

CARTE-11

Cette carte est un nombre REAL qui indique combien de temps on doit simuler pour atteindre ce qu'on appelle un état stable du système.

Si on a le résultat d'une simulation antérieure alors on peut se servir des CARTE-17 et CARTE-18 qui permettent d'initialiser le système. Cette pratique peut réduire considérablement le temps requis pour atteindre un état stable.

CARTE-12

Cette carte exprime le nombre de changements du taux que l'on veut effectuer. Elle est directement reliée à la CARTE-13.

CARTE-13

Chacune de ces cartes contient deux données: le temps et le taux. Le temps est la valeur absolue à laquelle on doit initialiser le nouveau taux. Par exemple, si après 300 secondes de simulation on veut doubler le trafic et ce pendant 200 secondes alors on aura:

2		(CARTE-12)	(nombre de CARTE-13)
300	2.0	(CARTE-13)	(on double le taux)
500	1.0	(CARTE-13)	(on revient au taux initial).

CARTE-14

C'est tout simplement le nombre de fois qu'on devra imprimer les résultats associés au système. En d'autres mots, c'est le nombre de CARTE-15.

CARTE-15

Cette carte contient le temps de la simulation où l'on veut faire imprimer les informations relatives à l'état du système. Ces informations incluent:

- le contenu des listes d'événements
- l'état du réseau
- les performances du réseau
- les performances des liens
- un résumé des performances

CARTE-16

Le nombre de type REAL indiqué sur cette carte représente le temps total de la simulation. A la fin de la simulation, les informations décrites à la CARTE-15 sont imprimées.

CARTE-17

Cette donnée permet d'initialiser le système à l'aide d'information antérieure. Elle indique le nombre de CARTE-18 qui suivent. Elle peut être égale à zéro.

CARTE-18

Chacune de ces cartes est constituée de quatre valeurs: le noeud origine, le noeud destination, le mode et le nombre de tronçons qu'on doit occuper. On ne peut pas initialiser des chemins, seulement des liens.

Si le nombre de tronçons qu'on veut occuper sur un lien est plus grand que le nombre de tronçons disponibles, on occupe tous les tronçons et on ne donne aucun message à l'usager.

Si l'usager veut faire plus d'une simulation avec le même réseau, cela lui est permis en ajoutant autant de fois qu'il veut le groupe de cartes suivant (CARTE-19 à CARTE-23).

CARTE-19

Cette carte permet d'atteindre un nouvel état stable du système. Elle peut être utile, par exemple, si on veut simuler le même système mais en augmentant le taux du trafic. Lors de l'état stable, tous les compteurs sont remis à zéro.

CARTE-20

Cette carte détermine le taux. Comme on vient de la mentionner, il peut être intéressant de simuler le même réseau mais avec un trafic différent. Il faut toutefois faire attention, les changements de taux qui ont été mis dans la liste d'événements lors de la première simulation et qui ne sont pas passés demeurent actifs.

CARTE-21

Encore ici on demande le nombre de fois qu'on devra imprimer les résultats. Cette carte correspond à la CARTE-14.

CARTE-22

Cette carte a la même fonction que la CARTE-15, elle indique les temps où devront avoir lieu les impressions mentionnées à la CARTE-15.

CARTE-23

Comme la CARTE-11, cette carte indique la durée de la simulation. A la fin de chaque simulation, toute l'information sur le système est imprimée.

On peut faire autant de simulations que l'on veut. Pour chacune d'entre-elles il s'agit d'ajouter une série de cartes allant de CARTE-19 à CARTE-23, à la suite les unes des autres.

5.2 Exemple d'utilisation

Cette section présente la façon de coder nos données d'entrée pour un réseau de dix noeuds et de 40 liens. L'information fournie sur le réseau est la suivante:

- la structure du réseau (figure 7)
- le nombre de tronçons pour chaque lien (table 1)
- le trafic en CCS (table 2)
- la table d'acheminement (table 3)
- la durée moyenne d'un appel qui est de 200 secondes.

En fait ce réseau est constitué de 10 centres de commutation dont 9 de classe 5 et un de classe 4 (noeud 10). C'est pourquoi, dans la table d'acheminement, on réfère si souvent au noeud 10. Si deux bureaux de classe 5 ne peuvent être reliés directement, alors on passe par le bureau de classe 4 (noeud 10).

Le réseau Bell

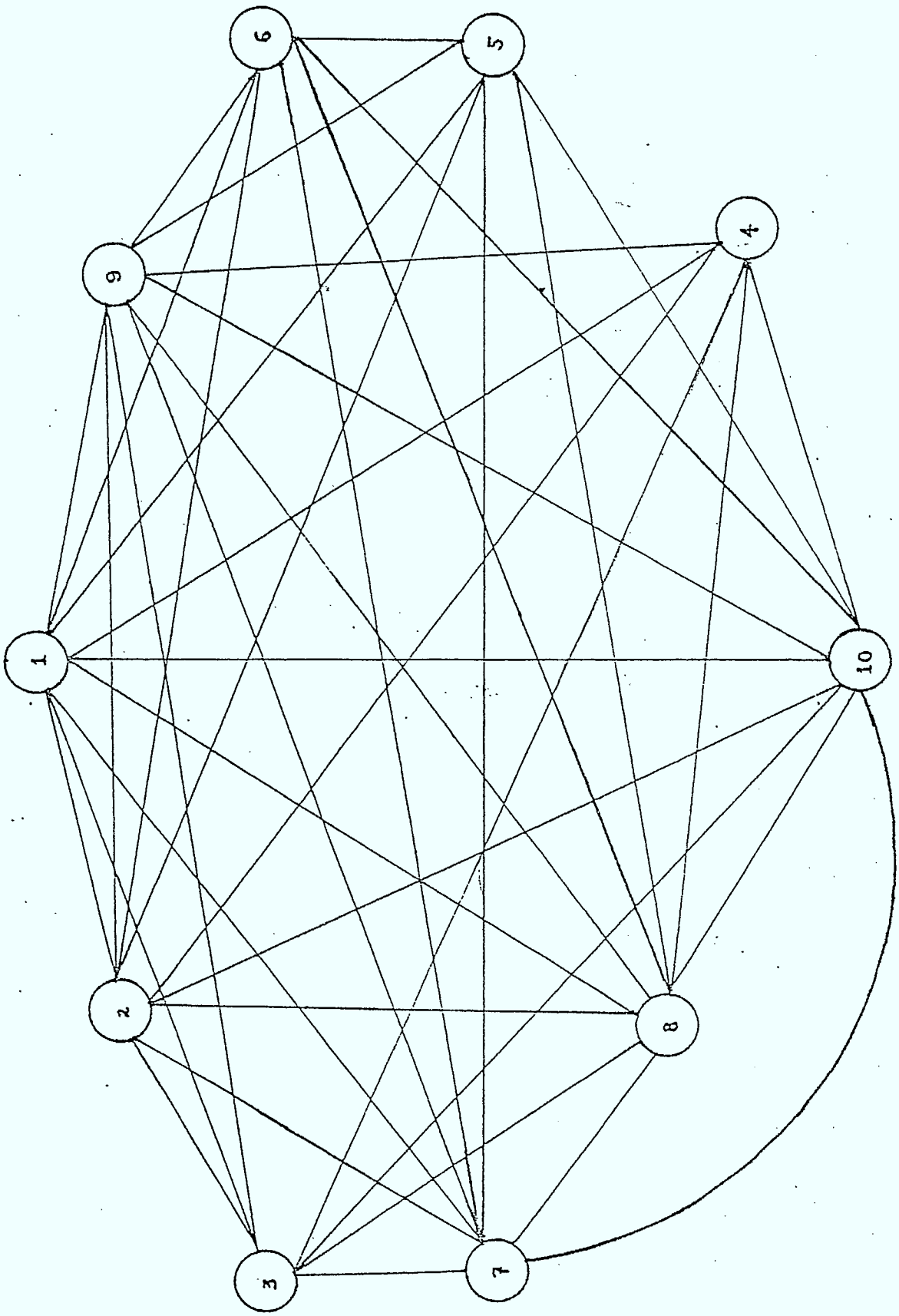


Figure 7

de \ vers	1	2	3	4	5	6	7	8	9	10
1	-	546	135	56	91	265	39	602	205	616
2		-	35	17	45	131	14	313	292	208
3			-	22	0	0	22	63	13	90
4				-	0	0	0	47	7	64
5					-	187	15	21	14	154
6						-	46	98	45	352
7							-	16	2	50
8								-	186	208
9									-	204
10										-

Table 1: Nombre de tronçons pour chaque lien.

vers de	1	2	3	4	5	6	7	8	9	10
1	-	11956	2848	1328	2238	5844	879	12578	3322	11726
2	10833	-	717	420	1007	2534	331	6851	5614	3803
3	2623	611	-	375	0	0	355	1302	231	1588
4	1354	359	368	-	0	0	0	821	137	819
5	1800	860	0	0	-	3743	238	449	364	2759
6	5441	2407	0	0	3962	-	948	2430	910	7081
7	922	332	349	0	376	848	-	430	0	779
8	12063	6401	1058	915	576	2269	373	-	3699	3879
9	5325	5595	272	162	426	918	88	3885	-	3727
10	11837	3545	1182	1062	2526	6007	637	3478	3482	-

Table 2: Matrice du trafic (en CCS)

vers de	1	2	3	4	5	6	7	8	9	10
1	-	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10
2	1,10	-	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10
3	1,10	2,10	-	4,10	10	10	7,10	8,10	9,10	10
4	1,10	2,10	3,10	-	10	10	10	8,10	9,10	10
5	1,10	2,10	10	10	-	6,10	7,10	8,10	9,10	10
6	1,10	2,10	3,10	10	5,10	-	7,10	8,10	9,10	10
7	1,10	2,10	3,10	10	5,10	6,10	-	8,10	9,10	10
8	1,10	2,10	3,10	4,10	5,10	6,10	7,10	-	9,10	10
9	1,10	2,10	3,10	4,10	5,10	6,10	10	8,10	-	10
10	1	2	3	4	5	6	7	8	9	-

Table 3: Table d'acheminement

Nous allons maintenant trouver chaque carte décrite dans les données d'entrée.

CARTE-1

Cette carte est simple c'est:

```
/  10        40
```

10 noeuds
40 liens

CARTE-2

Pour cet exemple, on choisit de travailler avec des secondes.

```
/  Secondes
```

CARTE-3

Etant donné qu'on ne nous donne pas d'information sur le temps de commutation on assume qu'il est négligeable (=0.0) On suppose aussi que le mode est direct et qu'un appel ne peut pas être réessayé. On a donc.

```
/  0.0      0.0      100.0
```

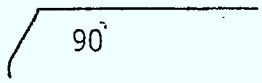
CARTE-4

La seule information qu'on a sur la durée de l'appel c'est qu'il dure en moyenne 200 secondes.

```
/  1.0        200.0 *
```

CARTE-5

Le nombre de liens du réseau est minimum i.e (N * (N - 1) car tous les liens sont en mode direct.



CARTE-6

Cette carte contient dans l'ordre, le noeud origine, le noeud destination, le mode, le nombre de tronçons et le temps inter-arrivée des appels.

Nous n'avons pas la dernière donnée mais nous pouvons la dériver à partir du trafic en CCS et de la durée moyenne d'un appel.

Trafic en Erlang = trafic en CCS/36.0

$$\text{Temps d'inter-arrivée} = \frac{\text{trafic en Erlang}}{\text{durée moyenne d'un appel}}$$

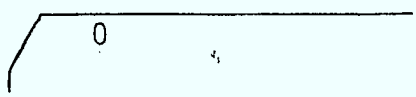
donc

$$\text{Temps d'inter-arrivée} = \frac{\text{trafic en CCS}}{36 \times 200}$$

La figure 8 montre les 90 CARTE-6. On voit qu'il n'y a pas de format à suivre.

CARTE-7

Il n'y a pas de réservation.



CARTE-6

```
 1  2  1    546  0.66
 1  3  1    135  2.74
 1  4  1     56  5.32
 1  5  1     91  4.00
 1  6  1    265  1.32
 1  7  1     39  7.81
 1  8  1    602  0.60
 1  9  1    205  1.35
 1 10  1    616  0.0
 2  1  1      0  0.60
 2  3  1     35 10.89
 2  4  1     17 20.05
 2  5  1     45  8.38
 2  6  1    131  2.99
 2  7  1     14 21.70
 2  8  1    313  1.12
 2  9  1    292  1.29
 2 10  1    208  0.0
 3  1  1      0  2.53
 3  2  1      0 10.04
 3  4  1     22 19.55
 3  5  1      0  0.0
 3  6  1      0  0.0
 3  7  1     22 20.64
 3  8  1     63  6.81
 3  9  1     13 26.43
 3 10  1     90  0.0
 4  1  1      0  5.42
 4  2  1      0 17.16
 4  3  1      0 19.20
 4  5  1      0  0.0
 4  6  1      0  0.0
 4  7  1      0  0.0
 4  8  1     47  7.87
 4  9  1      7 44.48
 4 10  1     64  0.0
 5  1  1      0  3.22
 5  2  1      0  7.15
 5  3  1      0  0.0
 5  4  1      0  0.0
 5  6  1    187  1.82
 5  7  1     15 19.14
 5  8  1     21 12.51
 5  9  1     14 16.89
 5 10  1    154  0.0
 6  1  1      0  1.23
 6  2  1      0  2.84
 6  3  1      0  0.0
 6  4  1      0  0.0
 6  5  1      0  1.92
 6  7  1     46  8.49
 6  8  1     98  3.17
 6  9  1     45  7.85
 6 10  1    352  0.0
 7  1  1      0  8.19
 7  2  1      0 21.78
 7  3  1      0 20.31
 7  4  1      0  0.0
 7  5  1      0 30.22
 7  6  1      0  7.59
 7  8  1     16 19.31
 7  9  1      2 82.02
 7 10  1     50  0.0
 8  1  1      0  0.57
 8  2  1      0  1.05
 8  3  1      0  5.53
 8  4  1      0  8.77
 8  5  1      0 16.03
 8  6  1      0  2.96
 8  7  1      0 16.75
 8  9  1    186  1.85
 8 10  1    208  0.0
 9  1  1      0  2.17
 9  2  1      0  1.28
 9  3  1      0 31.11
 9  4  1      0 52.63
 9  5  1      0 19.80
 9  6  1      0  7.91
 9  7  1      0  0.0
 9  8  1      0  1.95
 9 10  1    204  0.0
10  1  1      0  0.0
10  2  1      0  0.0
10  3  1      0  0.0
10  4  1      0  0.0
10  5  1      0  0.0
10  6  1      0  0.0
10  7  1      0  0.0
10  8  1      0  0.0
10  9  1      0  0.0
```

Figure 8

CARTE-8

Etant donné que la donnée précédente est zéro, il n'y a pas de CARTE-8 pour cet exemple.

CARTE-9

Comme on peut le voir dans la table 3 le nombre de choix maximum pour l'acheminement est de deux.

```
2
```

CARTE-10

Ces 90 cartes définissent la table d'acheminement (voir table 3). Sur chaque carte on retrouve l'origine, la destination et les deux choix dans l'ordre d'importance. La figure 9 donne une image de ces cartes.

CARTE-11

Pour que ce système atteigne un état stable on suppose à prime abord que le système doit fonctionner une heure (3600 secondes). Cette donnée est basée sur l'expérience que l'on a des réseaux.

```
3600
```

CARTE-10

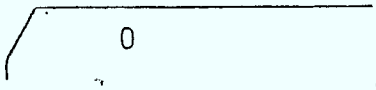
```
 1  2  2  10
 1  3  3  10
 1  4  4  10
 1  5  5  10
 1  6  6  10
 1  7  7  10
 1  8  8  10
 1  9  9  10
 1  10  10  0
 2  1  1  10
 2  3  3  10
 2  4  4  10
 2  5  5  10
 2  6  6  10
 2  7  7  10
 2  8  8  10
 2  9  9  10
 2  10  10  0
 3  1  1  10
 3  2  2  10
 3  4  4  10
 3  5  10  0
 3  6  10  0
 3  7  7  10
 3  8  8  10
 3  9  9  10
 3  10  10  0
 4  1  1  10
 4  2  2  10
 4  3  3  10
 4  5  10  0
 4  6  10  0
 4  7  10  0
 4  8  8  10
 4  9  9  10
 4  10  10  0
 5  1  1  10
 5  2  2  10
 5  3  10  0
 5  4  10  0
 5  6  6  10
 5  7  7  10
 5  8  8  10
 5  9  9  10
 5  10  10  0
 6  1  1  10
 6  2  2  10
 6  3  3  10
 6  4  10  0
 6  5  5  10
 6  7  7  10
 6  8  8  10
 6  9  9  10
 6  10  10  0
 7  1  1  10
 7  2  2  10
 7  3  3  10
 7  4  10  0
 7  5  5  10
 7  6  6  10
 7  8  8  10
 7  9  9  10
 7  10  10  0
```

```
 8  1  1  10
 8  2  2  10
 8  3  3  10
 8  4  4  10
 8  5  5  10
 8  6  6  10
 8  7  7  10
 8  9  9  10
 8  10  10  0
 9  1  1  10
 9  2  2  10
 9  3  3  10
 9  4  4  10
 9  5  5  10
 9  6  6  10
 9  7  10  0
 9  8  8  10
 9  10  10  0
10  1  1  0
10  2  2  0
10  3  3  0
10  4  4  0
10  5  5  0
10  6  6  0
10  7  7  0
10  8  8  0
10  9  9  0
```

Figure 9

CARTE-12

Pour une première simulation il n'est pas utile d'employer la carte pour changer le taux du trafic.



CARTE-13

Cette carte n'est pas utilisée car la CARTE-12 est initialisée à zéro.

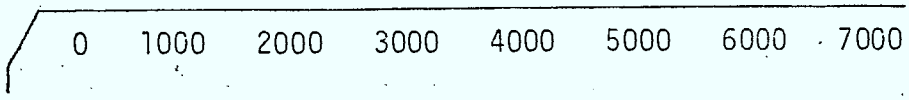
CARTE-14

Lorsqu'on code cette carte il faut avoir déterminé le temps de la simulation. On le suppose fixé à 8000 secondes.

Etant donné que c'est la première "passe" on va faire imprimer les résultats assez souvent pour pouvoir s'assurer si la liste d'événements est stabilisée. On décide donc de faire imprimer les résultats à toutes les 1000 secondes à partir de 0 inclusivement. On a donc 8 impressions à céduler.

CARTE-15

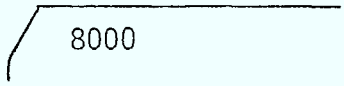
On exprime sur cette carte les temps où l'on veut faire imprimer des résultats.



On ne doit pas écrire 8000 car les résultats sont imprimés automatiquement à la fin de la simulation. Si on le fait tout de même on aura les résultats au temps 8000 en double.

CARTE-16

C'est le temps total de la simulation. Comme on l'a dit précédemment il est fixé à 8000.



CARTE-17

On n'a pas fait de simulation auparavant donc on ne peut initialiser le réseau.

CARTE-18

Il n'y a en pas car la CARTE-17 est égale à zéro.

Les cartes qui suivent sont facultatives. Elles permettent de recommencer la simulation en modifiant quelques paramètres. Pour les premiers tests avec un réseau il n'est pas conseillé d'utiliser cette option.

Il est possible qu'après quelques simulations l'usager ait besoin de tester son réseau pour plusieurs taux de trafic différents. A cette étape il a déjà fait des simulations et il peut fournir un état initial pour son réseau. Disons qu'il veut voir les implications des différents taux suivants:

0.5 0.75 1.00 1.25 1.50

Montrons tout d'abord les cartes nécessaires à l'initialisation du réseau. Ces données sont tirées des résultats d'une simulation précédente.

CARTE-17

Il y a 40 liens dans le réseau. On les initialise tous.

CARTE-18

Il y a 40 de ces cartes. Chacune contient quatre valeurs: l'origine, la destination, le mode et le nombre de tronçons qu'on doit occuper. On peut voir ces cartes sur la figure 10.

De plus, on doit faire quatre autres simulations; une pour chaque taux mentionné plus tôt. Pour ce faire on devra utiliser les CARTE-19 à CARTE-23.

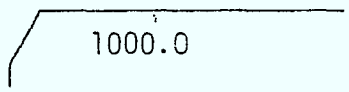
CARTE-18

```
1  2  1  0  538
1  3  1  0  127
1  4  1  0  55
1  5  1  0  91
1  6  1  0  265
1  7  1  0  38
1  8  1  0  591
1  9  1  0  204
1  10  1  0  331
2  3  1  0  29
2  4  1  0  17
2  5  1  0  45
2  6  1  0  118
2  7  1  0  14
2  8  1  0  313
2  9  1  0  284
2  10  1  0  195
3  4  1  0  20
3  7  1  0  14
3  8  1  0  51
3  9  1  0  13
3  10  1  0  22
4  8  1  0  47
4  9  1  0  4
4  10  1  0  50
5  6  1  0  181
5  7  1  0  14
5  8  1  0  18
5  9  1  0  13
5  10  1  0  92
6  7  1  0  45
6  8  1  0  91
6  9  1  0  43
6  10  1  0  177
7  8  1  0  16
7  9  1  0  1
7  10  1  0  28
8  9  1  0  181
8  10  1  0  179
9  10  1  0  114
```

Figure 10

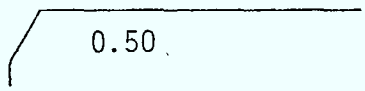
CARTE-19

Ayant fait d'autres simulations avec le réseau, il est possible de donner une valeur plus juste. Dans notre exemple, on fixe 1000 secondes pour atteindre un état stable.



CARTE-20

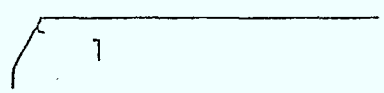
C'est cette carte qui nous permet de changer le taux.



Ici on suppose que les CARTE-1 à CARTE-18 provoquent une simulation avec le taux égal à 1.0.

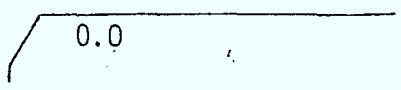
CARTE-21

Etant donné qu'on a déjà simulé le réseau il est moins important d'avoir des résultats intermédiaires; cela dépend des besoins de l'usager. Ici nous allons faire imprimer les résultats seulement au début de la simulation.



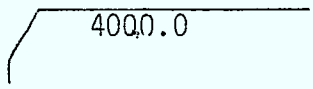
CARTE-22

Comme on vient de le dire l'impression est cédulée au début de la simulation i.e au temps 0.0.

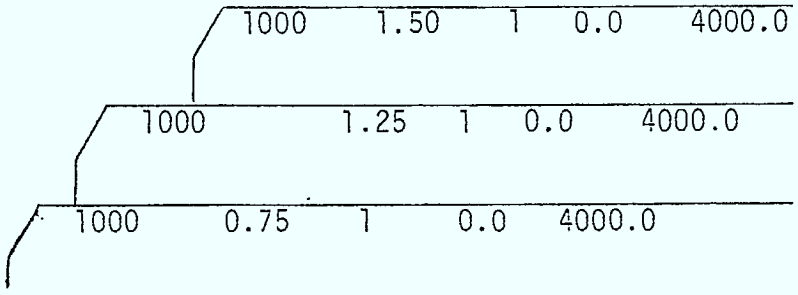


CARTE 23

Cette carte indique le temps total de cette simulation. On fixe sa valeur à 4000.0



Les mêmes cartes sont requises pour chaque taux différent. On peut écrire les données d'une simulation sur une seule carte. Les cartes pour les trois autres simulations vont comme suit:



L'usager aura remarqué que l'on n'a pas utilisé les liens réservés ni le mode retenu. Ce n'est pas sans raison. Les liens réservés sont implantés dans le programme mais les sorties sont en français et ne sont pas soignées. En ce qui concerne le mode retenu il n'est pas très pratique en ce sens qu'il utilise les files d'attente et que rien n'a été implanté pour les initialiser.

La partie qui suit présente les sorties du programme. Le réseau utilisé est le même que dans cette section sauf que les temps d'impression et la durée de la simulation ont été modifiés.

5.3 Les sorties

Cette partie présente les sorties du programme SIMRET. Tout d'abord on présente l'état du réseau au temps zéro i.e après le temps nécessaire pour atteindre l'état stable.

Il y a 3 parties

- "System description et user options" (figure 11)
- "System initial state" (figure 12)
- "Routing table" (figure 13)

chacune décrite dans la figure indiquée. Ces sorties sont fournies automatiquement au début de chaque simulation. Les sorties qu'on va décrire par la suite sont imprimées le nombre de fois inscrit sur la CARTE-14 (ou CARTE-21) et au temps indiqué sur les CARTE-15 (ou CARTE 22) et une fois encore à la fin de la simulation. Elles sont délimitées par

START OF TRACE # # # #

et

END OF TRACE # # # #

Elles contiennent 5 parties:

. Contents of the event list (figure 14)

. Network progress report (figure 15)

. Network performance statistics (figure 16)

. Link performance statistics (figure 17)

. Summary of network performances (figure 18)

TELEPHONE NETWORK SIMULATION

SYSTEM DESCRIPTION AND USER OPTIONS

NUMBER OF NODES	:	10	
NUMBER OF LINKS	:	40	
ROUTING STRATEGY	:	S.D.C.	
TIME UNIT	:	SECONDES	
SWITCHING TIME	:	0.	UNIT(S)
PERCENTAGE OF RETRY CALLS	:	100.00	%
INITIAL RATE OF TRAFFIC	:	1.00	
MEAN RETRY TIME	:	0.	UNIT(S)
TRANSCIENT TIME	:	1200.00	UNIT(S)
SIMULATION TIME	:	3000.0	UNITS

CALL CLASS	PROBABILITY	MEAN HOLDING TIME (UNITS)
	1.00	200.0

Figure 11
III-51

SYSTEM INITIAL STATE

ORIGIN	DEST.	MODE	NB OF BUSY TRUNKS	NB OF IDLE TRUNKS	NB OF CALLS IN QUEUE	INTERARRIVAL TIME
1	2	1	532	14	0	.660
1	3	1	133	2	0	2.740
1	4	1	49	7	0	5.320
1	5	1	89	2	0	4.000
1	6	1	253	12	0	1.320
1	7	1	31	8	0	7.810
1	8	1	593	9	0	.600
1	9	1	205	0	0	1.350
1	10	1	873	243	0	0.
2	1	1	0	0	0	.600
2	3	1	25	10	0	10.890
2	4	1	14	3	0	20.050
2	5	1	42	3	0	8.380
2	6	1	129	2	0	2.990
2	7	1	13	1	0	21.700
2	8	1	307	6	0	1.120
2	9	1	288	4	0	1.290
2	10	1	199	9	0	0.
3	1	1	0	0	0	2.530
3	2	1	0	0	0	10.040
3	4	1	12	10	0	19.550
3	5	1	0	0	0	0.
3	6	1	0	0	0	0.
3	7	1	18	4	0	20.640
3	8	1	61	2	0	6.810
3	9	1	11	2	0	26.430
3	10	1	50	40	0	0.
4	1	1	0	0	0	5.420
4	2	1	0	0	0	17.160
4	3	1	0	0	0	19.200
4	5	1	0	0	0	0.
4	6	1	0	0	0	0.
4	7	1	0	0	0	0.
4	8	1	41	6	0	7.670
4	9	1	6	1	0	44.400
4	10	1	34	30	0	0.
5	1	1	0	0	0	3.220
5	2	1	0	0	0	7.150

Figure 12
III-52

ORIGIN	DEST.	MODE	NB OF BUSY TRUNKS	NB OF IDLE TRUNKS	NB OF CALLS IN QUEUE	INTERARRIVAL TIME
5	3	1	0	0	0	0.
5	4	1	0	0	0	0.
5	6	1	178	9	0	1.820
5	7	1	14	1	0	19.140
5	8	1	21	1	0	12.510
5	9	1	13	1	0	16.890
5	10	1	72	82	0	0.
6	1	1	0	0	0	1.230
6	2	1	0	0	0	2.840
6	4	1	0	0	0	0.
6	5	1	0	0	0	0.
6	7	1	56	1	0	1.920
6	8	1	97	1	0	8.490
6	9	1	41	4	0	3.170
6	10	1	106	246	0	7.850
7	1	1	0	0	0	0.
7	2	1	0	0	0	0.190
7	3	1	0	0	0	21.780
7	4	1	0	0	0	20.310
7	5	1	0	0	0	0.
7	6	1	16	0	0	30.220
7	8	1	2	0	0	7.590
7	9	1	33	17	0	19.310
7	10	1	0	0	0	82.020
8	1	1	0	0	0	0.
8	2	1	0	0	0	.570
8	3	1	0	0	0	1.050
8	4	1	0	0	0	5.530
8	5	1	0	0	0	8.770
8	6	1	0	0	0	16.030
8	7	1	0	0	0	2.960
8	9	1	178	0	0	16.750
8	10	1	197	11	0	1.850
9	1	1	0	0	0	0.
9	2	1	0	0	0	2.170
9	3	1	0	0	0	1.280
9	4	1	0	0	0	31.110
9	5	1	0	0	0	52.630
9	6	1	0	0	0	19.800
9	7	1	0	0	0	7.910
9	8	1	0	0	0	0.
10	1	1	156	48	0	1.950
			0	0	0	0.
						0.

Figure 12 (suite)

ORIGIN	DEST.	MODE	NB OF BUSY TRUNKS	NB OF IDLE TRUNKS	NB OF CALLS IN QUEUE	INTERARRIVAL TIME
10	2	1	0	0	0	0.
100	2	1	0	0	0	0.
1100	2	1	0	0	0	0.
11100	2	1	0	0	0	0.
111000	2	1	0	0	0	0.
1110000	2	1	0	0	0	0.
11100	2	1	0	0	0	0.
1110	2	1	0	0	0	0.

Figure 12 (fin)

ROUTING TABLE

ORIGIN	DEST.	CHOIX NO. 1	CHOIX NO. 2
1	2	2	10
1	3	3	10
1	4	4	10
1	5	5	10
1	6	6	10
1	7	7	10
1	8	8	10
1	9	9	10
1	10	10	0
2	1	1	10
2	3	3	10
2	4	4	10
2	5	5	10
2	6	6	10
2	7	7	10
2	8	8	10
2	9	9	10
2	10	10	0
3	1	1	10
3	2	2	10
3	4	4	10
3	5	10	0
3	6	10	0
3	7	7	10
3	8	8	10
3	9	9	10
3	10	10	0
4	1	1	10
4	2	2	10
4	3	3	10
4	5	10	0
4	6	10	0
4	7	7	10
4	8	8	10
4	9	9	10
4	10	10	0
5	1	1	10

Figure 13
III-55

Figure 13 (fin)

Chaque partie va être discutée dans les paragraphes suivants:

a) Contents of the event list

Comme pour les 4 autres parties, au début de chaque page on retrouve le temps de simulation et le taux de trafic à ce temps.

En plus, cette partie nous instruit sur le nombre d'événements qui restent dans la liste (figure 14). Tous les événements y sont présents sauf END.OF.SIMULATION. Le nombre d'événements ARRIVAL est toujours le même tout au long de la simulation. Le nombre de "ROUTING" est égal à zéro dans l'exemple parce que le temps de commutation est nul.

Cette partie est utile pour vérifier si on a vraiment atteint un état stable. Lorsque le nombre de "ENDING" est presque constant sur une longue période on peut dire qu'on a atteint l'état stable du réseau.

b) Network progress report

Cette section donne à l'usager, pour chaque triplet origine-destination-mode, le nombre d'appels qui ont été essayés, le nombre qui ont été acheminés, le nombre qui ont été perdus, ceux qui sont terminés, le nombre qui sont dans les files d'attente (pour MODE =2) et le nombre de tronçons occupés (figure 15).

START OF TRACE

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

CONTENTS OF THE EVENT LIST

##### NUMBER OF ARRIVAL EVENTS :	61
##### NUMBER OF ROUTING EVENTS :	0
##### NUMBER OF ENDING EVENTS :	4047
##### NUMBER OF RECALL EVENTS :	0
##### NUMBER OF TRACES EVENTS :	0
##### NUMBER OF CHANGE (OF RATE) EVENTS :	0

TOTAL NUMBER OF EVENTS IN THE LIST : 4109

Figure 14

NETWORK PROGRESS REPORT.

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

ORIGIN	DEST	MODE	NB. CALLS	ROUTED	LOST	FINISHED	NB. IN QUEUE	NB. OF BUSY TRUNKS
1	2	1	4529	4429	100	4424	0	537
1	3	1	1121	1121	0	1115	0	134
1	4	1	553	552	1	534	0	54
1	5	1	736	736	0	733	0	85
1	6	1	2280	2280	0	2300	0	261
1	7	1	397	396	1	307	0	36
1	8	1	5163	4904	259	4800	0	600
1	9	1	2301	2301	0	2324	0	204
1	10	1	0	0	0	0	0	354
2	1	1	4951	4848	103	4832	0	0
2	3	1	276	268	8	260	0	33
2	4	1	157	150	7	156	0	16
2	5	1	365	362	3	360	0	45
2	6	1	1054	1044	10	1047	0	118
2	7	1	118	115	3	117	0	12
2	8	1	2606	2458	148	2471	0	310
2	9	1	2269	2244	25	2268	0	275
2	10	1	0	0	0	0	0	187
3	1	1	1166	1166	0	1193	0	0
3	2	1	293	293	0	208	0	0
3	4	1	183	183	0	170	0	22
3	5	1	0	0	0	0	0	0
3	6	1	0	0	0	0	0	0
3	7	1	128	128	0	126	0	16
3	8	1	482	472	10	480	0	59
3	9	1	109	109	0	109	0	8
3	10	1	0	0	0	0	0	30
4	1	1	509	509	0	511	0	0
4	2	1	179	177	2	172	0	0
4	3	1	172	172	0	169	0	0
4	5	1	0	0	0	0	0	0
4	6	1	0	0	0	0	0	0
4	7	1	0	0	0	0	0	0
4	8	1	405	386	19	382	0	44
4	9	1	76	76	0	77	0	7

Figure 15

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

ORIGIN	DEST	MODE	NB. CALLS	ROUTED	LOST	FINISHED	NB. IN QUEUE	NB. OF BUSY TRUNKS
4	10	1	0	0	0	0	0	49
5	1	1	962	962	0	947	0	0
5	2	1	423	419	4	423	0	0
5	3	1	0	0	0	0	0	0
5	4	1	0	0	0	0	0	0
5	6	1	1702	1702	0	1677	0	175
5	7	1	150	150	0	156	0	9
5	8	1	246	234	12	237	0	16
5	9	1	173	173	0	180	0	13
5	10	1	0	0	0	0	0	83
6	1	1	2468	2468	0	2447	0	0
6	2	1	1034	1018	16	1022	0	0
6	3	1	0	0	0	0	0	0
6	4	1	0	0	0	0	0	0
6	5	1	1537	1537	0	1538	0	0
6	7	1	344	341	3	326	0	45
6	8	1	883	850	33	858	0	94
6	9	1	393	393	0	363	0	43
6	10	1	0	0	0	0	0	135
7	1	1	346	345	1	347	0	0
7	2	1	138	136	2	137	0	0
7	3	1	137	137	0	146	0	0
7	4	1	0	0	0	0	0	0
7	5	1	104	104	0	108	0	0
7	6	1	371	369	2	371	0	0
7	8	1	147	142	5	142	0	16
7	9	1	41	41	0	42	0	2
7	10	1	0	0	0	0	0	30
8	1	1	5339	5075	264	5076	0	0
8	2	1	3009	2842	167	2837	0	0
8	3	1	525	511	14	502	0	0
8	6	1	331	318	13	316	0	0
8	5	1	200	190	10	187	0	0
8	6	1	1033	984	49	984	0	0
8	7	1	179	170	9	164	0	0
8	9	1	1658	1603	55	1590	0	176
8	10	1	0	0	0	0	0	207
9	1	1	1362	1362	0	1395	0	0

Figure 15 (suite)
III-60

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

ORIGIN	DEST	MODE	NB. CALLS	ROUTED	LOST	FINISHED	NB. IN QUEUE	NB. OF BUSY TRUNKS
9	2	1	2420	2385	35	2390	0	0
9	3	1	89	89	0	97	0	0
9	4	1	57	57	0	57	0	0
9	5	1	156	156	0	160	0	0
9	6	1	368	368	0	370	0	0
9	7	1	0	0	0	0	0	0
9	8	1	1485	1446	39	1468	0	0
9	10	1	0	0	0	0	0	85
10	1	1	0	0	0	0	0	0
10	2	1	0	0	0	0	0	0
10	3	1	0	0	0	0	0	0
10	4	1	0	0	0	0	0	0
10	5	1	0	0	0	0	0	0
10	6	1	0	0	0	0	0	0
10	7	1	0	0	0	0	0	0
10	8	1	0	0	0	0	0	0
10	9	1	0	0	0	0	0	0

Figure 15 (fin)
III-61

La dernière colonne sert à initialiser un réseau. Au lieu de reprendre la simulation à partir du départ, grâce à cette information, on peut continuer où on est rendu. On se sert alors des CARTE-17 et CARTE-18.

c) Network performance statistics

Comme dans la section précédente, on fournit pour chaque triplet origine-destination-mode les quatre informations suivantes (figure 16):

- la charge de trafic offert en CCS;
- la charge de trafic porté en CCS;
- le temps d'attente moyen dans une file;
- et le "grade of service" pour chaque origine-destination.

En ce qui concerne le temps d'attente moyen on remarque que cette valeur n'est pas nulle même si on n'utilise pas le mode retenu. En fait cette valeur devrait être zéro mais varie sensiblement à cause de l'énoncé ACCUMULATE en SIMSCRIPT. Il ne faut donc pas en tenir compte dans cet exemple.

Le "grade of service" indique la probabilité qu'un usager ne soit pas capable d'obtenir la communication lorsqu'il en a besoin.

NETWORK PERFORMANCE STATISTICS

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

ORIGIN	DEST	MODE	OFFERED LOAD (CCS)	CARRIED LOAD (CCS)	AVERAGE WAITING TIME (UNITS)	D.D.G.D.S.
1	2	1	11054.0664	10820.4258	.01	.0221
1	3	1	2847.1929	2847.1924	.00	0.
1	4	1	1350.9377	1350.3755	.04	.0018
1	5	1	1743.3401	1743.3399	.00	0.
1	6	1	5570.3867	5570.3828	.00	0.
1	7	1	884.4255	884.2300	.00	.0025
1	8	1	12656.2148	12042.6525	.00	.0502
1	9	1	5660.6875	5660.6875	.01	0.
1	10	1	0.	0.	0.	0.
2	1	1	11791.1075	11530.2969	.01	.0206
2	3	1	644.2842	627.7375	.01	.0290
2	4	1	360.7791	344.2520	.01	.0446
2	5	1	858.8682	857.0461	.00	.0082
2	6	1	2460.3452	2435.9839	.01	.0095
2	7	1	301.8511	297.1030	.03	.0854
2	8	1	6199.1094	5842.3633	.01	.0568
2	9	1	5381.2266	5300.2383	.01	.0110
2	10	1	0.	0.	0.	0.
3	1	1	2608.5139	2606.5134	.01	0.
3	2	1	769.8713	769.8713	.02	0.
3	4	1	405.7422	405.7419	.03	0.
3	5	1	0.	0.	0.	0.
3	6	1	0.	0.	0.	0.
3	7	1	0.	0.	0.	0.
3	8	1	335.3567	335.3564	.01	.0207
3	9	1	1122.0933	1099.0691	.02	0.
3	10	1	289.0798	289.0798	.06	0.
4	1	1	0.	0.	0.	0.
4	2	1	1291.0611	1291.0606	.02	.0112
4	3	1	421.3745	419.9934	.04	0.
4	5	1	445.4578	445.4575	.01	0.
4	6	1	0.	0.	0.	0.
4	7	1	0.	0.	0.	0.
4	8	1	0.	0.	0.	0.
4	9	1	9684.4399	922.6116	.01	.0469

Figure 16
III-63

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

ORIGIN	DEST	MODE	OFFERED LOAD (CCS)	CARRIED LOAD (CCS)	AVERAGE WAITING TIME (UNITS)	D.D.G.D.S.
4	9	1	179.0515	179.0515	.01	0.
4	10	1	0.	0.	0.	0.
5	1	1	2216.2732	2216.2727	.01	0.
5	2	1	1007.8542	998.3276	.01	.0095
5	3	1	0.	0.	0.	0.
5	4	1	0.	0.	0.	0.
5	6	1	4020.9880	4020.9873	-.01	0.
5	7	1	313.7771	313.7771	.00	0.
5	8	1	572.7681	559.9238	-.01	.0488
5	9	1	409.9807	409.9604	.01	0.
5	10	1	0.	0.	0.	0.
6	1	1	5979.2344	5979.2344	.00	0.
6	2	1	2426.0203	2407.8115	.01	.0155
6	3	1	0.	0.	0.	0.
6	4	1	0.	0.	0.	0.
6	5	1	3672.7803	3672.7795	-.00	0.
6	7	1	826.2336	822.0440	-.02	.0087
6	8	1	2047.2480	1986.5613	.02	.0374
6	9	1	962.7708	962.7705	-.04	0.
6	10	1	0.	0.	0.	0.
7	1	1	826.3555	811.0083	.00	.0029
7	2	1	351.5130	347.4617	.01	.0145
7	3	1	364.3398	364.3396	-.03	0.
7	4	1	0.	0.	0.	0.
7	5	1	232.7437	232.7436	-.09	0.
7	6	1	900.8972	895.7922	.00	.0054
7	8	1	361.5647	338.9755	.03	.0340
7	9	1	95.8313	95.8313	-.07	0.
7	10	1	0.	0.	0.	0.
8	1	1	13117.3125	12517.9023	-.00	.0494
8	2	1	6970.5020	6624.6211	-.01	.0555
8	3	1	1262.0913	1213.0291	-.04	.0267
8	4	1	901.9812	866.9771	-.02	.0393
8	5	1	542.3408	518.0156	.01	.0500
8	6	1	2448.3923	2343.4790	-.03	.0474
8	7	1	354.8135	342.2844	-.01	.0503
8	9	1	3987.0420	3851.9421	.01	.0338

Figure 16 (suite)
III-64

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

ORIGIN	DEST	MODE	OFFERED LOAD (CCS)	CARRIED LOAD (CCS)	AVERAGE WAITING TIME (UNITS)	D.D.G.O.S.
8	10	1	0.	0.	0.	0.
9	1	1	3372.6736	3372.6729	.01	0.
9	2	1	5831.5859	5749.6797	.00	.0145
9	3	1	176.6969	176.6969	.06	0.
9	4	1	156.2621	156.2621	.04	0.
9	5	1	374.1394	374.1394	.05	0.
9	6	1	863.7415	863.7412	.02	0.
9	7	1	0.	0.	0.	0.
9	8	1	3571.0125	3502.3611	.01	.0263
9	10	1	0.	0.	0.	0.
10	1	1	0.	0.	0.	0.
10	2	1	0.	0.	0.	0.
10	3	1	0.	0.	0.	0.
10	4	1	0.	0.	0.	0.
10	5	1	0.	0.	0.	0.
10	6	1	0.	0.	0.	0.
10	7	1	0.	0.	0.	0.
10	8	1	0.	0.	0.	0.
10	9	1	0.	0.	0.	0.

Figure 16 (fin)
III-65

d) Link performance statistics

Par "link" on entend un triplet origine-destination-mode avec un nombre de tronçons positifs. On donne pour chaque lien (figure 17) le nombre de tronçons, le nombre de tronçons disponibles en moyenne, le pourcentage d'utilisation des tronçons, - le trafic porté et la probabilité de blocage (correspond au G.O.S.)

e) Summary of Network performances

En dernier lieu on imprime un bilan qui indique le nombre d'appels total, ceux qui ont été acheminés, etc.

On fournit aussi la durée moyenne des appels, le temps d'attente moyen dans les files, le "grade of service" du réseau et les trafics offert et porté par le réseau (figure 18).

Il peut y avoir plusieurs simulations les unes à la suite des autres. Chaque simulation contient au moins les parties indiquées. La dernière simulation se termine par le message suivant:

FIN NORMALE DE LA SIMULATION # # # #

LINK PERFORMANCE STATISTICS

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

FROM NODE	TO NODE	MODE	NB. OF TRUNKS (INITIAL)	AVERAGE OF AVAILABLE TRUNKS	PERCENTAGE OF TRUNKS UTILISATION	CARRIED LOAD	LINK BLOCKING PROBABILITY
1	2	1	546	6.15	98.87	19481.27	.1349
1	3	1	135	4.70	96.52	4720.75	.1359
1	4	1	56	3.07	94.52	1933.47	.2434
1	5	1	91	3.39	96.27	3100.64	.1937
1	6	1	265	3.91	98.53	9430.41	.1656
1	7	1	39	2.70	93.08	1315.64	.2179
1	8	1	608	4.94	99.18	21578.87	.1442
1	9	1	205	3.37	98.36	7252.03	.2007
1	10	1	616	267.21	56.62	12521.99	0.
2	3	1	35	2.90	91.72	1166.90	.1670
2	4	1	17	1.69	90.05	537.48	.3228
2	5	1	45	3.25	92.78	1517.97	.1664
2	6	1	131	5.49	95.81	4440.64	.0799
2	7	1	14	1.44	89.72	439.44	.3386
2	8	1	313	4.60	98.53	11134.70	.1489
2	9	1	292	11.40	96.09	10007.31	.0090
2	10	1	208	20.47	90.16	8663.54	.1212
3	4	1	22	2.84	87.10	723.77	.1351
3	7	1	22	4.60	79.00	633.75	.1147
3	8	1	63	4.22	93.30	2099.07	.1010
3	9	1	13	2.62	79.86	366.35	.1787
3	10	1	90	45.28	49.68	1551.50	0.
4	8	1	47	4.34	90.76	1552.36	.1202
4	9	1	7	1.30	81.46	226.90	.3290
4	10	1	64	26.43	58.70	1411.43	.0013
5	6	1	187	5.58	97.01	6525.35	.1313
5	7	1	15	3.17	78.86	416.26	.1911
5	8	1	21	1.48	92.94	716.94	.3051
5	9	1	14	1.17	91.67	449.02	.3763
5	10	1	154	65.82	57.26	3191.13	0.
6	7	1	46	4.77	89.62	1493.50	.1319
6	8	1	98	2.65	97.29	3304.55	.2450
6	9	1	45	2.60	93.77	1525.10	.1553
6	10	1	352	209.44	40.50	8162.03	0.

Figure 17

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.80

FROM NODE	TO NODE	MODE	NB. OF TRUNKS (INITIAL)	AVERAGE OF AVAILABLE TRUNKS	PERCENTAGE OF TRUNKS UTILISATION	CARRIED LOAD	LINK BLOCKING PROBABILITY
7	8	1	16	1.80	88.74	496.93	.3068
7	9	1	2	.50	75.16	57.68	.4928
7	10	1	50	15.21	69.58	1227.48	.0708
8	9	1	186	4.78	97.43	6547.03	.1402
8	10	1	208	9.41	95.48	7062.23	.2326
9	10	1	204	70.96	65.21	4524.33	0.

Figure 17 (fin)

SUMMARY OF NETWORK PERFORMANCES

SIMULATION TIME : 3000.0 UNIT(S)
RATE OF TRAFFIC : 1.00

TOTAL SUM OF CALLS	:	60466
ROUTED CALLS	:	60978
STORED CALLS	:	0
RE-ROUTED CALLS	:	0
LOST CALLS	:	1432
RETRIED CALLS	:	0
RESERVED CALLS	:	0
FINISHED CALLS	:	60987

MEAN HOLDING TIME	:	200.80
MEAN WAITING TIME	:	-.00
GRADE OF SERVICE (G_0:S_0)	:	.022946
TOTAL OFFERED LOAD (GCS)	:	180203.81
TOTAL CARRIED LOAD (CCS)	:	146926.44

END OF TRACE

Figure 18

6. Conclusion

Pour conclure nous allons donner les résultats de deux expériences que nous avons fait avec le simulateur SIMRET. Chaque expérience a été faite avec un réseau différent ayant ses caractéristiques propres.

Le premier, qui a été présenté en exemple dans la section précédente est basé sur des informations de Bell Canada. Il contient 10 noeuds et 26 liens et représente 9 bureaux de classe 5 et un bureau de classe 4. La table d'acheminement est très simple mais le trafic sur le réseau très élevé.

Le second, qui n'est pas illustré ici, est le réseau AUTOVON qui contient 11 noeuds et 26 liens. Dans ce cas, la table d'acheminement permet des arbres d'acheminement assez complexes. Cependant le trafic est très faible comparé au premier.

Le réseau Bell a été simulé pour divers taux de trafic variant de 1.0 à 2.0. Chaque simulation a duré 2000 secondes. Le nombre d'appels générés est environ de 43500 et le trafic "porté" par le réseau de 156000 CCS. La figure 19 montre pour chaque taux calculé le trafic offert et le trafic porté par le réseau. On s'aperçoit que le réseau semble saturé à 163000 CCS.

La figure 20 illustre pour les mêmes taux le "grade of service" du réseau calculé à l'aide du simulateur SIMRET et à l'aide du modèle analytique ETARET.

Graphique du trafic offert et porté en fonction du tauc pour le réseau BELL

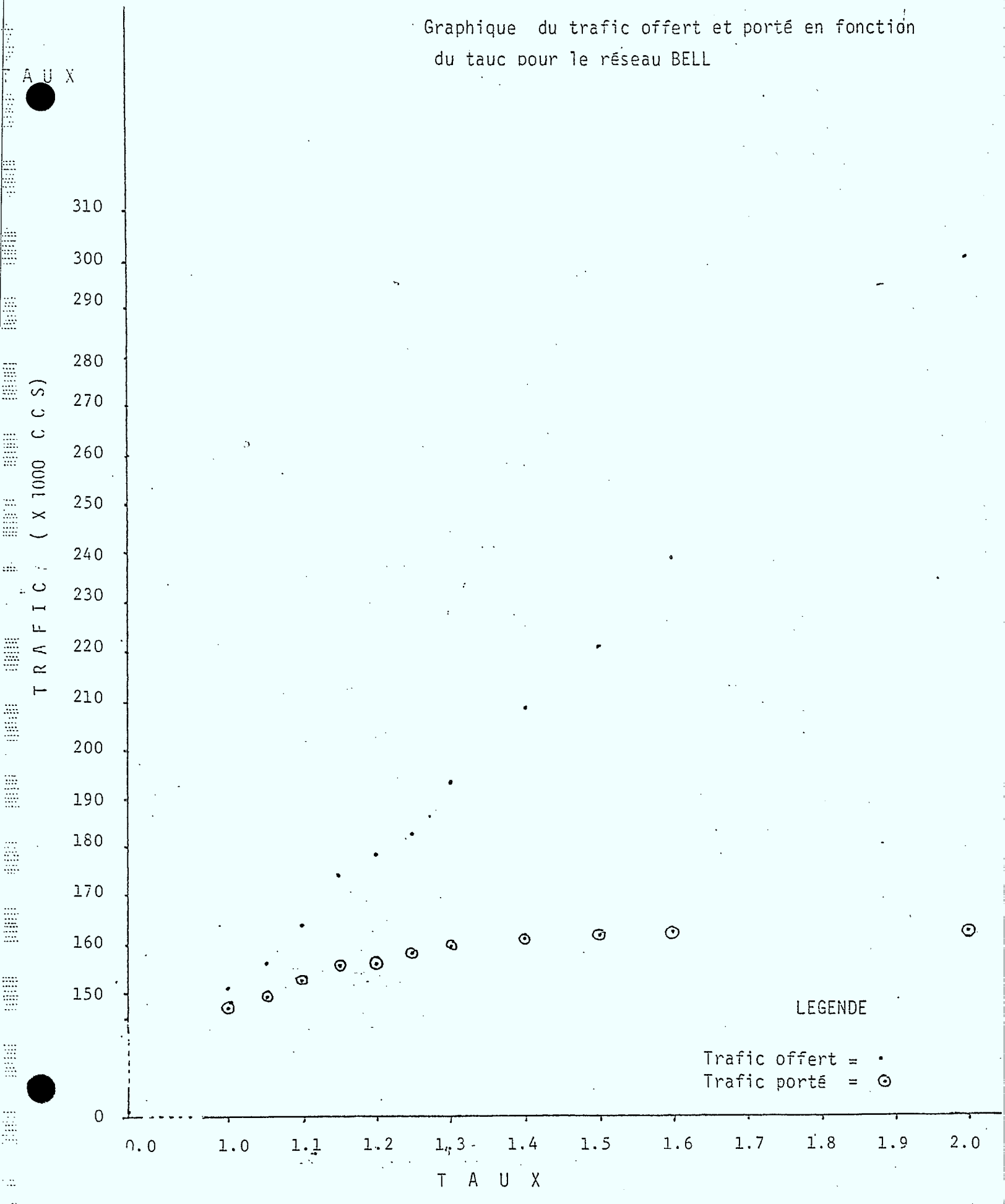


Figure 19

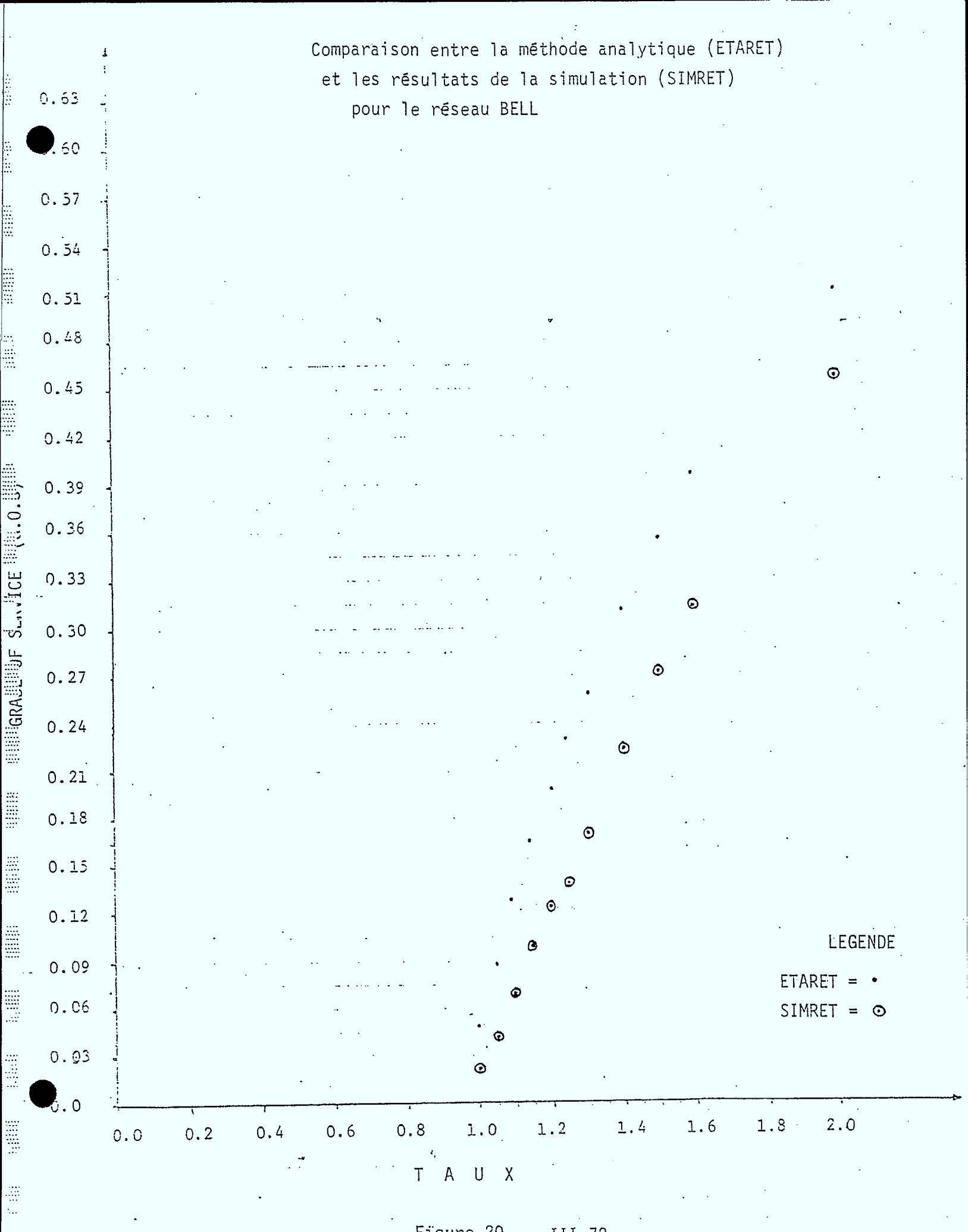


Figure 20 III-72

Les figures 21 et 22 donnent les mêmes imformations mais pour le réseau AUTOVON. Le taux du trafic varie entre 0.5 et 2.0, le temps d'une simulation est de 5000 secondes et le nombre d'appels n'est que de 4000 environ. Le trafic en CCS est égal à 7000.

On note aussi la similitude des résultats obtenus par le simulateur (SIMRET) et le modèle analytique (ETARET). Ceux-ci sont presque identiques même avec deux réseaux très différents.

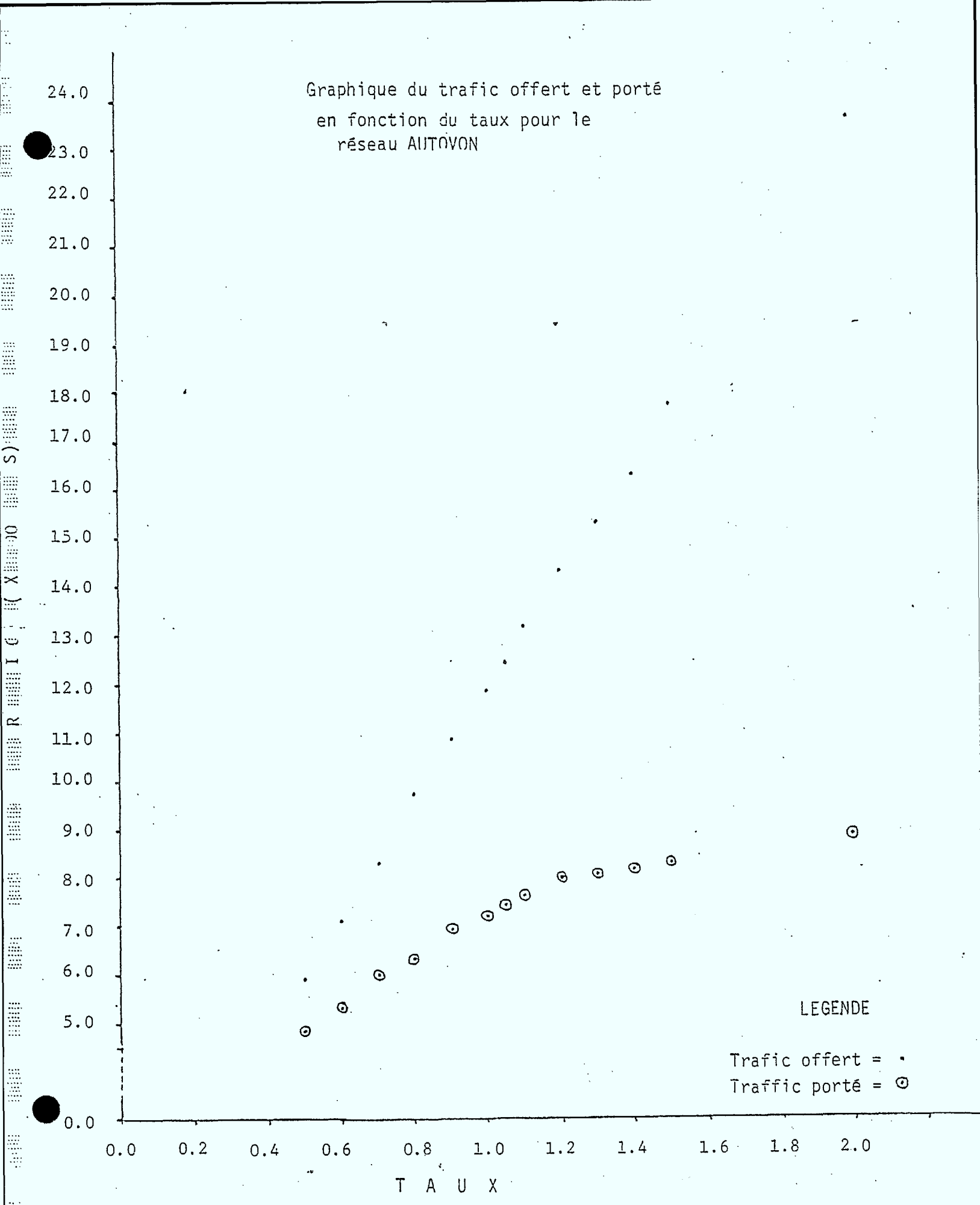

Graphique du trafic offert et porté
en fonction du taux pour le
réseau AUTOVON
24.0
23.0
22.0
21.0
20.0
19.0
18.0
17.0
16.0
15.0
14.0
13.0
12.0
11.0
10.0
9.0
8.0
7.0
6.0
5.0
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
T A U X
LEGENDE
Trafic offert = •
Traffic porté = ⊙

Figure 21

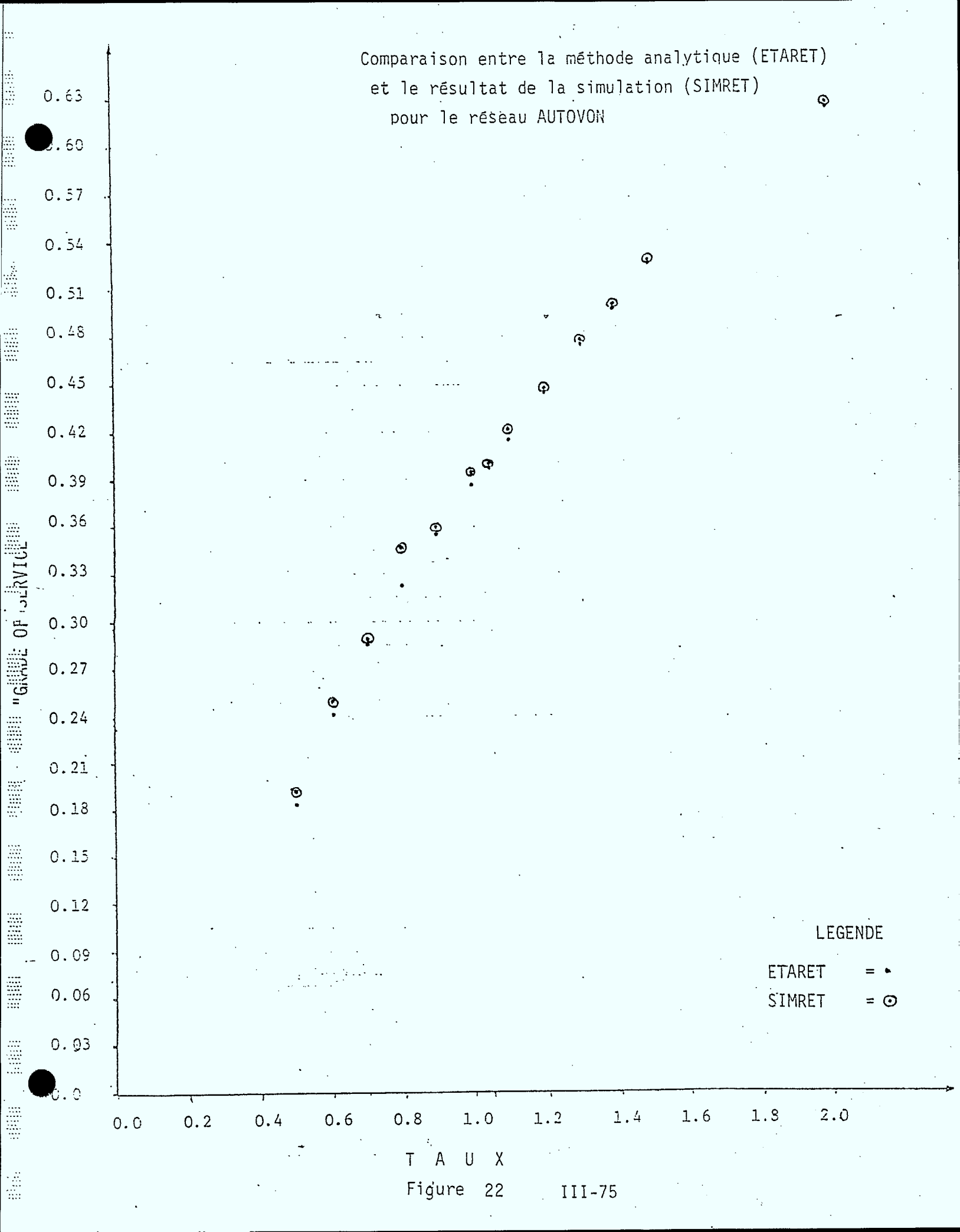

Comparaison entre la méthode analytique (ETARET)
et le résultat de la simulation (SIMRET)
pour le réseau AUTOVON
"GRADE OF SERVICE"
0.63
0.60
0.57
0.54
0.51
0.48
0.45
0.42
0.39
0.36
0.33
0.30
0.27
0.24
0.21
0.18
0.15
0.12
0.09
0.06
0.03
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
T A U X
LEGENDE
ETARET = •
SIMRET = ⊙

Figure 22

7. Liste du programme SIMRET

Cette section contient la liste du programme SIMRET écrit dans le langage SIMSCRIPT II.5

```
1
2     PREAMBLE
3
4       NORMALLY MODE IS INTEGER AND DIMENSION IS 0
5
6       THE SYSTEM HAS A CLASSE RANDOM STEP VARIABLE
7       DEFINE CLASSE AS A REAL, STREAM 5 VARIABLE
8
9       DEFINE DIRECT TO MEAN 1
10      DEFINE STORE.AND.FORWARD TO MEAN 2
11
12      DEFINE NB.NODE, NB.LK, NB.LINKS, NB.CHOIX, I.CALL AS VARIABLES
13      DEFINE RT  AS A 3-DIMENSIONAL ARRAY
14      DEFINE TAUX,TAUX,RREC,TIME AS REAL VARIABLES
15      DEFINE PC.LOST, QTIME, REATTEMPS.TIME, STEADY.STATE,
16             STOP.TIME, SWITCHED.TIME AND TIME.TR AS REAL VARIABLES
17      DEFINE UNITE AS ALPHA, 1-DIM ARRAY
18      DEFINE NB.CALL, ROUTED, STORED, LOST, REESSAI, RE.ROUTED,
19             RESERVED, FINISH, NB.TRK AND RES.TRK AS 1-DIM ARRAYS
20      DEFINE NB.CARRIED.LK.BLOCKED.LK.CALLI AS REAL,
21             1-DIM ARRAYS
22      DEFINE TRANS.TIME AS REAL VARIABLE
23      DEFINE HOLDING.TIME AS A REAL VARIABLE
24
25
26
27    TEMPORARY ENTITIES
28
29      EVERY PATH OWNS A PATH.QU
30      DEFINE PATH.QU AS A FIFO SET WITHOUT M AND  R ATTRIBUTES AND
31             WITHOUT RF,FB,FA,RL AND RS ROUTINES
32      EVERY LINK HAS A VALUE AND MAY BELONGS TO THE PATH.QU
33      EVERY CALL HAS A POSITION, AN ATIME
34             AND MAY BELONGS TO THE LINK.QU
35
36
37
38    PERMANENT ENTITIES
39
40      EVERY P.LINK HAS A L.SOURCE, A L.DEST, A L.MODE, A NB.TRUNKS,
41                   A D.CALL, A DUREE.CALL, A QD.QTIME, A I.A.TIME,
42                   A LK.LOAD
43                   AND OWNS A LINK.QU
44      DEFINE I.A.TIME, D.CALL, DUREE.CALL,QD.QTIME, LK.LOAD AS REAL
45             VARIABLES
46      DEFINE LINK.QU AS A SET RANKED BY POSITION WITHOUT L.P AND M
47             ATTRIBUTES AND WITHOUT FF, FL, FB, FA, RL AND RS ROUTINES
48
49      EVERY RES.LINK HAS A R.SOURCE, A R.DEST, A R.MODE
50             AND A  R.NB.TRK
51
52
53
```

```
   EVENT NOTICES INCLUDE INITIAL.STATE, TRACES AND END.OF.SIMULATION

      EVERY ARRIVAL HAS A NEW.CALL
      EVERY RECALL HAS A RE.CALL
      EVERY ROUTING HAS A ROUTE.CALL
      EVERY ENDING HAS A DIS.CALL, A DIS.PATH AND A PARAMETER
      EVERY CHANGE HAS A CHANGE.TAUX
      DEFINE CHANGE.TAUX AS REAL VARIABLE



      TALLY WQ AS THE MEAN OF QTIME
      ACCUMULATE LQ AS THE MEAN OF N.LINK.QU
      TALLY AVG.DUREE AS THE AVERAGE OF DUREE.CALL
      ACCUMULATE AVG.TRK AS THE AVERAGE OF NB.TRUNKS
      TALLY AV.D.CALL AS THE AVERAGE OF D.CALL
      ACCUMULATE AV.OD.QT AS THE MEAN OF OD.QTIME
      TALLY AV.LK.LOAD AS THE MEAN OF LK.LOAD
      TALLY AV.HOLDING.T AS THE MEAN OF HOLDING.TIME

   END
```

```
!!    PROGRAMME DE SIMULATION D'UN RESEAU TELEPHONIQUE.
!!                AUTEUR : MARIO LAVOIE
!!                       ETE  1980
!!
!!        PROGRAMME PRINCIPAL
!!
  MAIN
  DEFINE T AS A REAL VARIABLE
  RESERVE UNITE(*) AS 4
  LET ZERO = 0
  LET QTIME = 0.0
!!     LECTURE DES PARAMETRES
  READ NB.NODE, NB.LK
  START NEW CARD
  FOR I = 1 TO 4 READ UNITE(I) AS A 4
  READ SWITCHED.TIME
  READ REATTEMPS.TIME
  READ PC.LOST
  READ CLASSE
  LET N = NB.NODE
  LET NB.LINKS = N * N - N
  READ N.P.LINK
  RESERVE NB.CALL(*), ROUTED(*), STORED(*), LOST(*), REESSAI(*),
  RE.ROUTED(*), RESERVED(*), FINISH(*), NB.TRK(*)  AS N.P.LINK
  RESERVE NB.CARRIED(*), LK.BLOCKED(*), LK.CALL(*)
          AS N.P.LINK
!!
!!      CREATION DES ENTITES PERMANENTES.
!!
  CREATE ALL P.LINK
  FOR EACH P.LINK DO
      READ S.D.M.NB.T
      LET L.SOURCE (P.LINK) = S
```

```
54          LET L.DEST    (P.LINK) = D
55          LET L.MODE    (P.LINK) = M
56          LET NB.TRUNKS (P.LINK) = NB
57          LET NB.TRK    (P.LINK) = NB
58          LET I.A.TIME  (P.LINK) = T
59          LET LK.LOAD   (P.LINK) = 0.0
60          LET DD.QTIME  (P.LINK) = 0.0
61
62      LOOP
63
64  !!
65  !!        RESERVATIONS DES LIENS
66  !!
67
68      READ N.RES.LINK
69
70      IF N.RES.LINK # ZERO
71          GO TO !RT!
72      ELSE
73  !!      CONTINUE
74
75      RESERVE RES.TRK(*) AS N.RES.LINK
76      CREATE ALL RES.LINK
77
78      FOR EACH RES.LINK DO
79
80          READ S,D,M,NB
81          LET SO = MIN.#(S,D)
82          LET DE = MAX.#(S,D)
83
84          FOR EACH P.LINK WITH
85              L.SOURCE(P.LINK) # SO AND
86              L.DEST  (P.LINK) # DE AND
87              L.MODE  (P.LINK) # M
88              FIND L.POS # THE FIRST P.LINK
89
90          IF NONE
91              LET L.POS # 0
92
93          ALWAYS
94
95          IF L.POS GT ZERO
96              LET NB.TRUNKS(L.POS) = NB.TRUNKS(L.POS) - NB
97              LET NB.TRK(L.POS) = NB.TRUNKS(L.POS)
98
99          ALWAYS
100
101         LET R.SOURCE(RES.LINK) = S
102         LET R.DEST  (RES.LINK) = D
103         LET R.MODE  (RES.LINK) = M
104         LET R.NB.TRK(RES.LINK) = NB
105         LET RES.TRK (RES.LINK) = NB
106
```

```
107
108
109     LOOP
110
111   !!
112   !!        FORMATION DE LA ROUTING TABLE (RT)
113   !!
114
115   !RT!
116
117
118     READ NB.CHOIX
119     RESERVE RT(*,*,*)  AS N BY N BY NB.CHOIX
120
121     FOR I = 1 TO NB.LINKS DO
122
123           READ S,D
124           FOR J = 1 TO NB.CHOIX  READ RT(S,D,J)
125
126     LOOP
127
128
129
130   !!
131   !!      DEFINITION DU TEMPS REQUIS POUR ATTEINDRE  UN ETAT
132   !!      STABLE DU SYSTEME.
133   !!
134
135     READ STEADY.STATE
136     LET TRANS.TIME = STEADY.STATE
137
138     SCHEDULE AN INITIAL.STATE IN STEADY.STATE UNITS
139
140
141
142   !!
143   !!              TAUX DE CHANGEMENT LINEAIRE
144   !!
145     LET TIME = 0.0
146     LET TAUX = 1.0
147     LET TAUX.PREC = 1.0
148
149     READ NB.CARDS
150     IF  NB.CARDS EQ ZERO
151         GO TO !TR!
152     ELSE
153   !!      CONTINUE
154
155     FOR I = 1 TO NB.CARDS DO
156         READ TIME, TAUX
157         IF  TIME LT 0.0 OR TAUX LE 0.0
158             GO TO !NO.MODIF!
159         ELSE
```

```
60          SCHEDULE A CHANGE GIVEN TAUX IN (TIME+STEADY.STATE) UNITS
61      'NO.MODIF'
62    LOOP
63
64
65
66  !!
67  !!      IMPRESSION DU CONTENU DES LISTES D'EVENEMENTS
68  !!
69
70  'TR'
71
72
73    READ NB.TRACE
74
75    IF NB.TRACE = ZERO
76        GO TO 'STOP'
77    ELSE
78  !!      CONTINUE
79
80
81    FOR I = 1 TO NB.TRACE DO
82        READ TIME.TR
83        SCHEDULE A TRACES IN (TIME.TR + STEADY.STATE) UNITS
84    LOOP
85
86
87
88  !!
89  !!      FIN DE LA SIMULATION
90  !!
91
92  'STOP'
93
94
95    READ STOP.TIME
96
97    SCHEDULE A TRACES IN (STOP.TIME + STEADY.STATE) UNITS
98    SCHEDULE AN END.OF.SIMULATION IN (STOP.TIME + STEADY.STATE) UNITS
99
00
01
02  !!
03  !!      INITIALISATION DE LA SIMULATION
04  !!
05
06    PERFORM INIT
07    RELEASE INIT
08
09
10
11    START SIMULATION
12
13
14
15    END
```

```
1
2     ROUTINE  INIT
3
4   !!      ROUTINE QUI INITIALISE LES LIGNES.  ELLE DONNE  L'ETAT
5   !!      STABLE DU SYSTEME.
6
7
8     DEFINE T AS A REAL VARIABLE
9
10    LET PERM = ZERO
11    LET DIFF = ZERO
12    READ NB.INIT
13
14    IF  NB.INIT = ZERO
15        GO TO 'NO.INIT'
16    ELSE
17        FOR I = 1 TO NB.INIT DO
18            READ SS.DD.M.NB
19            LET S = MIN.F(SS,DD)
20            LET D = MAX.F(SS,DD)
21            FOR EACH P.LINK WITH
22                L.SOURCE (P.LINK) = S AND
23                L.DEST   (P.LINK) = D AND
24                L.MODE   (P.LINK) = M
25                FIND POS = THE FIRST P.LINK
26            IF  NONE
27                GO TO 'LOOP'
28            ELSE
29  !!            CONTINUE
30
31            IF  NB.TRUNKS(POS) IS LT NB
32                LET DIFF = NB - NB.TRUNKS(POS)
33                LET NB = NB.TRUNKS(POS)
34            ALWAYS
35
36            IF  NB LE ZERO
37                GO TO 'DIFF'
38            ELSE
39                LET NB.TRUNKS(POS) = NB.TRUNKS(POS) - NB
40                FOR J = 1 TO NB DO
41                    CREATE A CALL
42                    LET POSITION(CALL) = POS
43                    CREATE A PATH
44                    CREATE A LINK
45                    LET VALUE(LINK) = POS
46                    FILE LINK IN PATH.QU(PATH)
47                    LET D.CALL (POS) = EXPONENTIAL.F(CLASSE,7)
48                    LET DUREE.CALL(POS) = D.CALL(POS)
49                    LET HOLDING.TIME = DUREE.CALL(POS)
50                    SCHEDULE AN ENDING(CALL,PATH,PERM)
51                    IN DUREE.CALL(POS) UNITS
52                LOOP
53  !!
```

```
54   !!     ON VERIFIE SI ON A BIEN NB TRONCONS (OU APPELS) D'ENVOYES.
55   !!     POUR CE FAIRE ON TESTE LA VARIABLE DIFF.  SI DIFF > 0 ALORS
56   !!     CELA SIGNIFIE QU'ON A MANQUE DE TRONCONS  ET QU'IL FAUT VERIFIER
57   !!     S'IL Y A DES LIENS RESERVES.
58
59   'DIFF'
60             IF DIFF = ZERO
61                GO TO 'LOOP'
62             ELSE
63                FOR EACH RES.LINK WITH
64                    R.SOURCE (RES.LINK) = S  AND
65                    R.DEST   (RES.LINK) = D  AND
66                    R.MODE   (RES.LINK) = M
67                    FIND R.POS = THE FIRST RES.LINK
68                IF NONE
69                    GO TO 'LOOP'
70                ELSE
71   !!                CONTINUE
72
73
74   !!     SI ON VEUT INITIALISER PLUS DE TRONCONS QU'IL Y EN A DE
75   !!     DISPONIBLES, AUCUN MESSAGE N'EST DONNE ET ON LAISSE TOMBER
76   !!     LE NOMBRE DE TRONCONS EN SUS DE LA QUANTITE QUI EXISTE AU
77   !!     DEPART.
78   !!
79
80                LET DIFF = MIN.F(DIFF,R.NB.TRK(R.POS))
81                FOR JJ = 1 TO DIFF DO
82                    CREATE A CALL
83                    LET POSITION(CALL) = POS
84                    CREATE A PATH
85                    CREATE A LINK
86                    LET VALUE(LINK) = R.POS
87                    LET PERM = R.POS
88                    LET R.NB.TRK(PERM) = R.NB.TRK(PERM) - 1
89                    LET D.CALL(POS) = EXPONENTIAL.F(CLASSE,7)
90                    LET DUREE.CALL(POS) = D.CALL(POS)
91                    LET HOLDING.TIME = DUREE.CALL(POS)
92                    SCHEDULE AN ENDING(CALL,PATH,PERM)
93                    IN DUREE.CALL(POS) UNITS
94                LOOP
95   !!
96   'LOOP'
97        LOOP
98   'NO.INIT'
99
100
101    FOR EACH P.LINK DO
102
103         CREATE A CALL
104         LET POSITION(CALL) = R.LINK
105         LET T = I.A.TIME(P.LINK)
106
```

```
107          IF  T LE ZERO
108              GO TO 'NEXT'
109          ELSE
110              SCHEDULE AN ARRIVAL GIVEN CALL
111              IN EXPONENTIAL.F(T,Q) UNITS
112              ADD 1 TO NB.CALL(R,LINK)
113
114  'NEXT'
115
116    LOOP
117
118    RETURN
119    END
```

```
1
2     EVENT ARRIVAL GIVEN CALL
3
4
5     DEFINE T AS A REAL VARIABLE
6
7     LET ATIME(CALL) = TIME.V
8     LET POS = POSITION(CALL)
9
10    SCHEDULE A ROUTING GIVEN CALL IN SWITCHED.TIME UNITS
11
12    CREATE A CALL
13    LET POSITION(CALL) = POS
14    LET T = I.A.TIME(POS)
15    ADD 1 TO NB.CALL(POS)
16
17    SCHEDULE AN ARRIVAL GIVEN CALL
18    IN EXPONENTIAL.F(T,4) UNITS
19
20
21    END
```

```
1
2     EVENT  RECALL  GIVEN CALL
3
4
5     LET ATIME(CALL) = TIME.V
6     LET POS = POSITION(CALL)
7
8     SCHEDULE A ROUTING GIVEN CALL IN SWITCHED,TIME UNITS
9
10    ADD 1 TO NB.CALL(POS)
11
12
13    END
```

```
    1
    2     EVENT  ROUTING  GIVEN CALL
    3
    4   !!
    5   !!      CET EVENEMENT ACHEMINE L'APPEL SELON LA STRATEGIE
    6   !!      S.D.C. (SUCCESSIVE OFFICE CONTROL).
    7   !!
    8
    9     DEFINE X AS A REAL VARIABLE
   10
   11     LET GUIDE = -1
   12     LET PERM = 0
   13
   14     LET POS = POSITION(CALL)
   15     LET S = L.SOURCE(POS)
   16     LET D = L.DEST  (POS)
   17     LET M = L.MODE  (POS)
   18     LET C = 1
   19
   20     LET D.CALL(POS) = EXPONENTIAL.F(CLASSE,9)
   21
   22     CREATE A PATH
   23
   24   'DEBUT'
   25
   26   !!
   27   !!      TEST POUR EVITER LES BOUCLES INFINIES.
   28   !!
   29
   30     IF  N.PATH.QU  GE  NB.LK
   31         PRINT 1 LINE WITH FLAG THUS
               BOUCLE DANS LA ROUTING TABLE.  POSITION = ****
   32         GO TO 'NO.PATH'
   33     ELSE
   34   !!      CONTINUE
   35
   36
   37     LET N = RT(S,D,C)
   38
   39     IF  (N = 0)  OR  (C GREATER THAN NB.CHOIX)
   40         GO TO 'NO.PATH'
   41     ELSE
   42         CALL VERIFY GIVEN S,N,M YIELDING FLAG
   43
   44     IF  FLAG = 0
   45         IF  (C LT NB.CHOIX)
   46             LET C = C + 1
   47             GO TO 'DEBUT'
   48         ELSE
   49             GO TO 'NO.PATH'
   50     ELSE
   51         IF  (N = D)
   52             CREATE A LINK
```

```
53              LET VALUE(LINK) = FLAG
54              FILE LINK IN PATH.QU
55              CALL RESERV (PATH,GUIDE,D,CALL(POS))
56              LET DUREE.CALL(POS) = D.CALL(POS)
57              LET HOLDING.TIME = DUREE.CALL(POS)
58              ADD 1 TO ROUTED(POS)
59              LET QTIME = TIME.V - ATIME(CALL)
60              LET DD.QTIME(POS) = QTIME
61              SCHEDULE AN ENDING (CALL,PATH,PERM)
62              IN DUREE.CALL(POS) UNITS
63              RETURN
64          ELSE
65              CREATE A LINK
66              LET VALUE(LINK) = FLAG
67              FILE LINK IN PATH.QU
68              LET S = RT(S,D,C)
69              LET C = 1
70              GO TO 'DEBUT'
71
72  !!
73  !!          SI ON  NE TROUVE PAS DE CHEMIN...
74  !!
75
76  'NO,PATH'
77
78      FOR EACH LINK IN PATH.QU DO
79
80          REMOVE FIRST LINK FROM PATH.QU(PATH)
81          DESTROY LINK
82
83      LOOP
84
85
86      LET S = L.SOURCE(POS)
87      LET D = L.DEST  (POS)
88
89  !!
90  !!          TEST S'IL Y A DES RESERVATIONS.
91  !!
92
93
94      LET SO = MIN.F(S,D)
95      LET DE = MAX.F(S,D)
96
97      FOR EACH RES.LINK WITH
98          R.SOURCE(RES.LINK) = SO  AND
99          R.DEST  (RES.LINK) = DE  AND
100         R.MODE  (RES.LINK) = M
101         FIND R.POS = THE FIRST RES.LINK
102
103     IF  NONE
104         LET R.POS = 0
105     ALWAYS
```

```
105
106
107     IF R.POS NE ZERO
108         IF R.NB.TRK(R.POS) GT ZERO
109             LET R.NB.TRK(R.POS) = R.NB.TRK(R.POS) - 1
110             CREATE A LINK
111             LET VALUE(LINK) = R.POS
112             FILE LINK IN PATH.QU
113             LET PERM = R.POS
114             LET DUREE.CALL(POS) = D.CALL(POS)
115             LET HOLDING.TIME = DUREE.CALL(POS)
116             LET QTIME = TIME.V - ATIME(CALL)
117             LET QD.QTIME(POS) = QTIME
118
119             SCHEDULE AN ENDING(CALL,PATH,PERM)
120             IN DUREE.CALL(POS) UNITS
121             ADD 1 TO ROUTED(POS)
122             ADD 1 TO RESERVED(POS)
123
124             RETURN
125         ELSE                !!  AUCUN LIEN DE DISPONIBLE
126
127     ALWAYS                  !!  PAS DE RESERVATION.
128
129
130     DESTROY PATH
131
132 !!
133 !!          TRAITEMENT DU MODE
134 !!
135
136     IF L.MODE(POS) = DIRECT
137         LET X = UNIFORM.F(0.0,100.0,3)
138
139         IF X LE PC.LOST
140             ADD 1 TO LOST(POS)
141             DESTROY CALL
142             RETURN
143         ELSE
144             SCHEDULE A RECALL GIVEN CALL IN
145             EXPONENTIAL.F(REATTEMPS.TIME,1) UNITS
146 !!
147             !!  UN APPEL RE-ESSAYE EST CONSIDERE COMME N'AYANT PAS ATTENDU
148 !!
149             LET QTIME = 0.0
150             LET QD.QTIME = 0.0
151             ADD 1 TO REESSAI(POS)
152             RETURN
153     ELSE            !!  C'EST UN APPEL STORE-AND-FORWARD
154
155             FILE CALL IN LINK.QU(POSITION)
156             ADD 1 TO STORED(POS)
157     RETURN
158     END
```

```
    1
    2    ROUTINE TO VERIFY GIVEN N.SOURCE, NODEUD, M YIELDING POS
    3
    4
    5    LET POS = 0
    6    LET S = MIN.F(N.SOURCE,NODEUD)
    7    LET D = MAX.F(N.SOURCE,NODEUD)
    8
    9
   10    FOR EACH P.LINK
   11        WITH L.SOURCE(P.LINK) = S  AND
   12             L.DEST  (P.LINK) = D  AND
   13             L.MODE  (P.LINK) = M
   14        FIND POS = THE FIRST P.LINK
   15
   16    IF  NONE
   17        LET POS = 0
   18        RETURN
   19
   20    ELSE
   21
   22        ADD 1 TO LK.CALL(POS)
   23
   24        IF  NB.TRUNKS(POS) IS LE ZERO
   25            ADD 1 TO LK.BLOCKED(POS)
   26            LET POS = 0
   27        ALWAYS
   28
   29    RETURN
   30    END
```

```
  1
  2     ROUTINE TO RESERV GIVEN PATH,GUIDE, LK,DUREE
  3
  4
  5   !!
  6   !!      CETTE ROUTINE RESERVE (OU LIBERE) LES LIENS POUR L'APPEL
  7   !!      EN COURS.
  8   !!
  9   !!      LA VARIABLE PATH CONTIENT LES LIENS UTILISES PAR L'APPEL (CALL),
 10   !!      GUIDE INDIQUE SI ON DOIT RESERVER (-1) OU LIBERER (+1) UN LIEN.  LA
 11   !!      DERNIERE VARIABLE (LK,DUREE) C'EST LA DUREE DE L'APPEL EN COURS;
 12   !!      ELLE SERT POUR LES STATISTIQUES.
 13   !!
 14   !!
 15
 16     DEFINE LK,DUREE AS A REAL VARIABLE
 17
 18     IF  PATH,OU  IS EMPTY
 19         GO TO 'RETOUR'
 20     ALWAYS
 21
 22     FOR I = 1 TO N,PATH,OU(PATH)
 23         DO
 24             REMOVE FIRST LINK FROM PATH,OU(PATH)
 25             LET LK = VALUE(LINK)
 26             LET NB,TRUNKS(LK) = NB,TRUNK$(LK) + GUIDE
 27   !!
 28   !!        ON TRAITE LE "CARRIED LOAD" POUR CHAQUE LIEN.
 29   !!
 30             IF  GUIDE EQ -1
 31                 LET LK,LOAD(LK) = LK,DUREE
 32                 ADD 1 TO NB,CARRIED(LK)
 33             ALWAYS
 34   !!
 35
 36             FILE LINK IN PATH,OU(PATH)
 37         LOOP
 38
 39     RETURN
 40
 41   'RETOUR'
 42
 43         PRINT 1 LINE THUS
EMIN VIDE
 44
 45
 46     RETURN
 47     END
```

```
    1
    2     EVENT  ENDING  GIVEN @ALL,PATH,PERM
    3
    4
    5     LET POS = POSITION(CALL)
    6
    7     IF  PERM GT ZERO      '' APPEL RESERVE
    8         ADD 1 TO R.NB.TRK(PERM)
    9         GO TO 'SUITE'
   10     ELSE
   11         LET GUIDE = 1
   12         CALL RESERV(PATH,GUIDE)
   13
   14
   15   'SUITE'
   16
   17     FOR EACH LINK IN PATH,QU DESTROY LINK
   18     DESTROY PATH
   19     DESTROY CALL
   20
   21     IF  LINK.QU(POS) IS NOT EMPTY
   22         REMOVE FIRST CALL FROM LINK.QU(POS)
   23         SCHEDULE A ROUTING GIVEN CALL NOW
   24         ADD 1 TO RE.ROUTED(POS)
   25
   26     ALWAYS
   27
   28     ADD 1 TO FINISH(POS)
   29
   30
   31     END
```

```
EVENT  INITIAL,STATE

!!
!!     CET EVENEMENT IMPRIME L'ETAT INITIAL DU SYSTEME  ET
!!     REMET TOUS LES COMPTEURS A ZERO
!!

   WRITE PAGE.V AS /,/,/,S 120,"PAGE ",I 3,/,/,/,/,/,/
   WRITE A$ S 38,
         "T E L E P H O N E   N E T W O R K   S I M U L A T I O N",/,
         S 38,
         "=========================================",/,/,/,/,
         /,/,/,/,/,/,/
   WRITE A$ S 10,"SYSTEM DESCRIPTION AND USER OPTIONS",/,
            S 10,"--------------------------------",/,/,/,/,/
   PRINT 5 LINES WITH NB.NODE,NB.LK THUS
         NUMBER OF NODES              :  ****

         NUMBER OF LINKS              :  ****

         ROUTING STRATEGY             :  S.D.C.
   WRITE UNIT(1),UNIT(2),UNIT(3),UNIT(4) AS /,S 15,"TIME UNIT",
         S 17,"=  ",4 A 4,/,/,/
   PRINT 11 LINES WITH SWITCHED.TIME, PC.LOST,TAUX, REATTEMPS.TIME,
         TRANS.TIME, STOP.TIME THUS
         SWITCHING TIME               :  ******.**  UNIT(S)

         PERCENTAGE OF RETRY CALLS    :     ***.** %

         INITIAL RATE OF TRAFFIC      :     ***.**

         MEAN RETRY TIME              :  ******.**  UNIT(S)

         TRANSCIENT  TIME             :  ******.**  UNIT(S)

         SIMULATION TIME              :  *******.*  UNITS

   SKIP 3 LINES

   PRINT 1 DOUBLE LINE THUS
         CALL CLASS                   PROBABILITY     MEAN HOLDING TIME (UNITS)
   SKIP 2 LINES
   LET CL = F.CLASSE
   LET PROB = 0.00

   FOR I = 1 TO 500 WHILE CL GT ZERO DO
       LET PROB = PROB.A(CL) - PROB
       PRINT 1 LINE WITH PROB, RVALUE.A(CL) THUS
                                    *.***                  ******.*
       LET PROB = PROB.A(CL)
       LET CL = S.CLASSE(CL)
```

```
    LOOP
!!
!!      IMPRESSION DES LIENS
!!
  BEGIN REPORT ON A NEW PAGE
        LET FANION = 0
        BEGIN HEADING
            WRITE PAGE.V AS /,/,/,$ 120,"PAGE ",I 3,/,/,/,/
            IF  FANION EQ ZERO
                WRITE AS $ 15,"SYSTEM INITIAL STATE",/,
                      $ 15,"------------------",/,/,/,/,/,/
                LET FANION = 1
            ALWAYS

            PRINT 2 DOUBLE LINES THUS
ORIGIN          DEST.          MODE          NB OF BUSY TRUNKS          NB OF IDLE TRUNKS          NB OF CALLS IN QUEUE          INTERARRIVAL TIME

        SKIP 2 LINES
        END !! HEADING


        FOR EACH P,LINK DO
            IF  I.A.TIME(P,LINK) LT ZERO LET I.A.TIME(P,LINK) = 0.0
            ALWAYS


            PRINT 1 DOUBLE LINE WITH L.SOURCE(P,LINK), L.DEST(P,LINK),
                  L.MODE(P,LINK), NB.TRK(P,LINK)-NB.TRUNKS(P,LINK),
                  NB.TRUNKS(P,LINK),N.LINK.QU(P,LINK),I.A.TIME(P,LINK)
                  THUS
**                **              *               ****                ****                ****.***

        LOOP

    END  !!  REPORT


!!
!!      IMPRESSION DES LIENS RESERVES.
!!

    IF  N.RES.LINK  EQ  ZERO
        GO TO 'NXREP'
    OTHERWISE
!       CONTINUE
```

```
LINE  OACI SIMSCRIPT II.5  RELEASE 8H                              05/25/81  PAGE  31

   86
   87    BEGIN REPORT ON A NEW PAGE
   88
   89        BEGIN HEADING
   90        WRITE PAGE.V AS /,/,/,S 120,"PAGE ",I 3,/,/,/,/,/,/
   91        WRITE A$ S 15, "IMPRESSION DES LIENS RESERVES",/,
   92                 S 15, "-----------------------------",/,/,/,/,/
   93
   94        PRINT 1 DOUBLE LINE THUS
                                  SOURCE          DEST          MODE                          NB TRKS
   95        SKIP 2 LINES
   96        END  ''HEADING
   97
   98
   99    FOR EACH RES.LINK  DO
  100
  101        PRINT 1 DOUBLE LINE WITH R.SOURCE(RES.LINK),
  102        R.DEST(RES.LINK),  R.MODE(RES.LINK),
  103        R.NB.TRK(RES.LINK)  THUS
                                  **              **            **                            ***
  104
  105    LOOP
  106
  107    END  ''REPORT
  108
  109
  110  !NXREP!
  111
  112
  113  ''
  114  ''     IMPRESSION DE LA TABLE D'ACHEMINEMENT
  115  ''
  116
  117    WRITE PAGE.V AS 1 ,/,/,/, S 120, "PAGE " , I 3 , /,/,/,/,/,/
  118    WRITE A$ S 59,"ROUTING  TABLE",/,
  119             S 59,"--------------",/,/,/,/,/,/
  120    WRITE A$ S 20, "ORIGIN          DEST,    "
  121    FOR K=1 TO NB.CHOIX DO
  122        WRITE K AS 2 S , "CHOIX NO, "; I 2
  123    LOOP
  124
  125    SKIP 3 OUTPUT LINES
  126
  127
  128    FOR I = 1 TO NB.NODE DO
  129        FOR J = 1 TO NB.NODE DO
  130
  131            IF  I = J
  132               GO TO 'LOOP'
  133            ELSE
  134  ''          CONTINUE
  135
  136            WRITE I,J AS S 22, I 2, S 13, I 2, S 14
```

```
            FOR K = 1 TO NB.CHOIX DO
                WRITE RT(I,J,K) AS I 2, S 14
            LOOP
            SKIP 1 OUTPUT LINE
        !LOOP1
        LOOP
    LOOP


!!
!!      INITIALISATION DES COMPTEURS.
!!

    FOR EACH P.LINK DO

        LET NB.CALL(P.LINK) = 0.0
        LET ROUTED(P.LINK) = 0.0
        LET STORED(P.LINK) = 0.0
        LET LOST(P.LINK) = 0.0
        LET REESSAI(P.LINK) = 0.0
        LET RE.ROUTED(P.LINK) = 0.0
        LET RESERVED(P.LINK) = 0.0
        LET FINISH(P.LINK) = 0.0
        LET NB.CARRIED(P.LINK) = 0.0

        RESET TOTALS OF N.LINK.QU(P.LINK)
        RESET TOTALS OF NB.TRUNKS(P.LINK)
        RESET TOTALS OF DUREE.CALL(P.LINK)
        RESET TOTALS OF D.CALL(P.LINK)
        RESET TOTALS OF OD.QTIME(P.LINK)
        RESET TOTALS OF LK.LOAD(P.LINK)

    LOOP

    RESET TOTALS OF QTIME
    RESET TOTALS OF HOLDING.TIME
    LET I.CALL = N.EV.S(I.ENDING)


    END
```

```
EVENT  CHANGE  GIVEN TAUX

DEFINE TAUX AS A REAL VARIABLE

LET TAUX = TAUX.PREC/TAUX

FOR EACH P,LINK DO
    LET I.A.TIME(R.LINK) = I.A.TIME(P,LINK) * TAUX
LOOP

LET TAUX.PREC = TAUX.PREC / TAUX

RETURN
END
```

```
1
2       EVENT TRACE$
3
4
5     !!
6     !!      EVENEMENT DONNANT DE L'INFORMATION SUR L'ETAT DU $YSTEME
7     !!      A CERTAINS MOMENTS DE LA SIMULATION.
8     !!
9
10
11      DEFINE OFFERED.LOAD, CARRIED.LOAD, TOTAL.OFFER, TOTAL.CARRY,
12             PROB.BLOCK AS REAL VARIABLES
13      DEFINE PC.UTIL, PROB AND G.O,S AS REAL VARIABLES
14
15      LET S.NB.CALL   = I.CALL
16      LET S.ROUTED    = 0
17      LET S.STORED    = 0
18      LET S.RE.ROUTED = 0
19      LET S.LOST      = 0
20      LET S.REESSAI   = 0
21      LET S.RESERVED  = 0
22      LET S.FINISH    = 0
23
24
25
26
27      WRITE PAGE.V AS /,/,/, 8 120,"PAGE ",I 3,/,/,/
28      SKIP 5 OUTPUT LINES
29
30      PRINT 1 LINE THUS
                                                  ##### START   OF   TRACE #####
31      SKIP 10 OUTPUT LINES
32
33    !!
34    !!IMPRESSION DU CONTENU DE LA LISTE DES EVENEMENTS,
35    !!
36
37      PRINT 2 LINE WITH TIME.V - STEADY.STATE, TAUX   THUS
              SIMULATION TIME :   ******.* UNIT(S)
              RATE OF TRAFFIC :     ***.**
38      SKIP 5 LINES
39
40
41      LET NB1 = N.EV.$(I.ARRIVAL)
42      LET NB2 = N.EV.$(I.ROUTING)
43      LET NB3 = N.EV.$(I.ENDING)
44      LET NB4 = N.EV.$(I.RECALL)
45      LET NB5 = N.EV.$(I.TRACES)
46      LET NB6 = N.EV.$(I.CHANGE)
47      LET TOTAL = NB1 + NB2 + NB3 + NB4 + NB5 + NB6 + 1
48
49
50      PRINT 1+ LINES WITH NB1, NB2, NB3, NB4, NB5, NB6  THUS
```

```
LINE  GACI SIMSCRIPT II.5  RELEASE 8H                                   05/25/81  PAGE  37
                    CONTENTS OF THE EVENT LIST

                    ##### NUMBER OF ARRIVAL EVENTS          !   *****
                    ##### NUMBER OF ROUTING EVENTS          !   *****
                    ##### NUMBER OF ENDING EVENTS           !   *****
                    ##### NUMBER OF RECALL EVENTS           !   *****
                    ##### NUMBER OF TRACES EVENTS           !   *****
                    ##### NUMBER OF CHANGE (OF RATE) EVENTS !   *****

      SKIP 7 LINES

      PRINT 1 LINE WITH TOTAL THUS
                    ##### TOTAL NUMBER OF EVENTS IN THE LIST !  ******



''
''IMPRESSION DE L'ETAT DU RESEAU
''

      BEGIN REPORT ON A NEW PAGE

          LET FANION = 0
          BEGIN HEADING

              WRITE PAGE.V AS /,/,/,5 120,"PAGE ",I 3,/,/,/,/
              IF  FANION EQ ZERO
                  WRITE AS 3 55,"NETWORK PROGRESS REPORT",/,
                            5 55,"-------------------------",/,/
                  LET FANION = 1
              ALWAYS

      PRINT 2 LINE WITH TIME.V - STEADY-STATE- TAUX  THUS
            SIMULATION TIME !    ******.** UNIT(S)
            RATE OF TRAFFIC !      ***.**
              SKIP 3 LINES

              PRINT 2 DOUBLE LINES THUS
      ORIGIN        DEST        MODE        NB. CALLS       ROUTED       LOST       FINISHED       NB. IN       NB. OF BUSY
                                                                                                    QUEUE          TRUNKS
              SKIP 3 LINES

          END  ''  HEADING

          FOR EACH P.LINK DO
              PRINT 1 DOUBLE LINE WITH L.SOURCE(P.LINK), L.DEST(P.LINK),
                    L.MODE(P.LINK), NB.CALL(P.LINK), ROUTED(P.LINK),
```

```
LINE  OACI SIMSCRIPT II.5  RELEASE 8H                                   05/25/81  PAGE: 38

  85                  LOST(P,LINK), FINISH(P,LINK), STORED(P,LINK),
  86                  NB.TRK(P,LINK)-NB.TRUNKS(P,LINK)  THUS
        **            **          *          ******       ******     ****       ******       ******       *****
  87
  88
  89        LET S.NB.CALL = S.NB.CALL + NB.CALL(P.LINK)
  90        LET S.ROUTED = S.ROUTED + ROUTED(P.LINK)
  91        LET S.STORED = S.STORED + STORED(P.LINK)
  92        LET S.RE.ROUTED = S.RE.ROUTED + RE.ROUTED(P.LINK)
  93        LET S.LOST = S.LOST + LOST(P.LINK)
  94        LET S.REESSAI = S.REESSAI + REESSAI(P.LINK)
  95        LET S.RESERVED = S.RESERVED + RESERVED(P.LINK)
  96        LET S.FINISH = S.FINISH + FINISH(P.LINK)
  97
  98        LOOP
  99
 100    END  ''  REPORT
 101
 102
 103  ''
 104  ''IMPRESSION DES STATISTIQUES SUR LES PERFORMANCES DU RESEAU.
 105  ''
 106
 107        IF  (TIME.V - STEADY.STATE) LE ZERO
 108            GO TO 'GLOBAL'
 109        ELSE
 110  ''        CONTINUE
 111
 112
 113    BEGIN REPORT ON A NEW PAGE
 114
 115        LET PANION = 0
 116        BEGIN HEADING
 117
 118            WRITE PAGE.V AS /,/,/,$ 120,"PAGE ",I 3,/,/,/,/
 119            IF  PANION EQ ZERO
 120                WRITE AS $ 51,"NETWORK PERFORMANCE STATISTICS",/,
 121                          $ 51,"-------------------------",/,/
 122                LET PANION = 1
 123            ALWAYS
 124
 125    PRINT 2 LINE WITH TIME.V - STEADY.STATE, TAUX  THUS
           SIMULATION TIME !    *******.* UNIT(S)
           RATE OF TRAFFIC !      ***.**
 126           SKIP 3 LINES
 127
 128           PRINT 3 DOUBLE LINES THUS
  ORIGIN            DEST              MODE          OFFERED LOAD (CCS)      CARRIED LOAD (CCS)      AVERAGE WAITING TIME (UNITS)      O.D.G.O.S.
 129           SKIP 3 LINES
 130
 131        END  ''  HEADING
```

```
132
133         LET TOTAL.OFFER = 0.0
134         LET TOTAL.CARRY = 0.0
135
136         FOR EACH P.LINK DO
137
138             LET G.O.S = 0.0
139             IF  NB.CALL(P.LINK) NE ZERO
140                 LET G.O.S = LOST(P.LINK) / NB.CALL(P.LINK)
141             ALWAYS
142
143             LET OFFERED.LOAD = (AV.D.CALL(P.LINK)*NB.CALL(P.LINK)*36.0)
144                 / (TIME.V - STEADY.STATE)
145             LET TOTAL.OFFER = TOTAL.OFFER + OFFERED.LOAD
146             LET CARRIED.LOAD = (AVG.DUREE(P.LINK)*ROUTED(P.LINK)*36.0)
147                 / (TIME.V - STEADY.STATE)
148             LET TOTAL.CARRY = TOTAL.CARRY + CARRIED.LOAD
149
150             PRINT 1 DOUBLE LINE WITH L.SOURCE(P.LINK), L.DEST(P.LINK),
151                   L.MODE(P.LINK), OFFERED.LOAD, CARRIED.LOAD, AV.OD.QT,
152                   G.O.S    THUS
**                 **              *            *.****          **.****          **.**          *.****
153
154         LOOP
155
156     END '' REPORT
157
158
159 ''
160 ''IMPRESSION DES STATISTIQUES SUR LES PERFORMANCES DES LIENS.
161 ''
162
163     BEGIN REPORT ON A NEW PAGE
164
165         LET FANION = 0
166         BEGIN HEADING
167
168             WRITE PAGE.V AS /,/,/,S 120,"PAGE ",I 3,/,/,/,/
169             IF  FANION EQ ZERO
170                 WRITE AS S 52,"LINK PERFORMANCE STATISTICS",/,
171                       S 52,"--------------------------",/,/
172                 LET FANION = 1
173             ALWAYS
174
175     PRINT 2 LINE WITH TIME.V - STEADY.STATE, TAUX  THUS
            SIMULATION TIME I    ********.** UNIT(S)
            RATE OF TRAFFIC I      ***.**
176           SKIP 3 LINES
177
178           PRINT 3 DOUBLE LINE THUS
      FROM NODE     TO NODE     MODE     NB. OF TRUNKS     AVERAGE OF        PERCENTAGE      CARRIED     LINK BLOCKING
                                           (INITIAL)       AVAILABLE         OF  TRUNKS        LOAD       PROBABILITY
                                                            TRUNKS           UTILISATION
```

```
179             SKIP 3 LINES
180
181         END  '' HEADING
182
183         FOR EACH P.LINK WITH NB.TRK(P.LINK) GT ZERO DO
184
185             LET CARRIED.LOAD = (AV.LK.LOAD(P.LINK)*NB.CARRIED(P.LINK)
186                 * 36.0) / (TIME.V - STEADY.STATE)
187
188             IF  LK.CALL(P.LINK) LE ZERO
189                 GO TO 'BOUCLE'
190             ELSE
191 ''              CONTINUE
192             LET PROB.BLOCK = LK.BLOCKED(P.LINK) / LK.CALL(P.LINK)
193
194             LET PC.UTIL = 100.0 - (AVG.TRK(P.LINK)*100.0/NB.TRK(P.LINK))
195
196             PRINT 1 DOUBLE LINE WITH L.SOURCE(P.LINK), L.DEST(P.LINK),
197                   L.MODE(P.LINK),NB.TRK(P.LINK),AVG.TRK(P.LINK), PC.UTIL,
198                   CARRIED.LOAD, PROB.BLOCK  THUS
      **            **          *             ***             ***.**          ***.**          ****.**         *.****
199
200         'BOUCLE'
201         LOOP
202
203     END  '' REPORT
204
205
206 ''
207 ''RESULTATS GLOBAUX SUR LE RESEAU
208 ''
209
210     'GLOBAL'
211     WRITE AS *
212     WRITE PAGE.V AS /,/,/,S 120,"PAGE ",I 3,/,/,/
213     WRITE AS S 50,"SUMMARY OF NETWORK PERFORMANCES",/,
214           S 50,"--------------------------------",/,/,/,/,/,/
215     PRINT 2 LINE WITH TIME.V - STEADY.STATE, TAUX  THUS
            SIMULATION TIME I    ******.* UNIT(S)
            RATE OF TRAFFIC I      ***.**
216     SKIP 8 LINES
217
218     PRINT 15 LINES WITH S.NB.CALL, S.ROUTED, S.STORED, S.RE.ROUTED,
219           S.LOST, S.REESSAI, S.RESERVED AND S.FINISH THUS
                  TOTAL SUM OF CALLS I  ******

                  ROUTED CALLS       I  ******

                  STORED CALLS       I  ******

                  RE-ROUTED CALLS    I  ******

                  LOST CALLS         I  ******
```

```
                RETRIED CALLS          :  ******
                RESERVED CALLS         :  ******
                FINISHED CALLS         :  ******
220
221     SKIP 5 LINES
222
223
224     LET S.NB.CALL = S.NB.CALL - I.CALL
225
226         IF  S.NB.CALL EQ ZERO GO TO 'NO.CALL' ELSE
227     PRINT 9 LINES WITH AV.HOLDING,T.WQ,S.LOST/S,NB.CALL,TOTAL.OFFER,
228           TOTAL.CARRY  THUS
                MEAN HOLDING TIME              :  *****.**
                MEAN WAITING TIME              :  *****.**
                GRADE OF SERVICE (G.O.S.)      :  *.******
                TOTAL OFFERED LOAD (OCS)       :  *****.**
                TOTAL CARRIED LOAD (OCS)       :  *****.**
229
230     'NO.CALL'
231
232
233     SKIP 3 LINES
234         PRINT 1 LINE THUS
                                                ###### END   OF  TRACE ######
235
236     RETURN
237     END
```

```
    1
    2     EVENT  END.OF.SIMULATION
    3
    4
    5
    6   !!
    7   !!      ON PEUT REFAIRE UNE AUTRE SIMULATION AVEC LE MEME RESEAU.
    8   !!
    9
   10     IF  DATA IS ENDED  GO TO 'ARRET'
   11     ELSE
   12          READ TPS
   13          SCHEDULE AN INITIAL.STATE IN TPS UNITS
   14          LET STEADY.STATE = TIME.V + TPS
   15          LET TRANS.TIME = TPS
   16          READ TAUX
   17          SCHEDULE A CHANGE GIVEN TAUX NOW
   18          READ NB.TRACE
   19          IF   NB.TRACE EQ ZERO GO TO 'NO.TRACE'
   20               ELSE
   21
   22          FOR J = 1 TO NB.TRACE DO
   23               READ TIME.TR
   24               SCHEDULE A TRACES IN (TIME.TR + TPS ) UNITS
   25
   26          LOOP
   27   'NO.TRACE'
   28
   29          READ STOP.TIME
   30          SCHEDULE A TRACES IN (STOP.TIME + TPS) UNITS
   31          SCHEDULE AN END.OF.SIMULATION IN (STOP.TIME+TPS) UNITS
   32          RETURN
   33
   34
   35
   36   'ARRET'
   37
   38     WRITE  PAGE.V AS #,/,/,/,S 120, "PAGE ", I 3,/,/
   39
   40     SKIP 25 OUTPUT LINES
   41     PRINT 1 DOUBLE LINE THUS
                                              ##### FIN NORMALE DE LA SIMULATION #####
   42
   43     STOP
   44     END
```

BIBLIOGRAPHIE

Livres et sources spéciales

Gordon, Geoffroy, System Simulation, New-Jersey, Prentice-Hall, 1978, 2e édition, 324 pages

Kiviat, P.J., R. Villanueva, H.M. Markowitz SIMSCRIPT II.5 Programming Language, Los Angeles, CACI, 1975, 384 pages.

Lin, P.M., B.J. Leon, H. Thapar, C. Stewart, STARTUP-a statistical analysis and routing table updating program for european AUTOVON, School of electrical engineering, Purdue University, Ind., Tech. Rept. TR-EE 76-22, August 1976, 315 pages.

Notes on Distance Dialing, Section I-II-III, New-York, American Telephone and Telegraph Company, 1975.

Articles et périodiques

Lin, P.M., Benjamin J. Leon, Chris R. Stewart,
"Analysis of circuit-switched networks employing originating - Office control with spill-forward".
IEEE transactions on communications, Vol. COM-26, No 6, June 1978, pp. 754-765.

Weber, J.H. "A simulation study of routing and control in communications networks", The Bell System Technical Journal, Vol. 43, November 1964, pp. 2639-2676.

Weber, J.H., "Some traffic characteristics of communication networks with automatic alternate routing"., The Bell System Technical Journal, Vol 41, March 1962, pp. 769-796.

Wilkov, R.S., "An analytical approch to traffic analysis in maximally reliable communication networks" , Proc. of the UMR, 1970, pp. 22.5.1 - 22.5.7.